Supplement for "Uncertainty-aware State Space Transformer for Egocentric 3D Hand Trajectory Forecasting"

Wentao Bao [1*], Lele Chen[2], Libing Zeng[3], Zhong Li[2], Yi Xu[2], Junsong Yuan[4], Yu Kong[1]
[1]Michigan State University, [2]OPPO US Research Center,
[3]Texas A&M University, [4]University at Buffalo
{baowenta,yukong}@msu.edu, libingzeng@tamu.edu,
{lele.chen,zhong.li,yi.xu}@oppo.com, jsyuan@buffalo.edu

In this document, we provide more details of the data collection and annotation, model implementation, evaluation results, and visualizations.

A. Details of the Datasets

A.1. Annotation Workflow

Following the similar pipeline in the EgoPAT3D [8], we propose to obtain the 3D hand trajectory annotations based on egocentric RGB-D recordings. In the following paragraphs, we elaborate on each processing step based on RGB-D data from the EgoPAT3D [8] and H2O [6].

Clip Division The EgoPAT3D dataset consists of RGB-D data of hand-object manipulation in 14 indoor scenes. We leverage the provided manual clip divisions and the hand landmarks to obtain more accurate trajectory divisions. Specifically, let (s_m, e_m) denote the start and end of a manually annotated trajectory, and $\{t_s, \dots, t_e\}$ denote the indices of detected 3D hand landmarks, our trajectory start and end are determined by $\max(s_m, t_s)$ and $\min(e_m, t_e)$, respectively. This technique could mitigate the ambiguity of trajectory start and end. Then, we use them to obtain the RGB video clips from the raw recordings. The H2O dataset contains 184 long videos and each video is annotated with 3D poses of the left and right hand as well as the binary validity flag. The trajectory start and end are determined by the validity flag.

2D Trajectory For each clip, we found the hand trajectory is not stable if only using the centers of frame-wise hand landmarks as trajectory points. Therefore, for the EgoPAT3D dataset, we propose to leverage the optical flow model RAFT [11] to warp the hand landmark center as the 2D hand trajectory. Specifically, we apply the RAFT to the forward pass starting from the first 2D location $\mathbf{p}_1$ of the hand and backward pass starting from the last location $\mathbf{p}_T$ of the hand, resulting in the forward trajectory $\{\mathbf{p}_t^{(f)}\}_{t=1}^T$ and backward trajectory $\{\mathbf{p}_t^{(b)}\}_{t=1}^T$. Then, for each frame t, the ultimate 2D location is determined by a temporally weighted sum $\tilde{\mathbf{p}}_t = w_t\mathbf{p}_t^{(f)} + (1-w_t)\mathbf{p}_t^{(b)}$ where the weight w_t is temporally decreasing from 1.0 to a constant c by $w_t = c + (1-c)/(1+\exp(t-T/2))$. In practice, we set c to 0.3. The rationale of weighing is to mitigate the error accumulation from the RAFT model. It assigns more weight to the earlier locations by forward flow and more weight to the latter locations by backward flow, with the margin c between the two passes.

Local 3D Trajectory With the 2D hand trajectory, it is straightforward to obtain the 3D hand trajectory by fetching the depth of each trajectory point from the RGB-D clips. However, we noticed that due to the fast motion of the hand and camera, the recorded depth channels in those frames could be missing, i.e., depth values are zeros (see the red dots in Fig. 2). To obtain high-quality 3D hand trajectory annotations, we initially attempted to use the state-of-the-art depth estimation model NewCRFs [13] to estimate the missing depths from RGB frames. However, it cannot work well due to the camera motion that results in dynamic scenes in RGB frames. Instead, we found that a simple least-square fitting (LSF) by combining the third-order polynomial and sine functions, i.e., $z_\alpha(t) = \alpha_1 t^3 + \alpha_2 t^2 + \alpha_3 t + \alpha_4 + \alpha_5 \sin(\alpha_6 t)$, could repair the missing depth. For both EgoPAT3D and H2O, we apply the LSF to repair 3D hand trajectory depth. To enable successful depth fitting, we use at least 10 valid trajectory points to fit a multinomial model on each 3D hand trajectory that contains invalid depths.

Global 3D Trajectory Note that the 3D trajectory points from the previous step are defined in the local camera coordinate system. When the camera is moving in an egocentric view, using RGB videos to predict the local 3D trajectory

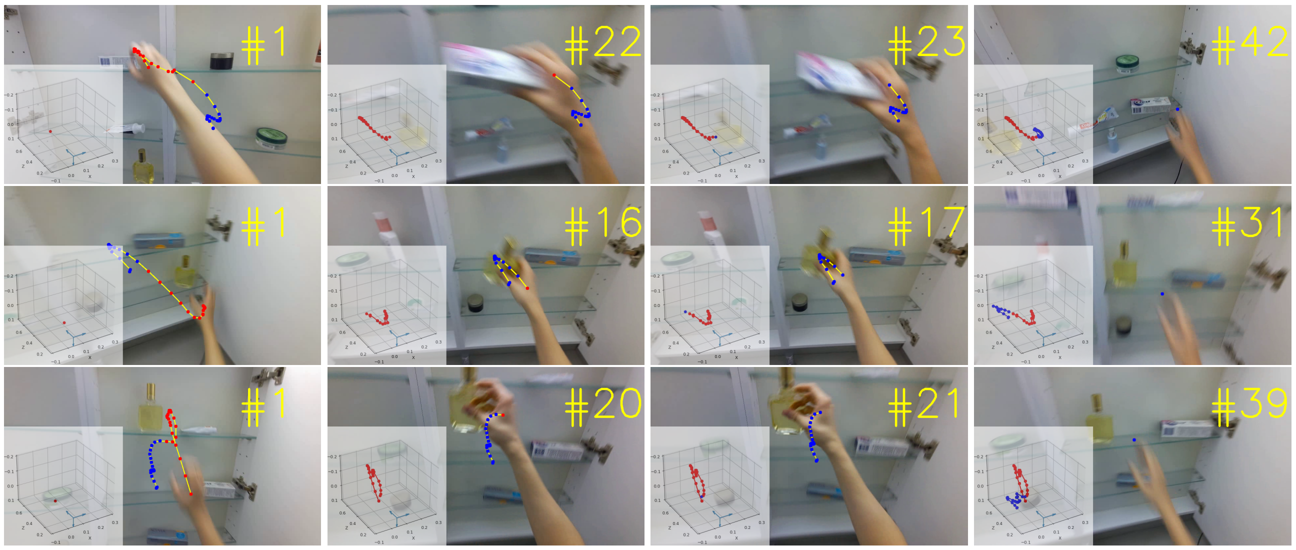


Figure 1: **Dataset Examples**. For each video (in a row), the global 3D trajectory and the projected 2D trajectory are visualized, where the past and future trajectory segments are in red and blue, respectively. Zoom in for more details.

will be ambiguous. In other words, distinct visual contents are forced to learn to predict numerically similar coordinates. To eliminate the ambiguity, similar to EgoPAT3D [8], we propose to transform the 3D trajectory targets into a global world coordinate system with reference to the first frame. This is a visual odometry procedure that computes the 3D homogeneous transformation $\mathbf{M}_t \in \mathbb{R}^{4\times4}$ between camera poses at two successive frames $t-1$ and t. Eventually, a local 3D trajectory point $\mathbf{p}_t^l$ is transformed as a global 3D trajectory point $\mathbf{p}_t^g$ by the accumulative matrix product $\mathbf{p}_t^g = \prod_{k=1}^t \mathbf{M}_k \mathbf{p}_t^l$. In experiments, we use the global 3D trajectory $\{\mathbf{p}_t^g\}_{t=1}^T$ as the ground truth for model training, evaluation, and visualization by default. Fig. 1 shows three video examples with global 3D trajectory annotations.

A.2. Camera Intrinsics and Poses

For both the EgoPAT3D and H2O, the camera intrinsics are fixed across all samples. Table 1 summarizes the camera intrinsics of the dataset we used in this paper. Note that the intrinsics are scaled with the factor 0.25 when we downscale the RGB videos to the input resolution. For camera poses of EgoPAT3D, we use Open3D [15] library to perform visual odometry[1] by using adjacent RGB-D pairs so that the camera motion is obtained. The camera poses of H2O dataset are given for each video frame.

B. Additional Implementation Details

Data Structure To enable efficient parallel training with batches of data input that contain videos of varying lengths, we adopt the mask mechanism in our implementation. Specifically, we set the maximum length of each video to 40 and 64 for EgoPAT3D and H2O, respectively. The lengths of the past observation and future frames are determined by the actual video length. For instance, when the observation ratio is set to 0.6, a sample with 35 frames in total has 21 observed frames, 14 unobserved frames, and 5 zero-padded frames. Since the visual background of RGB videos is relatively clean, we resize videos into the size of 64×64 in training and inference.

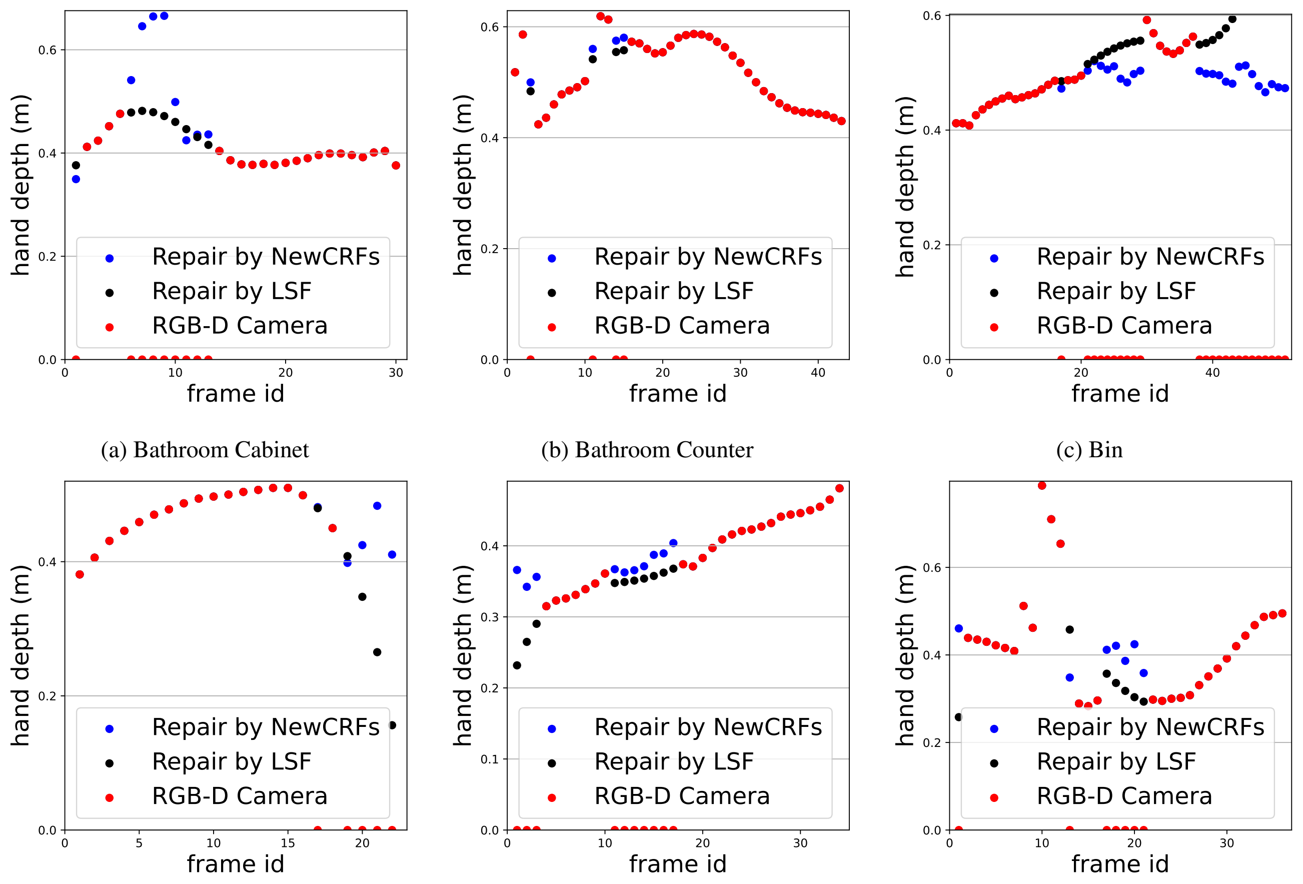
Table 1: Summary of camera intrinsics

EgoPAT3D	resolution	$H = 2160,\quad W = 3840$
	focal length	$f_x = 1808.203,\quad f_y = 1807.946$
	principle point	$o_x = 1942.287,\quad o_y = 1123.822$
H2O	resolution	$H = 720,\quad W = 1280$
	focal length	$f_x = 636.659,\quad f_y = 636.252$
	principle point	$o_x = 635.284,\quad o_y = 366.874$

Model Structure For the ResNet-18 backbone, we replace the global pooling layer after the last residual block with `torch.flatten`, in order to preserve as much visual contextual information as possible. When the visual prompt tuning (VPT) is utilized, the width of the padded learnable pixels is set to 5 as suggested by [4], resulting in 1380[2] additional parameters to learn. For the ViT backbone, we adopt the `vit/b16-224` architecture provided by TIMM, which is pre-trained on the ImageNet-21K

[1]In practice, we followed the EgoPAT3D to use the Open3D API (*RGBDOdometryJacobianFromHybridTerm*) to compute the 3D camera motion.

[2]For 64×64 input, the number of learnable parameters in prompt embeddings is computed by $(64 + 5 \times 2)^2 - 64^2 = 1380$.



(a) Bathroom Cabinet (b) Bathroom Counter (c) Bin

(d) Kitchen Cupboard (e) Microwave (f) Nightstand

Figure 2: Examples of comparison between the Least Square Fitting (LSF) and the depth estimation model NewCRFs [13] for repairing the noisy depth values from EgoPAT3D RGB-D data. It's clear that on this video dataset with dynamic background, a simple LSF with a multinomial model could achieve a much better depth repairing effect than the state-of-the-art deep learning model NewCRFs.

dataset. For either ResNet-18 or ViT-based frame encoder $f_{\mathcal{V}}$, the output feature is embedded by a two-layer MLP with 512 and 256 hidden units. For the trajectory encoder $f_{\mathcal{T}}$, we use a two-layer MLP with 128 and 256 hidden units. For both visual and trajectory transformer encoders, we utilize the standard transformer encoder architecture, which consists of 6 multi-head self-attention blocks where the number of heads is 8 and the MLP ratio is 4. For the decoder, we implement the three prediction branches, i.e., future trajectory prediction, uncertainty prediction, and velocity prediction, using three MLP heads, each of which consists of 128 and 3 hidden units. For trajectory and velocity prediction outputs, we use `tanh` activation, while for the uncertainty output, we use `softplus` activation. Besides, for the velocity prediction, layer normalization is applied to each hidden layer.

Learning and Inference In training, we set the δ parameter of Huber loss to $1e-5$, and set the γ coefficient of the velocity-based warping loss to 0.1. For the cosine learning rate scheduler, we adopt warm-up training in the first 10 epochs. For 500 training epochs in total, our model training can be completed within 5 hours on a single RTX A6000 GPU. In testing, we evaluate the predicted 3D trajectory in the global coordinate system by referring to the camera at the first time step, while visualizing the 2D trajectory by first projecting the global 3D trajectory into the local 3D trajectory, and then projecting the local 3D coordinates onto a video frame as 2D pixel coordinates.

C. Additional Evaluation Results

Full results on H2O-DT We additionally provide full experimental results by training models on **H2O-DT** and **H2O-DT** w/o depth repair in Table 2. It shows that our USST method could still achieve the best performance using training data with inaccurate trajectory annotations.

Table 2: **Results of models training on H2O-DT dataset**. We report all results of models trained by annotations from H2O-DT (left) and its version without depth repair (right), and tested on the accurate H2O-PT test set. All models are built with ResNet-18 backbone. Best and secondary results are viewed in bold **black** and **blue** colors, respectively.

Models	H2O-DT						H2O-DT (w/o depth repair)					
	ADE (↓)			FDE (↓)			ADE (↓)			FDE (↓)		
	3D$_{(3D)}$	2D$_{(3D)}$	2D$_{(2D)}$	3D$_{(3D)}$	2D$_{(3D)}$	2D$_{(2D)}$	3D$_{(3D)}$	2D$_{(3D)}$	2D$_{(2D)}$	3D$_{(3D)}$	2D$_{(3D)}$	2D$_{(2D)}$
DKF [5]	0.236	0.235	0.269	0.138	**0.030**	**0.020**	0.199	0.186	0.208	0.181	0.153	0.187
RVAE [7]	0.125	0.209	0.094	0.057	0.082	0.047	0.051	0.060	0.059	0.058	0.071	0.059
DSAE [12]	0.081	0.113	0.078	0.043	0.059	0.040	0.072	0.068	0.063	0.067	0.047	0.077
STORN [1]	0.091	0.100	0.070	0.245	0.078	0.040	0.067	0.061	0.054	0.135	0.097	0.121
VRNN [2]	0.080	0.092	0.068	0.042	0.035	0.039	0.065	0.063	0.054	0.133	0.087	0.087
SRNN [3]	0.087	0.097	0.076	0.124	0.072	0.045	0.055	0.059	0.061	0.083	0.089	0.135
AGF [14]	0.108	0.065	0.080	0.171	0.061	0.214	0.099	0.075	0.065	0.186	0.044	0.056
OCT [9]	0.360	0.473	0.350	0.348	0.362	0.520	0.381	0.519	0.403	0.403	0.521	0.505
ProTran [10]	0.080	0.082	0.099	**0.023**	0.031	0.107	0.070	0.093	0.064	0.162	0.146	0.041
USST	**0.033**	**0.041**	**0.041**	0.052	0.050	0.041	**0.032**	**0.041**	**0.040**	**0.053**	**0.041**	**0.041**

Table 3: **FDE results of 2D hand trajectory forecasting**. Compared models are built with ResNet-18 (R18) backbone. Best and secondary results are in bold **black** and blue colors, respectively.

Model	Seen (↓)	Unseen (↓)
DKF [5]	0.150	0.239
RVAE [7]	0.152	0.201
DSAE [12]	0.144	0.233
STORN [1]	0.145	0.266
VRNN [2]	0.155	0.237
SRNN [3]	0.157	0.198
OCT [9]	0.090	0.147
ProTran [10]	0.134	**0.049**
USST (R18)	0.075	0.107
USST (ViT)	**0.066**	0.114

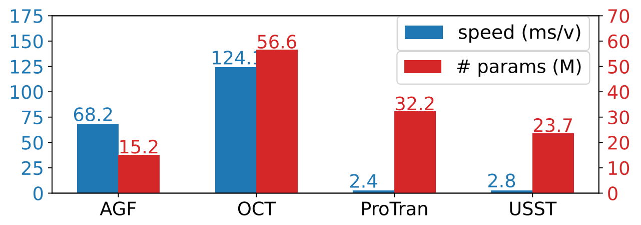


Figure 3: Inference speed in milliseconds/video (ms/v) and the number of model parameters in million (M), tested on a single RTX 6000Ada GPU with input video size 64 × 64 × 64.

FDE results on EgoPAT3D-DT We additionally provide the Final Displacement Error (FDE) results for 2D hand trajectory forecasting as shown in Table 3. Our method could achieve the best performance on the seen test data while being competitive on the unseen test data. Besides, ProTran shows the best result on the unseen data, which could be attributed to its extra trajectory supervision from the full observation of the latent Gaussian distributions.

Inference speed In Fig. 3, we compared with the Transformer-based methods. It shows the USST achieves competitive speed to ProTran while comparable model size to AGF. With certain improvements, our method could potentially benefit the rendering latency in AR/VR.

D. Additional Demos

In addition to the visualizations of the main paper, we additionally provide some examples as shown in Fig. 4 and 5. They show that our method could accurately predict the 3D and 2D hand trajectories in both seen and unseen scenarios.

References

[1] Justin Bayer and Christian Osendorfer. Learning stochastic recurrent networks. In *NeurIPS Workshop*, 2014. 4

[2] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *NeurIPS*, 2015. 4

[3] Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. In *NeurIPS*, 2016. 4

[4] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *ECCV*, 2022. 2

[5] Rahul G Krishnan, Uri Shalit, and David Sontag. Deep kalman filters. *arXiv preprint arXiv:1511.05121*, 2015. 4

[6] Taein Kwon, Bugra Tekin, Jan Stühmer, Federica Bogo, and Marc Pollefeys. H2O: Two hands manipulating objects for first person interaction recognition. In *ICCV*, 2021. 1

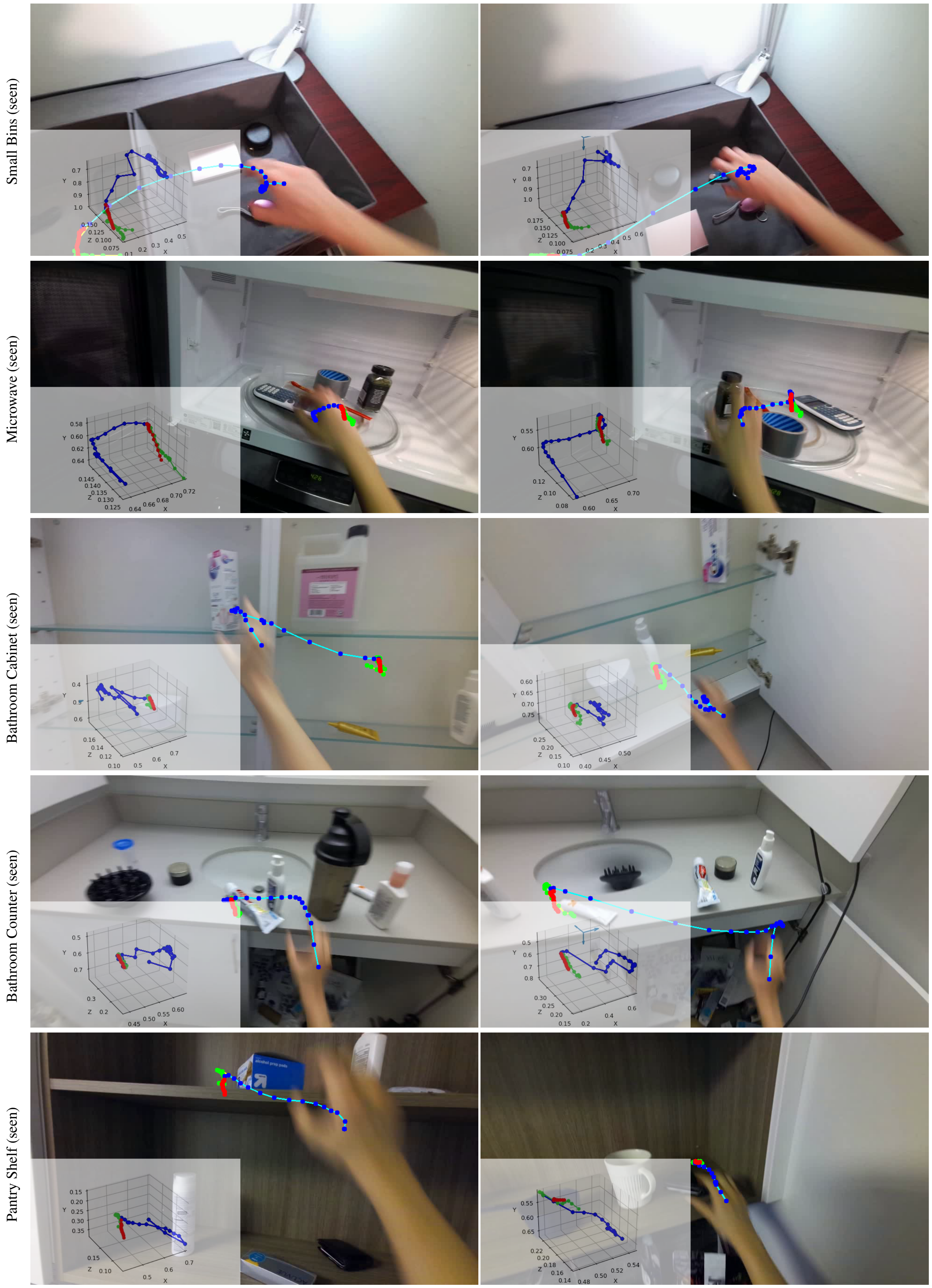


Figure 4: **Visualization on Seen data.** For each scene (in a row), we show two examples of the 2D and 3D trajectories on the first frame. The blue, green, and red trajectory points represent the past observed, future ground truth, and future predictions, respectively.

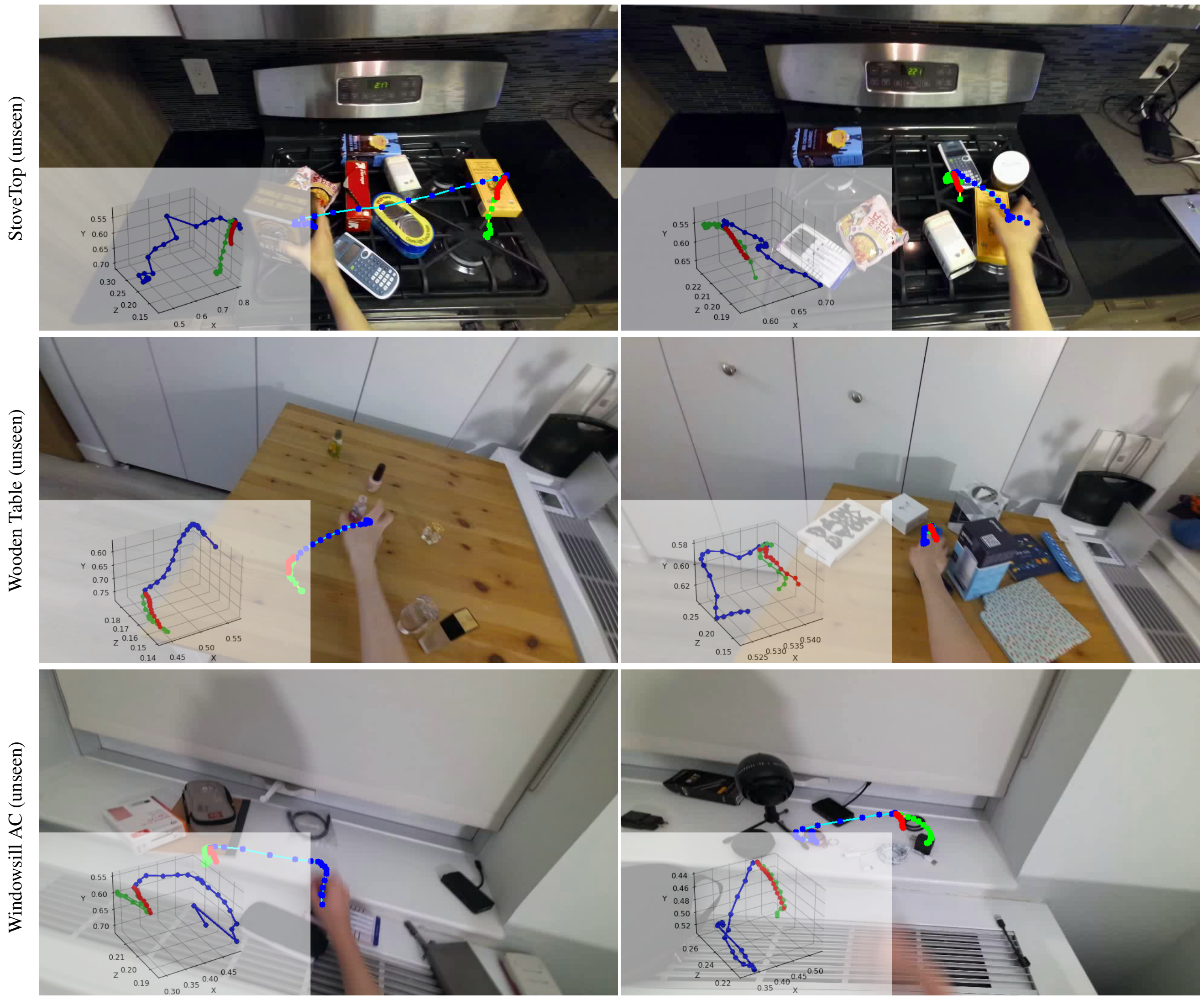


Figure 5: **Visualization on Unseen data.** For each scene (in a row), we show two examples of the 2D and 3D trajectories on the first frame. The blue, green, and red trajectory points represent the past observed, future ground truth, and future predictions, respectively.

[7] Simon Leglaive, Xavier Alameda-Pineda, Laurent Girin, and Radu Horaud. A recurrent variational autoencoder for speech enhancement. In *ICASSP*, 2020. 4

[8] Yiming Li, Ziang Cao, Andrew Liang, Benjamin Liang, Luoyao Chen, Hang Zhao, and Chen Feng. Egocentric prediction of action target in 3d. In *CVPR*, 2022. 1, 2

[9] Shaowei Liu, Subarna Tripathi, Somdeb Majumdar, and Xiaolong Wang. Joint hand motion and interaction hotspots prediction from egocentric videos. In *CVPR*, 2022. 4

[10] Binh Tang and David S Matteson. Probabilistic transformer for time series analysis. In *NeurIPS*, 2021. 4

[11] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020. 1

[12] Li Yingzhen and Stephan Mandt. Disentangled sequential autoencoder. In *ICML*, 2018. 4

[13] Weihao Yuan, Xiaodong Gu, Zuozhuo Dai, Siyu Zhu, and Ping Tan. NeW CRFs: Neural window fully-connected crfs for monocular depth estimation. In *CVPR*, 2022. 1, 3

[14] Ye Yuan, Xinshuo Weng, Yanglan Ou, and Kris M Kitani. Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In *ICCV*, 2021. 4

[15] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018. 2