SELF-SUPERVISED OBJECT-CENTRIC CROSS-VIEW REASONING FROM A SINGLE IMAGE

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce **Particle-based Object Representation Transport (PORT)**, a self-supervised object-centric model that directly transports object-centric latent representations extracted from a single source image to target viewpoints according to relative camera pose. Rather than merely generating novel-view images, PORT directly transports each object latent to predict its position, scale, depth, and appearance under different horizontal camera viewpoints. This enables PORT to construct viewpoint-conditioned multi-view object representations from a single image while preserving object correspondence across views, enabling downstream 3D reasoning by leveraging object latents across multiple viewpoints. We provide object-level evaluation metrics and a protocol for quantitatively assessing the accuracy of cross-view latent transport. We then demonstrate the utility of the resulting multi-view object representations in robotic manipulation and Gaussian Splatting, showing their applicability to both 3D scene representation and downstream tasks. Our code will be publicly available. Videos are available at `https://sites.google.com/view/particletransport`.

1 INTRODUCTION

Object-centric representations decompose a scene into a set of object-level entities that are each represented separately. This object-level decomposition captures compositional structure and has been shown to improve sample efficiency and generalization (Wiedemer et al., 2024; Zhang et al., 2026b). Consequently, object-centric models have been successfully applied to a wide range of downstream tasks, including video prediction (Daniel & Tamar, 2024), reinforcement learning (Haramati et al., 2024), and imitation learning (Qi et al., 2025). In robotics, representing individual objects and their states can provide a more structured basis for decision making and interaction than treating the scene as a single global representation (Haramati et al., 2024; Qi et al., 2025; Daniel et al., 2026).

However, many object-centric representations are learned primarily from 2D RGB images and do not explicitly capture geometric cues such as depth and 3D spatial relationships. Such geometric information is important for downstream tasks that require reasoning and decision making in 3D environments, like robotic manipulation (Shridhar et al., 2023; Ze et al., 2024). To incorporate richer geometric information, prior work has explored 3D object-centric representations through point-cloud-based models (Wang et al., 2022), neural rendering approaches (Luo et al., 2025; Stelzner et al., 2021b), models that operate on explicit 3D observations such as RGB-D images or voxel grids (Zhang et al., 2026a), and methods that leverage multi-view images (Li et al., 2020; Yuan et al., 2022). Yet, explicitly modeling 3D representations or processing multi-view images can increase representation complexity and computational cost, which may be undesirable for downstream settings that require efficient inference and policy learning (Tian et al., 2023; Zhang et al., 2026a).

Hence we ask the question: can we instead enrich object-centric representations with richer geometric information from a single RGB image, without relying on explicit 3D representations or multi-view images at inference time? In human visual reasoning, humans can anticipate how positions and spatial configuration of objects would change under a different viewpoint (Pinker & Finke, 1980), and this ability is consistent with humans forming and manipulating internal representations that preserve 3D spatial relationships (Pinker, 1980). Inspired by this capability, we hypothesize that learning to predict how object states—including position, scale, depth, and visibility—transform un-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

der changes in camera viewpoint, while preserving object correspondence across views, can equip 2D object-centric representations with 3D-relevant geometric structure.

In this work, we introduce **Particle-based Object Representation Transport (PORT)**, a self-supervised object-centric model for reasoning about object states across camera viewpoints from a single image. Given a viewpoint change, PORT directly transforms each object's latent representation to predict its position, scale, depth, visibility, and appearance at the target viewpoint. By maintaining source-object correspondence throughout this transformation, PORT constructs viewpoint-conditioned object-centric multi-view representations from a single image. It builds on the efficient particle-based object-centric representation introduced by Daniel et al. (2026). Existing evaluations of object-centric representations primarily focus on reconstruction quality or the quality of latents independently inferred at each viewpoint, making it difficult to assess how accurately a source object latent is transformed into a state at the target viewpoint corresponding to the same physical object. To address this, we introduce a dataset construction and evaluation protocol, together with object-centric metrics that measure how accurately object position and scale are predicted at target viewpoints while preserving source-object correspondence. We further compare 3D coordinates recovered from object locations predicted across multiple viewpoints with the ground truth to assess how well the learned representation reflects the underlying 3D geometry. Finally, we apply PORT to downstream tasks such as robotic manipulation and Gaussian Splatting (Kerbl et al., 2023) to evaluate whether the learned object-centric multi-view representation provides useful information for downstream tasks.

Our contributions are summarized as follows: (1) We propose PORT, a self-supervised object-centric model that generates viewpoint-conditioned multi-view representations from a single RGB image via camera-conditioned object transport; (2) We show that the learned representations capture 3D geometric information and introduce an object-centric dataset construction and evaluation protocol for measuring cross-view geometric consistency; and (3) We demonstrate the utility of transported representations for downstream tasks, including Gaussian Splatting and robotic manipulation, without requiring additional multi-view or explicit 3D observations at inference time.

2 Related Work

Our work learns to transform object-centric representations across camera viewpoints from a single image, using novel-view reconstruction as a self-supervised training signal. In the following, we review related works in self-supervised object-centric learning, multi-view and 3D object-centric representation learning, and single-image novel view synthesis.

Self-Supervised Object-Centric Representation Learning. Self-supervised object-centric learning aims to decompose visual scenes into latent entities corresponding to individual objects without object-level supervision. Existing approaches can be broadly categorized as follows: **Patch-based approaches** (Lin et al., 2020b; Crawford & Pineau, 2019; Stanić & Schmidhuber, 2019) infer object latents from local image regions; **Slot-based approaches** (Locatello et al., 2020; Burgess et al., 2019; Engelcke et al., 2020; Greff et al., 2019) assign visual information to a predefined set of latent slots through attention; **Particle-based approaches** (Daniel & Tamar, 2022; 2024; Daniel et al., 2026) represent objects using learned keypoint-centered latent particles. In particular, particle-based representations provide compact and interpretable object representations in which spatial and appearance-related properties are explicitly parameterized as separate attributes. DLP (Daniel & Tamar, 2022) introduced particles that separate object position from appearance, with subsequent variants incorporating additional attributes such as scale, depth, and presence. This explicit attribute-wise representation is particularly suitable for our setting, where geometric object states must be transformed according to changes in camera viewpoint. DDLP (Daniel & Tamar, 2024) extended particle representations to object-centric video prediction, and LPWM (Daniel et al., 2026) further developed them into a self-supervised object-centric world model with richer particle attributes and temporal dynamics. Our work builds on the DLPv3 representation introduced in LPWM, focusing on how object states transform under changes in camera viewpoint.

Object-Centric Multi-View and 3D Representation Learning. Prior work has extended object-centric representations to 3D through several directions. Object-centric neural rendering approaches construct object-level neural fields or radiance representations and render them from novel viewpoints (Yu et al., 2021; Qi et al., 2023; Stelzner et al., 2021a; Luo et al., 2025; Liu et al., 2024a),

while multi-view representation learning methods leverage observations from multiple viewpoints to learn viewpoint-consistent object representations (Sajjadi et al., 2022; Tian et al., 2023; Wang et al., 2026). Other approaches directly operate on explicit 3D observations such as RGB-D images or voxel grids to obtain object-level 3D representations (Wang et al., 2022; Zhang et al., 2026a). These approaches obtain richer object-level geometry by constructing implicit or explicit 3D representations, or by leveraging observations from multiple viewpoints. Consequently, an object's state at a novel viewpoint is typically derived through an intermediate 3D representation or from multi-view information, making the resulting geometry dependent on the quality of that representation or the availability of additional observations. Rather than requiring a complete 3D reconstruction or additional observations at inference time, we directly learn how the geometric state of each source object transforms as a function of the target camera condition.

Novel View Synthesis from a Single Image. Recent single-image novel view synthesis methods leverage strong priors from large pre-trained generative models to synthesize target views conditioned on relative camera pose, with subsequent work extending this paradigm to full-scene generation and improved multi-view consistency (Liu et al., 2023; Sargent et al., 2023; Seo et al., 2024; Liu et al., 2024b; Gao et al., 2024). Among these methods, MOVIS (Lu et al., 2025) particularly addresses object placement and cross-view inconsistency in compositional multi-object scenes by incorporating depth and object-mask cues. These methods primarily focus on generating high-quality target-view images through diffusion-based generation rather than learning their internal latent representations as persistent object-level representations. In contrast, our goal is to learn at the representation level how the state of the same object changes under camera viewpoint transformations. To this end, we maintain each source object as a persistent entity and transform its state across viewpoints conditioned on the relative camera pose. This transformation is learned without explicit geometric supervision, using novel-view reconstruction as a self-supervised training signal.

3 Background

Deep Latent Particles (DLP). DLP (Daniel & Tamar, 2022; 2024) is a self-supervised object-centric model formulated as a variational autoencoder (VAE Kingma & Welling (2014)), where the latent space is structured as a set of particles representing localized entities in an image. Given an RGB image $I \in \mathbb{R}^{3\times H\times W}$, DLP encodes it into M foreground particles $\{z_{\text{fg}}^m\}_{m=1}^M$ and a single background latent z_{bg}. Following the latest DLP formulation, DLPv3 (Daniel et al., 2026), each foreground particle is represented by a set of disentangled stochastic attributes $z_{\text{fg}} = [z_p, z_s, z_d, z_t, z_f] \in \mathbb{R}^{6+d_{\text{obj}}}$. Here, $z_p \in \mathbb{R}^2$ with $z_p \sim \mathcal{N}(\mu_p, \sigma_p^2)$ denotes the image-plane position; $z_s \in \mathbb{R}^2$ with $z_s \sim \mathcal{N}(\mu_s, \sigma_s^2)$ denotes the spatial scale; $z_d \in \mathbb{R}$ with $z_d \sim \mathcal{N}(\mu_d, \sigma_d^2)$ denotes the depth that specifies the compositing order; $z_t \in [0, 1]$ with $z_t \sim \text{Beta}(a, b)$ denotes the transparency that controls particle visibility; and $z_f \in \mathbb{R}^{d_{\text{obj}}}$ with $z_f \sim \mathcal{N}(\mu_f, \sigma_f^2)$ denotes the appearance feature. The background is represented by a separate latent $z_{\text{bg}} \in \mathbb{R}^{d_{\text{bg}}}$ with $z_{\text{bg}} \sim \mathcal{N}(\mu_{\text{bg}}, \sigma_{\text{bg}}^2)$, which captures global background appearance. The decoder generates a spatial appearance map and a corresponding alpha-channel mask for each foreground particle, which are composited according to the particle position, scale, depth, and transparency to reconstruct the input image. These particle-wise alpha masks provide a pixel-space decomposition of foreground entities and background. The entire DLP model is trained without object-level supervision by optimizing the evidence lower bound (ELBO).

4 Particle-based Object Representation Transport (PORT)

Our goal is to directly transport source-view object-centric latents into their corresponding target-view latents, given a source RGB image $I^{\text{src}} \in \mathbb{R}^{3\times H\times W}$ and a source-to-target relative camera azimuth $\Delta_{\text{cam}} \in (-\pi, \pi]$. As DLP outputs an *unordered set of particles*, so particle m encoded from view A does not necessarily represent the same object as particle m encoded from view B, and no ground-truth correspondence is available during training. So, rather than inferring a target-view representation and matching it to the source, we propose **Particle-based Object Representation Transport (PORT)** that transports each source particle to its target-view state while retaining its index, so that object-level correspondence is preserved by construction.

PORT builds on the standard DLP encoder E_ϕ and decoder D_θ, and introduces a camera-conditioned **ParticleTransporter** for transporting source-view object latents to a target viewpoint. The overall

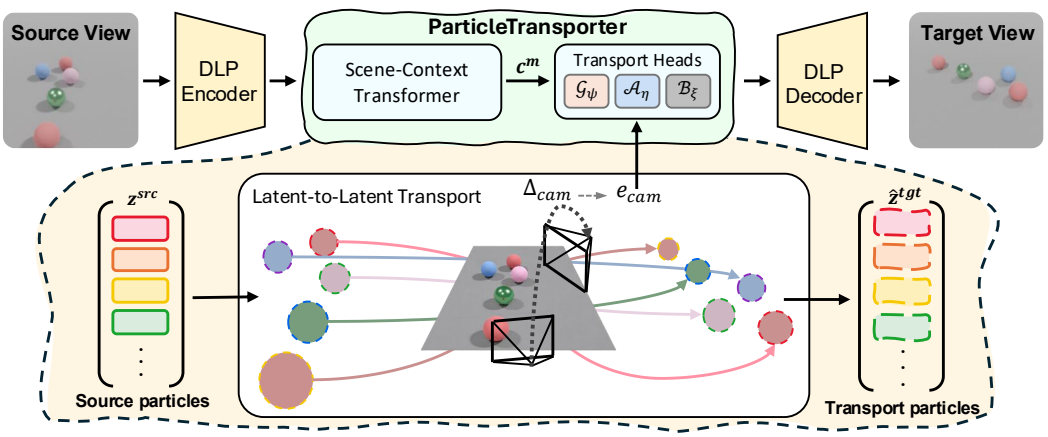


Figure 1: **Overview of PORT.** Source-view object-centric latents extracted by the DLP encoder are directly transported to their corresponding target-view latent states conditioned on the relative camera change Δ_{cam}, while preserving cross-view object correspondence. The transported latents are then rendered by the same DLP decoder to reconstruct the target view. The bottom illustration depicts the latent-to-latent transport from source-view object latents to their target-view states.

pipeline is illustrated in Figure 1. Given a source image, the encoder infers the source-view foreground latents $Z_{\text{fg}}^{\text{src}} = \{z_{\text{fg}}^m\}_{m=1}^M$ together with a source background latent $z_{\text{bg}}^{\text{src}}$. These source latents are decoded to reconstruct the source view, while ParticleTransporter transforms them according to the relative viewpoint change, producing the target-view latents $\hat{Z}_{\text{fg}}^{\text{tgt}} = \{\hat{z}_{\text{fg}}^m\}_{m=1}^M$ and $\hat{z}_{\text{bg}}^{\text{tgt}}$. The transported target-view latents are then rendered by the same decoder to reconstruct the corresponding target view.

4.1 PARTICLETRANSPORTER

ParticleTransporter directly transports each source foreground particle to the target viewpoint while preserving its cross-view correspondence, and transforms the background latent accordingly. In this process, the particle position, scale, depth/compositing order, transparency, and appearance are transformed into their target-view states. We encode the relative azimuth Δ_{cam} as the camera feature $e_{\text{cam}} = [\Delta_{\text{cam}}, \sin\Delta_{\text{cam}}, \cos\Delta_{\text{cam}}]$, which conditions the transport process. ParticleTransporter consists of a Scene-Context Transformer and three transport branches: Geometry and Visibility ($\mathcal{G}_\psi$), Appearance ($\mathcal{A}_\eta$), and Background ($\mathcal{B}_\xi$).

Scene-Context Transformer: Under a change in camera viewpoint, objects' attributes do not change in isolation; how each object transforms depends on its spatial arrangement relative to the other objects in the scene. To incorporate this scene-level relational information, the Scene-Context Transformer constructs a contextual representation from the full set of source particles, which is used for geometric and visibility transport. Specifically, the position z_p^m, scale z_s^m, depth z_d^m, and appearance feature z_f^m of each foreground particle are concatenated and processed by a Transformer encoder:

$$\{c^m\}_{m=1}^M = \text{TransformerEnc}\left(\{u^m\}_{m=1}^M\right), \qquad u^m = [z_p^m, z_s^m, z_d^m, z_f^m].$$

Geometric Attribute and Visibility Transport ($\mathcal{G}_\psi$): Position z_p^m, scale z_s^m, and depth/compositing order z_d^m can change jointly with the camera viewpoint. We therefore group them as *geometric attributes* and jointly predict their target-view transformations using a shared geometry network. Specifically, the geometry network takes the source geometric attributes, camera feature e_{cam}, and scene-context feature c^m, and predicts residual changes $\Delta z_p^m, \Delta z_s^m, \Delta z_d^m$ for each attribute together with a visibility logit g^m:

$$(\Delta z_p^m,\ \Delta z_s^m,\ \Delta z_d^m,\ g^m) = \mathcal{G}_\psi\left([z_p^m,\ z_s^m,\ z_d^m,\ e_{\text{cam}},\ c^m]\right),$$
$$(\hat{z}_p^m,\ \hat{z}_s^m,\ \hat{z}_d^m,\ \hat{z}_t^m) = (z_p^m + \Delta z_p^m,\ z_s^m + \Delta z_s^m,\ z_d^m + \Delta z_d^m,\ z_t^m \sigma(g^m)).$$

Here, $\mathcal{G}_\psi$ consists of a shared MLP followed by attribute-specific linear prediction heads. The sigmoid function $\sigma(\cdot)$ converts the visibility logit $g^m \in \mathbb{R}$ into a soft visibility gate $\sigma(g^m) \in (0, 1)$, which modulates the contribution of particle m to the target-view rendering.

Appearance Transport ($\mathcal{A}_\eta$): For appearance transport, ParticleTransporter predicts the target appearance $\hat{z}_f^m$ from the source appearance z_f^m, source and transported positions $(z_p^m, \hat{z}_p^m)$, and camera feature e_{cam}, without directly using the scene-context feature c^m. This avoids directly mixing appearance information across particles while allowing the appearance transformation to depend on the particle's predicted geometric displacement. Even under the same viewpoint change, different particles can undergo different image-plane displacements depending on their source positions, leading to different view-dependent appearance changes:

$$\Delta z_f^m = \mathcal{A}_\eta\big([z_p^m,\, \hat{z}_p^m,\, z_f^m,\, e_{\text{cam}}]\big)\,, \qquad \hat{z}_f^m = z_f^m + \Delta z_f^m.$$

Background Transport ($\mathcal{B}_\xi$): Background transport independently transforms the source background latent z_{bg} into its target-view state $\hat{z}_{\text{bg}}$, conditioned on e_{cam}:

$$\Delta z_{\text{bg}} = \mathcal{B}_\xi([z_{\text{bg}},\, e_{\text{cam}}])\,, \qquad \hat{z}_{\text{bg}} = z_{\text{bg}} + \Delta z_{\text{bg}}.$$

This allows the model to capture viewpoint-dependent changes in global background appearance independently of the foreground particles. We use the same decoder D_θ to render both the source and transported particle representations. The encoded source particles are decoded to reconstruct the source image, while the transported foreground particles and background latent are decoded to reconstruct the corresponding target view, thereby avoiding explicit matching in the latent space. Using the same decoder encourages the transported particles to preserve the same latent structure and semantics as the source particles.

4.2 Optimization and Training

During training, we assume access to a dataset of multi-view scenes, where each scene is observed from multiple viewpoints with known relative camera azimuths. During inference, PORT requires only a single source image. Given such a dataset, the entire model is trained end-to-end, optimizing the encoder, ParticleTransporter, and decoder jointly. For the source view, we maintain the original ELBO objective of DLP (Daniel et al., 2026), consisting of a reconstruction term and a KL-divergence term. Specifically, we minimize a reconstruction loss $\mathcal{L}_{\text{rec}}^{src}$ between I^{src} and its reconstruction $\hat{I}^{src}$, and regularize the source-view latent posterior toward the DLP prior using the KL-divergence objective $\mathcal{L}_{\text{KL}}$.

We construct a set of transported reconstruction targets that includes the source view itself with $\Delta_{\text{cam}} = 0$ together with the available target views. Let N denote the number of transported reconstruction targets, including the identity target. We define the transported target-view reconstruction loss as

$$\mathcal{L}_{\text{rec}}^{tgt} = \frac{1}{N} \sum_{j=1}^{N} \ell_{\text{rec}} \left(I_j^{tgt}, \hat{I}_j^{tgt} \right),$$

where ℓ_{rec} denotes the image reconstruction loss. Including the identity target encourages ParticleTransporter to preserve the source representation when no viewpoint change is applied.

Finally, we apply a transport Chamfer loss $\mathcal{L}_{\text{TC}}$ to align the unordered sets of transported and target-view particles without requiring costly explicit one-to-one matching. For this loss, the target images are encoded by the same encoder under stop-gradient, such that the inferred target particles are used only as reference particles for transport alignment. We provide the mathematical definition of the Chamfer-distance regularization in Appendix A.2.

The final training objective is

$$\mathcal{L} = \lambda_{\text{rec}} \left(\mathcal{L}_{\text{rec}}^{src} + \mathcal{L}_{\text{rec}}^{tgt} \right) + \lambda_{\text{KL}} \mathcal{L}_{\text{KL}} + \lambda_{\text{TC}} \mathcal{L}_{\text{TC}}.$$

We implement PORT in PyTorch (Paszke et al., 2019) and optimize all components end-to-end using Adam (Kingma & Ba, 2014) with a learning rate of 8×10^{-5}. Full loss formulations, dataset-specific loss weights, training schedules, optimization settings and additional hyperparameters are provided in Appendix A.2.

Dataset	Method	Pos. [px] (S/U/A) ↓	Scale [%] (S/U/A) ↓	Identity Acc. (S/U/A) ↑
TP-OBJ3D	LPWM (Oracle)	3.8 / 5.4 / 5.4	**19.3** / 19.7 / 18.0	–
	LPWM (Tracked)	39.7 / 39.9 / 39.9	33.0 / 32.9 / 32.2	0.276 / 0.292 / 0.292
	PORT	**2.3 / 2.6 / 2.6**	21.9 / **18.1 / 15.9**	**0.998 / 0.998 / 0.998**
TP-GROCERY	LPWM (Oracle)	3.7 / 3.7 / 3.7	15.8 / 15.9 / 15.9	–
	LPWM (Tracked)	35.4 / 35.5 / 35.5	41.9 / 42.5 / 42.3	0.306 / 0.304 / 0.305
	PORT	**3.5 / 3.5 / 3.5**	**10.9 / 10.9 / 10.9**	**0.995 / 0.995 / 0.995**
CLEVR3D	LPWM (Oracle)	9.8 / 13.3 / 13.3	**16.5 / 21.8** / 23.3	–
	LPWM (Tracked)	37.4 / 38.5 / 38.5	73.3 / 64.1 / 60.8	0.375 / 0.305 / 0.305
	OSRT (Tracked)	6.5 / 44.6 / 44.3	20.0 / 53.2 / 35.5	0.989 / 0.392 / 0.397
	PORT	**6.1 / 7.3 / 7.3**	23.1 / 21.9 / **19.8**	**0.997 / 0.996 / 0.996**

Table 1: Cross-view object representation results. S/U/A denotes performance on Seen, Unseen, and All viewpoints, respectively. Position error is reported in pixels, scale error in percent, and identity accuracy as a fraction. Oracle evaluation independently matches predicted representations to ground-truth objects at each target view, whereas Tracked evaluation preserves the object identity assigned in the source view across viewpoints.

5 Experiments

We first describe the datasets and object-centric evaluation protocol used throughout our experiments (Sec. 5.1). We then evaluate (i) how accurately our model transports individual object states across viewpoints (Sec. 5.2), (ii) the extent to which its learned 2D object-centric representation captures 3D geometric structure (Sec. 5.3), (iii) the utility of this representation for downstream tasks (Sec. 5.4), and (iv) key design choices of PORT through ablation studies (Appendix A.9).

5.1 Datasets and Evaluation Metrics

Datasets: To enable direct evaluation of cross-view object-centric transport, we develop a dataset generation protocol that provides consistent object-level annotations across viewpoints solely for evaluation and use it to construct two datasets: **TP-OBJ3D** and **TP-GROCERY**. TP-OBJ3D extends the setting of OBJ3D (Lin et al., 2020a) to a multi-view setup rendered with Kubric (Greff et al., 2022), providing persistent object identities and geometric annotations across views. To evaluate transport beyond simple primitive shapes, we further construct TP-GROCERY using the same generation protocol with more realistic everyday object geometry, drawing assets from ShapeNetV2 (Chang et al., 2015) and the YCB Object Set (Calli et al., 2015). We additionally use CLEVR3D (Stelzner et al., 2021a) as an external benchmark to evaluate generalization in a distinct object-centric scene setting. Finally, we use PandaPush (Haramati et al., 2024; Makoviychuk et al., 2021) to evaluate the downstream utility of the learned representation for robotic manipulation. Detailed dataset descriptions and the generation protocol are provided in Appendix A.3.2.

Metrics: To evaluate whether object representations are correctly transported across viewpoints, we introduce three object-level metrics. **Position Error** measures the distance between predicted and ground-truth object locations in the target view, reported in pixels. **Scale Error** measures the discrepancy between predicted and ground-truth source-to-target changes in object extent using bounding-box width and height and is reported as a multiplicative deviation (%). **Identity Accuracy** measures the fraction of object representations that remain associated with the same physical object across viewpoints. Since our model also generates target-view images, we additionally report PSNR, SSIM, and LPIPS. To further evaluate 3D geometric consistency, we triangulate the transported 2D object locations across multiple viewpoints and measure the distance between the estimated 3D position and the ground-truth object center. Further details are provided in Appendix A.3.3.

5.2 Cross-View Object-Centric Transport

We first evaluate how accurately PORT predicts the object-centric representation at a target viewpoint from a single source view. On TP-OBJ3D, TP-GROCERY, and CLEVR3D, we evaluate our object-level position, scale, and identity metrics, together with novel-view reconstruction quality.

	Novel-View Reconstruction				3D Triangulation			
Dataset	**Method**	PSNR ↑	SSIM ↑	LPIPS ↓	**Method**	E_{3D} ↓	E_{xy} ↓	z Bias
TP-OBJ3D	Zero-1-to-3	**38.49**	**0.976**	0.092	LPWM (Oracle)	0.267	0.103	−0.227
	LPWM	35.66	0.967	0.028	LPWM (Tracked)	2.733	2.472	−0.408
	PORT	37.50	**0.976**	**0.015**	**PORT**	**0.167**	**0.055**	−0.154
TP-GROCERY	Zero-1-to-3	**32.21**	**0.945**	0.119	LPWM (Oracle)	**0.232**	0.126	−0.131
	LPWM	29.81	0.924	**0.079**	LPWM (Tracked)	2.395	2.204	−0.172
	PORT	30.16	0.922	0.086	**PORT**	0.248	**0.116**	−0.165
CLEVR3D	Zero-1-to-3	**35.89**	**0.964**	**0.040**	OSRT (Tracked)	0.449	0.329	−0.272
	LPWM	28.72	0.907	0.083	LPWM (Oracle)	0.646	0.452	−0.369
	OSRT	32.90	0.941	0.295	LPWM (Tracked)	2.565	2.475	−0.049
	PORT	28.60	0.901	0.088	**PORT**	**0.401**	**0.085**	−0.387

Table 2: Novel-view reconstruction and 3D triangulation results across datasets. For triangulation, Tracked preserves the object assignment established in the source view, whereas Oracle independently establishes correspondence at each target view.

Baselines: Our main baselines include LPWM (Daniel et al., 2026), a particle-based latent world model; OSRT (Sajjadi et al., 2022), an object-centric novel-view synthesis model; and Zero-1-to-3 (Liu et al., 2023), a diffusion-based single-image novel-view synthesis model. LPWM uses particle-based representations containing explicit object states such as position and scale, but does not constrain a particular particle to remain associated with the same physical object across viewpoints (Appendix A.4.2). We therefore evaluate LPWM under both *Oracle*, which independently rematches predicted particles to ground-truth objects at each target view, and *Tracked*, which preserves the object assignment established in the source view. As Zero-1-to-3 provides no object-level representation, we compare against it only on image-space metrics.

Results: As shown in Table 1, PORT preserves correspondence without target-view oracle matching while maintaining low position and scale errors and high identity accuracy, matching or outperforming LPWM's oracle geometric accuracy in several settings. On CLEVR3D, OSRT is accurate at viewpoints observed during training but degrades at unseen viewpoints, whereas PORT remains stable across the full viewpoint range, suggesting that it learns how object states transform with relative viewpoint changes rather than relying on viewpoint-specific associations (Appendix A.4.1). Unlike diffusion-based NVS methods such as Zero-1-to-3, which primarily focus on camera-conditioned generation for visual fidelity rather than learning compact, persistent object-level latents, PORT targets these representations and is evaluated primarily on whether it supports geometric reasoning and downstream tasks.

As shown in Figure 2 (left), from a single source image, PORT generates target views at different relative azimuths, while each transported particle remains associated with the same object as its position and scale change with the viewpoint.

5.3 Evaluating 3D Geometric Information in the Learned Representation

We evaluate whether the learned representation captures 3D geometric structure from two perspectives: whether transported 2D object locations triangulate to a consistent 3D position, and whether a linear probe can effectively predict 3D position and depth from the frozen representation.

Recovering 3D Geometry from Transported Object Locations. We triangulate the predicted 2D locations of the same transported object across multiple viewpoints to estimate a single underlying 3D position. As shown in Table 2, PORT achieves low 3D position error across datasets, with particularly accurate estimates in the xy plane. Most of the remaining error is concentrated along the z axis as a systematic bias, which we analyze in Appendix A.5.2. The triangulated object positions can also be used to construct an object-level depth map from a single source image; detailed results and the full procedure are provided in Appendix A.10.2. These results show that the object locations predicted by PORT across viewpoints are geometrically consistent and sufficiently accurate to recover the underlying 3D position and depth.

Probe Input		Dim.	TP-OBJ3D			CLEVR3D		
			3D Pos. ↓	Raw Depth ↓	Rel. Depth ↓	3D Pos. ↓	Raw Depth ↓	Rel. Depth ↓
Patch AE (local)†		802	0.311	0.419	0.180	0.273	0.292	0.183
OSRT (slot)†		1,538	0.284	0.425	0.228	0.259	0.307	0.171
LPWM	Source only	10	0.462	0.282	0.306	0.324	0.276	0.215
	+ Rollout	20	0.455	0.301	0.299	0.320	0.301	0.214
PORT	Source only	10	0.396	0.244	0.217	0.316	0.297	0.193
	+ Transported	**20**	**0.140**	**0.191**	**0.076**	**0.066**	**0.225**	**0.045**

Table 3: Linear probing results on TP-OBJ3D and CLEVR3D. 3D-position error measures the Euclidean distance to the ground-truth 3D object center. Raw-depth error measures the depth difference from the source-view camera, while relative-depth error measures the depth difference relative to the scene center. † denotes methods additionally provided with the ground-truth 2D object centroid as an oracle positional cue.

Setting	Success (%) ↑	Rep. Time (ms) ↓
EC-Diffuser (2R)	77.1 ± 1.82	73.8
PORT (1R+1T)	77.1 ± 1.88	37.2
PORT (1R+2T)	79.6 ± 2.12	37.5

Table 4: PandaPush manipulation results. R/T denote real-view/transported particle representations. We report the success rate (mean $\pm$ std. over five seeds) and representation-construction time.

Linear Probing of 3D Information. While triangulation evaluates the geometric consistency of cross-view 2D locations, linear probing evaluate how well 3D position and depth are encoded in the representation space. We freeze all trained models and train only a linear regressor on their representations to predict each object's 3D position and depth. Details of the probing protocol are provided in Appendix A.6. As shown in Table 3, compared with using the source-view representation alone, augmenting it with transported object states substantially reduces the 3D-position error from 0.396 to 0.140 on TP-OBJ3D and from 0.316 to 0.066 on CLEVR3D, indicating that the transported latents encode additional information about the object's underlying 3D position and depth. PORT also achieves low 3D-position and depth errors with a considerably lower-dimensional representation than OSRT and Patch AE, a non-object-centric autoencoder that represents an image as a grid of local latent features rather than explicit object entities. In contrast, augmenting LPWM with camera-conditioned rollouts yields little improvement, suggesting that correspondence-preserving cross-view states, rather than additional viewpoint predictions alone, make 3D information easier to recover with a linear probe. Additional probing analyses and input ablations are provided in Appendix A.6.2, while a detailed comparison with LPWM rollouts is provided in Appendix A.6.3.

5.4 Downstream Tasks with Transported Representations

Robotic Manipulation with Transported Representations. We evaluate whether particle representations transported by PORT from a single-view observation can substitute for an additional real camera observation in downstream manipulation on PandaPush (Haramati et al., 2024). Our baseline, EC-Diffuser (Qi et al., 2025), is an entity-centric diffusion-based behavioral cloning policy that uses object-centric particle representations from two real-view observations. We retain one real view and replace the second with particles transported by PORT from the source view. Because correspondence is preserved by index, corresponding particles are concatenated along the feature dimension and provided to the same EC-Diffuser policy architecture, reducing the number of input entities. Additional details are reported in Appendix A.7.

As shown in Table 4, with success rates evaluated over five seeds, PORT achieves the same 77.1% success rate as EC-Diffuser using two real views, while requiring only a single real camera observation and reducing the cost of constructing the multi-view representation by approximately half. Moreover, adding particle representations transported to an additional target viewpoint yields a success rate of 79.6% with almost no additional representation-construction cost. These results show that transported particle representations can efficiently substitute for additional real-view rep-

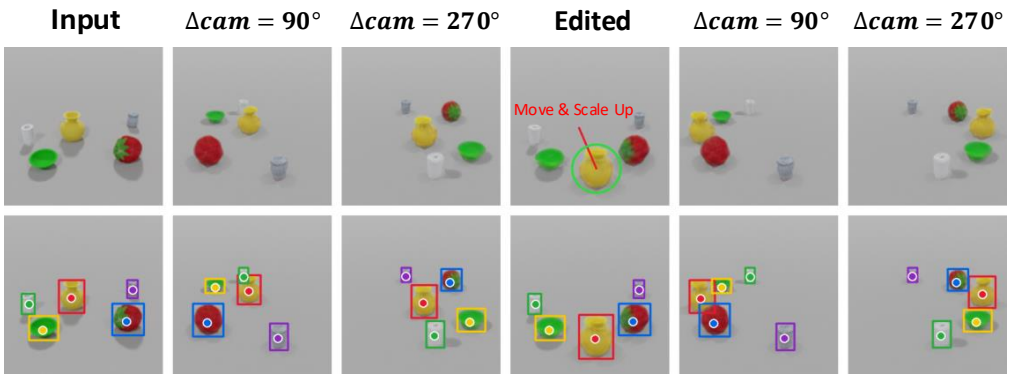


Figure 2: **Cross-view transport and latent-space editing on TP-GROCERY.** Left: from a single source image, PORT generates target views at relative azimuths $\Delta_{\text{cam}} = 90°$ and $270°$ (top), while the transported particles (bottom; color indicates the same particle across views) preserve cross-view correspondence as their positions and scales change with viewpoint. Right: after editing the jar in the source latent space by moving and enlarging it, the edit is consistently reflected in all target views. Videos are available on the project website.

resentations while maintaining manipulation performance. Qualitative rollouts are provided in Appendix A.10.3, and videos are available on our project website.

We further analyze object localization under robot-arm occlusion to understand why transported representations can substitute for real target-view observations (Appendix A.7.1). Details of the manipulation task, data collection, policy training, and evaluation protocol are provided in Appendix A.7, while the ablation on multi-view representation encoding is reported in Appendix A.9.3.

Gaussian Splatting with Transported Views. As PORT transports source object latents to multiple target viewpoints and decodes an RGB image at each, multi-view observations can be generated from a single source image and used directly by an existing Gaussian Splatting pipeline. As our particle-based representation also yields particle-wise mask maps at each viewpoint, object-level 3D masks can be constructed from self-supervised object masks without separate segmentation models or per-view mask annotations. Results and experimental settings are provided in Appendix A.8.

Object-Level Scene Editing. Because PORT transports an interpretable particle representation, object-level edits in the source view, such as moving, rescaling, or removing an object, can be propagated consistently to target viewpoints (Figure 2, right) and to the resulting 3D scene (Appendix A.8.3). The consistency of these edits across viewpoint changes suggests that PORT transports object states according to the viewpoint condition rather than simply reproducing a fixed scene configuration. Additional qualitative videos are available on our project website.

6 Conclusion

We introduce Particle-based Object Representation Transport (PORT), which directly transports object-centric representations from a single source image to target viewpoints conditioned on relative camera pose. By transforming the geometric and appearance attributes of latent particles while preserving object correspondence, PORT goes beyond novel-view generation and enables reasoning about the states of the same objects across viewpoints. We show that these transported representations can support reasoning about 3D geometric information and can be used as imagined multi-view representations for downstream tasks such as robotic manipulation.

Limitations & Future Work: PORT currently considers only azimuthal viewpoint changes and does not model changes in camera elevation. In addition, PORT inherits the limitations of DLP, such as decomposition of scenes with complex backgrounds and moving cameras. Failure cases, discussed in Appendix A.10.5, include objects that are not visible in the source view but become visible from a target viewpoint. Future work includes exploring extensions to more diverse camera poses and moving-camera settings, scaling to more diverse datasets, and applications to a broader range of downstream tasks.

REFERENCES

Christopher P. Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. MONet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019.

Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M. Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 International Conference on Advanced Robotics (ICAR)*, pp. 510–517, 2015. doi: 10.1109/ICAR.2015.7251504.

Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

Eric Crawford and Joelle Pineau. Spatially invariant unsupervised object detection with convolutional neural networks. In *AAAI*, 2019.

Tal Daniel and Aviv Tamar. Unsupervised image representation learning with deep latent particles. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 4644–4665. PMLR, 17–23 Jul 2022.

Tal Daniel and Aviv Tamar. DDLP: Unsupervised object-centric video prediction with deep dynamic latent particles. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856.

Tal Daniel, Carl Qi, Dan Haramati, Amir Zadeh, Chuan Li, Aviv Tamar, Deepak Pathak, and David Held. Latent particle world models: Self-supervised object-centric stochastic dynamics modeling. In *International Conference on Learning Representations (ICLR)*, 2026.

Martin Engelcke, Adam R. Kosiorek, Oiwi Parker Jones, and Ingmar Posner. GENESIS: Generative scene inference and sampling with object-centric latent representations. In *International Conference on Learning Representations (ICLR)*, 2020.

Ruiqi Gao, Aleksander Holynski, Philipp Henzler, Arthur Brussee, Ricardo Martin-Brualla, Pratul Srinivasan, Jonathan T Barron, and Ben Poole. Cat3d: Create anything in 3d with multi-view diffusion models. *arXiv preprint arXiv:2405.10314*, 2024.

Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. In *International Conference on Machine Learning*, pp. 2424–2433. PMLR, 2019.

Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, et al. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3749–3761, 2022.

Dan Haramati, Tal Daniel, and Aviv Tamar. Entity-centric reinforcement learning for object manipulation from pixels. In *International Conference on Learning Representations*, volume 2024, pp. 29980–30014, 2024.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, George Drettakis, et al. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

Nanbo Li, Cian Eastwood, and Robert Fisher. Learning object-centric representations of multi-object scenes from multiple views. *Advances in neural information processing systems*, 33:5656–5666, 2020.

Zhixuan Lin, Yi-Fu Wu, Skand Peri, Bofeng Fu, Jindong Jiang, and Sungjin Ahn. Improving generative imagination in object-centric world models. In *International conference on machine learning*, pp. 6140–6149. PMLR, 2020a.

Zhixuan Lin, Yi-Fu Wu, Skand Vishwanath Peri, Weihao Sun, Gautam Singh, Fei Deng, Jindong Jiang, and Sungjin Ahn. SPACE: Unsupervised object-oriented scene representation via spatial attention and decomposition. In *International Conference on Learning Representations (ICLR)*, 2020b.

Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9264–9275. IEEE, 2023.

Yu Liu, Baoxiong Jia, Yixin Chen, and Siyuan Huang. Slotlifter: Slot-guided feature lifting for learning object-centric radiance fields. In *European Conference on Computer Vision (ECCV)*, 2024a.

Yuan Liu, Cheng Lin, Zijiao Zeng, Xiaoxiao Long, Lingjie Liu, Taku Komura, and Wenping Wang. Syncdreamer: Generating multiview-consistent images from a single-view image. In *International conference on learning representations*, volume 2024, pp. 27676–27697, 2024b.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Ruijie Lu, Yixin Chen, Junfeng Ni, Baoxiong Jia, Yu Liu, Diwen Wan, Gang Zeng, and Siyuan Huang. Movis: Enhancing multi-object novel view synthesis for indoor scenes. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26767–26778. IEEE, 2025.

Rundong Luo, Hong-Xing Yu, and Jiajun Wu. Unsupervised discovery of object-centric neural fields. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL `https://openreview.net/forum?id=ScEv13W2f1`.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Steven Pinker. Mental imagery and the third dimension. *Journal of Experimental Psychology: General*, 109(3):354–371, 1980. doi: 10.1037/0096-3445.109.3.354.

Steven Pinker and Ronald A. Finke. Emergent two-dimensional patterns in images rotated in depth. *Journal of Experimental Psychology: Human Perception and Performance*, 6(2):244–264, 1980. doi: 10.1037/0096-1523.6.2.244.

Carl Qi, Dan Haramati, Tal Daniel, Aviv Tamar, and Amy Zhang. EC-diffuser: Multi-object manipulation via entity-centric behavior generation. In *The Thirteenth International Conference on Learning Representations*, 2025.

Di Qi, Tong Yang, and Xiangyu Zhang. Slot-guided volumetric object radiance fields. *Advances in Neural Information Processing Systems*, 36:65766–65783, 2023.

Mehdi SM Sajjadi, Daniel Duckworth, Aravindh Mahendran, Sjoerd Van Steenkiste, Filip Pavetic, Mario Lucic, Leonidas J Guibas, Klaus Greff, and Thomas Kipf. Object scene representation transformer. *Advances in neural information processing systems*, 35:9512–9524, 2022.

Kyle Sargent, Zizhang Li, Tanmay Shah, Charles Herrmann, Hong-Xing Yu, Yunzhi Zhang, Eric Ryan Chan, Dmitry Lagun, Li Fei-Fei, Deqing Sun, and Jiajun Wu. ZeroNVS: Zero-shot 360-degree view synthesis from a single real image. *CVPR, 2024*, 2023.

Junyoung Seo, Kazumi Fukuda, Takashi Shibuya, Takuya Narihira, Naoki Murata, Shoukang Hu, Chieh-Hsin Lai, Seungryong Kim, and Yuki Mitsufuji. Genwarp: Single image to novel views with semantic-preserving generative warping. *arXiv preprint arXiv:2405.17251*, 2024.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pp. 785–799. PMLR, 2023.

Aleksandar Stanić and Jürgen Schmidhuber. R-sqair: relational sequential attend, infer, repeat. *arXiv preprint arXiv:1910.05231*, 2019.

Karl Stelzner, Kristian Kersting, and Adam R Kosiorek. Decomposing 3d scenes into objects via unsupervised volume segmentation. *arXiv preprint arXiv:2104.01148*, 2021a.

Karl Stelzner, Kristian Kersting, and Adam R Kosiorek. Decomposing 3d scenes into objects via unsupervised volume segmentation. *arXiv preprint arXiv:2104.01148*, 2021b.

Stephen Tian, Yancheng Cai, Hong-Xing Yu, Sergey Zakharov, Katherine Liu, Adrien Gaidon, Yunzhu Li, and Jiajun Wu. Multi-object manipulation via object-centric neural scattering functions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9021–9031, 2023.

Kuanning Wang, Ke Fan, Yuqian Fu, Siyu Lin, Hu Luo, Daniel Seita, Yanwei Fu, Yu-Gang Jiang, and Xiangyang Xue. Ocra: Object-centric learning with 3d and tactile priors for human-to-robot action transfer. *arXiv preprint arXiv:2603.14401*, 2026.

Tianyu Wang, Miaomiao Liu, and Kee Siong Ng. Spatially invariant unsupervised 3D object-centric learning and scene decomposition. In *European Conference on Computer Vision (ECCV)*, pp. 120–135. Springer, 2022.

Thaddäus Wiedemer, Jack Brady, Alexander Panfilov, Attila Juhos, Matthias Bethge, and Wieland Brendel. Provable compositional generalization for object-centric learning. In *International Conference on Learning Representations*, volume 2024, pp. 20137–20164, 2024.

Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, et al. gsplat: An open-source library for gaussian splatting. *Journal of Machine Learning Research*, 26(34):1–17, 2025.

Hong-Xing Yu, Leonidas J Guibas, and Jiajun Wu. Unsupervised discovery of object radiance fields. *arXiv preprint arXiv:2107.07905*, 2021.

Jinyang Yuan, Bin Li, and Xiangyang Xue. Unsupervised learning of compositional scene representations from multiple unspecified viewpoints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8971–8979, 2022.

Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. *arXiv preprint arXiv:2403.03954*, 2024.

Ellina Zhang, Madhavan Iyengar, Amir Zadeh, Chuan Li, David Held, Deepak Pathak, and Tal Daniel. 3d-DLP: Self-supervised 3d object-centric scene representation learning. In *Forty-third International Conference on Machine Learning*, 2026a. URL `https://openreview.net/forum?id=vIotI25gJz`.

Weipu Zhang, Adam Jelley, Trevor McInroe, Amos Storkey, and Gang Wang. Object-centric world models from few-shot annotations for sample-efficient reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 91366–91394, 2026b.

A Appendix

A.1 AI use Statement

We used large language models (LLMs) to assist with writing and language refinement. All AI-assisted content was reviewed and approved by the authors.

A.2 Loss and Training Details

The complete training objective defined in the main paper consists of source-view reconstruction, transported target-view reconstruction, DLP latent regularization, and a transport Chamfer loss. In this section, we provide the detailed formulation of each loss term. The reconstruction and latent regularization terms follow the single-image formulation of DLPv3/LPWM, while the transport Chamfer loss provides additional cross-view particle-space alignment.

Reconstruction Loss. Given a source image I^{src}, the source particles inferred by the encoder are rendered by the decoder to obtain the source reconstruction $\hat{I}^{\mathrm{src}}$. The source-view reconstruction loss is defined as

$$\mathcal{L}_{\mathrm{rec}}^{\mathrm{src}} = \ell_{\mathrm{rec}}\left(I^{\mathrm{src}}, \hat{I}^{\mathrm{src}}\right),$$

where ℓ_{rec} denotes the image reconstruction loss.

For each source view, we construct a set of N transported reconstruction targets consisting of the available target views together with the source view itself under the identity camera condition $\Delta_{\mathrm{cam}} = 0$. The transported target-view reconstruction loss is

$$\mathcal{L}_{\mathrm{rec}}^{\mathrm{tgt}} = \frac{1}{N}\sum_{j=1}^{N} \ell_{\mathrm{rec}}\left(I_j^{\mathrm{tgt}}, \hat{I}_j^{\mathrm{tgt}}\right),$$

where $\hat{I}_j^{\mathrm{tgt}}$ is obtained by transporting the source particles to the j-th target viewpoint and rendering the transported representation with the same decoder. The identity target encourages ParticleTransporter to preserve the source latent state when no viewpoint change is applied.

DLP Latent Regularization. The source posterior is regularized using the standard DLPv3/LPWM (Daniel et al., 2026) latent objective. Specifically, the position-related latent variables, scale, depth, and appearance features are regularized using Gaussian priors, while the transparency attribute is regularized using a Beta prior. The background latent is regularized separately with a Gaussian prior. The keypoint proposal follows the position-prior formulation of DLP.

Transport Chamfer Loss. The transport Chamfer loss $\mathcal{L}_{\mathrm{TC}}$ provides particle-space alignment between the transported representation and the representation inferred directly from the corresponding target image. It is a bidirectional Chamfer distance between the two particle sets, in which nearest neighbors are found by 2D position, distances are computed over all particle attributes, and each pair is weighted by the transparency of the reference particle.

For each target image I_j^{tgt}, we obtain a reference foreground particle set $\tilde{Z}_j^{\mathrm{tgt}}$ using the same encoder. Stop-gradient is applied to this branch such that the target particles serve only as reference particles for transport alignment. Let $\hat{Z}_j^{\mathrm{tgt}}$ denote the transported foreground particle set. We match the two particle sets using bidirectional nearest-neighbor matching based on their 2D positions. Omitting the target-view index j for clarity, the two matching functions are

$$\pi(i) = \arg\min_k \left\|\hat{z}_p^{\,i} - \tilde{z}_p^{\,k}\right\|_2^2, \qquad \rho(k) = \arg\min_i \left\|\hat{z}_p^{\,i} - \tilde{z}_p^{\,k}\right\|_2^2.$$

For each matched pair, we minimize the discrepancy in position, scale, depth, transparency, and appearance. Each pair is weighted by the transparency of the target-side particle so that inactive

Table 5: Dataset-specific training configurations.

Configuration	TP-OBJ3D	CLEVR3D	TP-GROCERY	PandaPush
Resolution	128×128	128×128	128×128	128×128
Keypoint proposals	64	64	64	64
Posterior particles M	64	64	64	64
Transport particles L	12	12	12	25
Top-k	10	10	10	20
Reconstruction loss	MSE + 0.1 VGG	MSE + 0.1 VGG	MSE + 0.1 VGG	MSE
Reconstruction λ_{rec}	1.0	1.0	1.0	1.0
KL λ_{KL}	0.08	0.08	0.08	0.04
Transport Chamfer λ_{TC}	0	0	0.1	0.1
Batch size	16	16	2	8
Epochs	220	100	220	100

target particles contribute less to the alignment objective. For the j-th target view, we define

$$\mathcal{L}_{\text{TC}}^{(j)} = \sum_i \tilde{z}_t^{\pi(i)} \sum_{\text{att}\in\{p,s,d,t,f\}} \left\| \hat{\tilde{z}}_{\text{att}}^{i} - \tilde{z}_{\text{att}}^{\pi(i)} \right\|_2^2 + \sum_k \tilde{z}_t^{k} \sum_{\text{att}\in\{p,s,d,t,f\}} \left\| \hat{\tilde{z}}_{\text{att}}^{\rho(k)} - \tilde{z}_{\text{att}}^{k} \right\|_2^2 .$$

The transport Chamfer loss is then averaged across the target views:

$$\mathcal{L}_{\text{TC}} = \frac{1}{N} \sum_{j=1}^{N} \mathcal{L}_{\text{TC}}^{(j)}.$$

The matching is not assumed to be injective and is recomputed for each training sample. Gradients are propagated only through the transported branch. For datasets where particle-space alignment is not used, we set $\lambda_{\text{TC}} = 0$.

Training and Optimization Details. Training begins with a short warm-up stage in which ParticleTransporter is disabled and only the source-view DLP objective is optimized. After warm-up, ParticleTransporter is enabled, and the encoder, ParticleTransporter, and decoder are optimized jointly end to end. We use Adam (Kingma & Ba, 2014) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-6}$. The learning rate is fixed to 8×10^{-5} throughout training, with no weight decay or learning-rate scheduling. All transport MLPs use a hidden dimension of 256. The scene-context transformer uses a model dimension of 64 with one Transformer layer, four attention heads, a feed-forward dimension of 128, and dropout of 0.1, with pre-normalization. All experiments were conducted using two NVIDIA RTX 4500 Ada Generation or RTX 5000 Ada Generation GPUs. Training each PORT model required approximately 1–2 days, depending on the dataset configuration.

Here, L denotes the number of particles used for transport, while Top-k denotes the number retained after confidence-based filtering, for example using particle variance scores. Dataset-specific training configurations, including the number of transported particles, reconstruction objective, use of the transport Chamfer loss, batch size, and training duration, are summarized in Table 5.

A.3 Dataset and Evaluation Details

A.3.1 Multi-view Transport Dataset Generation Protocol

To quantitatively evaluate object-centric representation transport across viewpoint changes, we build a multi-view dataset generation protocol using Kubric (Greff et al., 2022) as the rendering and simulation backend. The goal of this protocol is to preserve object identity consistently across viewpoints while providing object-level geometric annotations, including 2D and 3D object positions. This enables direct evaluation of how accurately individual object states are transported across viewpoints, beyond pixel-level reconstruction quality.

Scene generation. Each scene is composed of either primitive objects or 3D mesh assets from ShapeNetCore.v2 (Chang et al., 2015). In the primitive setting, we use spheres and cubes and sample their colors, materials, scales, positions, and orientations. For ShapeNet-based scenes, the object categories, number of objects, object scale, and category sampling strategy can be controlled for each dataset configuration. Objects are placed within a predefined workspace using rejection sampling to avoid substantial overlap, and each object is assigned a random rotation around the vertical axis while being placed on the ground plane. This design allows us to generate scenes ranging from simple primitive compositions to more complex scenes containing diverse 3D mesh objects within the same generation framework.

Multi-view camera sampling. For each episode, we construct a camera ring around a fixed scene and place multiple cameras along the ring. We first sample the camera radius and elevation and then randomly sample an initial azimuth θ_0 for each episode. The azimuth of the v-th viewpoint is defined as

$$\theta_v = \theta_0 + v\Delta\theta,$$

where $\Delta\theta$ denotes the angular spacing between adjacent viewpoints for a given dataset configuration. Consequently, the absolute camera orientation varies across episodes, while the relative camera transformation between any pair of viewpoints is precisely controlled. All viewpoints share the same camera intrinsics, such that the variation across views is induced by changes in camera extrinsics. This construction allows the model to be trained and evaluated over controlled relative viewpoint changes without depending on a fixed absolute orientation.

Cross-view object annotations. Within an episode, each physical object retains the same instance identity across all viewpoints. For every view, we store the RGB image and an instance segmentation mask, where the segmentation ID associated with each object remains consistent across views. This provides direct correspondence between the same object in source and target views without requiring additional object-matching heuristics.

We additionally store the camera intrinsic and extrinsic parameters, the world-space 3D position of each object, and its projected 2D position in each view. The 2D position is obtained by projecting the object's 3D location through the corresponding camera model and is therefore defined even when the object is partially occluded by another object. These annotations support evaluation of transported object position, persistent cross-view identity, viewpoint-dependent scale changes, and 3D geometry recovered from multiple transported viewpoints.

Deterministic scene regeneration and viewpoint extension. Each episode is generated deterministically from a fixed random seed. Reusing the same seed therefore reproduces the same object identities, appearances, poses, placements, and overall scene configuration. This allows an existing scene to be re-rendered from additional camera viewpoints without changing its underlying object configuration, while the metadata for the newly rendered views can be appended to the existing episode.

This functionality enables multi-view configurations with different angular resolutions to be generated from the same scene distribution. For example, sparse configurations can use larger angular intervals, whereas denser configurations for analysis or visualization can sample the same 360° camera trajectory at finer angular intervals. The protocol therefore allows scene complexity and viewpoint density to be controlled independently within a unified framework for evaluating cross-view transport.

Overall, the proposed generation protocol combines (1) controllable scene composition, (2) precisely specified relative camera transformations, (3) persistent cross-view object identity, and (4) object-level 2D and 3D geometric ground truth. These properties provide the basis for constructing evaluation datasets that directly measure how accurately object representations are transported across viewpoint changes.

A.3.2 Dataset Details

We evaluate on TP-OBJ3D, TP-Grocery, CLEVR3D, and PandaPush. Here, we provide additional details on dataset construction and the ground truth used for evaluation.

TP-OBJ3D is a multi-view extension of OBJ3D (Lin et al., 2020a), constructed to support object-level cross-view evaluation. Scenes are rendered with Kubric (Greff et al., 2022) at 128×128 resolution and contain multiple spheres and cubes with colors and materials sampled following the OBJ3D setup. Each scene is observed from six static cameras arranged on a ring with 60° azimuth spacing. The camera radius, elevation, and base azimuth are randomized per scene, while the scene configuration remains fixed across views. Camera intrinsics and extrinsics are stored for every view, allowing the relative camera transformation to be recovered exactly. The training and validation splits contain 4,000 and 1,000 scenes, respectively. We additionally construct a held-out evaluation split of 1,000 scenes containing cross-view consistent instance identities, instance segmentation masks, object 3D centers, and their projected 2D centers in each view. These object-level annotations are used exclusively for evaluation and are not used during training.

TP-GROCERY is a multi-view dataset constructed to evaluate object-level cross-view transport on more complex everyday objects beyond simple primitives. It follows the same scene and rendering protocol as TP-OBJ3D, while replacing spheres and cubes with five ShapeNetV2 (Chang et al., 2015) categories—bowl, jar, bottle, can, and trash bin—together with eight scanned fruit objects from the YCB Object Set (Calli et al., 2015). The YCB subset consists of fixed instances of apple, orange, lemon, peach, pear, plum, strawberry, and banana, which are preprocessed to follow the same coordinate and normalization convention as the ShapeNet assets.

Each scene contains 3–6 objects selected from different categories. Rather than sampling object categories independently, we cycle through a shuffled category order so that objects within the same scene come from distinct categories. Objects are placed upright on the ground plane with randomized yaw, using rejection sampling to avoid overlapping footprints. Scenes are rendered with Kubric (Greff et al., 2022) at 128×128 resolution using the same neutral background and lighting configuration as TP-OBJ3D.

Each scene is observed from 12 static cameras arranged on a ring with 30° azimuth spacing, covering the full 360°. The camera radius, elevation, and base azimuth are randomized per scene, while the angular spacing between views remains fixed, allowing the relative camera transformation between any source–target pair to be recovered exactly. Camera intrinsics and extrinsics are stored for every view.

TP-GROCERY contains 4,000 training scenes, 500 validation scenes, and 1,000 held-out test scenes, generated using disjoint random seeds. For all splits, we store cross-view consistent instance identities, instance segmentation masks, depth, object world-space 3D centers, analytically projected 2D centers for every view, and complete camera calibration. The projected 2D centers are defined independently of visibility, enabling object localization to be evaluated even under partial or complete occlusion. These object-level annotations are used exclusively for evaluation and are not used to train either the representation or transport model.

CLEVR3D (Stelzner et al., 2021a) consists of scenes composed of CLEVR primitives observed from multiple viewpoints. Each scene contains up to ten objects, including spheres, cylinders, and cubes with two possible sizes. Each scene is rendered from three cameras separated by approximately 120° in azimuth. Small perturbations in camera azimuth, radius, and elevation are applied independently across scenes, while all cameras are oriented toward the scene center. The original images are rendered at 240×320 resolution; we take the central 240×240 crop and resize it to 128×128 for both training and evaluation. We train on scenes containing at most six objects and evaluate on the held-out test split (scene index $\geq 85{,}000$), restricted to scenes containing at most six objects and complete observations from all three viewpoints.

All object-level ground truth is derived directly from the simulator metadata provided with CLEVR3D, rather than from learned detectors or cross-view matching heuristics. The metadata contains the world-space 3D position, shape, size, and orientation of each object, together with the camera positions and per-object instance segmentation masks. Using this information, we reconstruct the corresponding look-at camera model and analytically project the stored 3D center of each object into each viewpoint to obtain 2D position ground truth. We validate the reconstructed projection against the pixel coordinates provided by the dataset and obtain a maximum discrepancy of 0.375 pixels in our 128×128 image space.

For scale evaluation, we compute an amodal 2D bounding box by projecting the known 3D geometry of each object into the corresponding camera view. Specifically, we project sampled surface

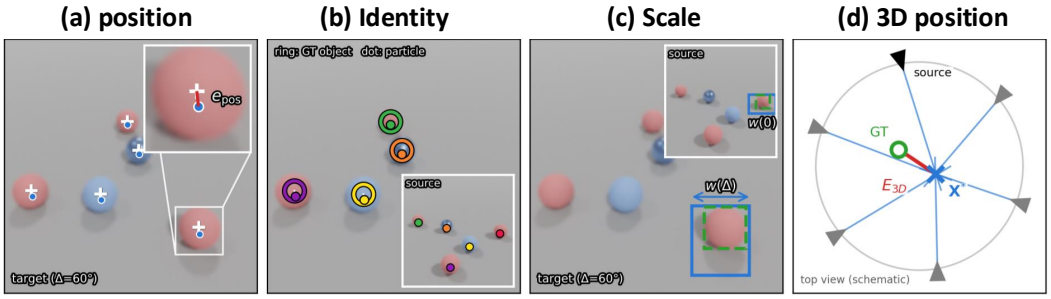


Figure 3: **Evaluation Metric overview.** From left to right: (a) position error, (b) identity preservation, (c) scale consistency, and (d) 3D position error estimated via multi-view triangulation.

points for spheres, rotated 3D corners for cubes, and points sampled along the two end circles for cylinders, using the object size and orientation stored in the metadata. Because these quantities are computed from the underlying simulator state, they remain well-defined even when an object is partially occluded by another object. Persistent object identity is obtained from the provided instance segmentation masks, where the same simulator entity index identifies the same object across all three viewpoints. The stored world-space object centers are additionally used as ground truth for evaluating 3D positions recovered through triangulation.

The reconstructed analytic camera model also allows us to evaluate geometric transport at intermediate azimuths between the three rendered viewpoints. During training, the model observes only the three approximately $120°$-separated viewpoints, while additional azimuths used for continuous-view transport analysis remain unseen during training.

PandaPush (Haramati et al., 2024; Qi et al., 2025) is an Isaac Gym-based robotic manipulation environment (Makoviychuk et al., 2021) in which a Franka Panda robot pushes three colored cubes toward target configurations. We retain the original physical environment, robot dynamics, action space, and task definition, while modifying only the camera configuration to match our view-transport setting.

The original EC-Diffuser setup uses two fixed cameras placed at different positions and heights. We do not use these cameras. Instead, we configure cameras on a common ring such that they share the same radius and height and differ only in azimuth. Specifically, the cameras are placed at a radius of $0.8\,\mathrm{m}$ and a height of $1.75\,\mathrm{m}$, with azimuths of $0°$, $-45°$, and $-90°$. This configuration isolates azimuthal viewpoint changes and aligns with our transport formulation, which transforms source-view object representations according to relative camera azimuth. We configure these viewpoints using the ring-camera functionality provided in the released EC-Diffuser code.

All cameras use a resolution of 128×128 and a $35°$ field of view. We generate 2,448 episodes by rolling out the officially released pretrained EC-Diffuser policy checkpoint (Qi et al., 2025), with each episode containing 100 simulation frames. At every timestep, the three cameras simultaneously observe the same simulator state, ensuring that the multi-view observations are temporally aligned and share identical robot and object configurations. Although we also collected a denser camera-ring configuration during dataset development, all PandaPush experiments reported in this paper use only these three viewpoints.

We use PandaPush as a downstream benchmark to evaluate whether the learned object-centric representation and transported views are useful for robotic manipulation, rather than for object-level geometric evaluation.

A.3.3 Evaluation Metrics

Source-object matching. For each source view, we first establish correspondences between inferred foreground object representations and ground-truth objects. We use the term *object representation* to refer generically to an indexed object-level representation produced by a model; in our model, each object representation corresponds to a particle. We convert each representation's normalized 2D position to pixel coordinates in the 128×128 image space, and perform an injective Hungarian

assignment between active foreground object representations and observable ground-truth objects using Euclidean pixel distance as the matching cost. Among the resulting representation–object pairs, we retain only those with a source-view localization error of at most 8 pixels as valid source anchors.

Once a source object representation k is anchored to a ground-truth object i, we track the same representation index k after transport to all target viewpoints. Thus, correspondence is not re-established independently at each target view; instead, the source-view representation–object association defines which physical object the transported representation is expected to represent. This source anchoring is shared across the position, scale, identity, and triangulated 3D position metrics.

Image reconstruction. We report PSNR, SSIM, and LPIPS to evaluate target-view image reconstruction quality. These metrics provide the standard image-space comparison used by novel-view synthesis methods, while the object-level metrics below evaluate properties that are not directly exposed by image-space representations.

2D position error. For an object representation k anchored to ground-truth object i in the source view, we measure the Euclidean distance between the transported representation position $\hat{p}_k(\Delta)$ at target viewpoint Δ and the ground-truth image-plane center $c_i(\Delta)$ of the same object:

$$e_{\mathrm{pos}}(k, i, \Delta) = \|\hat{p}_k(\Delta) - c_i(\Delta)\|_2 \,.$$

The error is computed in pixels in the 128×128 image space and aggregated over evaluated objects and target viewpoints. This metric measures how accurately each object's position is transported across viewpoint changes while preserving the source-established correspondence.

Identity accuracy. Accurate target-view localization alone does not guarantee that an object representation associated with a physical object in the source view continues to represent the same object after a viewpoint change. We use the term *object representation* to refer generically to an indexed object-level representation produced by a model; in our model, each object representation corresponds to a particle.

At each target viewpoint, transported object representations are matched to observable ground-truth objects using an injective Hungarian assignment based on Euclidean pixel distance. If object representation k is anchored to ground-truth object i in the source view, identity is considered correctly preserved only when the same representation index k is assigned to the same object i in the target view. Identity accuracy is defined as the fraction of evaluated source-anchored object representations whose identities are preserved:

$$\mathrm{Acc}_{\mathrm{id}} = \frac{\#\text{ correctly preserved identities}}{\#\text{ evaluated source-anchored representations}}.$$

The target-view assignment is injective, so each ground-truth object can be assigned to at most one predicted representation. Consequently, swaps in object identity across representations are counted as identity errors. This metric therefore measures whether the physical-object correspondence established in the source view is preserved by the same representation index after transport.

Scale-consistency error. We evaluate whether the 2D extent of each object representation changes consistently with the projected extent of the corresponding ground-truth object across viewpoints. We define the ground-truth scale using the analytic amodal 2D bounding box obtained by projecting the object's known 3D geometry into each camera view. Let $w_i^{\mathrm{GT},s}$ and $h_i^{\mathrm{GT},s}$ denote the ground-truth bounding-box width and height of object i in the source view, and $w_i^{\mathrm{GT},t}$ and $h_i^{\mathrm{GT},t}$ those in the target view. The per-axis ground-truth scale ratios are

$$\rho_i^w = \frac{w_i^{\mathrm{GT},t}}{w_i^{\mathrm{GT},s}}, \qquad \rho_i^h = \frac{h_i^{\mathrm{GT},t}}{h_i^{\mathrm{GT},s}}.$$

This definition captures not only perspective-induced size changes but also viewpoint-dependent changes in the object's projected silhouette.

On the model side, object extent is measured from the axis-aligned bounding box of the pre-compositing alpha mask produced by each object representation. The mask is taken before depth

compositing across representations, and pixels with alpha values of at least 0.05 are included. Let w_k^s and h_k^s denote the bounding-box width and height of representation k in the source view, and w_k^t and h_k^t those after transport to the target view. We measure the discrepancy between the predicted and ground-truth source-to-target extent ratios in log-ratio space:

$$e_{\text{scale}}(k,i) = \frac{1}{2}\left(\left|\log_2 \frac{w_k^t/w_k^s}{\rho_i^w}\right| + \left|\log_2 \frac{h_k^t/h_k^s}{\rho_i^h}\right|\right).$$

The log-ratio formulation treats over- and under-estimation symmetrically in multiplicative scale, while the use of source-to-target ratios makes the metric independent of the absolute scale of the model-side readout. Let m denote the median log-ratio error over all evaluated pairs. We report the final scale error as a multiplicative deviation in percentage:

$$E_{\text{scale}}(\%) = 100\,(2^m - 1)\,.$$

Ground-truth amodal bounding boxes are computed from the known 3D geometry and orientation of each object. For TP-OBJ3D, we project surface samples for spheres and the eight oriented corners for cubes. For CLEVR3D, we additionally project points along the two end circles of cylinders. We validate the analytic bounding boxes against rendered segmentation masks for unoccluded objects; on the dense TP-OBJ3D split, the median analytic-to-mask bounding-box ratios for cubes are 0.997 and 1.000 for width and height, respectively. Analytic projections similarly agree closely with rendered object extents for spheres, cylinders, and cubes in CLEVR3D.

Oracle-correspondence evaluation. In the default object-level evaluation, we track the index of each object representation anchored to a ground-truth object in the source view across target viewpoints. This protocol therefore evaluates geometric accuracy while preserving the source-established cross-view correspondence. Some baselines do not maintain a consistent object-representation index across viewpoints, making the same index-tracked protocol inapplicable. To evaluate the geometric representation quality of such models, we additionally use an oracle-correspondence protocol.

In the oracle setting, active object representations are independently matched to observable ground-truth objects at each target viewpoint using an injective Hungarian assignment with Euclidean 2D position as the matching cost. The resulting target-view correspondences are used to compute position and scale errors. For triangulation, the 2D position of the representation independently matched to the same ground-truth object at each viewpoint is used to construct the corresponding camera ray. Identity accuracy is computed only under the index-tracked protocol.

A.4 Cross-View Generalization and Correspondence Analysis

A.4.1 Viewpoint Generalization

We evaluate cross-view generalization on the CLEVR3D test set by sweeping the relative camera viewpoint at 1° intervals, covering both the viewpoints observed during training (0°, 120°, and 240°) and the unseen viewpoints between them. Since OSRT slots do not explicitly encode 2D position, we obtain a positional readout for each slot from the weighted centroid of its decoder attention map when rendering from the queried camera. As shown in Figure 4, OSRT produces strong reconstructions and meaningful object-level readouts near the viewpoints observed during training, but its reconstruction quality and object-level position, scale, and identity measurements degrade substantially at unseen viewpoints.

This behavior can be understood from how the two models represent viewpoint changes. In OSRT, the slot representation itself is view-agnostic, while viewpoint variation is modeled implicitly within the decoder conditioned on the query rays. With only three sparse training viewpoints, the decoder therefore receives direct supervision only around a narrow distribution of ray configurations and must generalize to unseen ray configurations between them. In contrast, PORT explicitly transports low-dimensional object attributes, including position, scale, and depth, as a function of the relative camera viewpoint, while its decoder is agnostic to the target camera. Consequently, unseen viewpoints are represented through explicit transformations of the object latents within the same latent space rather than through unseen camera queries to the decoder. As further illustrated in Figure 5,

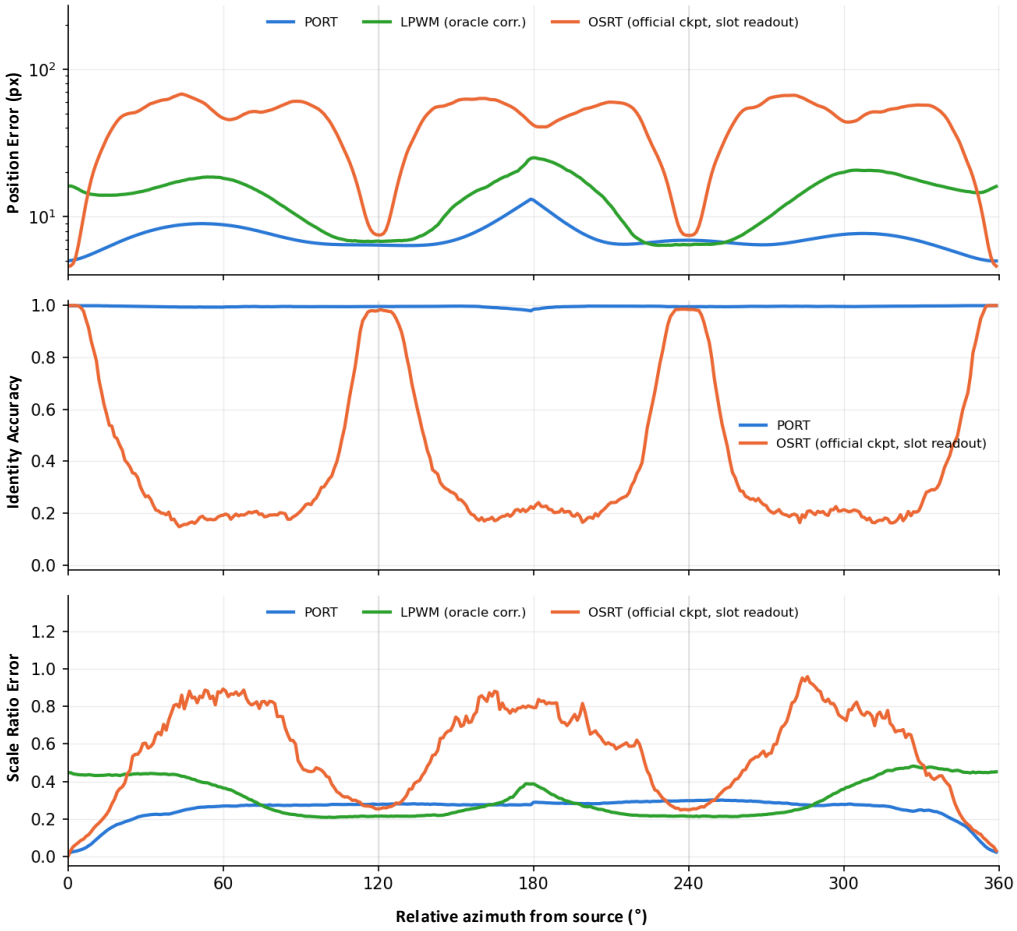


Figure 4: **Object-level evaluation across relative camera azimuths on CLEVR3D.** From top to bottom, we report position error, identity accuracy, and scale ratio error, comparing PORT with LPWM (oracle correspondence) and OSRT (official checkpoint). Gray vertical lines indicate the relative viewpoints used for training. PORT maintains high identity consistency while stably transporting position and scale across the full azimuth range, whereas the comparison methods degrade as the target viewpoint moves farther from the training viewpoints.

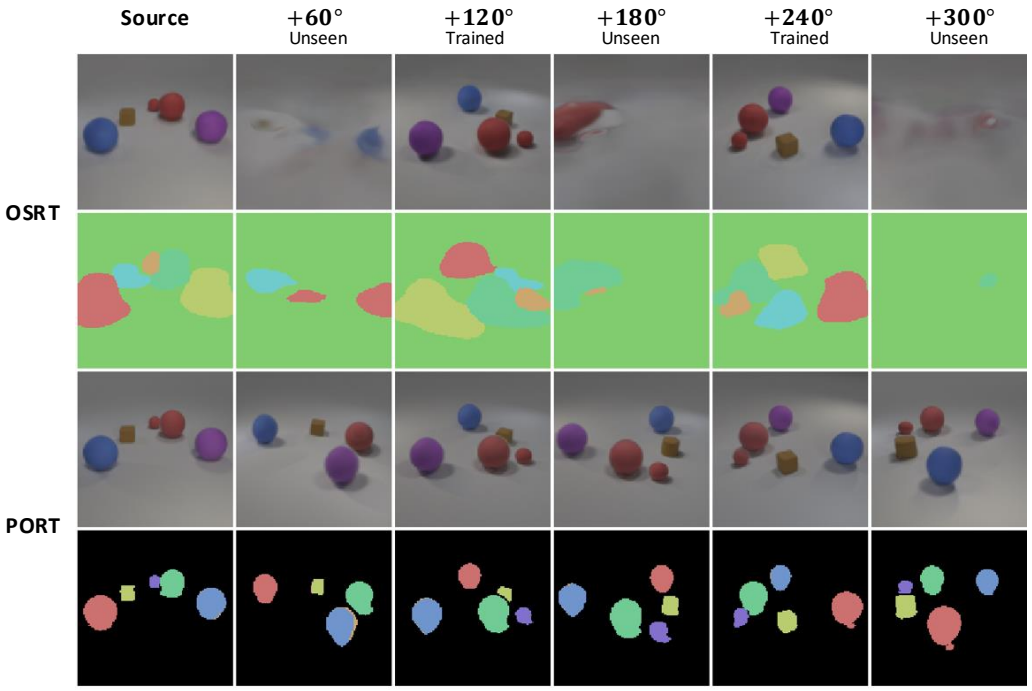


Figure 5: **Qualitative comparison on CLEVR3D.** The top two rows show OSRT's generated target-view images and mask maps, while the bottom two rows show PORT's generated target-view images and mask maps. OSRT degrades at unseen viewpoints, whereas PORT maintains coherent generation and object-level decomposition.

OSRT degrades substantially at unseen viewpoints, whereas PORT maintains stable object representations and reconstructions throughout the continuous viewpoint sweep. These results show that strong novel-view reconstruction at the training viewpoints does not by itself guarantee a persistent object representation under continuous viewpoint changes.

A.4.2 Particle Correspondence under Viewpoint Changes

In LPWM (Daniel et al., 2026), particle identity is defined by the image patch from which the particle originates rather than by a particular physical object. This induces a particle-grid regime in which each particle is associated with a local region around its originating patch, and object features can be transferred to neighboring particles when the object's image-space projection moves outside that region. Consequently, under viewpoint changes that induce large image-space displacements, the same particle index in the source and target views is not guaranteed to correspond to the same physical object.

This behavior is further related to the fact that the LPWM encoder processes each frame independently and does not explicitly establish cross-view associations. LPWM also replaces explicit tracking with this grid-based representation and is primarily designed for settings with relatively small camera motion. In contrast, PORT directly transports each source particle to the target viewpoint conditioned on the relative camera pose, thereby preserving particle identity across views. Figure 6 illustrates this difference, while Figure 7 shows how the lack of persistent correspondence can lead to incorrect state reassignment and degraded cross-view generation after object-level editing. These design choices in LPWM therefore do not enforce persistent object-level correspondence across viewpoints.

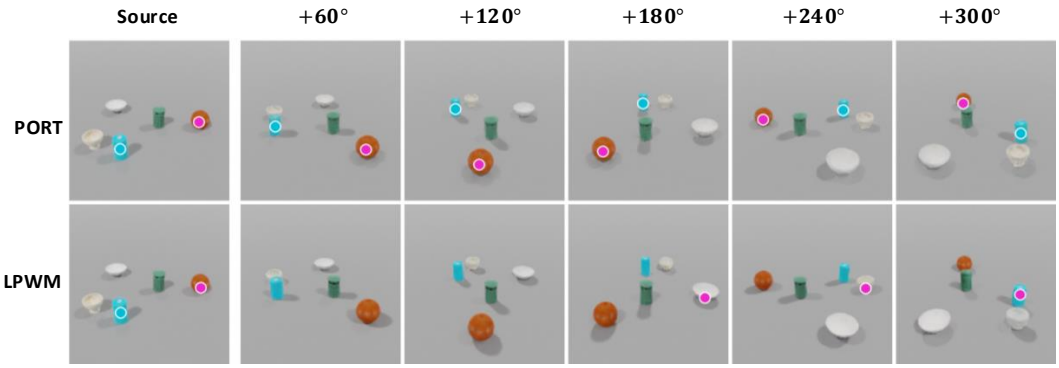


Figure 6: **Particle-to-object correspondence across viewpoints.** We visualize how the particles selected in the source view correspond to objects across target viewpoints. PORT preserves the identities of the selected source particles, such that each particle consistently tracks the same physical object across all views. In contrast, LPWM does not maintain persistent particle-to-object correspondence during viewpoint rollout, causing the assignment of a particle to change across views.

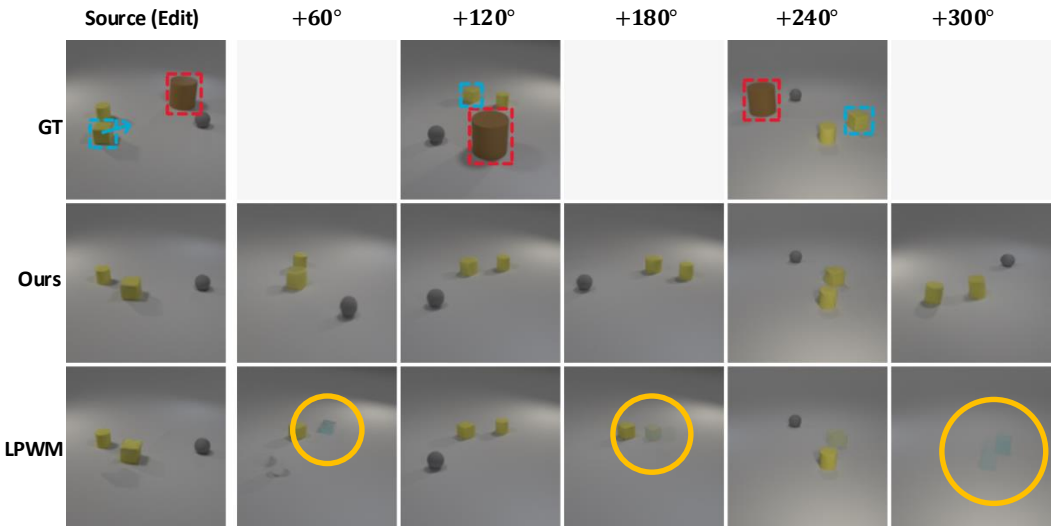


Figure 7: **View-consistent object-level scene editing.** We remove the cylinder marked by the red box and move the yellow cube marked by the blue box in the source view, and propagate the edited representation across target viewpoints. PORT consistently preserves both edits across viewpoint changes. In contrast, LPWM does not maintain persistent particle-to-object correspondence, causing edited object states to be incorrectly propagated and producing visible artifacts across views, highlighted by the yellow circles. The available ground-truth views at $+120°$ and $+240°$ are shown only as references to the original, unedited scene; ground truth is not available for the edited scene.

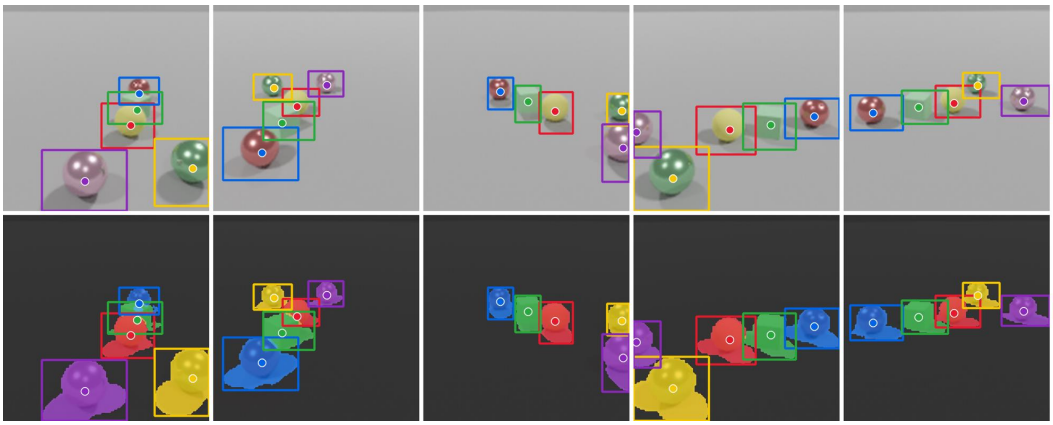


Figure 8: **Particle position and scale visualization on TP-OBJ3D.** The DLP encoder occasionally assigns part of an object's cast shadow to the same particle as the object itself, causing the predicted particle center to shift downward from the true object center in the image plane.

A.5 Triangulation Details

A.5.1 Triangulation Protocol

In the main paper, we evaluate whether the 2D locations of the same object transported across multiple viewpoints are consistent with a single underlying 3D position. We first match particles to ground-truth objects in the source view and then transport the matched particle to each target viewpoint. The predicted 2D object center at each viewpoint is converted into a 3D camera ray using the calibrated camera parameters, and the object's 3D position is estimated by computing the least-squares intersection of the resulting rays. On TP-OBJ3D, we use the predicted locations from all six ring views and measure the Euclidean distance between the triangulated 3D position and the ground-truth object center. All 3D errors are reported in world-coordinate units, where the object radius is 0.5. As a sanity check, applying the same triangulation procedure to the ground-truth 2D projections yields only numerical-precision error, confirming the correctness of the camera projection and triangulation implementation.

A.5.2 Analysis of the z-Axis Bias

A substantial portion of the 3D localization error reported in the main paper arises from a systematic bias along the z axis. As illustrated in Figure 8, the DLP encoder occasionally captures part of an object's cast shadow as part of the particle representing that object. This causes the predicted 2D particle center to shift slightly downward from the true object center in the image plane. Because this downward localization bias is repeated across viewpoints, it results in a consistent offset along the z axis in the triangulated 3D position.

A.6 Linear Probing Details

A.6.1 Probing Protocol

Following the linear-probing approach of Daniel & Tamar (2022), we freeze all learned representations and train only a linear regressor to predict object-level 3D position and depth. Since object ordering is not fixed in general multi-object scenes, we first match objects to representation units in the source view and independently probe each matched object representation.

The train, validation, and test splits are constructed at the scene level so that different views of the same scene never appear across different splits. We evaluate 3D position in a coordinate frame aligned with the source-camera azimuth. For depth, we use the object's depth relative to the scene center as the primary target, removing the camera-to-scene-center distance from the raw camera-axis depth.

Probe Input	**Dim.**	**TP-OBJ3D**	**CLEVR3D**
Source position only	2	0.485	0.446
Source particle attributes	10	0.396	0.316
Source position + transported positions	12	0.151	0.099
Source attributes + transported states	**20**	**0.140**	**0.066**

Table 6: Per-object 3D-position probing with progressively richer inputs. Most of the improvement comes from adding correspondence-preserving transported object states, while the remaining particle attributes provide a smaller additional gain.

Representation	Dim.	TP-OBJ3D 3D ↓	CLEVR3D 3D ↓
Patch AE	8,192	1.526	1.652
OSRT	196k / 230k	1.934	**1.344**
LPWM, source only	124	1.166	1.412
LPWM, + camera-action rollout	244	1.166	1.414
PORT, source only	124	1.173	1.389
PORT, + transported views	244	**1.119**	1.360

Table 7: Whole-scene 3D-position probing. Particle representations are sorted by their predicted image-space x coordinate before flattening to provide an observable ordering. Unlike per-object probing, this protocol additionally requires a linear map to route an unordered object representation to ordered output slots.

For PORT and LPWM, the source representation contains the particle position, scale, depth, presence, and appearance attributes. When cross-view information is included, we additionally use the object states predicted at other viewpoints. Patch AE does not provide explicit object-level units, so we probe the local feature centered at each object centroid together with its 2D centroid position. OSRT provides object-centric slots; we therefore evaluate both the object-aligned slot representation alone and the same slot augmented with its decoded 2D centroid, allowing us to separate the information contained in the slot latent from that provided by an explicit positional readout.

A.6.2 Additional Linear-Probing Analysis

What contributes to 3D-position probing performance? To identify which components of the representation contribute to 3D-position prediction, we progressively add information to the per-object probe input while keeping the matched object population fixed. Table 6 summarizes the results. Using only the source-view 2D particle position yields errors of 0.485 and 0.446 on TP-OBJ3D and CLEVR3D, respectively. Adding the remaining source particle attributes—scale, depth, presence, and appearance—reduces the errors to 0.396 and 0.316. The largest improvement, however, comes from adding the transported positions of the same object across viewpoints: even without the remaining particle attributes, the error drops to 0.151 and 0.099. Using the full source attributes together with the transported states further reduces the error to 0.140 and 0.066. These results show that most of the linearly accessible 3D-position information comes from the correspondence-preserving cross-view object states produced by transport, rather than from simply increasing the dimensionality of the source-view representation.

Whole-scene probing. In addition to per-object probing, we evaluate whether a linear regressor can jointly predict the 3D positions of all objects from the complete scene representation (Table 7). For particle-based representations, particles are sorted by their predicted source-view image-space x-coordinate and flattened into a single vector, without using ground-truth object identities. For OSRT, we use its full scene-level set representation for whole-scene probing, whereas the per-object results in the main paper use object-aligned slots, with and without the decoded object position.

On TP-OBJ3D, PORT achieves an error of 1.173 using only the source representation, which decreases to 1.119 when transported views are included. The error similarly decreases from 1.389 to 1.360 on CLEVR3D. In contrast, adding the camera-conditioned rollout to LPWM produces almost no change on either dataset. On CLEVR3D, OSRT achieves the lowest whole-scene error of 1.344.

Unlike per-object probing, whole-scene probing requires the linear regressor not only to extract 3D information from the representation, but also to resolve the correspondence between input particles and ordered output object slots. As a result, the protocol is sensitive to particle ordering and generally yields higher error than per-object probing. Consistent with this interpretation, sorting particles by their source-view image-space x-coordinate substantially improves performance compared with using the original particle ordering. Thus, whole-scene probing reflects not only representation quality but also the consistency of particle ordering.

A.6.3 Comparison with LPWM Rollouts

To isolate the effect of transported cross-view states, we compare PORT with LPWM, which uses the same particle-based representation family. Unlike PORT, LPWM does not directly move a persistent object particle across viewpoints. Instead, each particle remains anchored to a local patch or spatial carrier, and its next state is predicted conditioned on the camera transformation. As the viewpoint changes, information about the same physical object may therefore be transferred to a different particle, and a particle index is not explicitly constrained to represent the same object across views. LPWM can still generate the target-view reconstruction, but its rollout does not guarantee persistent cross-view object correspondence.

Using only the source-view representation, PORT and LPWM achieve comparable 3D-position probing performance. However, their behavior differs substantially when predicted multi-view states are added. For PORT, the error decreases from 0.396 to 0.140 on TP-OBJ3D and from 0.316 to 0.066 on CLEVR3D. In contrast, LPWM changes only from 0.462 to 0.455 and from 0.324 to 0.320, respectively. These results suggest that simply adding predicted states from multiple viewpoints is not sufficient. Instead, correspondence-preserving cross-view object states are important for making 3D position linearly accessible.

A.7 Manipulation Details

Task. We use the PandaPush task from EC-Diffuser Qi et al. (2025), a goal-conditioned multi-object manipulation environment simulated in IsaacGym. A Franka Panda robot is required to push three colored cubes to their corresponding target locations specified by a goal image. The goal is provided as an RGB image showing the cubes at their target positions with the robot arm retracted from view, rather than as explicit coordinates. At each control step, the policy predicts a three-dimensional end-effector displacement, and each episode lasts for 100 steps. An episode is considered successful if, at the final step, all cubes lie within 4 cm of their respective target positions.

Camera and Observation Setup. The original EC-Diffuser setup uses front and side cameras placed at different positions and heights. Since PORT models viewpoint changes through relative camera azimuth, we instead use a ring-camera configuration in which all cameras share the same height and radius and differ only in azimuth. We use three views: a front view, a side view, and an intermediate view between them. Each observation and goal image is encoded with the same frozen DLP encoder, and the resulting real-view or PORT-transported particles are used as policy inputs.

Policy Inputs and Baselines. In Table 4, R denotes particles obtained by encoding a real camera image with the frozen DLP encoder, while T denotes particles produced by transporting particles from one real view to another viewpoint using PORT. The **2R** setting is the EC-Diffuser baseline and uses two independently encoded real images from the side and front cameras. Because the two images are encoded independently, the same particle index is not guaranteed to represent the same object in both views. We therefore provide the two particle sets to the policy following the original EC-Diffuser input format.

PORT uses only the side-view image as a real observation. In **1R+1T**, the side-view particles are transported to the front viewpoint. In **1R+2T**, we additionally transport them to the intermediate viewpoint. Since PORT keeps the particle index fixed during transport, particle i in the source and transported views refers to the same source object representation. We therefore concatenate the features of same-index particles across views and use the resulting vector as one object token for the policy. The goal image is processed in the same way.

Baselines and Policy Inputs. We use EC-Diffuser (Qi et al., 2025) as our main baseline. EC-Diffuser converts the RGB observation and goal image into object-centric particle sets using a frozen DLP encoder, and feeds these particles to an entity-centric diffusion policy. Each particle represents object-level state variables such as position, scale, depth, and appearance, and the policy predicts manipulation actions conditioned on the particle representations of the current observation and the goal. In the standard multi-view PandaPush setting, EC-Diffuser independently encodes two real camera images from the front and side views. Because the two views are encoded independently, particle indices are not explicitly aligned across views, and the two particle sets are provided to the policy as separate entities together with view information. The **2R** setting in Table 4 corresponds to this two-real-view EC-Diffuser baseline.

For the PORT-based inputs, we use only a single real camera observation. In Table 4, R denotes particles obtained by encoding a real image with the DLP encoder, while T denotes particles obtained by transporting the source particles to a target viewpoint using PORT. In **1R+1T**, the real side-view particles are combined with particles transported to the front viewpoint, while **1R+2T** additionally includes particles transported to the intermediate viewpoint. Because PORT preserves particle indices during transport, the same particle index across viewpoints corresponds to the same source object representation. We therefore concatenate the features of same-index particles across views to form a single object token, which is then provided to the same entity-centric policy. The goal image is processed using the same encoding and transport procedure as the observation.

Training and Evaluation Protocol. We first train PORT on the multi-view observations from PandaPush and then use the trained PORT together with the frozen DLP encoder to construct the policy inputs. For all comparison settings, we use the same EC-Diffuser policy architecture and training setup, while keeping both the DLP encoder and PORT frozen during policy training. All settings are trained using the same five random seeds to ensure a consistent comparison. Evaluation is performed in closed loop for 100-step episodes. An episode is considered successful if, at the final step, all cubes are within $4\,\text{cm}$ of their corresponding target positions. We report the mean and standard deviation of the success rate over the five seeds.

Representation-Construction Cost. To compare the computational cost of using additional real views with transporting particles, we measure the time required to construct the representations used as policy inputs. For a real view, we measure the time required to encode an RGB image into particle representations using the frozen DLP encoder. For a transported view, we measure the PORT forward pass that transforms the source particles to a target viewpoint. All timings are measured under the same hardware and inference setting, excluding camera capture and policy inference. Since the cost of PORT transport is substantially smaller than that of DLP encoding, additional viewpoints can be constructed with little extra computation by transporting source particles instead of encoding additional real images. The *Rep. Time* reported in Table 4 denotes this representation-construction cost.

A.7.1 Occlusion Analysis

To further understand why transported representations can substitute for real target-view observations, we analyze object localization under robot-arm occlusion. The results are summarized in Table 8. At the front viewpoint, the real-view representation localizes 85.6% of objects, whereas the transported representation localizes 91.8%. Moreover, 11.2% of objects are localized by the transported representation despite being missed by the corresponding real-view representation. Because PORT transports representations of objects already observed in the side-view source image rather than re-detecting them from the target-view image, these object states can remain available to the downstream policy under target-view occlusion.

A.8 Gaussian Splatting from Transported Views

PORT can transport an object-centric representation from a single source observation to multiple target viewpoints. In this experiment, we investigate whether the resulting multi-view observations are sufficiently consistent to serve directly as input to an existing 3D scene representation that relies on multi-view consistency. Our goal is not to propose a new Gaussian Splatting method or to improve 3D reconstruction performance itself. Instead, we examine whether a single source observation can

Table 8: Object localization under robot-arm occlusion.

Metric	Localized (%) ↑
Real-view localization	85.6
Transported-view localization	91.8
Recovered despite real-view miss	11.2

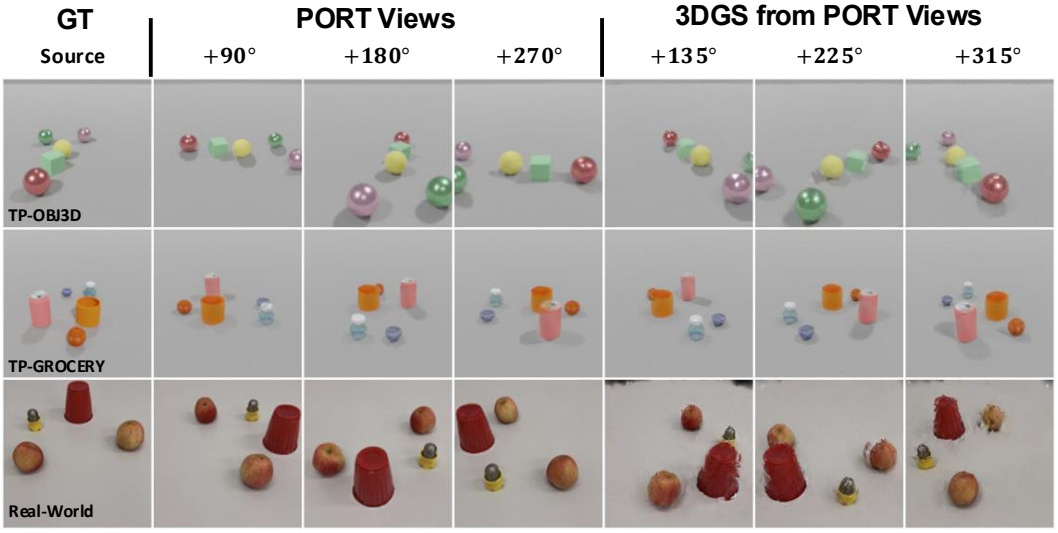


Figure 9: RGB Gaussian Splatting from PORT-generated views. Starting from a single source image, PORT generates multi-view RGB observations that are used to fit a 3D Gaussian scene. We show PORT views and 3DGS renderings from different viewpoints on TP-OBJ3D, TP-GROCERY, and real-world scenes.

be expanded by PORT into multi-view RGB images and object masks that can be directly connected to an existing 3D scene modeling pipeline. This provides an additional demonstration of whether the cross-view object representations learned by PORT can support 3D scene construction beyond 2D novel-view synthesis.

Experimental Setup. For each scene, we use only a single source RGB image as input. The source image is encoded into object-centric particles using the frozen DLP encoder, and PORT transports these particles to target viewpoints covering the full 360° camera ring. Specifically, we generate 72 viewpoints at 5° intervals. Each transported particle set is rendered by the frozen DLP decoder into either a 128×128 RGB image or particle-wise alpha masks. The resulting multi-view sequence, together with the corresponding camera parameters, is used to fit a whole-scene 3D Gaussian Splatting (3DGS) representation (Kerbl et al., 2023) with the gsplat implementation (Ye et al., 2025), using its default densification schedule. Because each target viewpoint is defined directly by its relative camera motion from the source camera, the camera pose of every generated view is known by construction and does not require separate pose estimation. We apply the same procedure to TP-OBJ3D, TP-GROCERY, and real-world scenes.

A.8.1 RGB Gaussian Splatting

Figure 9 shows RGB Gaussian Splatting results on TP-OBJ3D, TP-GROCERY, and real-world scenes. Using only multi-view images generated from a single source image, we fit a 3DGS scene and render it over a full 360° camera trajectory.

We further evaluate this quantitatively on TP-OBJ3D and TP-GROCERY. We use 20 scenes from each dataset and keep the 3DGS configuration fixed across all settings, changing only the images used for fitting. The GT setting fits 3DGS using all available real views and evaluates it at the same

Table 9: **Quantitative evaluation of RGB Gaussian Splatting.** Each entry reports PSNR / SSIM / LPIPS, averaged over 20 scenes. GT N fits 3DGS using all available real views and evaluates it at the same camera poses. All PORT-based settings use only one real source image. "PORT direct" evaluates PORT-generated target views without 3DGS, while the remaining settings fit 3DGS using different numbers of PORT-generated views.

Dataset	GT N	PORT direct (no 3DGS)	PORT N + 3DGS	PORT $2N$ + 3DGS	PORT 72 + 3DGS
TP-OBJ3D ($N = 6$)	53.12 / 0.997 / 0.072	38.95 / 0.983 / 0.029	38.88 / 0.982 / 0.083	38.28 / 0.979 / 0.108	35.97 / 0.969 / 0.145
TP-GROCERY ($N = 12$)	50.90 / 0.996 / 0.082	30.89 / 0.934 / 0.103	30.93 / 0.935 / 0.162	30.97 / 0.935 / 0.168	30.99 / 0.935 / 0.172

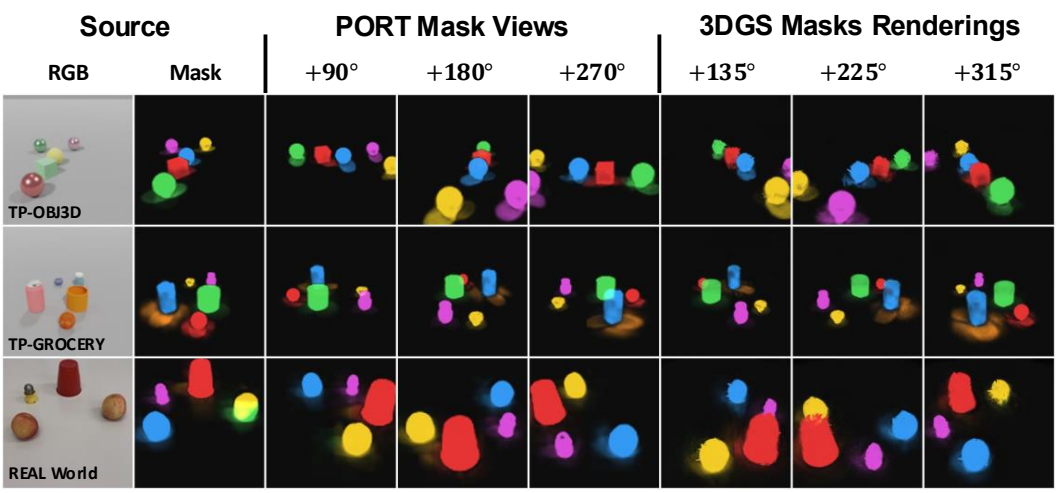


Figure 10: Self-supervised 3D object masks from PORT-generated particle masks. The transported particle masks are used to fit a 3D Gaussian representation, enabling object-level renderings across viewpoints on TP-OBJ3D, TP-GROCERY, and real-world scenes.

camera poses. In contrast, all PORT-based settings observe only a single real source image. "PORT direct" evaluates the target-view images generated by PORT without 3DGS, while the remaining settings fit 3DGS using N, $2N$, or 72 PORT-generated views.

As shown in Table 9, 3DGS fitted using N PORT-generated views produces PSNR and SSIM values similar to those of the target views generated directly by PORT. On TP-OBJ3D, PORT direct and the N-view 3DGS achieve 38.95 dB and 38.88 dB, respectively, while on TP-GROCERY they achieve 30.89 dB and 30.93 dB. These results show that multi-view observations generated from a single source image can be used to fit an existing 3DGS representation.

Increasing the number of generated views does not further improve reconstruction metrics at the real camera poses. Since all generated views are derived from the same single source observation, adding more views does not introduce additional real visual information or improve the quality of each individual target-view observation. Nevertheless, given that all PORT-based settings originate from only one real image, these results demonstrate the potential to connect PORT-generated multi-view observations to an existing 3D scene representation.

A.8.2 Self-Supervised 3D Object Masks.

Because PORT uses a particle-based representation, it can generate not only RGB images but also particle mask maps at target viewpoints. Following the same procedure as RGB Gaussian Splatting, we use the multi-view particle masks generated by PORT to construct a 3D object-mask representation. Figure 10 shows the resulting mask-based Gaussian representations rendered over the full 360° viewpoint range on TP-OBJ3D, TP-GROCERY, and real-world scenes.

An important advantage of this formulation is that it does not require additional segmentation supervision. Constructing an object-aware 3D representation from conventional multi-view RGB observations typically requires object segmentations to be obtained separately for each viewpoint. In

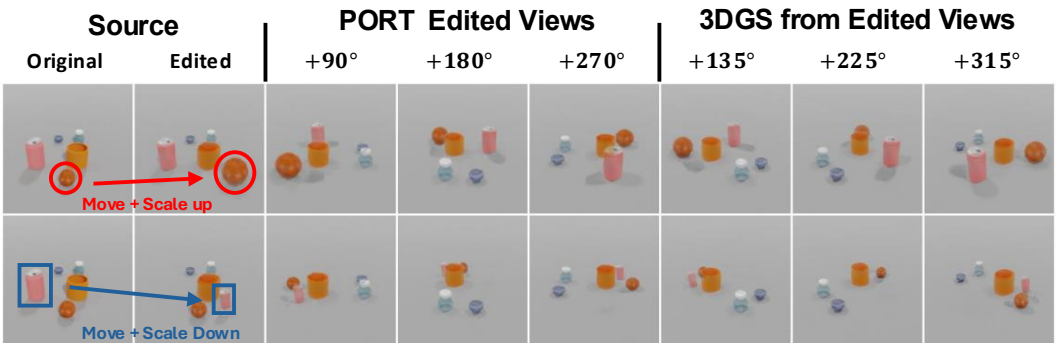


Figure 11: Object-level editing with transported views and Gaussian Splatting. We modify the position and scale of selected objects in the source representation, transport the edited representation across viewpoints, and fit a 3D Gaussian scene from the resulting views.

Table 10: **Ablation of scene-context usage on TP-OBJ3D.** We compare the reference PORT model with a variant without inter-particle scene context and a variant that additionally routes scene context to the appearance head.

Metric	Full Context	w/o Scene Context	+ Context to Appearance
PSNR ↑	**37.53**	32.05	36.43
LPIPS ↓	**0.023**	0.058	0.029
Position error (px) ↓	**2.56**	3.22	2.87
3D position error ↓	**0.163**	0.195	0.174
Ray-RMS ↓	**0.032**	0.083	0.036
Identity accuracy ↑	0.998	0.997	**0.999**
Scale error (%) ↓	14.7	15.0	**14.1**

contrast, PORT obtains object masks directly from its self-supervised particle decomposition and transports them across viewpoints. This allows multi-view object masks to be generated from a single source observation without ground-truth mask annotations or an external segmentation model, and subsequently extended into an object-level 3D representation.

A.8.3 Object-Level Editing.

PORT's particle-based representation allows individual objects to be directly edited in the source view and the resulting edits to be transported consistently across target viewpoints. Figure 11 shows examples in which a selected object is removed or moved, after which the edited particle representation is transported to multiple viewpoints. We further use these transported views to construct a 3D Gaussian scene, demonstrating that source-level edits are reflected consistently not only across the generated multi-view observations but also in the resulting 3D representation.

A.9 Ablation Analysis

A.9.1 Scene-Context Transformer

PORT uses a scene-context transformer to model relational information among source particles before transporting their attributes. Through self-attention, each particle can incorporate information from other particles in the scene, and the resulting context is provided to the transport heads for geometric attributes such as position, scale, and depth. To evaluate the contribution of this relational context, we train a variant without scene context, preventing each particle from accessing information from other particles. The variant is trained from scratch using the same training data, objectives, and optimization recipe as the full model; the only difference is the absence of inter-particle relational context.

Table 11: **Ablation of explicit particle correspondence encoding for robotic manipulation.** Explicitly adding a shared particle-index embedding across views does not improve over simple feature concatenation in the current PORT formulation.

Policy input	Success rate (%) ↑
Real + imagined (concatenation)	**77.1**
+ Particle identity embedding	63.5

As shown in Table 10, removing scene context substantially degrades novel-view reconstruction and geometric localization. The 2D position error increases from 2.56 to 3.22 pixels, while the triangulated 3D position error increases from 0.163 to 0.195. The largest difference appears in ray-RMS, which measures the geometric consistency of the transported positions across views, increasing from 0.032 to 0.083. In contrast, identity accuracy and scale error remain nearly unchanged. These results indicate that relational scene context primarily contributes to accurate geometric localization and cross-view geometric consistency, whereas object identity is preserved by the structured per-particle transport itself.

A.9.2 Scene Context for Appearance Transport

PORT uses the relational context produced by the scene-context transformer to transport geometric attributes such as position, scale, and depth, but does not directly provide this context to the appearance feature transport. To analyze this design choice, we train a variant in which the same scene context is additionally provided to the appearance head. In the reference model, appearance transport is conditioned on the source appearance feature together with geometric and camera information, whereas the variant additionally receives inter-particle scene context.

As shown in Table 10, adding scene context to the appearance head does not improve reconstruction or perceptual quality. PSNR decreases from 37.53 to 36.43, while LPIPS increases from 0.023 to 0.029, and identity accuracy remains near saturation for both models. We therefore do not observe evidence that scene-level relational information provides additional benefit for appearance transport. Together with the previous ablation, this suggests that the primary contribution of scene context lies in geometric transport rather than appearance modeling.

A.9.3 Explicit Particle Correspondence Encoding

In the downstream manipulation experiments in 5.4, we concatenate the real-view and imagined-view particle representations before providing them to the policy. While this exposes the policy to object-centric observations from both viewpoints, it does not explicitly encode the cross-view particle correspondence preserved by PORT. We therefore evaluate whether explicitly providing this correspondence can further improve downstream performance.

Specifically, we assign a learnable embedding $\mathbf{e}_k$ to each particle index k and add it to the corresponding particle representation. Let $\mathbf{z}_{t,v,k}$ denote the representation of particle k at time t and view v. The representation provided to the policy is

$$\tilde{\mathbf{z}}_{t,v,k} = \mathbf{z}_{t,v,k} + \mathbf{e}_k. \tag{1}$$

Since PORT preserves particle indices during view transport, corresponding real- and imagined-view particles receive the same embedding.

As shown in Table 11, explicit particle identity encoding does not outperform simple concatenation in the current setting. PORT is designed to preserve correspondence across viewpoints, but does not yet enforce persistent particle correspondence across time. Consequently, reusing the same particle-index embedding across temporal observations and the goal can introduce incorrect identity associations, despite being valid across views. We therefore find simple concatenation to be more effective in the current formulation. Extending PORT to preserve object correspondence along the temporal dimension may make explicit identity embeddings more beneficial by allowing the policy to exploit both cross-view and temporal correspondence.

A.10 Qualitative Results

We provide qualitative results of PORT. We first present novel view synthesis and latent particle decomposition results, illustrating how object-centric representations from the source view are transported and reconstructed at target viewpoints. We also visualize manipulation rollouts on PandaPush, where PORT-generated imagined-view latent particles are used as policy observations, demonstrating how the transported representations can support downstream policy learning. We further demonstrate the controllability and interpretability of the latent particles through object-level scene editing, where the edited scene remains consistent across viewpoint changes. Finally, we present representative failure cases and discuss the current limitations of PORT. Video versions of these results, including continuous 360° transport, manipulation rollouts, and Gaussian Splatting renderings, are available on our project website (`https://sites.google.com/view/particletransport`).

A.10.1 Novel View Synthesis and Latent Particle Decomposition

We present novel view synthesis and latent particle decomposition results on TP-OBJ3D, CLEVR3D, TP-GROCERY, PandaPush, and real-world scenes in Figure 12– 16. Each example shows the latent particles extracted from the source image, the particles transported to a target viewpoint, and the corresponding target-view reconstruction. These results qualitatively demonstrate that PORT maintains object-centric decomposition while producing viewpoint-dependent representations across scenes with varying complexity and domains. In addition, as shown in Figure 17– 19, PORT can estimate object-level depth from a single source image by triangulating object positions transported across multiple viewpoints.

A.10.2 Object-Level Depth from a Single Image

PORT can recover object-level depth from a single source image without a learned depth predictor. We first transport the active particles extracted from the source image to multiple virtual viewpoints around the source camera, obtaining multi-view 2D positions for each particle. Each predicted position is back-projected to a world-space ray using the known camera intrinsics and pose, and a 3D position is recovered for each particle through the same least-squares triangulation procedure used in our triangulation evaluation 5.3.

To visualize depth, we assign the distance between each triangulated 3D position and the camera center to the decoded alpha region of the corresponding particle in the selected viewpoint. Overlapping particles are combined using an alpha-weighted average. No depth supervision, learned depth predictor, or ground-truth 3D information is used in this procedure. Figure 17–19 shows the resulting triangulation-based depth estimates. Since PORT represents scenes using object-level particles, the resulting depth is limited to foreground objects and does not recover fine-grained intra-object surface depth or depth for the background and floor.

A.10.3 Policy Learning with Transport Latent Particles

Figure 20– 21 shows manipulation rollouts on PandaPush in which PORT-generated imagined-view latent particles are used as policy observations. The policy receives one real-view observation together with additional-view representations transported by PORT and manipulates the objects toward their target positions. These results qualitatively demonstrate that PORT's transported object representations can serve as effective multi-view observations for downstream control, beyond their use for novel-view reconstruction.

A.10.4 Interactive Scene Editing

PORT supports object-level scene editing through its interpretable latent particle representation while preserving the edited object state across viewpoints. As shown in Figure 22, we move the yellow ball upward in the source view and transport the edited particle to different target viewpoints. The edited object remains visible at a 40° camera rotation, becomes occluded around 82° and 200°, and reappears near 345°. This demonstrates that PORT remains robust to viewpoint changes even after object-level editing. Despite moving the object to a configuration not observed during training,

the transported object undergoes consistent viewpoint-dependent transformations: it becomes occluded at the appropriate viewpoints and reappears as the camera continues to rotate. This behavior suggests that PORT learns how to transport object latents according to camera viewpoint, rather than memorizing fixed scene configurations.

A.10.5 FAILURE CASES

PORT also transports only the particles observed in the source image. Therefore, as illustrated in Figure 23, it cannot recover objects that are completely hidden in the source view but become visible from a target viewpoint. This limitation reflects the inherent uncertainty of reasoning from a single observation. Future work could explore representations that explicitly group multiple particles belonging to the same object, together with generative mechanisms for modeling scene content that is unobserved in the source view.

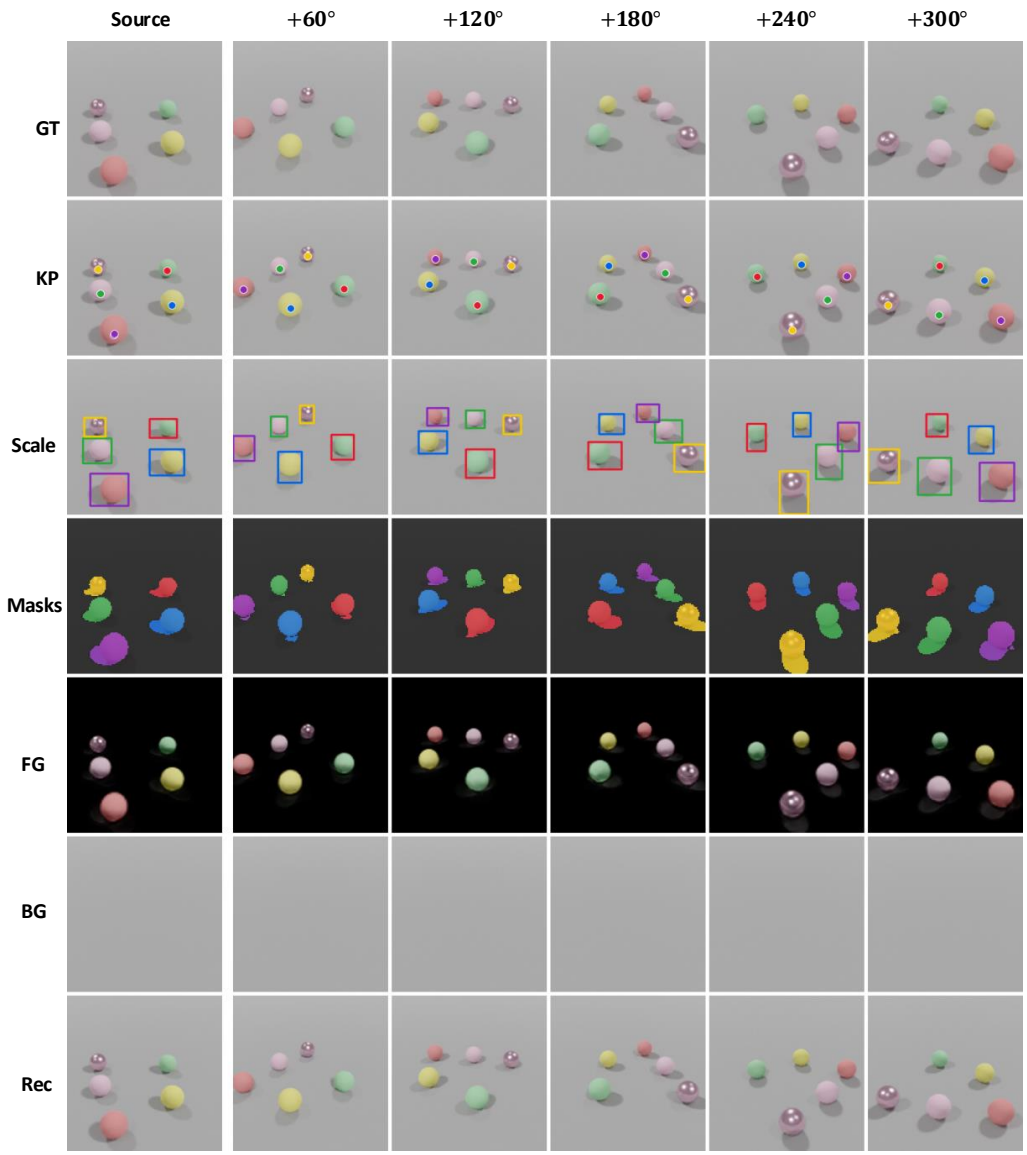


Figure 12: **Qualitative results on TP-OBJ3D.** GT: ground-truth image; KP: particle keypoints; Scale: particle scales; Masks: per-particle masks; FG: foreground; BG: background; Rec: reconstruction.

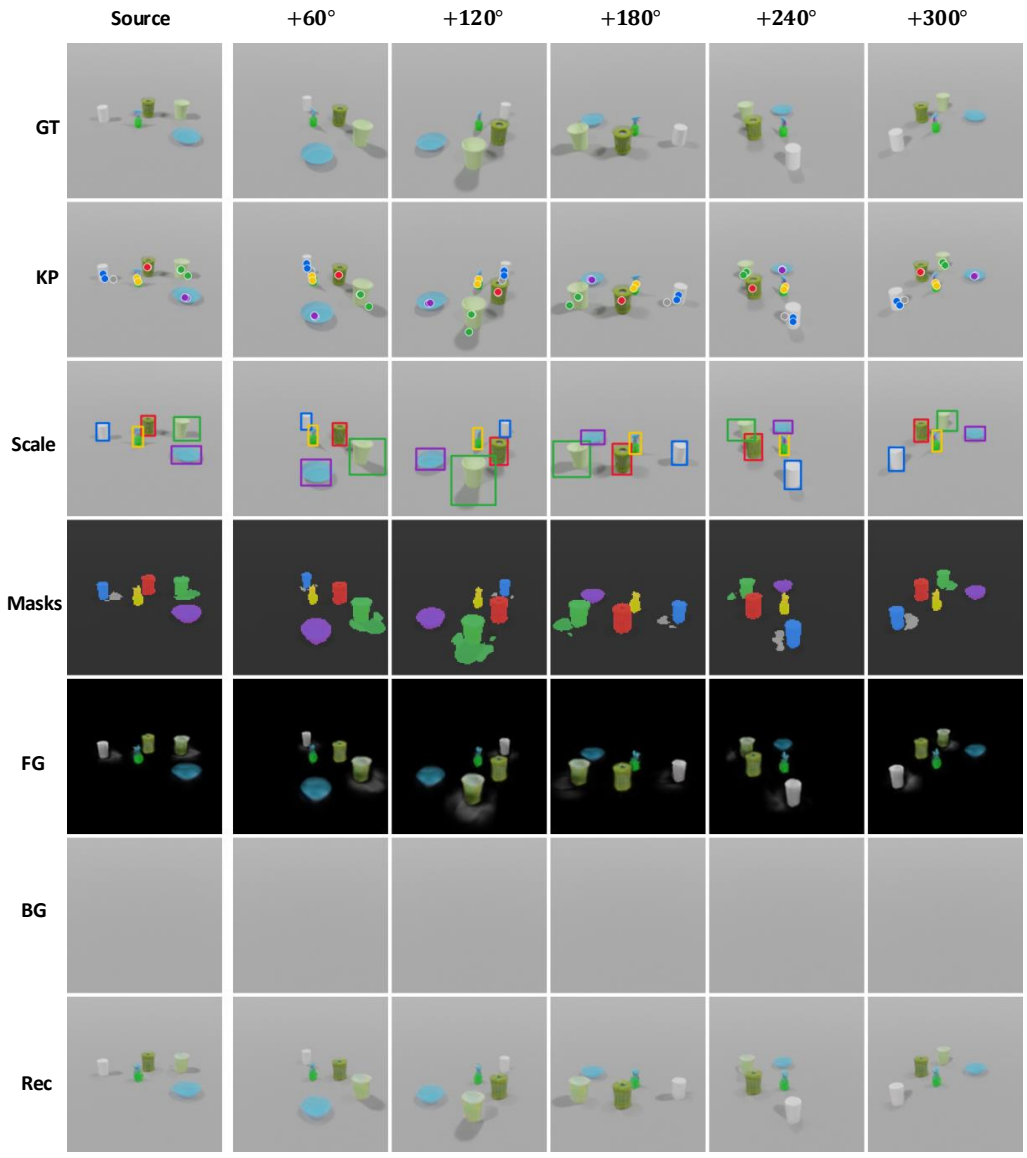


Figure 13: **Qualitative results on TP-GROCERY.** GT: ground-truth image; KP: particle keypoints; Scale: particle scales; Masks: per-particle masks; FG: foreground; BG: background; Rec: reconstruction.

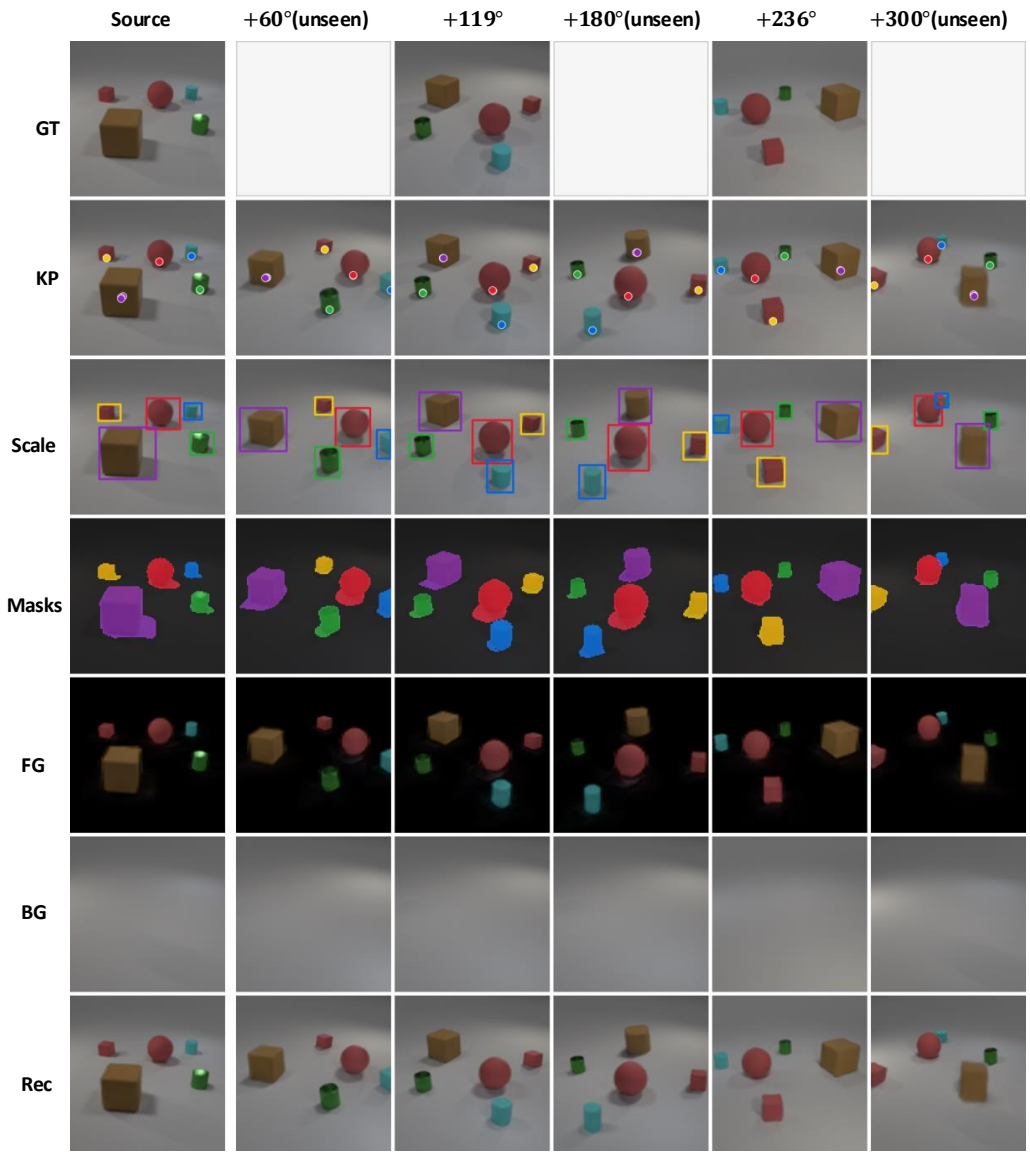


Figure 14: **Qualitative results on CLEVR3D.** GT: ground-truth image; KP: particle keypoints; Scale: particle scales; Masks: per-particle masks; FG: foreground; BG: background; Rec: reconstruction.

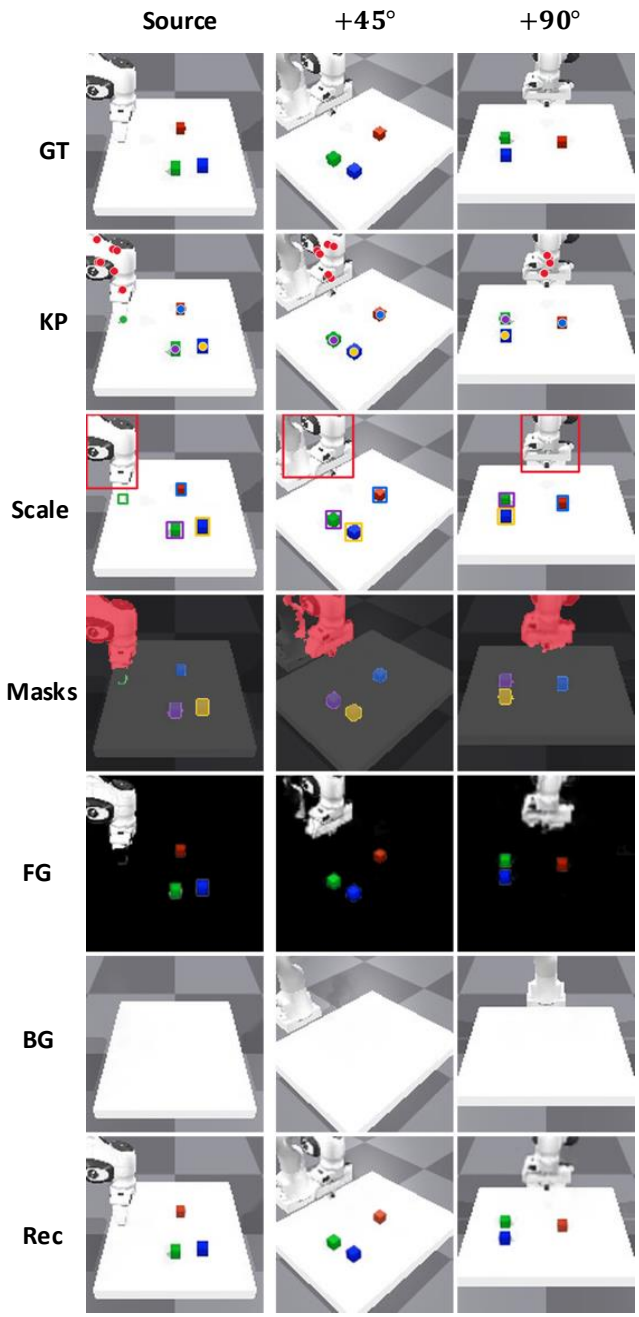


Figure 15: **Qualitative results on Panda Push.** GT: ground-truth image; KP: particle keypoints; Scale: particle scales; Masks: per-particle masks; FG: foreground; BG: background; Rec: reconstruction.

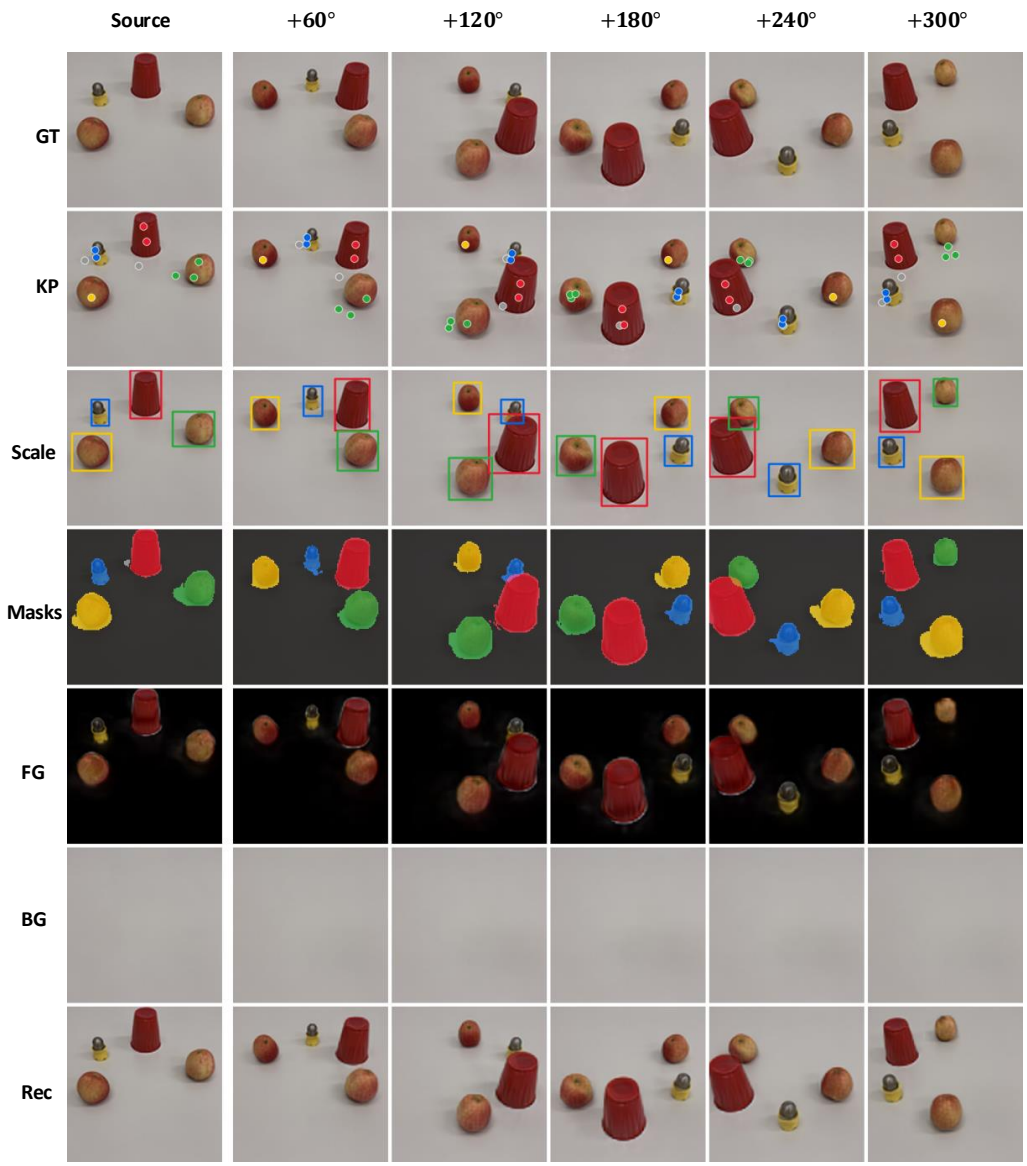


Figure 16: **Qualitative results on Real-world.** GT: ground-truth image; KP: particle keypoints; Scale: particle scales; Masks: per-particle masks; FG: foreground; BG: background; Rec: reconstruction.

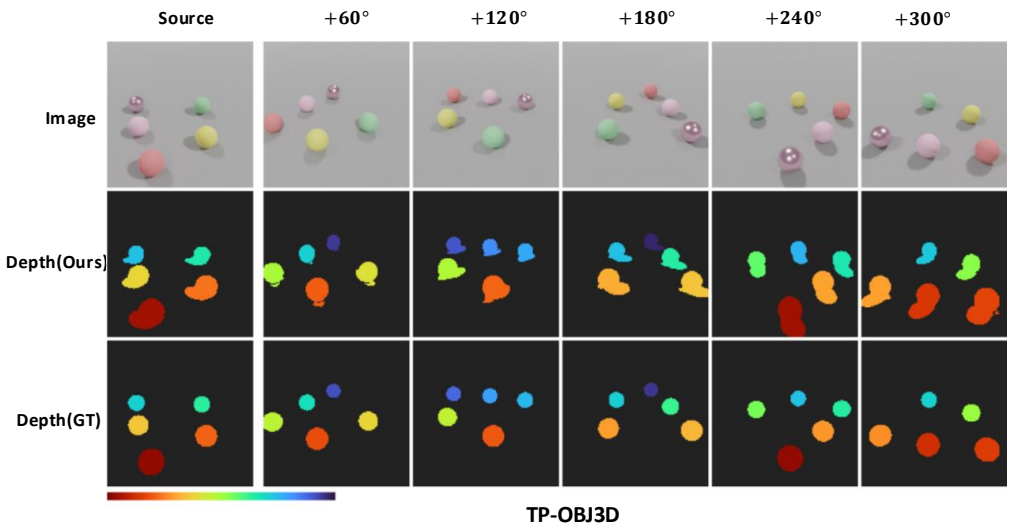


Figure 17: **Object-level depth estimation on TP-OBJ3D.** Object depth is approximated by triangulating PORT-transported object positions across multiple viewpoints.

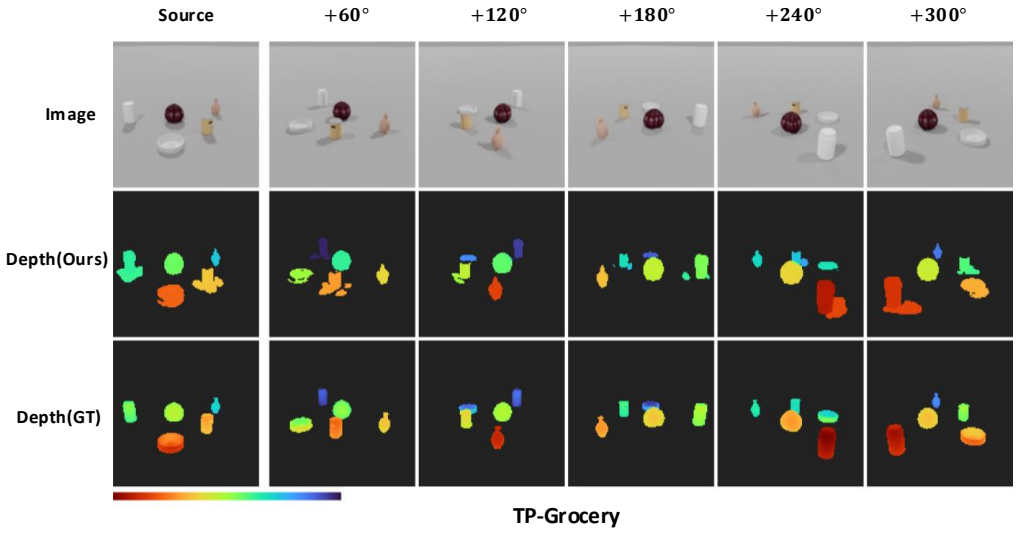


Figure 18: **Object-level depth estimation on TP-Grocery.** Object depth is approximated by triangulating PORT-transported object positions across multiple viewpoints.

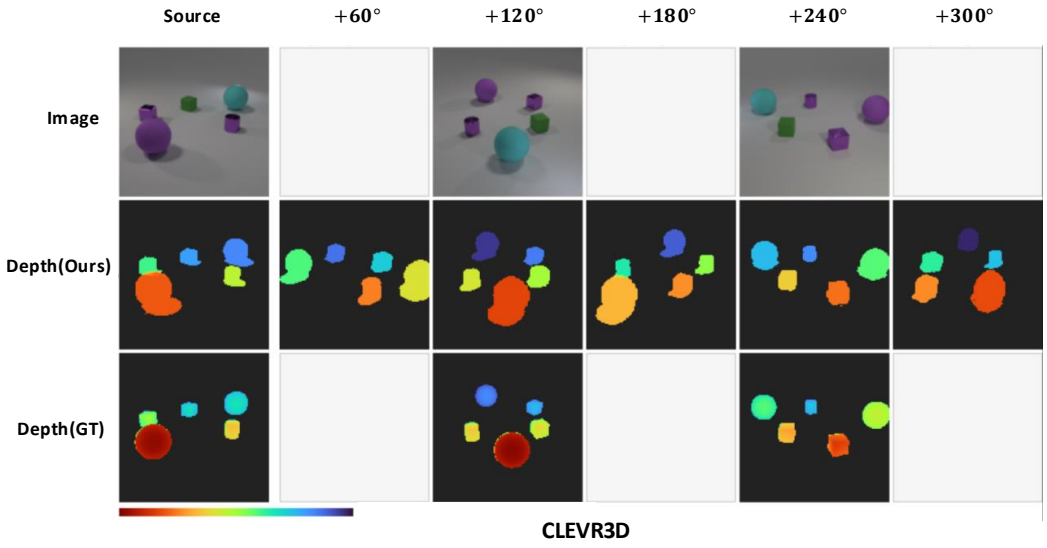


Figure 19: **Object-level depth estimation on CLEVR3D.** Object depth is approximated by triangulating PORT-transported object positions across multiple viewpoints.

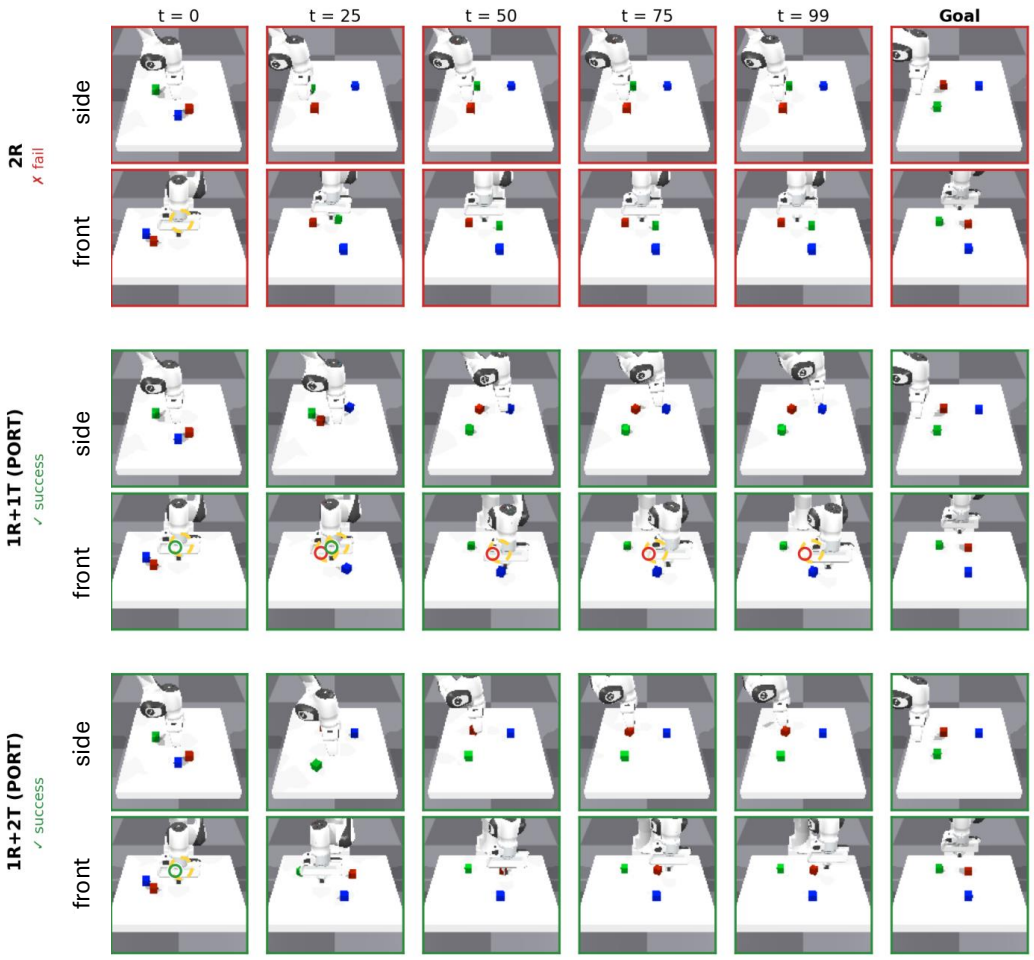


Figure 20: **PandaPush policy rollout with PORT-generated imagined views.** We show a representative manipulation rollout using transported latent particles as additional policy observations. The yellow dashed circle highlights a frame in which the cube is occluded.

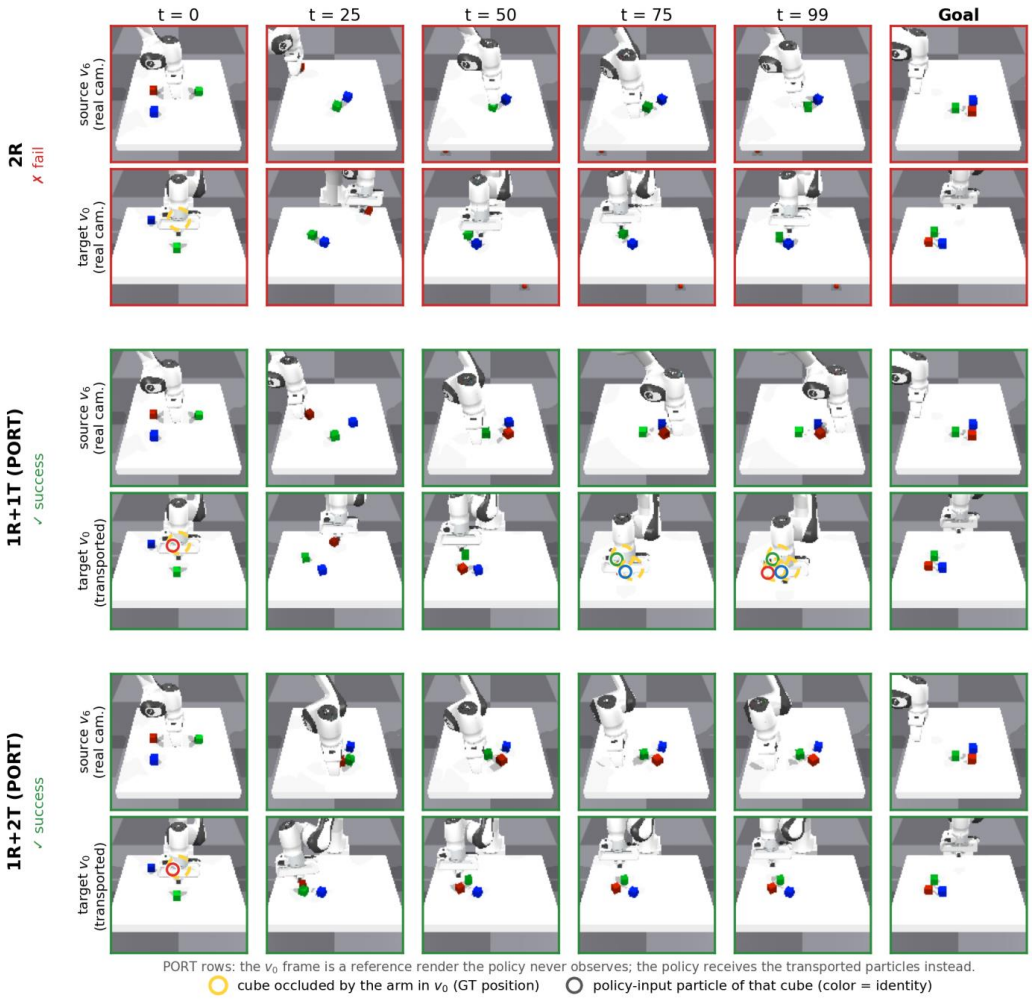


Figure 21: **PandaPush policy rollout with PORT-generated imagined views(2).** We show a representative manipulation rollout using transported latent particles as additional policy observations. The yellow dashed circle highlights a frame in which the cube is occluded.

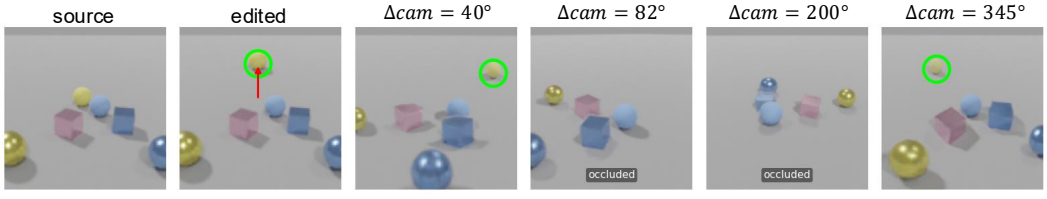


Figure 22: **Object-level scene editing across viewpoints.** We move the yellow ball upward in the source view and transport the edited particle to multiple target viewpoints. PORT consistently preserves the edited object state under viewpoint changes, including viewpoint-dependent occlusion and reappearance, even for an edited scene configuration not observed during training. This qualitatively indicates that PORT learns viewpoint-conditioned object transformations rather than memorizing fixed scene configurations.

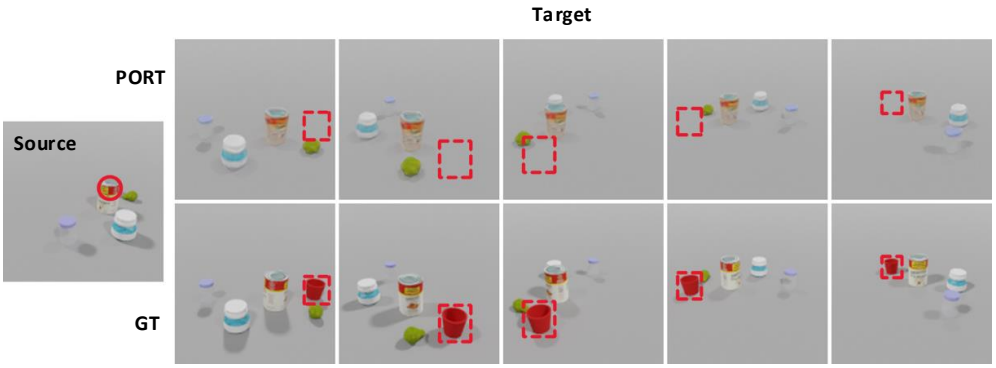


Figure 23: **Failure case: objects unobserved in the source view cannot be recovered.** PORT transports only the particles observed in the source image. As a result, objects that are completely hidden in the source view but become visible from target viewpoints are not generated. Red dashed boxes indicate the missing object in the target views.