Supplementary Material for "Predicting the Physical Dynamics of Unseen 3D Objects"

Davis Rempe Srinath Sridhar He Wang Leonidas J. Guibas
Stanford University

1. Introduction

In this document we present additional details, discussion, and results for "Predicting the Physical Dynamics of Unseen 3D Objects" that were omitted due to space constraints. Please refer to the main paper for an overview of our proposed method and the primary results. In Section 2, we provide additional details on the implementation and data collection, Section 3 discusses toppling and its stochastic nature, and Section 4 provides additional results on simulated and real-world data.

Video: In addition to this document, there is a supplementary video which provides a succinct overview of the proposed method and shows qualitative results. We *highly encourage* the reader to watch this video as it gives a better idea of our simulation and real-world data, along with the accuracy of predicted trajectories from our model.

2. Implementation Details

Here we provide additional details of our data pipeline and methods.

Simulation Procedure and Datasets: The simulation pipeline is introduced in Section 4 of the main paper. Prior to simulation, some pre-computation is done on object shapes to extract accurate physical parameters, namely the mass and moment of inertia. To calculate these values, we voxelize each shape using a grid with a cell side length of 2.5 cm. From this we approximate the volume of the object which can be used to calculate the mass given density. Additionally, we compute a discretized approximation of the moment of inertia about the shape's principle axes. These calculated mass and moment of inertia values are used directly to parameterize the simulated rigid bodies within the the Bullet physics engine [1]. Calculating physical parameters in this way ensures consistency across all simulated shapes rather than relying on the physics engine to calculate them using a mesh collider which can be extremely inconsistent and inaccurate. For each performed simulation, the amplitude of the randomly applied impulse is scaled by the object mass to ensure similar distributions for small and large objects alike. Every simulated object and the ground plane has a friction coefficient of 0.5. For each object, we sample a point cloud from the surface to use as input to our model. To ensure uniform sampling, we first oversample (by a factor of 3) the mesh surface area. We then sub-sample these points using furthest point sampling to obtain our final 1024 points. Each simulation runs until the object velocity is below a set threshold for a certain amount of time or it exceeds the maximum simulation time (7 seconds in our data). Since objects may be at rest for multiple timesteps at the end of a simulation, the randomly sampled windows during training will not always have the object coming to rest at the same time (i.e. at the last time step).

We synthesize multiple categories of datasets to train and evaluate our models with the following distribution: `Primitives` (13,550 total simulations, 6,779 include toppling, 197 unique object instances), `Bottles` (13,079 simulations, 6,550 toppling, 154 instances), `Mugs` (13,009 simulations, 1,011 toppling, 37 instances), `Trashcans` (13,018 simulations, 1,658 toppling, 47 instances), `Speakers` (13,059 simulations, 1,688 toppling, 358 instances), and `Combined` (union of others).

Each dataset is split into training (80%) and test sets (20%) by unique objects. This means no objects (or scaled versions of them) that are seen during training are in the test set. During training, about 10% of the objects in the training split are set aside as validation data for early stopping. Since the applied forces are randomly chosen during simulation, the test sets also contain new initial conditions (linear and angular velocities) though they are from the same distribution as the training data.

Real-world Data Experiments: Experimental results on real-world motion capture data are detailed in Section 6.5 of the main paper and Section 4.3 of this document. We collect 66 trajectories of a small cardboard box using an Optitrack [2] motion capture system which uses spherical markers to track objects. The system outputs 6D pose information for the box at 120 Hz, which we down-sample to 30 Hz by averaging using a sliding window with a size of 4 steps to smooth the data. Using 6D pose at each timestep, we derive the change in object state (3D position, rotation, linear velocity, and angular velocity) data to train our net-

work. We trim each trial so that the first step of the sequence is at its maximum linear velocity; this is a rough heuristic to determine when the object has left the hand applying the initial force (see video) and is freely sliding. We sample a full point cloud using a virtual version of the box (7.3 cm $\times$ 6.0 cm $\times$ 17.0 cm) to use as input to our model.

Real-world data is not annotated with per-frame stability information, so we train a modified version of our model that does not output a toppling classification at each step. Additionally, we found that not using batch normalization (likely because of the smaller batch sizes) in the shape processing branch and modifying all terms of the loss function to use the absolute 2-norm (making each loss term equivalent to the numerator of $\mathcal{L}_{\mathbf{P}}$ in Equation 1 of the main paper rather than the whole relative loss) gave the best results. Lastly, when the real-world trained model rolls out sequences, it uses one additional frame of ground truth input to start the roll out (it takes in the ground truth velocities at the second timestep in addition to initial velocities), which we found improved performance.

Model Architecture and Training: We use batch normalization following every layer in the shape processing branch. The state prediction branch uses 3 stacked LSTM cells, each with a hidden layer size of 1024. We train all branches of our network jointly using the Adam [4] optimization algorithm with a starting learning rate of 0.001 which is exponentially decayed to 1×10^{-5} during training. In the shape processing branch, PointNet weights are pretrained on the ModelNet40 [5] classification task, then fine-tuned during our training process. We directly supervise the state prediction branch outputs at every timestep for sequences of 15 steps during training. We train the network for 1000 epochs with a batch size of 64 on an NVIDIA V100. The model weights which result in the lowest validation split loss throughout training are used as the final model. In total, our network architecture has more than 20 million parameters.

3. Toppling Discussion

In reality, the motion of objects sliding (and possibly toppling) on a plane is macroscopically stochastic due to imperfections in the planar surface, micro surface interactions, and a non-uniform coefficient of friction [6]. The outcome of planar contact is also very sensitive to initial conditions [3]. Though the reality of contact interactions differs from approximations by simulation engines (like Bullet [1] which we use), we find the simulated outcome of toppling is still extremely sensitive to perturbations in initial conditions, but less so for sliding with no toppling. Therefore, in the main paper and this supplement we show quantitative results on non-toppling examples to focus evaluation on object shape generalization. In future work, we plan to explore modeling distributions of motion rather than direct regression to capture the probabilistic nature of toppling and sliding in the real world.

Data Freq.	**v**	$\boldsymbol{\omega}$	**P**	$\lvert\boldsymbol{\theta}\rvert$
15 Hz	8.65%	7.59%	8.10%	0.52%
30 Hz	**4.92%**	**3.25%**	**6.13%**	**0.43%**

Table 1. *Relative* roll-out errors training on a `Speakers` dataset with data sampled at 15 Hz against 30 Hz.

4. Additional Results

In this section, we present additional results and experiments omitted from the main paper due to space constraints. As in the main paper, unless otherwise noted all presented quantitative evaluation is performed on non-toppling test set sequences.

4.1. Effect of Sampling Rate

We train our model on a modified `Speakers` dataset which uses a step size of 30 Hz rather than 15 Hz as in prior experiments to evaluate whether using fine-grained data may improve performance. We train on the new data with 30-step sequences so the network still sees 1 second of data for each training example. To make a fair comparison, we compute the *relative* roll-out error for each trained model. Results are shown in Table 1. The larger sampling rate nearly doubles performance for velocity, but gives a less drastic improvement for position and rotation which ultimately decides the quality of rolled out trajectories.

4.2. Friction Generalization

In Section 6.2 of the main paper, we demonstrate our model's ability to implicitly identify friction in order to accurately extrapolate future motion by training on simulated data with random friction coefficients. Here we present additional visualizations from this experiment. Roll-out errors using a varying number of *additional* velocity steps as input are shown in Figure 1 (corresponding to Table 2 in the main paper). Figure 2 shows qualitative examples using 5 additional ground truth velocity steps as input to the model.

In detail, the additional velocity steps are fed in sequentially to the LSTM during test-time roll out. For example, if using 3 additional input steps, the model is given initial velocities (as usual) for the first step, then for the 3 following steps the LSTM inputs are ground-truth observed velocities (rather than its own predictions as done for all other experiments). Following these initial 4 steps (first step and additional 3), the model rolls out trajectories as usual using its own velocity predictions at each step.

4.3. Real-world Data

Results on real-world motion capture data are presented in Section 6.5 of the main paper. We present additional qualitative examples here in Figure 3. Please refer to the attached video for additional examples of roll-out using the model trained on real-world data.

4.4. Comparison to MLP Baseline

A comparison of our proposed model to that of an MLP baseline on the `Speakers` dataset are presented in the main paper in Section 6.3. This modified model uses 5 fully-connected (each size 1024) layers as the state prediction branch rather than an LSTM. We present qualitative results comparing our method to this baseline in Figure 4. The trajectories rolled out by the LSTM (shown in green) closely match ground truth simulation (in grey), while the MLP baseline (in blue) struggles. As discussed in the paper, this indicates a memory mechanism is beneficial to predicting object dynamics.

4.5. Object Generalization

Experiments detailing our model's ability to generalize to unseen object shapes are presented in Section 6.1 of the main paper. Here we present additional qualitative results for the *Leave Out* trained models in Figure 5. Again, we urge the reader to see the video for additional examples. In Figure 6 we break down roll-out errors for each training procedure into individual object datasets.

References

[1] Bullet physics engine. https://pybullet.org. 1, 2

[2] Optitrack motion capture. https://optitrack.com/. 1

[3] A. Ajay, J. Wu, N. Fazeli, M. Bauzá, L. P. Kaelbling, J. B. Tenenbaum, and A. Rodriguez. Augmenting physical simulators with stochastic neural networks: Case study of planar pushing and bouncing. In *International Conference on Intelligent Robots and Systems (IROS)*, 2018. 2

[4] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference for Learning Representations (ICLR)*, 2015. 2

[5] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. 2

[6] K. Yu, M. Bauzá, N. Fazeli, and A. Rodriguez. More than a million ways to be pushed: A high-fidelity experimental data set of planar pushing. *International Conference on Intelligent Robots and Systems (IROS)*, 2016. 2

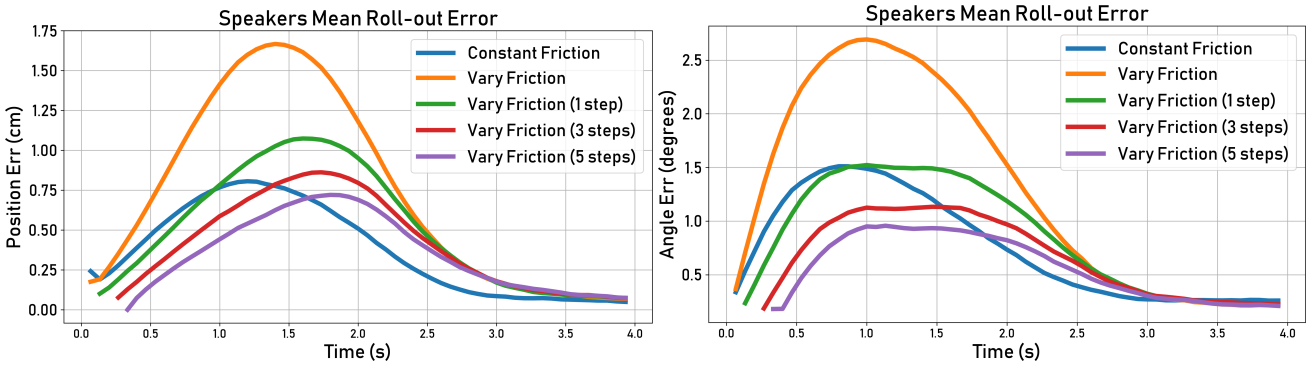

Figure 1. Roll-out errors for friction generalization experiments. Each curve shows the mean roll-out error over all evaluation sequences using either the constant or varied friction `Speakers` data. The *Vary Friction* curves in green, red, and purple use additional ground truth velocity steps at the beginning of their roll-outs to aid friction identification before making predictions. As indicated by the purple curve, using 5 additional ground truth steps to start the roll-out brings performance on varied-friction data very close to that of the model trained on constant friction data for both position and rotation angle.

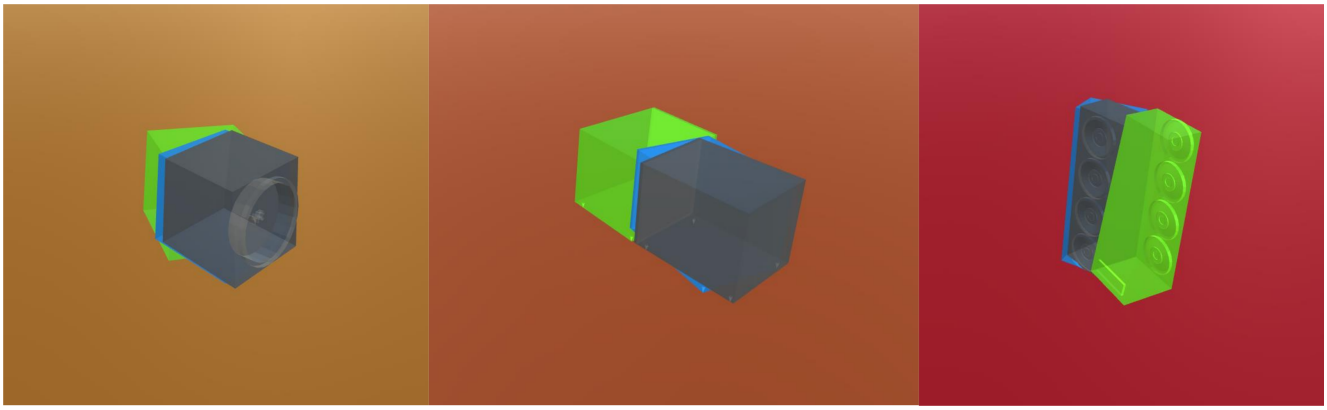

Figure 2. Friction generalization qualitative performance. The final frames of 3 rolled out sequences are shown with friction increasing from left to right. Ground truth simulation is in grey, the model prediction using no additional ground truth steps as input is shown in green, and using 5 additional ground truth steps during roll out is in blue. With lower friction (left), the vanilla roll out tends to underestimate sliding, while overestimating for higher friction (right). Using 5 additional steps of ground truth velocities as input helps the model identify the friction coefficient and make accurate future predictions.

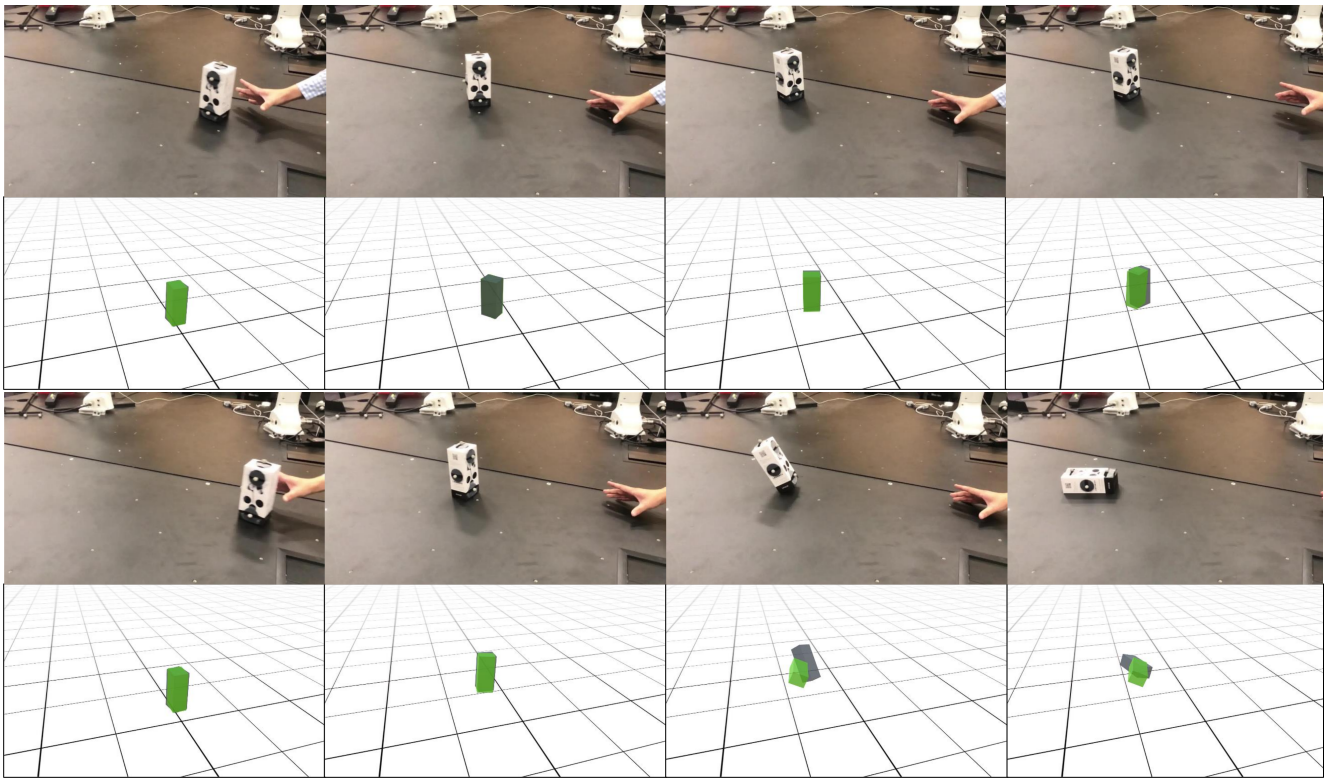

Figure 3. Real-world data performance. Shown are frames pulled from 2 model-predicted sequences with novel initial velocities from real-world motion capture data: one sliding (top) and one toppling (bottom). The model is trained on only 56 sequences of the shown cardboard box, and still predicts accurate trajectories for unseen initial conditions.

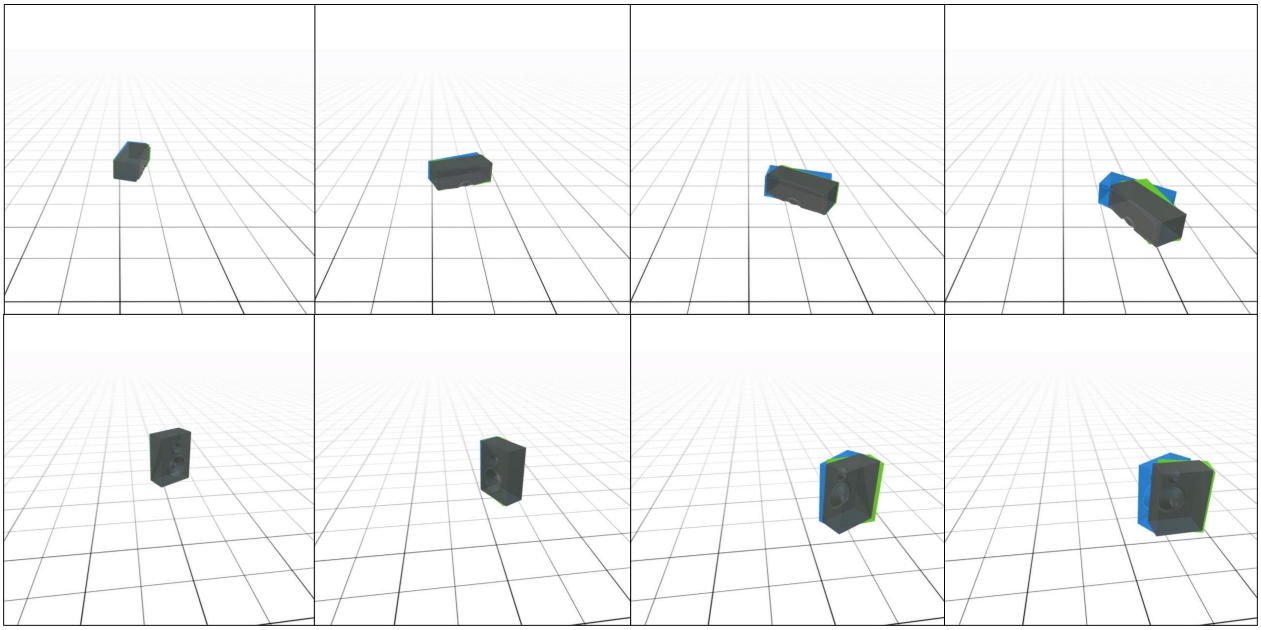

Figure 4. Performance against MLP baseline. Two rolled out sequences are shown for our proposed model (shown in green) and the MLP baseline (shown in blue) which replaces the state prediction LSTM with a simple MLP. Grey is the ground truth simulation given the same initial conditions. Qualitatively, the proposed model achieves roll-out closer to ground truth which indicates the memory mechanism inherent to the LSTM is useful for dynamics predictions.

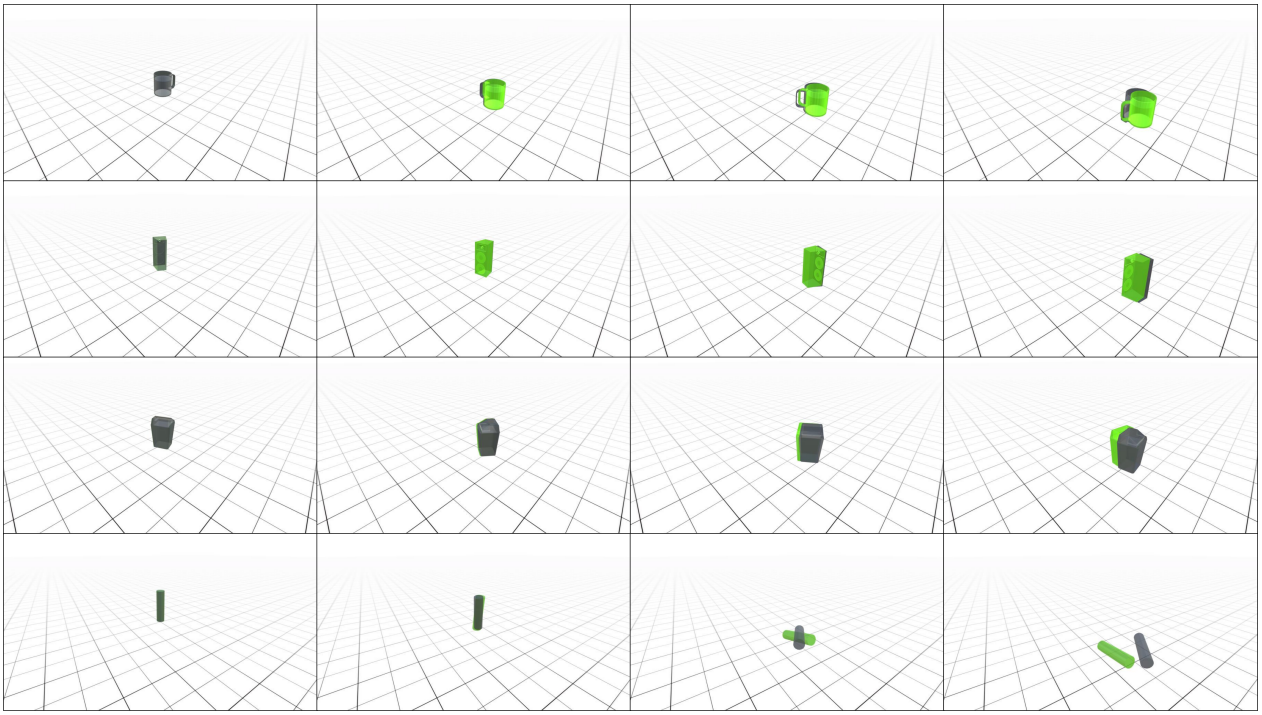

Figure 5. *Leave Out* trained model-predicted trajectories for 4 different sequences. The model roll-out is shown in green and ground truth simulation in grey. From top row to bottom, the object is from the `Mugs`, `Speakers`, `Trashcans`, and `Cylinders` dataset. The bottom row shows a toppling example.

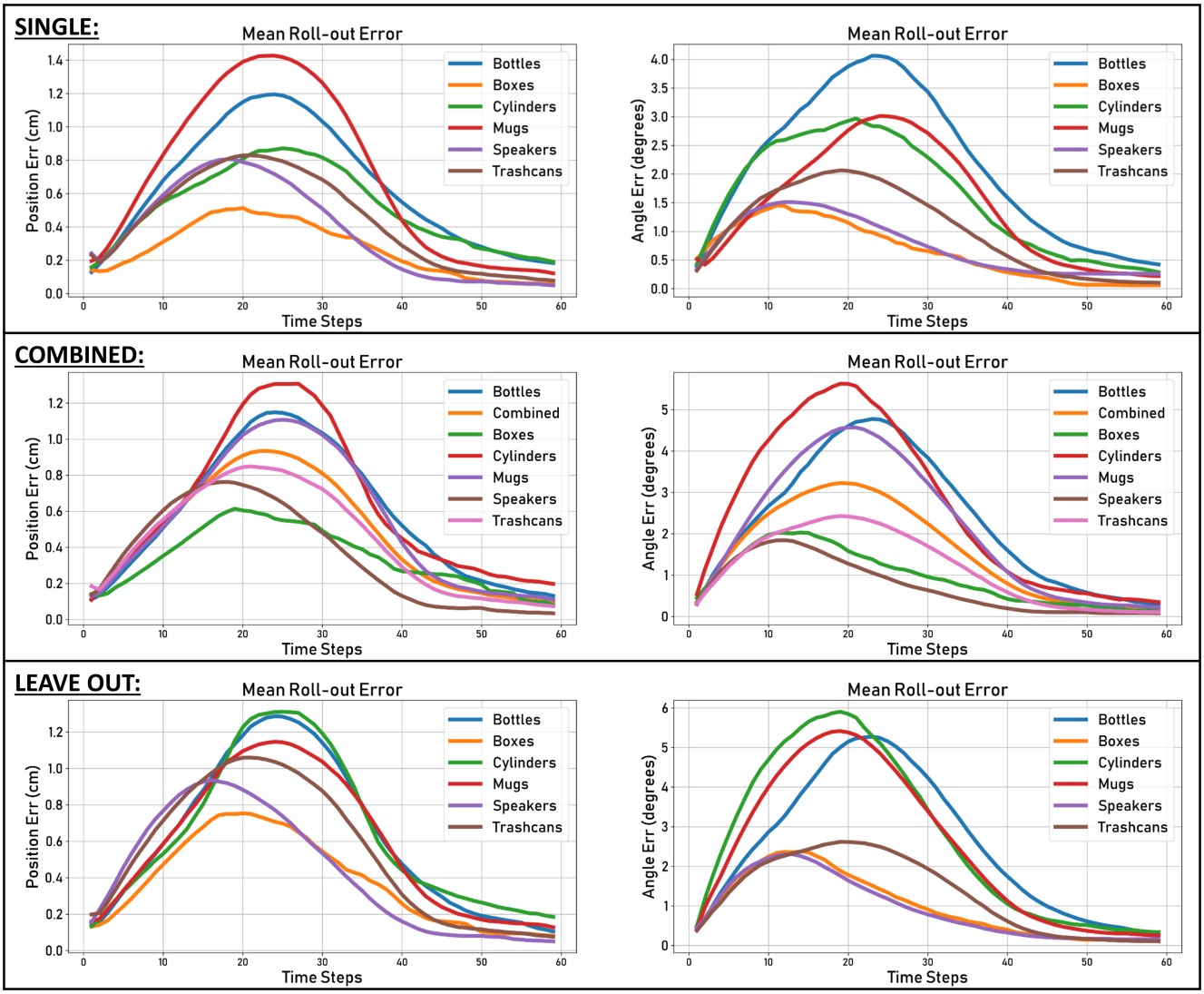

Figure 6. Object generalization performance on individual datasets. Mean roll-out errors for the *Single*, *Combined*, and *Leave Out* training procedures split by individual object category datasets.