GMT: Goal-Conditioned Multimodal Transformer for 6-DOF Object Trajectory Synthesis in 3D Scenes

Supplementary Material

In this supplementary material, we provide detailed descriptions of our implementation, dataset processing procedures, and additional experimental results. The document is organized as follows. We first outline the implementation details in Sec. 6, then the overall dataset processing workflow in Sec. 7, followed by the dataset-specific pre-processing steps applied to the Aria Digital Twin (ADT) [35] and HD-EPIC [38] datasets. Next, in Sec. 8, we present several failure cases arising from complex motion patterns, along with additional qualitative results.

6. Implementation Details

Our model is implemented in PyTorch [36] and is trained on a single NVIDIA A6000 GPU. The training process utilizes the AdamW optimizer with a learning rate of 1×10^{-4}, weight decay of 5×10^{-4}, and exponential learning rate decay with a factor of 0.99.

For data usage, 90% of the available sequences are used for training, with the remaining 10% split equally between validation and testing. Each training sample consists of a full scene point cloud, semantic fixture bounding boxes and labels, and a 6-DOF object trajectory. Trajectories are uniformly resampled to 200 frames, and the first 30% of the sequence is used as the input history. layers dimension

The multimodal Transformer encoder consists of 6 layers with 8 attention heads per layer. The input trajectory is embedded in a 128-dimensional feature space; the global scene point cloud feature has 128 dimensions, and per-point features have 64 dimensions. Semantic fixture bounding boxes are projected to 128 dimensions, while CLIP [42] embeddings for object categories and semantic labels are linearly projected to the same dimension. The goal state feature is also projected to 128 dimensions to ensure compatibility in the fusion space. The latent array within the Transformer is 256-dimensional.

During training, the loss function combines translation, orientation, reconstruction, and destination losses, with weights λ_{trans}, λ_{ori}, λ_{rec}, and λ_{dest} set to 1.0.

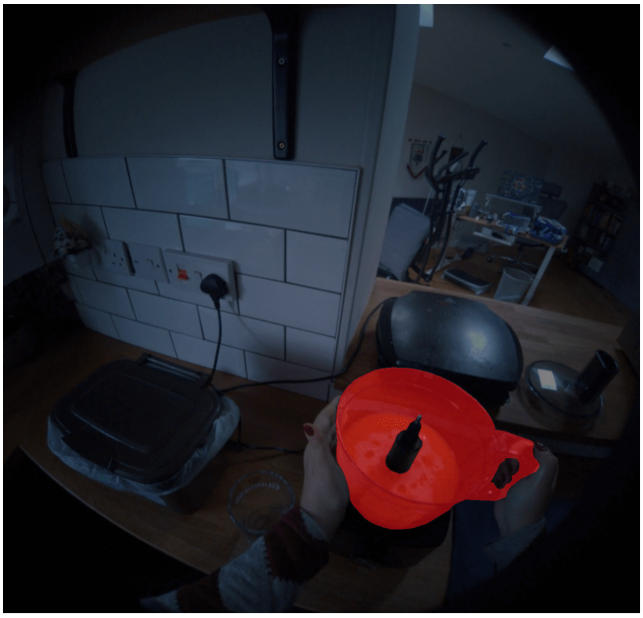
7. Dataset Processing

General Processing. We begin by processing the trajectory data. Since trajectories in ADT are overly dense, we downsample them by retaining one point every five frames to improve computational efficiency while preserving essential motion cues. To support training with multiple batches, we fix the predicted trajectory length to 200. Trajectories shorter than 200 frames are padded, whereas longer ones are truncated. An attention mask ensures that only valid trajectory points contribute to the training objective. We further apply two filtering rules to discard unrealistic motion: an object is considered to be moving only when its velocity exceeds 0.05 m/s, and segments are labeled static when an object remains still for more than three consecutive frames. These criteria help preserve only meaningful trajectories for both training and evaluation.

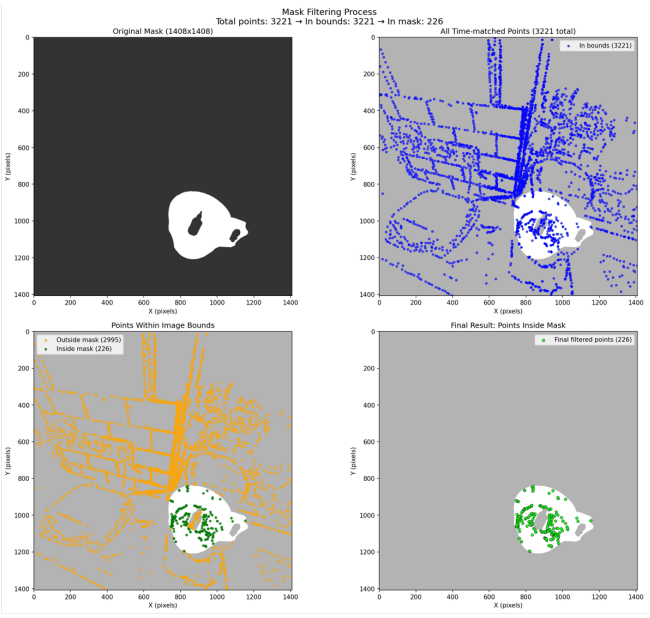
Next, instead of using the full scene point cloud, we extract only the local region around each trajectory to reduce computational overhead. For every trajectory point, we define a spherical neighborhood with a radius of 1 m and collect all points within this region, ensuring that only relevant scene geometry is retained. Similarly, fixture information is extracted only from regions near the trajectory, as distant geometry has limited influence on downstream tasks. These extracted fixture features are then incorporated into the model input. We now describe dataset-specific processing procedures for ADT and HD-EPIC.

Aria Digital Twin Dataset. Since ADT does not include action-level semantic annotations, we use only object categories as descriptive information. Certain trajectories are removed because they do not meet the requirements of our robotic manipulation setting. For example, trajectories in which an object is held or moved within an extremely limited spatial region for extended periods are excluded, as they lack meaningful interaction patterns and do not reflect practical robotic behaviors.

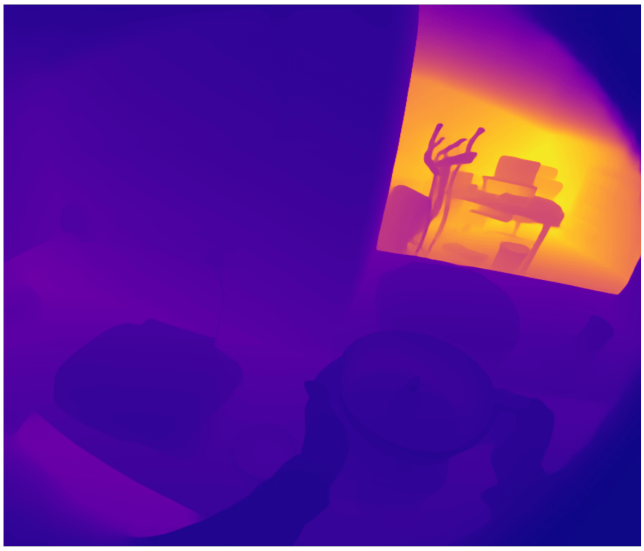
HD-EPIC Dataset. In contrast to ADT, the HD-EPIC dataset does not provide bounding boxes or semantic labels for scene objects. Large static structures such as countertops and drawers are manually reconstructed in Blender and aligned with the scene point cloud to serve as static bounding boxes. For small manipulable objects such as coffee machines and knives, HD-EPIC provides start/end timestamps of object motion, 2D masks, and 3D object centers. Using this information, we first align the timestamps with SLAM data and obtain sparse 2D–3D correspondences using MPS data collected with Aria glasses. We then estimate monocular depth using UniK3D [39], perform linear depth alignment with the correspondences to recover the true scale, and reconstruct 3D object bounding boxes. These *dynamic bounding boxes*, together with static ones, are used for model training. Fig. 5 illustrates the full processing pipeline.



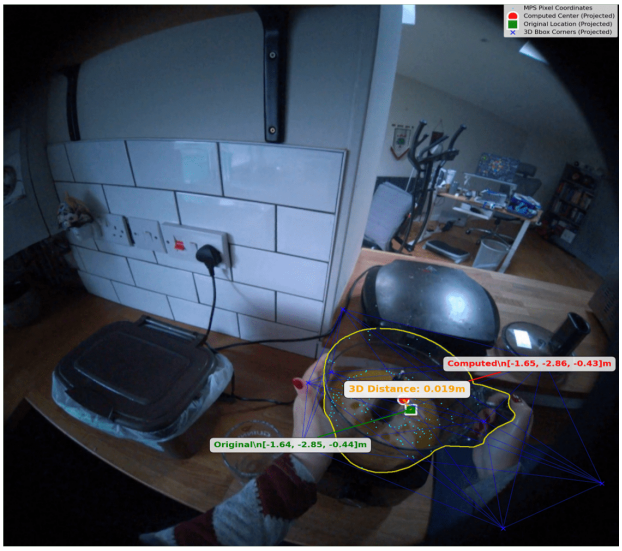
(a) Given 2D annotation



(b) 2D-3D correspondence filtering



(c) metrics depth estimation by Unik3D



(d) 3D bbox caculation

Figure 5. **3D bounding box reconstruction in the HD-EPIC dataset.** (a): input RGB frame with object mask. (b): mask filtering and sparse 2D-3D correspondences from SLAM and MPS data. (c): monocular depth estimation from UniK3D. (d): final 3D bounding box recovered after depth alignment and scaling. This pipeline enables accurate localization of small objects (e.g., bowls, cups) in cluttered scenes.

Since the dataset provides only pick-up and drop-off annotations for each object, we generate dense object trajectories using hand-tracking. We compute the transformations of the manipulating hand over time and apply these transformations to the provided initial object position, producing a complete trajectory for each object.

The hand trajectory extraction pipeline integrates multiple perception systems to process egocentric video. The pipeline begins with a bootstrap stage that establishes hand–object correspondence by computing the 3D Euclidean distance between detected hand positions and the annotated object center in world coordinates to identify the primary

manipulating hand. Subsequent hand positions are obtained using Project Aria's MPS [13]. For frames in which both hands are confidently detected, we leverage Hands23 [9] to disambiguate which hand is physically interacting with the object. Hands23 infers binary contact states for each hand, enabling reliable determination of the manipulating hand even when both hands appear in view. When Hands23 outputs are ambiguous (e.g., both hands detected in contact), the system maintains temporal consistency by defaulting to the initially selected primary hand. Temporal coherence is further enforced via a sliding-window filter (window size = 3), which suppresses spurious frame-to-frame switching.

For orientation estimation, we construct a 6D rotation representation [64] derived from the geometric structure of the hand. Specifically, we use Singular Value Decomposition (SVD) to compute the hand coordinate frame, where the primary axis aligns with the wrist-to-palm vector and the palm normal defines the facing direction, both obtained from MPS. This 6D parameterization ensures continuity across the rotation manifold and avoids the singularities present in Euler angles and quaternions.

We demonstrate the effectiveness of the reconstructed object trajectories in Fig. 6, where the recovered 3D object location is projected onto the input RGB frames for visual verification.

8. Additional Qualitative Results

To complement the results presented in the main paper, we provide additional qualitative examples for both the ADT and HD-EPIC datasets. Figures 7 and 8 illustrate representative failure cases and their likely causes, while Figs. 9 and 10 present additional successful predictions.

Figure 6. **Object Position computation using hand-tracking** We demonstrate the object positions. depicted as Yellow along with the orientation and the hand that is interacting with the object at the particular frame sampled at intervals during the entire period of the moving object.

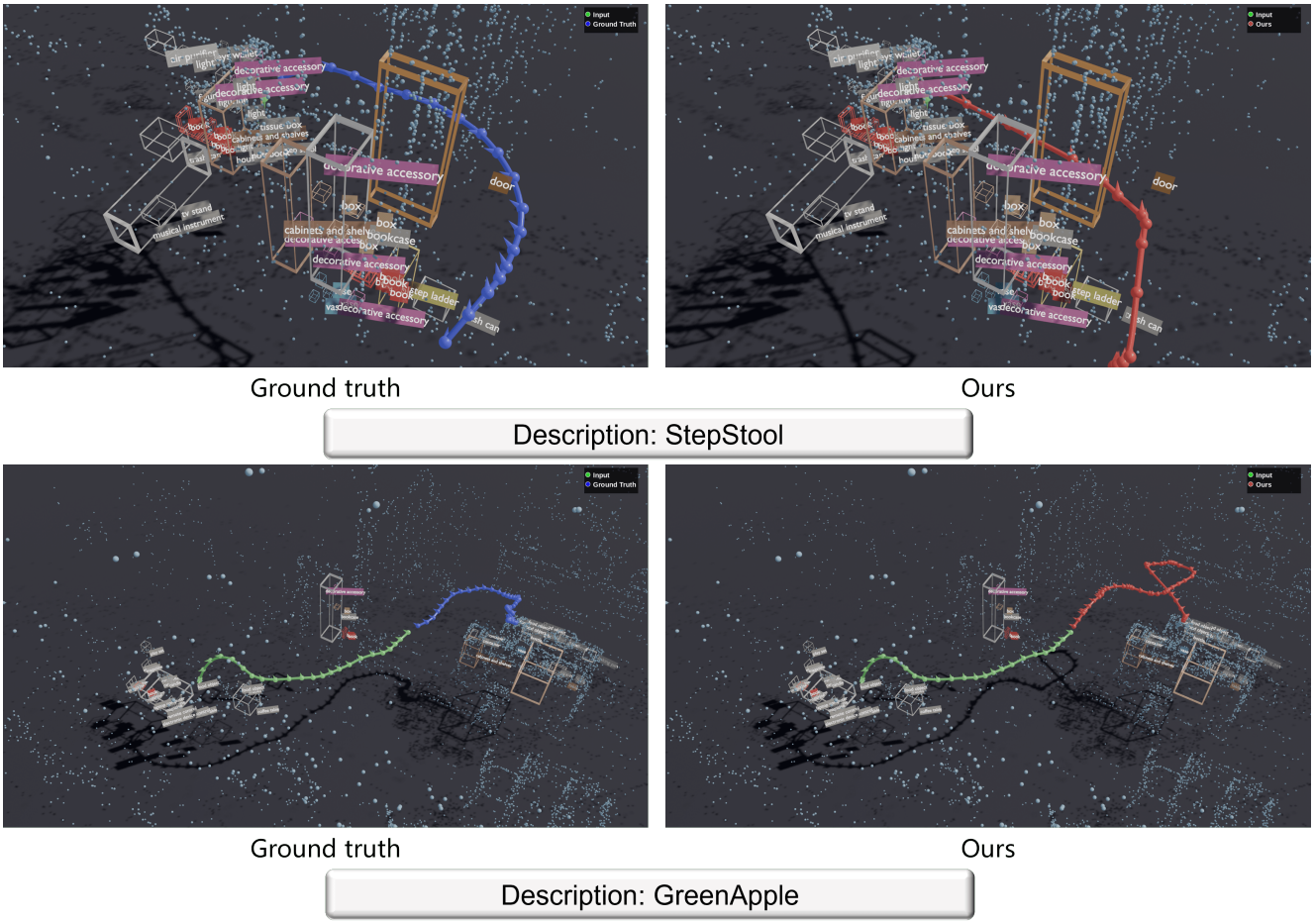


Figure 7. **Failure Cases in the ADT dataset.** We can observe that sometimes, despite being goal-conditioned, the generated trajectory may be longer than the ground truth trajectory and may overshoot the destination.

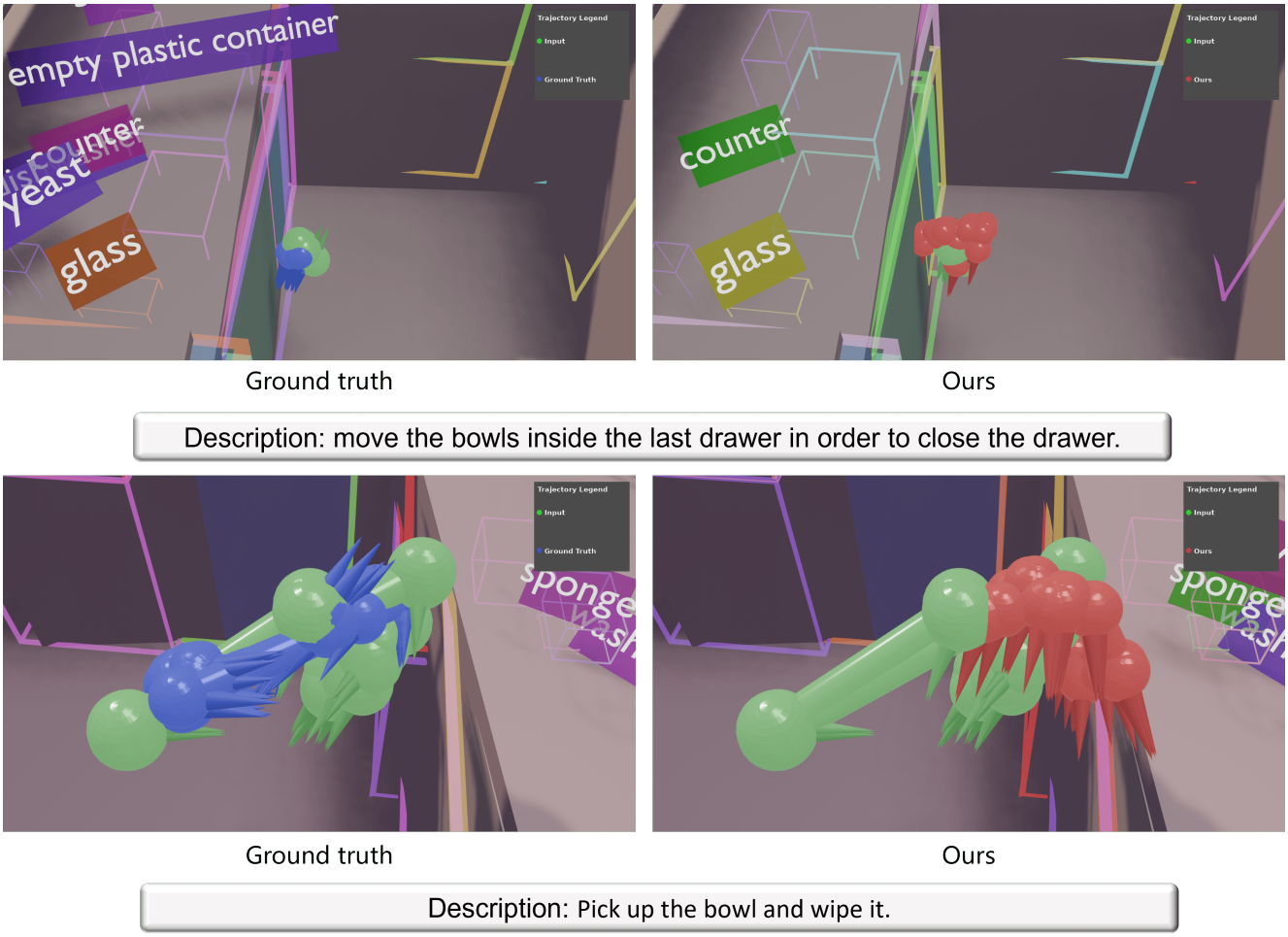


Figure 8. **Failure cases in the HD-EPIC dataset.** Our method suffers from adding redundant motion for inputs don't have significant change in their positions. Such motion os observed for small object trajectories that are often interacted with for a very small duration.

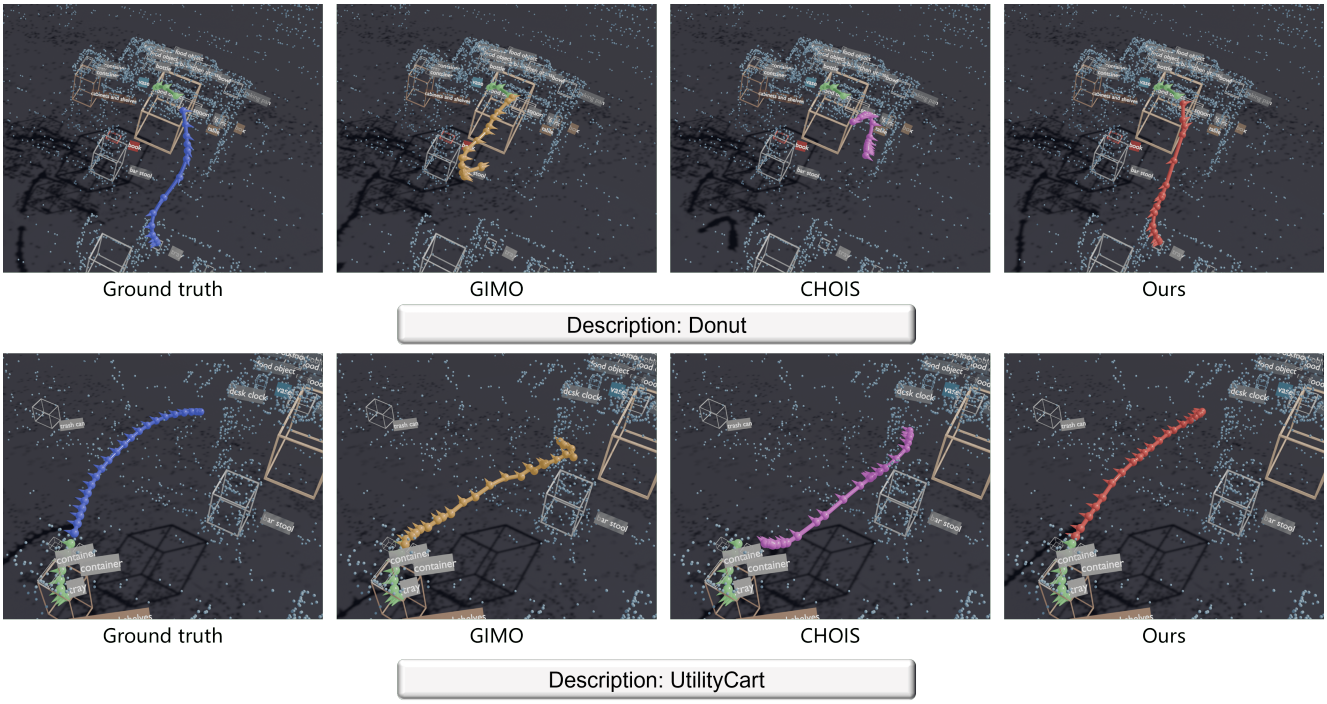
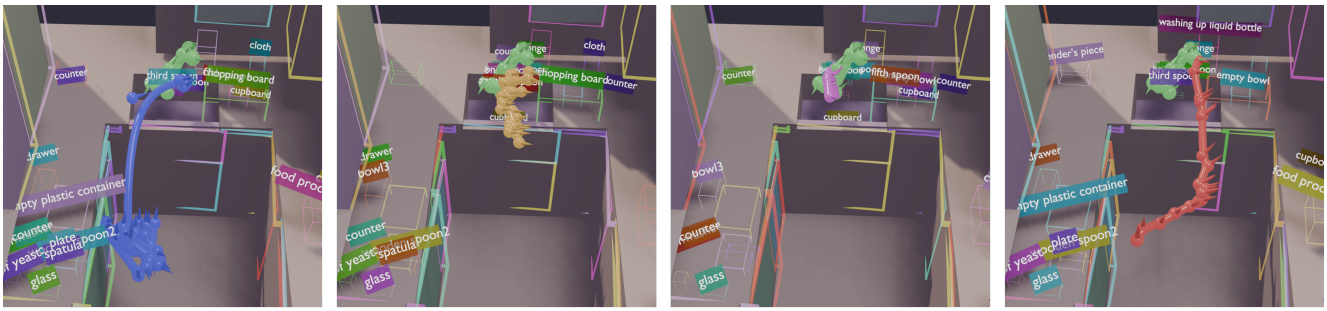
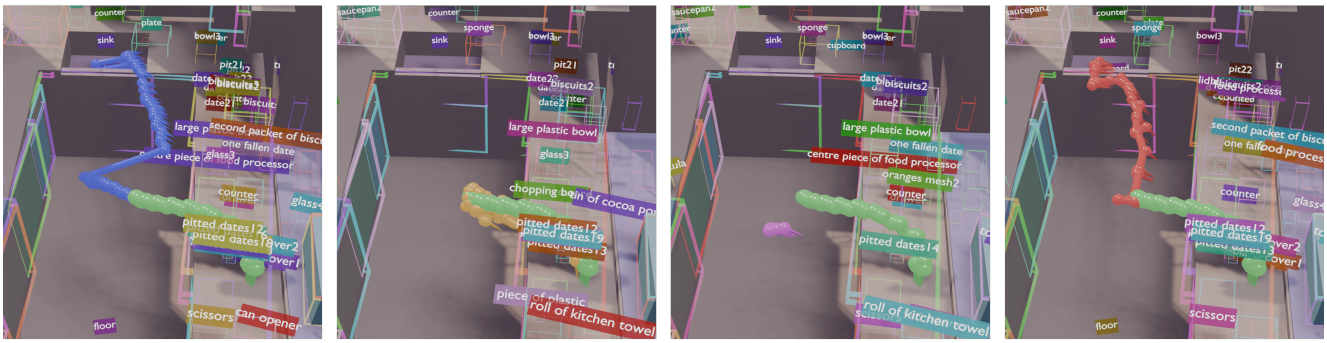


Figure 9. **Qualitative results in the ADT dataset.**



Description: pick up another spoon that's soaking in the sink. Remove what's stuck on the spoon by brushing the finger over it while soaking it in the water. Open the top cutlery drawer. Put the knife into a slot in the cutlery drawer.



Description: pick up an empty bowl from the countertop using the left hand pick up an empty plate from the countertop using the right hand this action is mostly off screen put the plate inside the sink using the right hand put the bowl inside the sink using the left hand this action is mostly off screen

Figure 10. **Qualitative results in the HD-EPIC dataset.**