

359 Appendix

Table of Contents

362	A Training Details	12
363	B Simulation Details	12
364	C Multi-modality Analysis	15
365	D Occlusion Analysis	15
366	E Real-world Equipment	15
367	F Workspace	17
368	G Foreground Segmentation	17

372 A Training Details

373 **Model Structure:** FlowBotHD consists of a current state encoder, a history state encoder and a
 374 denoiser. Both current and history state encoder are based on PointNet++ [38], where the history
 375 encoder only includes the PointNet++ encoding module (i.e. no decoder) that outputs a global latent
 376 of 128 dimensions. We kept the same model architecture and hyperparameter set as the original
 377 PointNet++ paper except for using a regression head for the history encoder output instead of the
 378 original segmentation head. We inject the history latent into every layer of the PointNet++ frame-
 379 work in the decoder (see Figure 1). This is performed through a hadamard product of the current
 380 point cloud encoding with the latent encodings at every decoding layer to produce the final output
 381 of the model. We base the denoiser on a DiT with 5 layers, 4 heads, and a hidden size of 128.

382 **Dataset Details:** Similar to the original FlowBot3D paper [1], we create a randomly opened dataset
 383 (RO) in which we randomly sample the view perspective, the target joint, and the open ratio. To open
 384 objects from the fully closed state, we create another mixed dataset (MD) in which we half of the
 385 objects are fully closed and the other half randomly opened. To enable history-aware training, we
 386 generate a dataset with 1/3 of the objects fully closed, 1/3 randomly opened but without history and
 387 the final third randomly opened and with history at random interval.

388 **Training Process:** We train the model for 450 epochs using AdamW optimizer with a weight decay
 389 of $1e-5$. We use a learning rate of $1e-4$, and a batch size of 128. We evaluate the model every 20
 390 epochs, and save the checkpoint with the best winner-take-all RMSE metric. The Winner-take-all
 391 metric means that we repeatedly make predictions for each sample 20 times and record the best
 392 RMSE.

393 B Simulation Details

394 We implement a simulation environment with PyBullet that simulates operating a suction gripper to
 395 operate PartNet-Mobility objects. We first roll out the simulation with ground truth flow and filter
 396 out samples that can't open with the ground truth motion, due to errors in the simulator. We repeat
 397 each sample for 5 times during evaluation. The final test object categories we evaluated on after
 398 filtering and repeating are listed in Table 2.

399 To manipulate the objects, we first attach the suction gripper to the object. This is implemented by
 400 first teleporting the gripper to a position close to the target grasp point and then gradually moving

	Oven	20		Microwave	10		Table	245		Phone	5
	Bucket	10		Window	55		Furniture	765		Door	30
	Box	20		Kettle	25		Dishwasher	45		Laptop	40
	Toilet	55		Safe	12		WashMachine	25		TrashCan	35
	KitchenPot	25		Refrigerator	75		FoldingChair	15		Stapler	20

Table 2: Simulation Objects and Sample Counts

401 towards the target grasp point until contact is detected. After contact, we apply a physical force
402 constraint between the gripper and the object. The move action is then implemented as applying a
403 certain velocity along the predicted direction.

404 **Hyperparameter in Policy:** There are 3 hyperparameters to set in Algorithm 1: the threshold for
405 switch grasp point δl , the threshold for consistency check $\delta \theta_{cc}$ and the threshold for good movement
406 δm . We set $\delta l = 0.2$, $\delta \theta_{cc} = 30^\circ$ and $\delta m = 1e-2$ in simulation. Table 3 demonstrates the quantitative
407 simulation results for the normalized distance metric, computed in the same way as in Eisner et al.
408 [1].

	SG	CC	HF	AVG _c	AVG _s																				
Baselines																									
FlowBot (RO)	×	×	×	0.250	0.166	0.23	0.54	0.29	0.29	0.00	0.90	0.10	0.23	0.35	0.31	0.10	0.15	0.18	0.10	0.10	0.48	0.39	0.07	0.11	0.08
FlowBot (RO)	✓	×	×	0.220	0.117	0.16	0.53	0.29	0.22	0.00	0.43	0.10	0.15	0.45	0.14	0.05	0.09	0.18	0.09	0.27	0.31	0.46	0.07	0.15	0.26
FlowBot (MD)	×	×	×	0.183	0.143	0.31	0.19	0.06	0.22	0.00	0.06	0.14	0.21	0.47	0.30	0.09	0.12	0.23	0.12	0.09	0.24	0.63	0.07	0.05	0.08
FlowBot (MD)	✓	×	×	0.159	0.098	0.26	0.08	0.07	0.09	0.00	0.06	0.08	0.14	0.56	0.07	0.05	0.09	0.16	0.09	0.07	0.35	0.60	0.07	0.13	0.16
Ablations																									
Ours- No History	×	×	×	0.255	0.176	0.29	0.57	0.18	0.19	0.16	0.36	0.14	0.23	0.46	0.28	0.18	0.14	0.20	0.31	0.07	0.53	0.54	0.07	0.09	0.10
Ours- No History	✓	×	×	0.178	0.113	0.26	0.36	0.09	0.09	0.00	0.06	0.13	0.10	0.38	0.17	0.08	0.09	0.12	0.21	0.31	0.29	0.45	0.07	0.10	0.20
Ours- No History	✓	✓	×	0.176	0.110	0.28	0.28	0.19	0.10	0.08	0.06	0.10	0.12	0.75	0.11	0.07	0.10	0.10	0.08	0.10	0.11	0.45	0.07	0.11	0.27
Ours- No Diffusion	✓	×	×	0.265	0.213	0.35	0.67	0.55	0.15	0.00	0.51	0.14	0.19	0.73	0.31	0.15	0.20	0.31	0.37	0.11	0.08	0.55	0.07	0.12	0.08
Ours- No Diffusion	✓	×	✓	0.194	0.183	0.39	0.48	0.40	0.11	0.00	0.45	0.18	0.17	0.04	0.30	0.11	0.18	0.28	0.35	0.24	0.08	0.48	0.07	0.15	0.07
Ours																									
FlowBotHD	✓	×	×	0.181	0.139	0.55	0.50	0.13	0.05	0.04	0.39	0.12	0.15	0.33	0.08	0.15	0.10	0.40	0.11	0.23	0.09	0.52	0.07	0.10	0.09
FlowBotHD	✓	×	×	0.103	0.086	0.33	0.35	0.11	0.05	0.00	0.06	0.06	0.02	0.15	0.08	0.07	0.08	0.25	0.09	0.08	0.10	0.27	0.07	0.07	0.08
FlowBotHD	✓	×	✓	0.110	0.096	0.53	0.52	0.13	0.05	0.04	0.10	0.06	0.15	0.24	0.07	0.05	0.07	0.21	0.13	0.23	0.11	0.49	0.07	0.12	0.10
FlowBotHD	✓	✓	✓	0.096	0.072	0.25	0.26	0.05	0.05	0.00	0.20	0.06	0.11	0.38	0.07	0.04	0.06	0.18	0.07	0.07	0.09	0.23	0.07	0.09	0.07

Table 3: Normalized Distance Metric Results (\downarrow): Normalized distances to the goal articulation joint angle after a full rollout of the methods. The lower the better.

409 We demonstrate the simulation process in Fig 4. We visualize the flow predictions (the first row),
410 the simulation process (the last row, x-axis is the step number, the y-axis is the open ratio), and the
411 trajectory plot along with policy signals (the middle plot). In these visuals, we can see our model
412 and policy’s pattern of opening an object: make trials at the beginning, once the door is opened a
413 bit, make consistent predictions based on past history and consistency check.

414 The first example is quite smooth, our model succeeds to move the door at the first step. Then it
415 continues to make consistent and history-aware predictions until the door is fully opened. Consis-
416 tency check didn’t have to filter out much predictions due to the good prediction quality, and history
417 is always updated at each step, meaning that we’re always using the previous step’s history for the
418 current step’s prediction. In the second example, our model makes several trials at the fully closed
419 state. History is not updated during these steps and remains none. Once the door is opened, the his-
420 tory information and the consistency check enable the model to execute consistent actions. We can
421 see from the background bar plot that for step 11, the consistency check filters out 12 inconsistent
422 predictions, demonstrating the effectiveness of applying consistency check. Also, we can see that at
423 step 6, the door is opened a bit, but the history is not updated due to the small movement (indicating
424 the prediction is not good enough) which demonstrates how the history filter works.

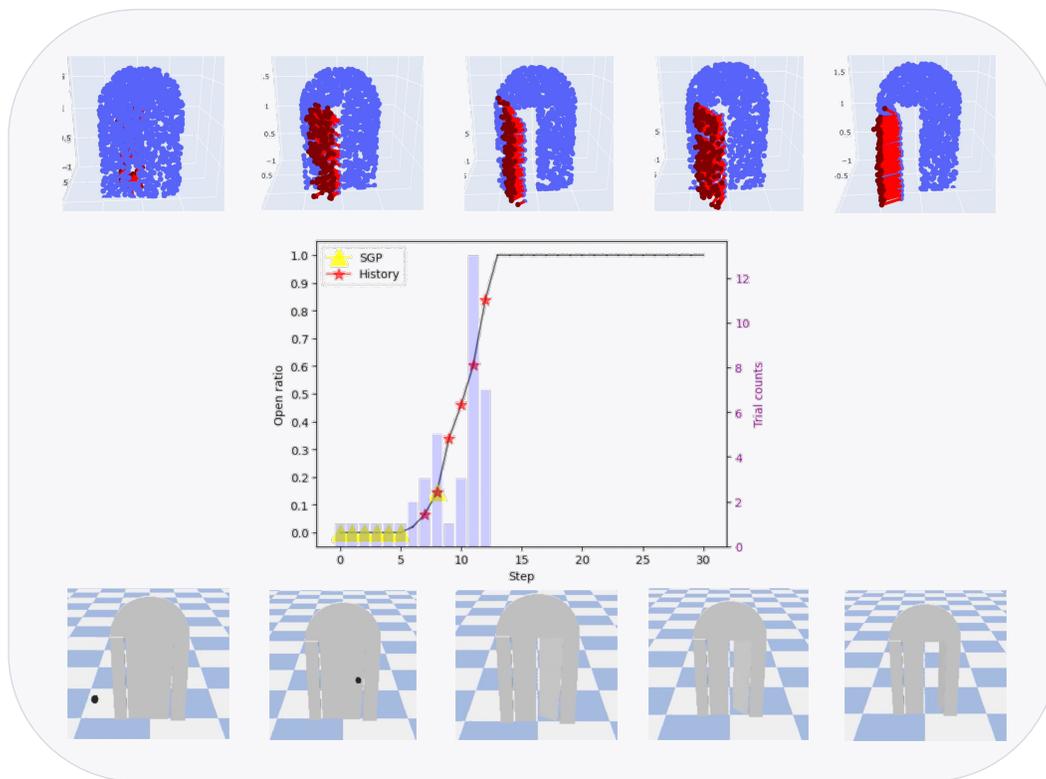
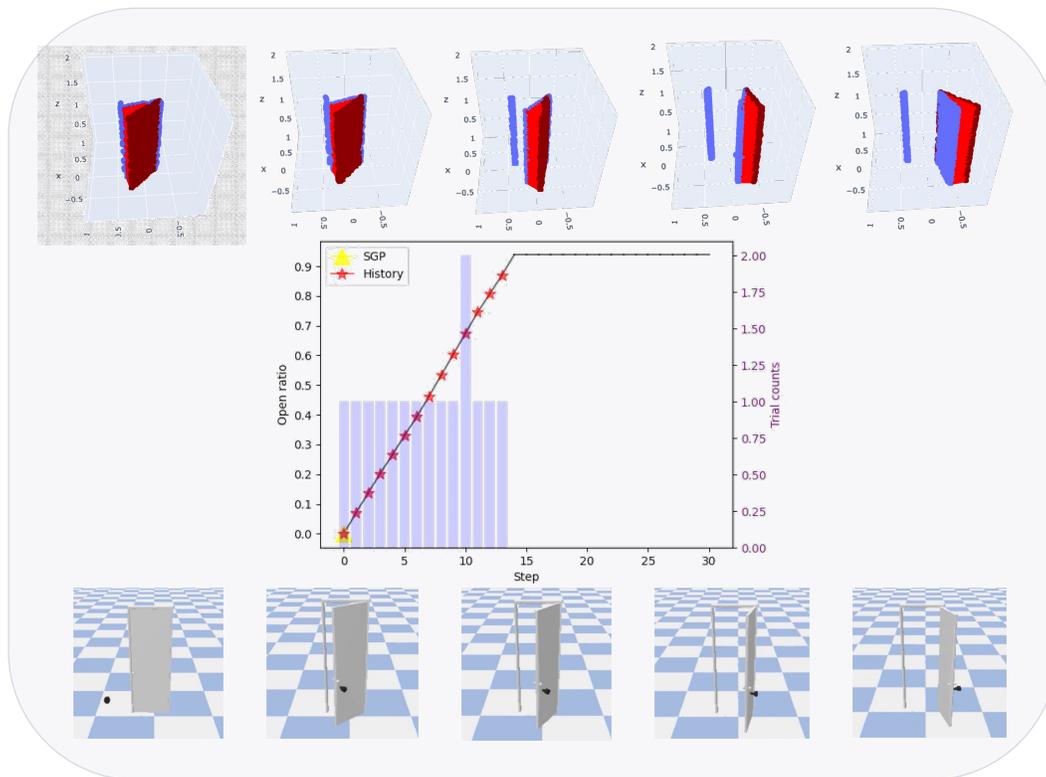


Figure 4: Simulation Visualizations: The plot in the middle is a simulation trajectory plot with x-axis as step number, and left y axis as the open ratio. We visualize the history update signal with red polygons, step with red polygon means updating this step's prediction as the latest history. Yellow triangle represents the switch grasp point (SGP) signal, meaning that this step requires a new grasp point. The bar plot on the background corresponds to number of trials we take to generate a prediction that satisfies the consistency check trial. The axis for the bar plot is the right y-axis.

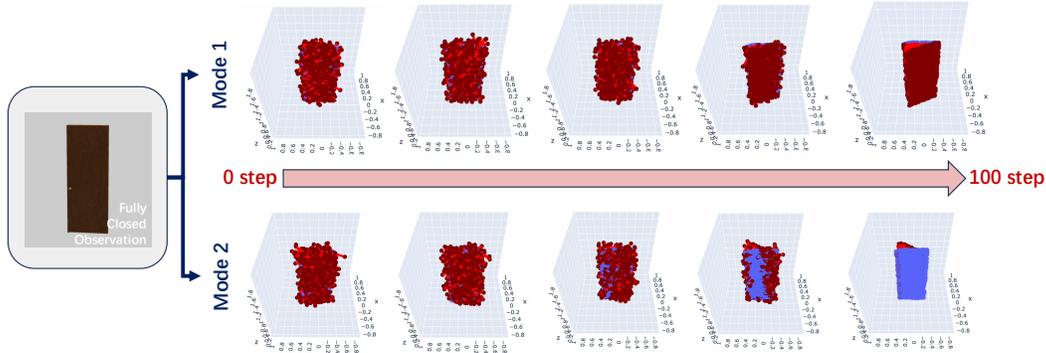


Figure 5: Multi-Model Diffusion Process Visualization: We visualize the denoising process (100 steps). We can see that the flow starts from pure random noises and gradually converges to different modes: Pull and Push.

425 C Multi-modality Analysis

426 To specifically analyze our model’s improvements on handling multi-modality, we evaluated base-
 427 line FlowBot3D trained on randomly opened dataset (RO) / mixed dataset (MD) and our model
 428 FlowBotHD on doors. We open each test door from 0% to 100%. We average the metrics within
 429 10% open to compute the performance for closed doors, and we average the metrics above 10%
 430 open to compute the performance for open doors. As we can see from Table 4, our model outper-
 431 forms the baselines by a large margin across all metrics. The observation that FlowBot3D trained on
 432 randomly opened dataset outperforms the one trained on the mixed dataset shows that multi-modal
 433 training data can confuse the regression models’ training process.

Model	Cosine (↑)			RMSE(↓)			MAG(↓)		
	FlowBot3D (RO)	FlowBot3D (MD)	FlowBotHD	FlowBot3D (RO)	FlowBot3D (MD)	FlowBotHD	FlowBot3D (RO)	FlowBot3D (MD)	FlowBotHD
Closed (<10% open)	0.2033	0.3129	0.8046	0.4426	0.4069	0.1833	0.2468	0.2150	0.1047
Randomly Open (>10% open)	0.7351	0.2720	0.9089	0.2218	0.4617	0.1246	0.1782	0.2215	0.0919

Table 4: Multi-Modality Analysis: We compare our model with baselines on doors. The cosine metric means the cosine similarity between the predicted flow and the ground truth flow, the higher the better. RMSE metric means the root mean square error between prediction and ground truth, the lower the better. MAG metric means the flow magnitude error, the lower the better.

434 Fig 5 visualizes the denoising process that produces different modes based on the same fully closed
 435 door, demonstrating our model’s ability of preserving multi-modality for ambiguous examples.

436 D Occlusion Analysis

437 We analyze our model’s performance under occlusions on different object categories. A door exam-
 438 ple is included in the Fig 2 of the main paper, and we include occluded fridge and furniture examples
 439 in Fig 6. We can see from the visualizations that with history, FlowBotHD is able to produce stable
 440 and robust predictions regardless of the occlusions while FlowBot3D makes less consistent predic-
 441 tions when severely occluded.

442 E Real-world Equipment

443 In our real-world experiments, the point cloud data of the door comes from an Azure Kinect Depth
 444 Camera. The door is custom built out of plywood, with four pairs of hinges. This allows for it to be
 445 configured to open in all four of the possibilities of a standard door (forwards to the right, forwards
 446 to the left, backwards to the right, and backwards to the left). The door’s current configuration is
 447 determined by two thin allen key shaped metal rods, which connect a pair of hinges. Those hinges

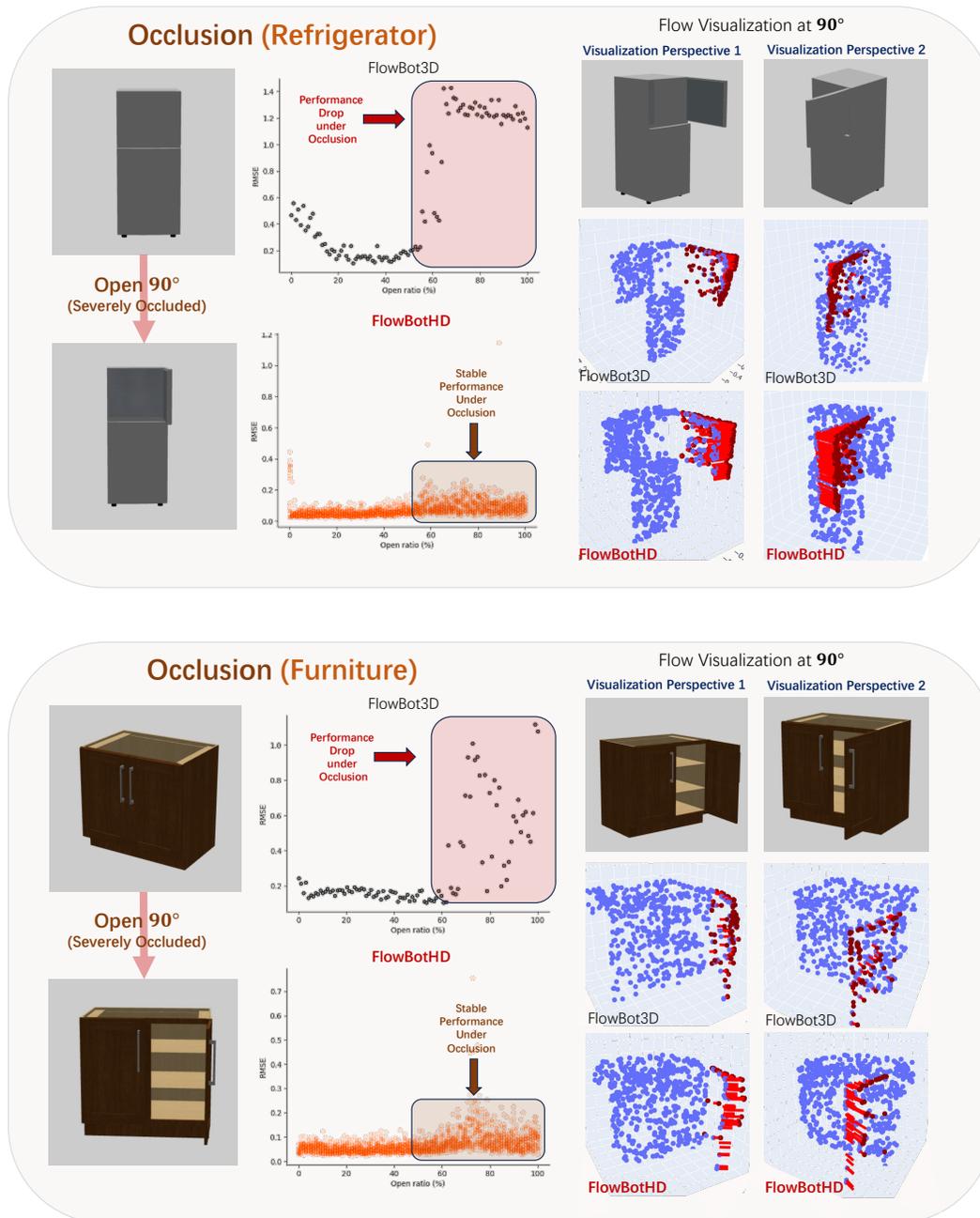


Figure 6: Occlusion analysis: We open the object to different angles, make predictions and plot the Cosine Similarity metric (\uparrow) and RMSE metric (\downarrow) against the open ratio. We also include flow visualizations from different viewing perspective to intuitively show the quality comparison of the predictions. We can see from the visualization that FlowBotHD can make good and consistent predictions even under severe occlusions.

448 are then effectively in use and determine how the door opens. The door's frame is 25 cm by 8.5 cm
449 by 43.8 cm, with a 31.75 cm by 36 cm by 1.25 cm stabilizing base. The door itself is 16.83 cm by
450 1.75 cm by 43.5 cm, with nothing on its front or back face to differentiate which way it opens. To
451 ensure it fit in our workspace, we built it to be much smaller than a normal door, then scaled the
452 point cloud up by two before passing it into the model.

453 **F Workspace**

454 We constructed our workspace in a 1.3 m by 1.3 m by 1.1 m space with Vention beams. The Azure
455 Kinect Camera was placed so that it pointed toward the center of the workspace at an angle that
456 allowed the door to be seen clearly.

457 **G Foreground Segmentation**

458 To isolate the door in the point cloud, we had to programmatically segment the table and background
459 points. This was simply done by thresholding the x , y , and z values of the points to effectively crop
460 all points outside of the box where the object would be. For example, all points with z value below
461 0.02 were removed, as that marks the top of the table.