Supplementary Material

Contents

A Implementation Details

In this section, we introduce the implementation details of ROSE. To be specific, we present the details of the Real2Sim transfer of human videos, Sim2Real policy training, and real-world setup in Appendix A.1, Appendix A.2, and Appendix A.3, respectively.

A.1 Real-to-Sim Transfer

We use the videos with a resolution of 512×288 containing 60$\sim$300 frames in 30 FPS (spanning 2s to 10s). We first reconstruct the 3D point clouds by running MegaSaM [31], detailed as follows.

Point Cloud Reconstruction For every frame, we follow MegaSaM [31] to get the affine-invariant monocular disparity map with Depth-Anything V2 [68], a camera pose in the world coordinate, and the focal length estimation obtained with UniDepth V2 [42]. With this information, we align the disparity to the metric scale, which is converted to the pixel-aligned 3D point clouds coordinated in the world frame. Afterwards, we use GeoCalib [58] on the first frame to estimate the scene's gravity direction. We then rotate the camera rig so that the estimated gravity is to the negative of the z-axis in a right-hand coordinate system, resulting in gravity-aligned point clouds in the world frame. To further avoid residual artifacts, we apply edge dilation to remove colors at boundaries and depth-gradient pruning to remove point clouds with corresponding gravity exceeding a certain threshold of 0.8 for reducing depth discontinuities.

For every image, we use Ground-SAM-2 [48] with DINO-X-Track[49] to obtain binary masks of objects in the scene, used for object removal and reconstruction. For the scene point cloud, we fuse the point clouds by random sampling over the video to leverage complementary information to reduce the inaccurate point clouds caused by occlusions and partial scenes after object removal. The random sampling is used for balancing every frame's contribution. On average, the point cloud reconstruction would take 3 minutes to process a 5-second casual video.

3D Mesh Reconstruction For the object mesh, we use TRELLIS [66] to reconstruct the 3D mesh, and we set the α channel's threshold to 0.2 to help reconstruct dark objects. For the scene mesh, we use a three-stage method. In the first stage, we run Neural Kernel Surface Reconstruction (NKSR) [20] for surface fitting. To close the small gaps created by mask removal while preserving high-frequency geometry, we set the detail level to 0.4 and use a single MISE iteration. Afterwards, to make the mesh orientable, two-manifold, and self-intersection-free for simulator usage, we wrap the resulting mesh with CGAL alpha-wrapping algorithm [44]. The α value is set to 400. Finally, each mesh vertex v of $\mathcal{M}_{\text{wrap}}$ inherits the RGB value $c(\cdot)$:

$$c(v) = \sum_{p \in \mathcal{N}_k(v)} w(p, v)\, c(p), \quad w(p, v) \propto \frac{1}{\|p - v\|_2}.$$

This is the inverse-distance-weighted average of its k-nearest neighbours $\mathcal{N}_k(v)$ in the point cloud. In this work, we use $k = 3$ with a maximum distance of 5cm.

Improved Foundation Pose Foundation pose [63] requires that the scale of the mesh and the scale, unprotected from the depth map, be the same in order to ensure accurate pose estimation. Therefore, we align the trellis mesh $\mathbf{M}^{\text{trellis}}$ with the scene mesh $\mathbf{M}^{\text{scene}}$ by the following transformation:

$$\mathbf{M}^{\text{trellis-align}} = \mathbf{G}_0 \mathcal{Q} \mathbf{s} \mathbf{M}^{\text{trellis}} f, \tag{6}$$

where s is the scale factor, $\mathbf{G}_0$ is the camera pose, $\mathcal{Q}$ is the object pose, and f is the focal length.

For pose tracking, we begin by using the scale s estimated in the first frame to adjust the scale of $\mathbf{M}^{\text{trellis}}$. We then follow the approach outlined in FoundationPose [63]. The object pose $\mathcal{Q}_i$ is initialized using the previously estimated pose $\mathcal{Q}_{i-1}$, and the refinement network is applied to further refine $\mathcal{Q}_i$, yielding the final estimation of the object pose for the current frame.

A.2 Sim-to-Real Policy

Simulation Environment Setup We use RoboVerse [14] as our simulation platform to establish the data collection pipeline. Specifically, we adopt the IsaacGym branch for mesh loading, policy execution, and reinforcement learning, while leveraging the IsaacLab branch for high-fidelity rendering and vision-based policy training.

Simulation and control parameters follow the default settings provided by RoboVerse. For both objects and scenes, we set the friction coefficient to 0.5. To ensure accurate collision detection and reliable physics simulation, we apply convex decomposition to the scene geometry.

Robotic Data Collection based on Motion Planning We utilize GSNet [40] and cuRobo [53] as the core components of our motion planning pipeline, and conduct all simulations within the Isaac Gym environment. After reconstructing the scene (as described in Appendix A.1), we load the reconstructed environment and the target object mesh into the simulator. The object is represented using a high-resolution mesh, preserving geometric detail necessary for accurate grasp prediction and motion planning.

Using GSNet, we extract both the surface point cloud and a predicted grasp pose for the target object. These predictions are then used to initialize an inverse kinematics (IK) solver provided by cuRobo, which computes a feasible trajectory for the robot's end effector to reach the designated grasping configuration. During this process, we account for kinematic constraints, joint limits, and potential collisions in the environment.

Upon reaching the grasp pose, the robot executes a grasp based on a set of hand-crafted heuristics, which evaluate grasp stability using factors such as contact normals and finger placement. Once the grasp is completed, the robot follows a trajectory generated by a previously introduced motion prediction model, which guides the object to a specified goal position or task-specific location.

Robotic Data Collection based on Reinforcement Learning Our method adopts a two-stage policy for robotic manipulation. In the first stage, we employ a reinforcement learning (RL) approach to train a policy that guides the robot's end effector toward the object and performs a grasp once proximity is sufficiently close. To facilitate generalization across different grippers or robotic hands, we design a simple yet broadly applicable reward function. This reward encourages the end effector to reduce its distance to the target object and penalizes undesired motions, without relying on gripper-specific parameters, making it adaptable to a wide range of hardware configurations.

During deployment, we execute the learned grasping policy for a fixed horizon of 50 steps, under the assumption that a successful grasp is achieved by the end of this phase. In the second stage, we switch to a motion planning phase using cuRobo [53]. The robot follows a precomputed trajectory that guides the grasped object to its goal location or completes the assigned task. This two-stage setup decouples grasp acquisition from subsequent manipulation, allowing each component to be optimized independently while ensuring end-to-end effectiveness.

A.3 Real-World Experimental Setups

Franka Setting For our real-world experiments, we use a Franka Emika Panda robotic arm equipped with a Robotiq 2F-85 adaptive gripper. This setup provides a reliable and widely used platform for evaluating grasping and manipulation policies in physical environments. The robot is controlled via a high-level interface that integrates seamlessly with our planning and control pipeline.

To reconstruct the environment, we capture RGB-D data using both an iPhone 16 Pro and a DJI Osmo Pocket 3. These consumer-grade devices offer high-resolution color and depth sensing capabilities, allowing for efficient and accessible scene scanning.

B Real-to-sim Benchmark

B.1 benchmark evaluation metric details

Our proposed benchmark is based on three primary evaluations: scene reconstruction similarity, object reconstruction similarity, and object trajectory reconstruction.

Uniform Sampling. Uniform sampling extracts a point cloud from a mesh by taking N points drawn *i.i.d.* over the surface of the mesh $\mathbf{M}$. Unless stated otherwise, $N = 10{,}000$ in all our experiments.

Symmetric Chamfer Distance. Given two point clouds A and B, symmetric chamfer distance is defined as:

$$\mathrm{ChamferDist}(A,B) = \frac{1}{|A|}\sum_{a\in A}\min_{b\in B}\|a-b\|_2^2 \;+\; \frac{1}{|B|}\sum_{b\in B}\min_{a\in A}\|b-a\|_2^2.$$

Scene Reconstruction Similarity. To evaluate scene reconstruction similarity, we first construct a point cloud $\mathbf{P}^{\text{scene}}$ through uniform sampling the reconstructed mesh. We then align $\mathbf{P}^{\text{scene}}$ with each frame's ground-truth point cloud $\mathbf{P}_i^{\text{gt-scene}}(i=1,\dots,F)$ by optimizing an SE(3) transformation to yield $\hat{\mathbf{P}}^{\text{scene}}$. Finally, we compute the average chamfer distance across the aligned frame point cloud $\hat{\mathbf{P}}^{\text{scene}}$ and the ground-truth scene point clouds.

$$\mathbf{E}_{\text{scene}} = \frac{1}{F}\sum_{i=1}^{F}\text{ChamferDist}(\mathbf{P}_i^{\text{gt-scene}}, \hat{\mathbf{P}}^{\text{scene}})$$

Object Reconstruction Similarity. Similarly to scene reconstruction similarity, we construct point clouds $\mathbf{P}^{\text{gt-obj}}$ and $\mathbf{P}^{\text{obj}}$ by uniformly sampling the ground-truth and reconstructed object meshes respectively. We then align the point clouds by optimizing an SE(3) transformation to yield $\hat{\mathbf{P}}^{\text{obj}}$. Finally, we evaluate object reconstruction similarity as the chamfer distance between the aligned point clouds.

$$\mathbf{E}_{\text{obj}} = \text{ChamferDist}(\mathbf{P}^{\text{gt-obj}}, \hat{\mathbf{P}}^{\text{obj}})$$

Trajectory Reconstruction. We evaluate object trajectory reconstruction using three metrics: absolute pose error translation (APE$_{trans}$), relative pose error translation (RPE$_{trans}$) and relative pose error rotation (RPE$_{rot}$). We compute these between the ground truth trajectory $\mathbf{Q}^{\text{gt-obj}}$ and the reconstructed trajectory $\mathbf{Q}^{\text{obj}}$, after aligning the reconstructed trajectories scale and SE(3) transformations.

The absolute pose error (APE) measures the deviation between corresponding poses and is defined as:

$$e_{\text{ape}}(\mathbf{Q}^{\text{gt-obj}}, \mathbf{Q}^{\text{obj}}) = \frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{Q}_{xyz}^{\text{gt-obj}} - \hat{\mathbf{Q}}_{xyz}^{\text{obj}}\right\|_2$$

where $\mathbf{Q}_{xyz}$ denotes the translation component of pose $\mathbf{Q}_i$ and $\hat{\mathbf{Q}}$ denotes the aligned trajectory.

The relative pose error (RPE) measures the local consistency of motion between consecutive poses. For an interval $\Delta = 1$, we define the relative transformations as:

$$\mathbf{T}_i^{\text{gt-rel}} = (\mathbf{Q}_i^{\text{gt-obj}})^{-1}\mathbf{Q}_{i+\Delta}^{\text{gt-obj}}$$
$$\mathbf{T}_i^{\text{obj-rel}} = (\hat{\mathbf{Q}}_i^{\text{obj}})^{-1}\hat{\mathbf{Q}}_{i+\Delta}^{\text{obj}}$$

The translation component of the RPE at time i is given by:

$$e_{\text{rpe,trans}}(i) = \left\|\text{trans}\left((\mathbf{T}_i^{\text{gt-rel}})^{-1}\mathbf{T}_i^{\text{obj-rel}}\right)\right\|_2$$

where $\text{trans}(\cdot)$ extracts the translation part of a transformation.

The rotation component of the RPE is given by:

$$e_{\text{rpe,rot}}(i) = \arccos\left(\frac{\text{trace}\left(\text{rot}\left((\mathbf{T}_i^{\text{gt-rel}})^{-1}\mathbf{T}_i^{\text{obj-rel}}\right)\right) - 1}{2}\right)$$

where $\text{rot}(\cdot)$ extracts the rotation matrix component of a transformation.

B.2 Data Details

We select 5 representative tasks from CALVIN [36] implemented in RoboVerse [14], covering the basic manipulation tasks on rigid objects: lift, push, rotate, "pick and place", and unstack blocks. These tasks are performed on top of a delicate desk, allowing for evaluating the proposed pipeline by reconstructing both the scene (desk) and the objects in interest (blocks).

B.3 Qualitative Results

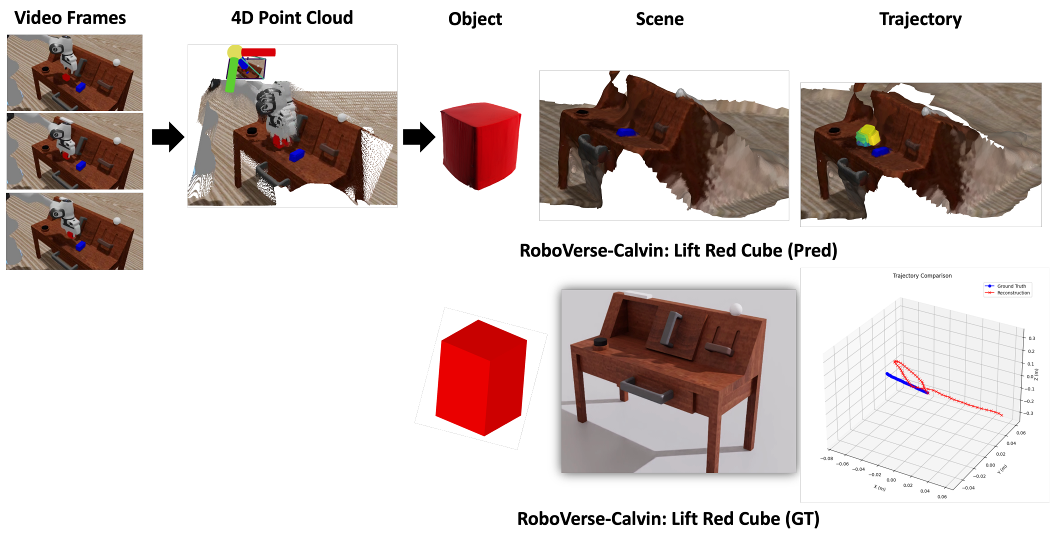

Figure 4: **Qualitative examples of our real-to-sim benchmark.**