# EgoFlow: Gradient-Guided Flow Matching for Egocentric 6DoF Object Motion Generation

## Supplementary Material

In this supplementary material, we provide additional details and analyses to complement the main paper. The supplementary is organized as follows:

First, in Sec. 6, we describe the dataset preprocessing steps to convert raw data into a consistent format for training and evaluation. In Sec. 7, we provide architectural hyperparameters of our framework, including layer configurations, hidden dimensions, and attention head counts. Next, we further analyze the effect of optimization steps during inference in Sec. 8, comparison of different generation paradigms in Sec. 9, and the impact of goal conditioning in Sec. 10.

Finally, a proof-of-concept application of our framework for robot manipulation is presented in Sec. 11.

## 6. Dataset Preprocessing

In this section, we provide more details on how we preprocess the datasets used in our experiments into a consistent format.

**HD-EPIC [25].** The HD-EPIC dataset lacks explicit annotations such as object bounding boxes or semantic labels. To compensate for this, large static elements (e.g., tables, drawers) are manually modeled in Blender and aligned with the scene's point cloud to define fixed bounding boxes. For smaller, movable items like coffee makers and knives, the dataset includes temporal segments marking object motion, along with 2D segmentation masks and estimated 3D centers. Using these cues, we synchronize timestamps with SLAM outputs and extract sparse 2D-3D correspondences via MPS data captured by Aria glasses. Monocular depth maps are then inferred with DepthAnythingv2 [49], and depth values are linearly scaled using the correspondences to recover metric scale. Finally, we reconstruct each object's 3D bounding box.

**HD-EPIC Validation on ADT [23].** Since the HD-EPIC dataset provides only sparse annotations of object positions, we developed an algorithm to generate dense annotations of object motion, as described in Sec. 4.2. To validate the effectiveness of this algorithm, we applied it to the Aria Digital Twin dataset, which offers dense ground-truth trajectories for dynamic objects across their motion sequences. Table 5 summarizes the evaluation results on this synthetic dataset. As shown, for sufficiently long trajectories, the 3D Euclidean error between the ground-truth and the predicted object positions remains minimal. Nevertheless, we exclude this dataset from our main experiments, as its limited size and lack of real-world diversity prevent it from providing meaningful training data for generative models.

| #Traj. | ADE Mean (cm) | ADE Median (cm) | Traj. Length Mean (m) | Traj. Length Median (m) |
|---|---|---|---|---|
| 505 | 20.35 | 15.76 | 3.90 | 3.06 |

Table 5. **ADT Statistics.** We study the effectiveness of our object position estimation approach and verify it on the ADT dataset.

Additionally, we project the computed 3D object locations during their motion onto egocentric video frames to verify the algorithm's effectiveness. We demonstrate this in Fig. 5 and Fig. 6. We also provide supporting videos of these object motions for a more holistic evaluation.

**Ego-Exo4D [8] & HOT3D [1].** For the Ego-Exo4D dataset, egocentric video streams were divided into short clips of approximately four seconds centered on annotated timestamps. Segments involving explicit object-hand interactions were automatically selected. An open-vocabulary segmentation model (Grounded-SAM [30]) was employed to identify the manipulated object in the initial frame, and the object was subsequently tracked throughout the clip using SpaTracker [44]. Depth maps were estimated for all frames using the DepthAnythingv2 [49], and RGB-D images were converted into point clouds.

For the HOT3D dataset, ground-truth 6DoF object trajectories captured with OptiTrack infrared cameras were utilized directly. Since temporal boundaries and textual descriptions were not provided in the original dataset, videos were divided into four second clips and action intervals were automatically localized using GPT-4o-assisted temporal annotation. Depth maps were further estimated to align all trajectories with the same coordinate convention as the Ego-Exo4D data.

## 7. Architecture Details

We provide more details on the architecture hyperparameters of our proposed model in Tab. 6.

The model uses 768-dimensional hidden representations across all stages with 0.1 dropout. Stage 1 contains 3 bidirectional Mamba layers with SSM state dimension 256. Stage 2 contains 6 Transformer layers with 12 attention


Figure 5. We project our object position calculated by our object position estimation algorithm using the hand poses of MPS as described in Sec. 4.2 onto the egocentric video frames to show the correctness and accuracy of our approach.

heads per layer. Stage 3 contains 3 bidirectional Mamba layers with SSM state dimension 256.

The conditioning pipeline consists of: PointNet++ producing 512-dimensional scene features, motion encoder producing 256 dimensions, bounding box encoder with 4-head attention producing 256 dimensions, CLIP ViT-B/32 producing 256-dimensional category embeddings, semantic bbox encoder producing 256 dimensions, goal pose encoder producing 256 dimensions, and a Perceiver module with 256 latent dimensions, 4 attention heads, and 4 layers. The concatenated conditioning vector is 1792 dimensions.

Figure 6. We plot our object position calculation algorithm on the ADT dataset. Since ADT has rich annotations, it works as an ideal demonstration of the effectiveness of our algorithm to generate dense object motions on the HD-EPIC dataset.

## 8. Analysis of Optimization Steps

We analyze the trade-off between guidance optimization and inference efficiency (Tab. 7 and Fig. 7). Test-time optimization primarily affects collision avoidance: collision rate decreases from 11.9% (0 steps) to 2.5% (50 steps), while trajectory errors remain largely stable (ADE: 0.273-0.279m, FDE: 0.102-0.119m). Noticeably, ADE gets slightly worse in this case, as the model is often encouraged to slightly deviate from its path to produce more collision-free trajectories, and hence it goes through more unoccupied space to avoid other objects, while noticeably reaching its destination. Inference time scales approximately linearly from 0.254s to 0.480s per trajectory. We use 50 steps for our

| Component | Value |
|---|---|
| *Core Architecture* | |
| Hidden dimension | 768 |
| Dropout | 0.1 |
| *Stage 1: Bidirectional Mamba* | |
| Number of layers | 3 |
| SSM state dimension | 256 |
| *Stage 2: Transformer* | |
| Number of layers | 6 |
| Attention heads | 12 |
| *Stage 3: Bidirectional Mamba* | |
| Number of layers | 3 |
| SSM state dimension | 256 |
| *Conditioning Encoders* | |
| Scene encoder (PointNet++) | 512 |
| Motion encoder | 256 |
| Bounding box encoder | 256 |
| BBox attention heads | 4 |
| Category encoder (CLIP ViT-B/32) | 256 |
| Semantic bbox encoder | 256 |
| Goal pose encoder | 256 |
| Motion-bbox Perceiver latent | 256 |
| Perceiver attention heads | 4 |
| Perceiver attention layers | 4 |
| Total conditioning dimension | 1792 |

Table 6. Model architecture hyperparameters.

| Steps | ADE↓ | FDE↓ | Geodesic↓ | Coll.↓ | Time(s)↓ |
|---|---|---|---|---|---|
| 1 | **0.273** | 0.119 | 1.151 | 11.7% | 0.259 |
| 5 | **0.273** | 0.119 | 1.151 | 11.0% | 0.277 |
| 10 | **0.273** | 0.119 | 1.151 | 10.0% | 0.301 |
| 20 | **0.273** | 0.119 | 1.151 | 8.2% | 0.345 |
| 30 | **0.273** | 0.120 | 1.151 | 6.7% | 0.391 |
| 40 | **0.273** | 0.120 | 1.151 | 4.6% | 0.436 |
| 50 | 0.279 | **0.102** | **1.141** | **2.5%** | 0.480 |

Table 7. Ablation study of optimization steps on HD-EPIC. Time denotes inference time per trajectory in seconds.

experiments to ensure collision-free generation.

## 9. Analysis of Generation Paradigms

We compare our flow matching approach against diffusion-based generation using the same architecture and training data (Tab. 8). Flow matching achieves substantially better trajectory quality (ADE: 0.279m vs 0.658m, FDE: 0.102m vs 0.549m) and is nearly an order of magnitude faster. When applying guidance to diffusion, we observe a fundamental trade-off: collision rate drops to nearly 1% but trajectory errors worsen (ADE: 0.692m, FDE: 0.660m) and

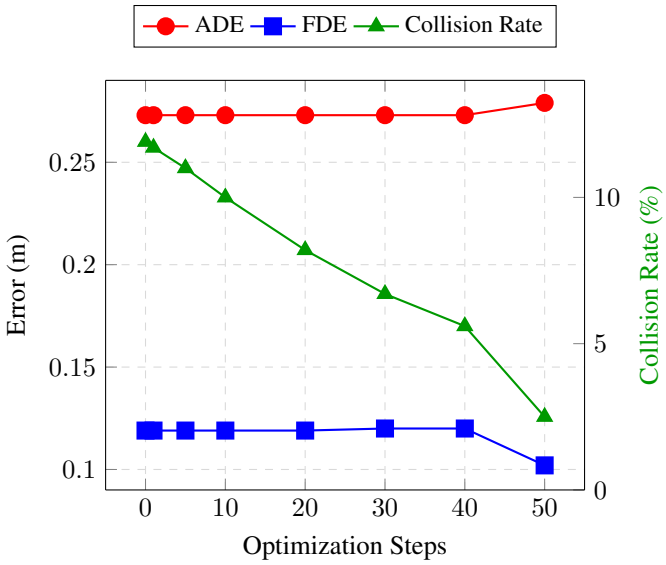



Figure 7. Effect of optimization steps on motion quality metrics.

inference time doubles to 5.561s. This degradation occurs because guidance is applied at every denoising step, causing the model to overemphasize collision avoidance and reroute trajectories into empty regions rather than toward task-relevant endpoints. In contrast, flow matching's deterministic straight-path interpolation allows guidance to refine constraints without derailing the overall motion plan. These results motivate our choice of flow matching for physically plausible object motion generation where both accuracy and constraint satisfaction are critical. Fig. 8 further illustrates the qualitative differences, showing that flow matching produces trajectories more faithful to the ground truth and conditioning.

## 10. Analysis of Goal Conditioning

Here, we show the qualitative comparison in Fig. 9 for the effect of having the end goal conditioned to the model as an input. Additionally, we show the **multiple possible paths** that the object could take based on the history and the conditioning. Supporting the quantitative results in Sec. 4.4, we show that the ADE and FDE are worsened as the object fails to reach its end goal.

## 11. Application

As a proof-of-concept, we further explored the potential of our framework in real-world applications. Given predicted object trajectories, we employed inverse kinematics (IK) to generate robot manipulator motions that follow the planned object paths. We utilized the Mink solver [52] on Mujuco [39] to compute joint configurations for Tidybot [42] with Leap hand [32], ensuring that the end-effector main-

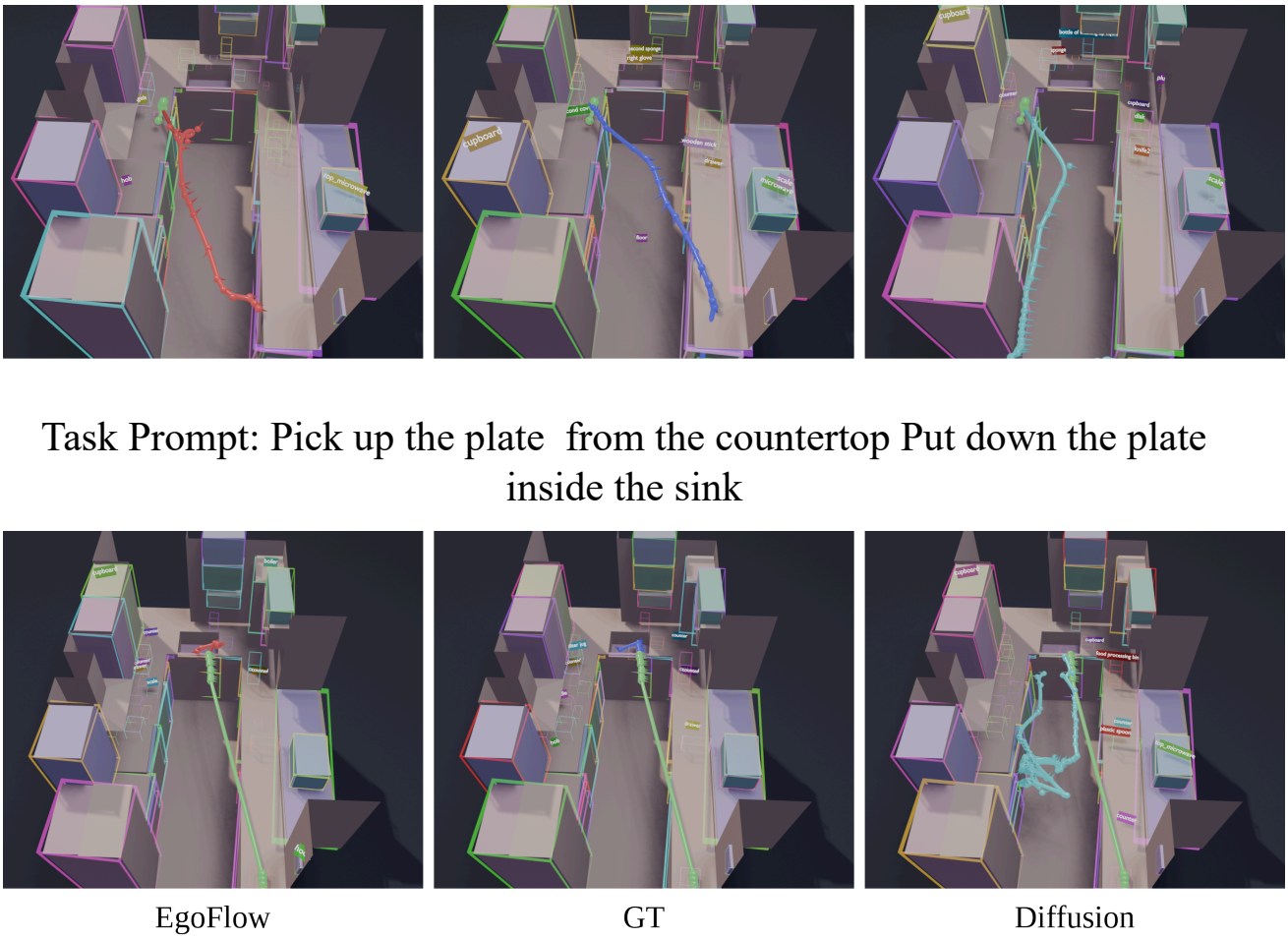



Figure 8. **Flow Matching vs Diffusion.** We show a qualitative comparison of using flow matching vs a diffusion paradigm while using the same model as the backbone. We observe that flow matching is more faithful to the ground truth and conditionings due to its simpler objective and more straight-line paths from noise to data distribution over the complex denoising steps of its diffusion counterpart. The green part of the trajectory shows the history, while Red, Blue and Cyan depict the flow matching, ground truth, and the diffusion-based predictions respectively.

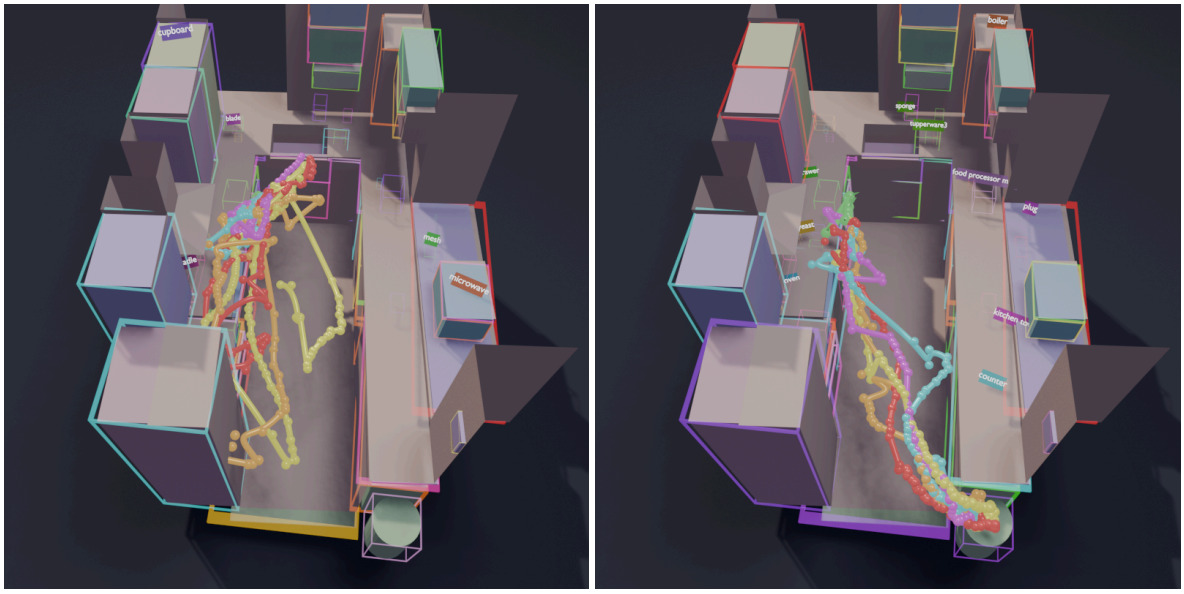
| Model | ADE↓ | FDE↓ | Geodesic↓ | Coll.↓ | Time(s)↓ |
|---|---|---|---|---|---|
| Diffusion w/o Guidance | 0.658 | 0.549 | 1.437 | 25.0% | 2.745 |
| Diffusion | 0.692 | 0.660 | 1.483 | **0.96%** | 5.561 |
| Ours (Flow Matching) | **0.279** | **0.102** | **1.141** | 2.5% | **0.480** |

Table 8. **Diffusion vs Flow Matching.** Comparison of diffusion-based and flow matching approaches on HD-EPIC. Time denotes inference time per trajectory in seconds.

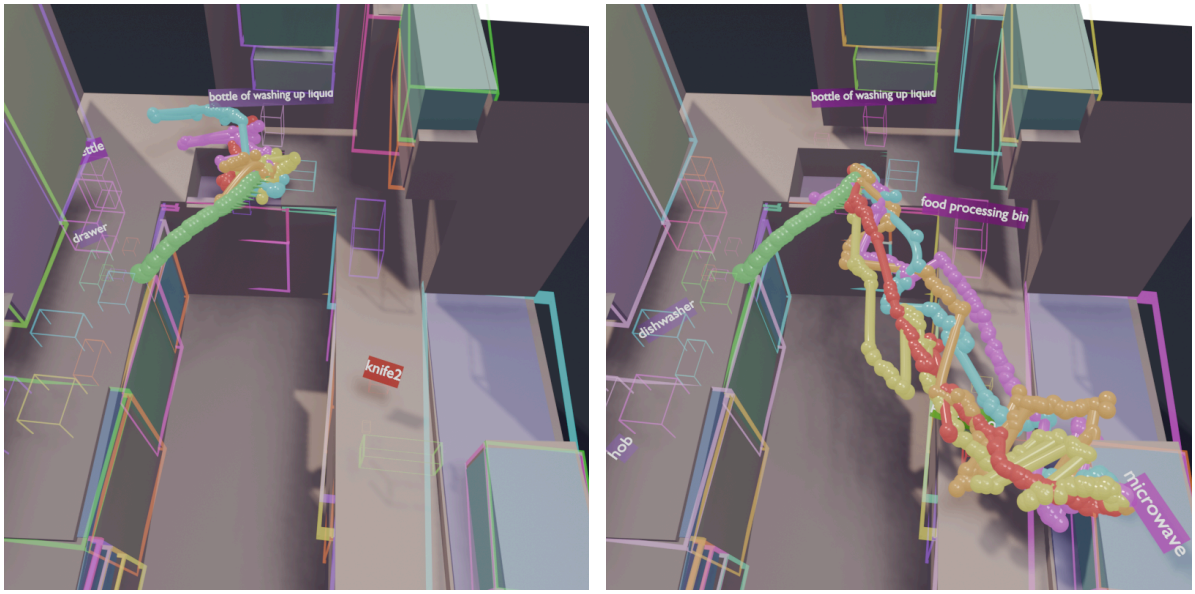
tained a fixed grasp pose relative to the object throughout the manipulation. For visualization, we rendered the object trajectories with a mobile robot in Mujoco, as shown in Fig. 10.

It is worth noting that the current demonstrations still have several limitations. First, we assume that the grasp

Task Prompt: pick up the kitchen towel that is on the countertop using the left hand. Throw the paper towel into the trash bin.



Task Prompt: Pick up the scale from the countertop using both hands. Put down the scale on top of the microwave on the shelf using the right hand.




Without Goal Conditioning

EgoFlow

Figure 9. **Analysis of Goal Conditioning** We show a qualitative comparison of the effect of having goal conditioning as input to the model. The green part of the trajectory shows the history, while the other colors such as red, orange, cyan, yellow and purple show the **multiple possible paths** the object motion could take based on the history to the end goal.

pose is known and remains fixed during manipulation, which does not always hold in real-world scenarios. Shifts in the grasp pose can lead to infeasibility in IK solutions. Second, dynamic feasibility is not explicitly enforced for high DoF robots such as humanoids or quadrupeds, which may cause balance loss or physically implausible motions

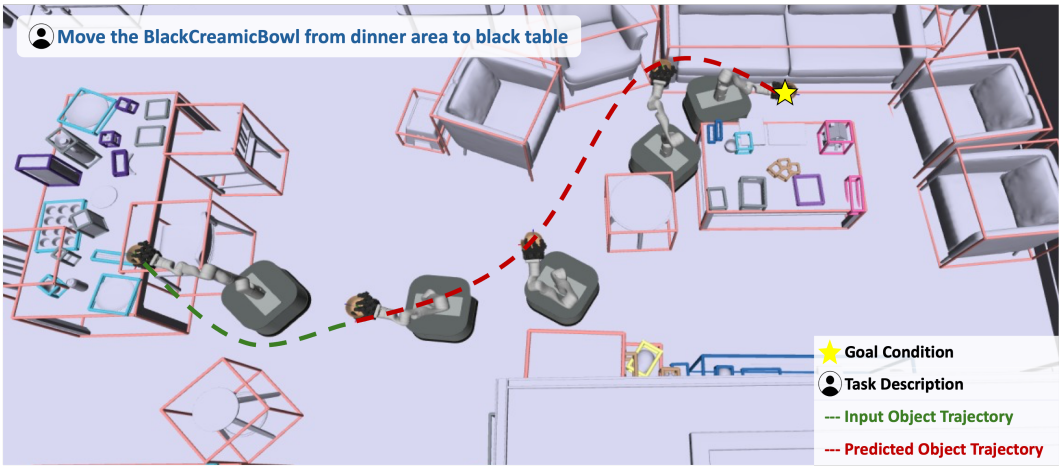



Figure 10. Visualization of robot mobile manipulation using object trajectories. The robot follows the object paths while maintaining a fixed grasp pose.

during execution. Third, our current framework focuses on object-level collision avoidance and does not incorporate base motion planning, which is essential for achieving coordinated whole-body control. Overall, successful mobile manipulation requires the integration of grasp pose estimation, base motion planning, and dynamic feasibility constraints, which we leave as future work.