One-shot Visual Imitation via Attributed Waypoints and Demonstration Augmentation - Supplementary Material

Matthew Chang[1] and Saurabh Gupta[1]

CONTENTS

[1]University of Illinois Urbana-Champaign, Illinois, USA. Emails: {mc48,saurabhg}@illinois.edu.
Project website with additional details: https://matthewchang.github.io/awda_site/

S1. Motor Primitives Details

For the experiments in this paper we utilize four motor primitives: free-space motion, carrying, dropping, and grasping. We normalize the action spaces for all environments to behave like the Meta-world environments. They have 4 degrees of freedom, 3-DOF delta end-effector position control, and 1-DOF for desired gripper closure. We implement 3-DOF delta end-effector position control by driving the joints to a configuration that achieves the desired end-effector position as obtained through inverse kinematics.

Free-space Motion: Given the current end-effector position and a destination, for each time step we move the end-effector directly towards the destination point in a straight line, using the abstracted delta end-effector position control space defined above.

Carrying: Carrying is implemented using the above free-space motor primitive, just keeping the gripper closed throughout the movement.

Dropping: Dropping is implemented by simply opening the gripper in place.

Grasping: We utilize depth images from an eye-in-hand camera for the grasping primitive. At a high level, the grasping primitive attempts to localize the nearest object to the gripper, position the gripper above that object, then drop down to perform the grasp. We first outline the steps of the motor primitive, and then describe the object localization procedure.

1) Determine the target object position using depth images (described below).
2) Move the gripper to a point 15cm above the estimated object position, recomputing the object position at each time-step.
3) After reaching a point above the object, move the gripper down to the estimated object z position offset by some small value (3cm for the panda gripper, but this should be adjusted based on the agent end effector).
4) Close the gripper.
5) Lift 15cm.

Next, we describe the object localization procedure to recover the target object position (step 1 above):

1) Mask out background values, defined as having a depth greater than 1 meter.
2) Determine the ground plane distance by taking the median of the remaining depth values.
3) Produce a mask for potential objects, taking pixels 1cm or more above the floor plane.
4) Compute connected components of the potential object mask to get individual object proposals.
5) Select the connected component with the centroid closest to the center of the frame as the closest object (target object for grasping).
6) Use the camera intrinsics to project the depth point of the centroid back into 3D world coordinates. This is the estimated object position.

A. Mining Attributes

Whether or not the attribute of *"is there an object in the gripper"* is present in at every timestep t is mined directly from the sequence of joint angles and commanded actions $\{(s_1, a_1)\dots\}$. We determine that an object is held when the commanded action is to close the gripper, but the gripper jaws do not close. At each timestep we annotate if it potentially contains a grasp with a variable g_t.

$$g_t = c_t \wedge (s^g_{t+d} - s^g_t > \delta)$$

Where c_t is a boolean indicating if the gripper was commanded to close on frame t, and s^g_t is the component of the robot at time t that corresponds to the distance between the gripper jaws. d and δ are tuned based on the timescale of the data, and dimensions of of the robot. For the Meta-world data, we use $d = 10$ and $\delta = -0.05$, for the T-OSVI data we use $d = 1$ and $\delta = -0.05$. For BC-Z, we use $d = 1$, $\delta = -0.07$. The final sequence of potential grasp frames $g_0, \dots, g_T$ is smoothed with a convolutional filter to give the final attribute annotations.

In addition to the grasping primitive used in our experiments, we have validated that this approach can work well for other attributes. We are able to detect a "pressing" attribute (*i.e.* button presses) using a similar methodology as for detecting object grasps. We identify frames in which the commanded action is to move the end-effector in a certain direction, but end-effector does not move, or accelerate in a direction consistent with the commanded action. This means there must be some object obstructing the end-effector, *i.e.* the end-effector is pressing into something. We visualize the results of this technique in Figure S2. We show the first pressing frame detected in trajectories from four different Meta-World tasks which require pressing (`coffee-button-v2`, `button-press-v2`, `handle-press-v2`, and `button-press-topdown-v2`). We find that this form of automatic mining is able to accurately identify frames in which the agent is performing button presses or other pressing-like actions.

S2. Implementation Details

A. Architecture

The architecture has three modules as shown in Figure S3: an convolutional image feature extractor, a transformer with self-attention layers for temporal processing, and a multi-headed MLP for waypoint prediction. These first two modules (feature extractor and transformer) are identical to the T-OSVI architecture. Our architecture differs only in the waypoint prediction heads. We notate the execution context observation as o_1, as only the first frame is used in our method. For the baselines, we use a frame-stack of 6 images $\{\dots, o_{t-1}, o_t\}$, following T-OSVI.

Image feature extraction: Visual features are extracted by a pre-trained ResNet-18 [2]. We share weights between the feature extractor for $\mathbf{v}$, and o_1, frames. We remove the last two layers of the ResNet and use the last spatial features.

Self-Attention Layers: We pass these spatial features into the self-attention module. We stack spatial features together along the time dimension, with o_1 (and any frame-stacking for non-waypoint baselines) placed after $\mathbf{v}$. We add sinusoidal

TABLE S1
DEFINITIONS OF PHRASES AND TERMS.

Term	Definition
Task	High level objective for the agent, *i.e.* open the window, or move the white block to the first bin
Novel Task	A task which does not appear in the training dataset
Task Instance	A unique configuration of objects in the scene in which a task must be performed
Task Context	The objects and elements in the scene in which a task is performed. One task context may admit many different tasks, i.e. the same blocks could be pushed, pulled, stacked etc.
Demonstration ($\mathbf{v}$)	A video containing only RGB images of a task being successfully performed. This may differ from the test time environment in agent embodiment, and will feature a different task instance.
Robot Trajectory $\{(o_1, s_1, a_1), \ldots\}$	A sequence of RGB images $\{o_1, \ldots, o_T\}$ along with robot states $\{s_1, \ldots, s_T\}$ (*e.g.* joint angles, joint velocities) and commands $\{a_1, \ldots, a_T\}$ (*e.g.* commanded torques, or end-effector destinations)
Attributed Waypoint	A point in $3 + k$ space indicating a point in $\mathcal{R}^3$ and the presence of up to k attributes. See main paper Section IV-A
AWDA	Attributed Waypoints and Demonstration Augmentation (our method)
ADM	Asymmetric Demonstration Mixup, see main paper Section IV-B
T-OSVI	Transformers for one-shot visual imitation [1]
TS	Trajectory Synthesis: Generating additional training samples featuring free-space motion, see main paper Section IV-B

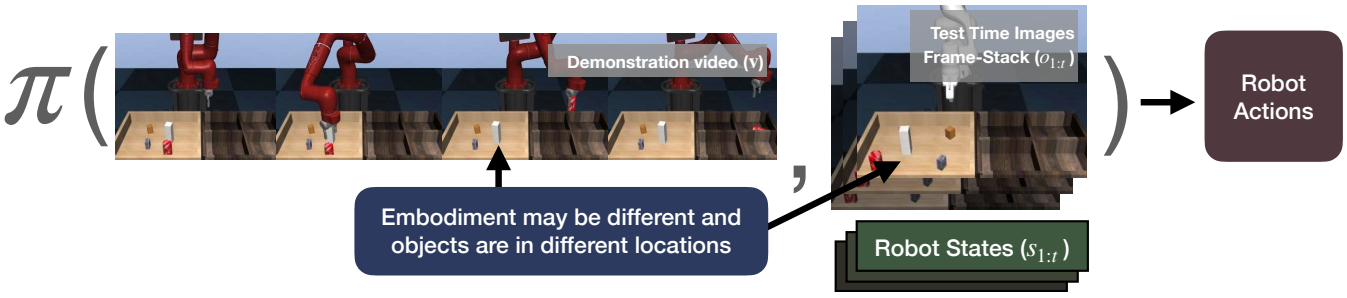

Fig. S1. **The one-shot visual imitation problem**: Given a video demonstration ($\mathbf{v}$) of an agent performing a novel task (task not seen during training), and the sequence of images for the current execution so far ($o_{1:t}$), the policy $\pi(a_t|\mathbf{v}, o_{1:t}, s_{1:t})$ must predict the next action to take to accomplish the task depicted in the video demonstration. Note that the agent embodiment, and the locations of the objects may differ between the video demonstration and the test time environment.

positional encodings to the features, treating time and space as a single dimension. We then pass these spatial features with positional encodings into a multi-headed self-attention module. We create key, query, and value tensors by applying 3D convolutions with kernel size 1. We then apply dropout, a residual connection, and batch normalization after each self-attention layer. Following T-OSVI, all methods use 4 attention heads and 2 layers of self-attention. Finally, after the attention layers, we apply a spatial softmax, followed by a two-layer MLP. This projects the features from the final time-step, corresponding to the o_1, into a fixed length vector.

Waypoint Prediction: We pass the final features (fixed-length vector corresponding to o_1 from above) through a multi-headed 2-layer MLP, *i.e.*, there is a separate layer to transform the features into waypoints for task-driven samples and synthesized samples. We select predictions from one head or the other based on the dataset of origin (task-driven or synthesized) of each trajectory in the batch.

B. Loss Computation

We linearly interpolate between the waypoints to produce a set of points amendable for use with Soft Dynamic Time Warping (SDTW) [39] with a fixed temperature parameter

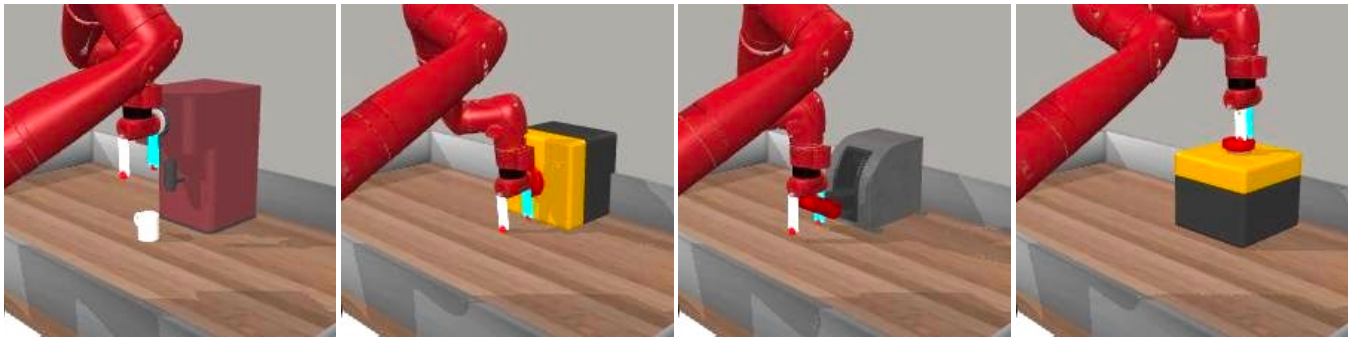

Fig. S2. The first frames in which a "pressing" attribute is detected in four trajectories from four different task in the Meta-World task-set, from left to right: `coffee-button-v2`, `button-press-v2`, `handle-press-v2`, and `button-press-topdown-v2`

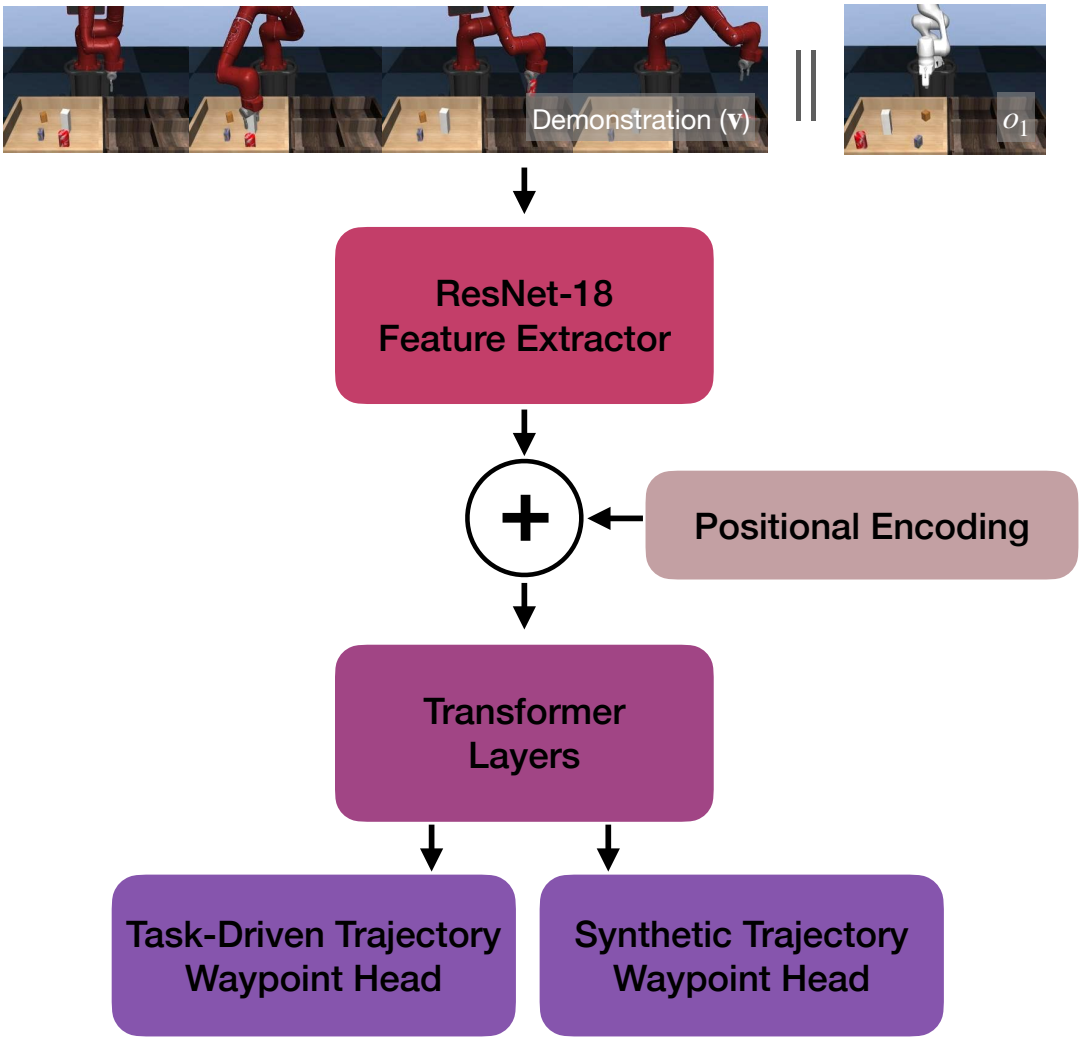

Fig. S3. Outline of our model architecture. Following T-OSVI [1], we extract features using ResNet-18. These features remain spacial, and have a sinusoidal positional encoding added, before being processed by a transformer module. Finally, the temporally processed features are projected down into waypoints by two separate heads, one to predict waypoints for task-driven trajectories, and one for trajectories synthesized as described in Section 4.2.

of 0.001. All experiments downstream are conducted using 5 waypoints. However, at training time we predict a total of 5 trajectories consisting of 1, 2, 3, 4, and 5 waypoints respectively, for a total of 15 waypoints. The SDTW loss against ground truth is computed independently for each of these 5 trajectories. The final loss is the mean across the computed SDTW distance for all 5 trajectories.

C. Inference Speed

When running our model on a single modern GPU (Nvidia A40), our model is able to make a forward pass through the network to predict waypoints in 7 ms. After waypoints are predicted, using the motor primitives to command the agent requires less than 2ms per environment step.

TABLE S2
HYPER-PARAMETERS FOR TRAINING AND EVALUATION

Hyperparameter	**Value**
Image size (Meta-world)	(224, 224)
Image size (Pick-and-place)	(240, 320)
Learning Rate	0.0005
Batch Size	30
Non-Local Attention Layers	2
Attention Heads per Layer	4
# of Waypoints for Evaluation	5
Evaluation Episodes per Snapshot per Task	20
γ (SDTW Temperature)	0.001
# of Demonstration ($\mathbf{v}$) frames	10
Optimizer	Adam
Post-Transformer Hidden Dimension	256

S3. DAGGER PROBLEM CONTINUED ANALYSIS

One may wonder about the role of action sampling and frame-stacking in our analysis of the DAgger problem in Section 3. We investigate this by evaluating the baseline (with frame-stack 6) by taking the mean action and mode action. Additionally, we train a model using no frame-stacking, relying only on the current frame for prediction. We find that none of these modifications address the failure mode studied in Section 3.

When training with no frame-stack we see the same poor performance, with this model achieving a reaching success of 0.12. We believe this is because, in any state where the recent frames dictate which object to reach, only the current frame is sufficient to make the same prediction, *i.e.*, the model can just look at which object the gripper has moved towards from the current frame. It doesn't need the past frames for this. Additionally, we find the model with no frame-stack achieves a significantly *lower* overall success rate than the baseline down to 0.01 from 0.1. This is because the frame-stack is useful for performing the grasp on the object. Demonstrations pause the griper above the object before grasping, and the model is unable to fit this behavior well without the previous frames: it pauses above the object and never descends to attempt a grasp. Using the mean or mode action did not resolve the reaching issue either, achieving a reaching success of 0.11 and 0.10 respectively.

S4. GRASPING FAILURE DETAILS

Grasping analysis results (as reported in Section 3, and above) are computed over 440 evaluations on the best snapshot from the action-prediction baseline T-OSVI. We consider the object to be successfully reached when the end effector comes within 4cm of the object and no grasp has yet been attempted in the current episode. A grasp is considered successful when the object has been reached, and then the object is raised at least 5cm off the ground.

As mentioned in Section 3, even when the right object is reached, grasping control in an end-to-end learned model with such limited data is often unreliable. In Figure S4 we visualize two of the typical grasping failure modes: missing the grasp, or acquiring an unstable grasp which drops the object.

S5. EXPERIMENTAL DETAILS AND RESULTS

In Table S3 we report mean success rates with standard error over the 10 evaluated snapshots of our method. In addition to methods reported in the main paper, we include **only waypoints**, which has neither additional data, nor our asymmetric demonstration mixup augmentation, and **T-OSVI +depth**, which is T-OSVI augmented to include the same eye-in-hand depth images that our method uses for grasping. We do this by adding a separate ResNet for depth image feature extraction, and perform late-fusion, combining RGB and depth image features into one vector before transformer layers. This performs significantly worse, likely because this adds more input and more capacity to over-fit, following our analysis in Section 3. In Table S4, we report the error bars for the MOSAIC results from main paper Table II. We follow the same procedure as other environments, reporting the mean and standard error of success rates over the last 10 snapshots from training. However, the results from MOSAIC [3] only report standard deviation over the last 3 snapshots from training. For fairer comparison we have computed the estimated standard error based on their reported standard deviations assuming normality of the underlying samples.

S6. VISUALIZATIONS

Figure S5 shows the waypoint predictions of our model on novel tasks. Notice that our method mimics the overall solution scheme of the demonstration while adapting to the different environment configuration in the current scene. We visualize trajectories from our trajectory synthesis method from Section 4.2, in Figure S6.

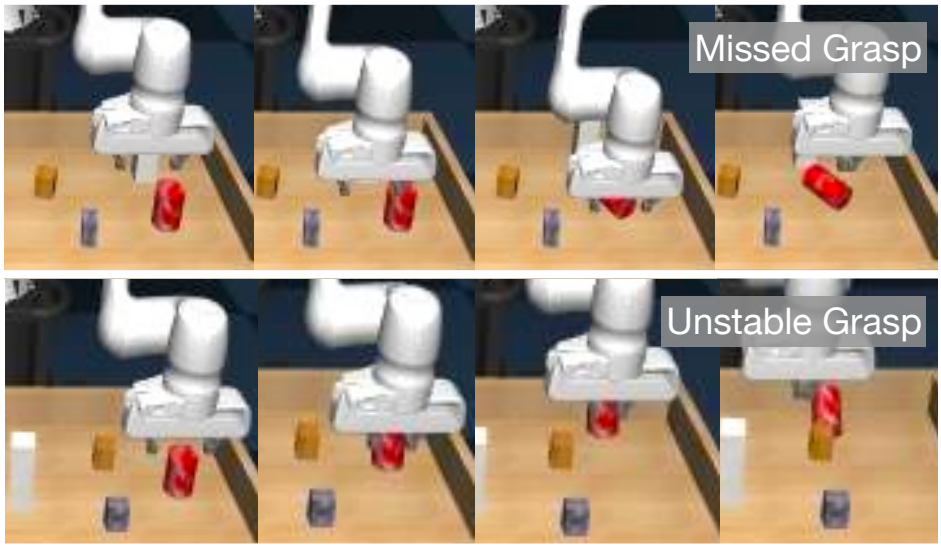

Fig. S4. The two most common grasping failure modes: missing the grasp entirely, and aquiring an unstable grasp.

TABLE S3

WE REPORT MEAN SUCCESS RATES AND STANDARD ERROR ON HELD-OUT TASKS ON THE PICK-AND-PLACE AND META-WORLD BENCHMARKS.

	DAML [4]	**T-OSVI [1]**	**T-OSVI +depth**	**Full-1**	**Full-2**	**single head**	**no AD**	**no ADM**	**no way points**	**only way points**
Asymm. Demo Mixup (ADM)?				✓	✓	✓	✓	✗	✓	✗
Additional Data (AD) Source?				TS	BC-z	TS	✗	TS	TS	✗
Waypoints?				✓	✓	✓	✓	✓	✗	✓
Pick-and-place	$.01 \pm .00$	$.10 \pm .05$	$.00 \pm .00$	$1.0 \pm .00$	$1.0 \pm .00$	$.99 \pm .00$	$1.0 \pm .00$	$.98 \pm .01$	$.01 \pm .02$	$.98 \pm .01$
Meta-world [easy]	$.04 \pm .01$	$.50 \pm .08$	$.02 \pm .01$	$.66 \pm .05$	$.73 \pm .08$	$.33 \pm .06$	$.74 \pm .05$	$.03 \pm .01$	$.33 \pm .10$	$.14 \pm .08$
Meta-world [hard]	$.08 \pm .03$	$.07 \pm .03$	$.02 \pm .01$	$.30 \pm .04$	$.17 \pm .04$	$.29 \pm .03$	$.11 \pm .04$	$.19 \pm .06$	$.06 \pm .02$	$.02 \pm .01$
Meta-world [all]	$.06 \pm .01$	$.28 \pm .06$	$.02 \pm .01$	$.48 \pm .05$	$.45 \pm .08$	$.31 \pm .03$	$.42 \pm .08$	$.11 \pm .04$	$.19 \pm .06$	$.08 \pm .04$

TABLE S4

SUCCESS RATES ON HELD-OUT TASKS FROM MOSAIC [3] TASK-SET

Task	Door	Drawer	Press Button	Stack Block	Basketball	Nut Assembly	All
MOSAIC [3][1]	0.05 ± 0.018	0.15 ± 0.038	$\mathbf{0.05} \pm 0.018$	0	0	0	0.04 ± 0.008
Full-1	$\mathbf{0.10} \pm 0.02$	$\mathbf{0.29} \pm 0.033$	0.01 ± 0.005	0	0	$\mathbf{0.02} \pm 0.01$	$\mathbf{0.07} \pm 0.007$

[1] The results from MOSAIC [3] only report standard deviation over the last 3 snapshots from training, where we report standar error over the last 10 snapshots for a lower variance estimate. For fairer comparison we have computed the estimated standard error based on their reported standard deviations assuming normality of the underlying samples.

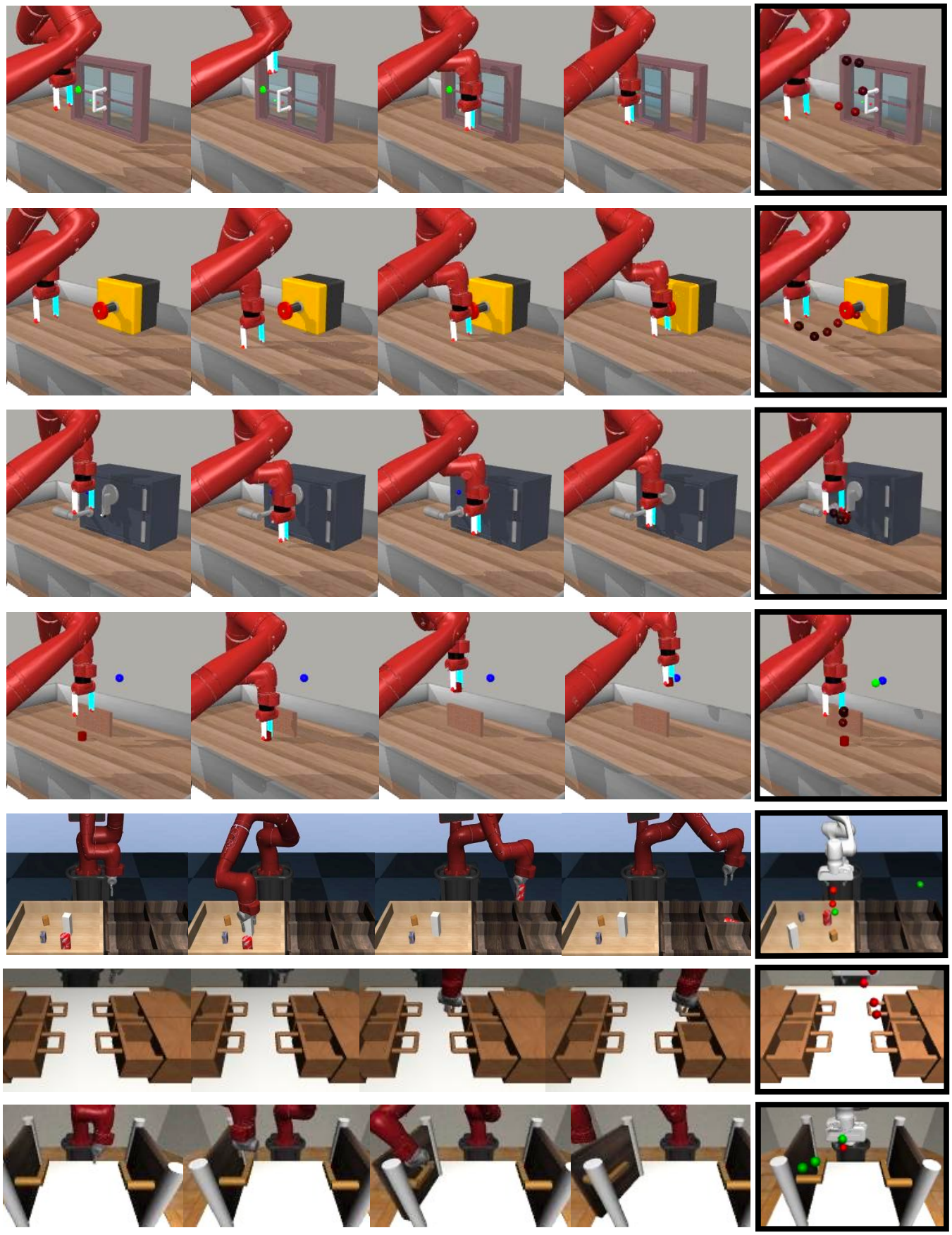

Fig. S5. We visualize example waypoint predictions on held-out tasks on Metaworld, Pick-and-place, and MOSAIC. Each row is a different novel task that the model has not seen during training. The frames on the left represent the video demonstration, and the frame on the right shows the waypoints predicted in a new setting. Red spheres represent free-space waypoints. The increasing order of waypoints is indicated by a brightening of the color. Green spheres indicate the end point of line segments attributed with the *object grasped* attribute. This means the grasping primitive should be invoked at the previous waypoint if it is red, and the object should be carried to the green waypoint.

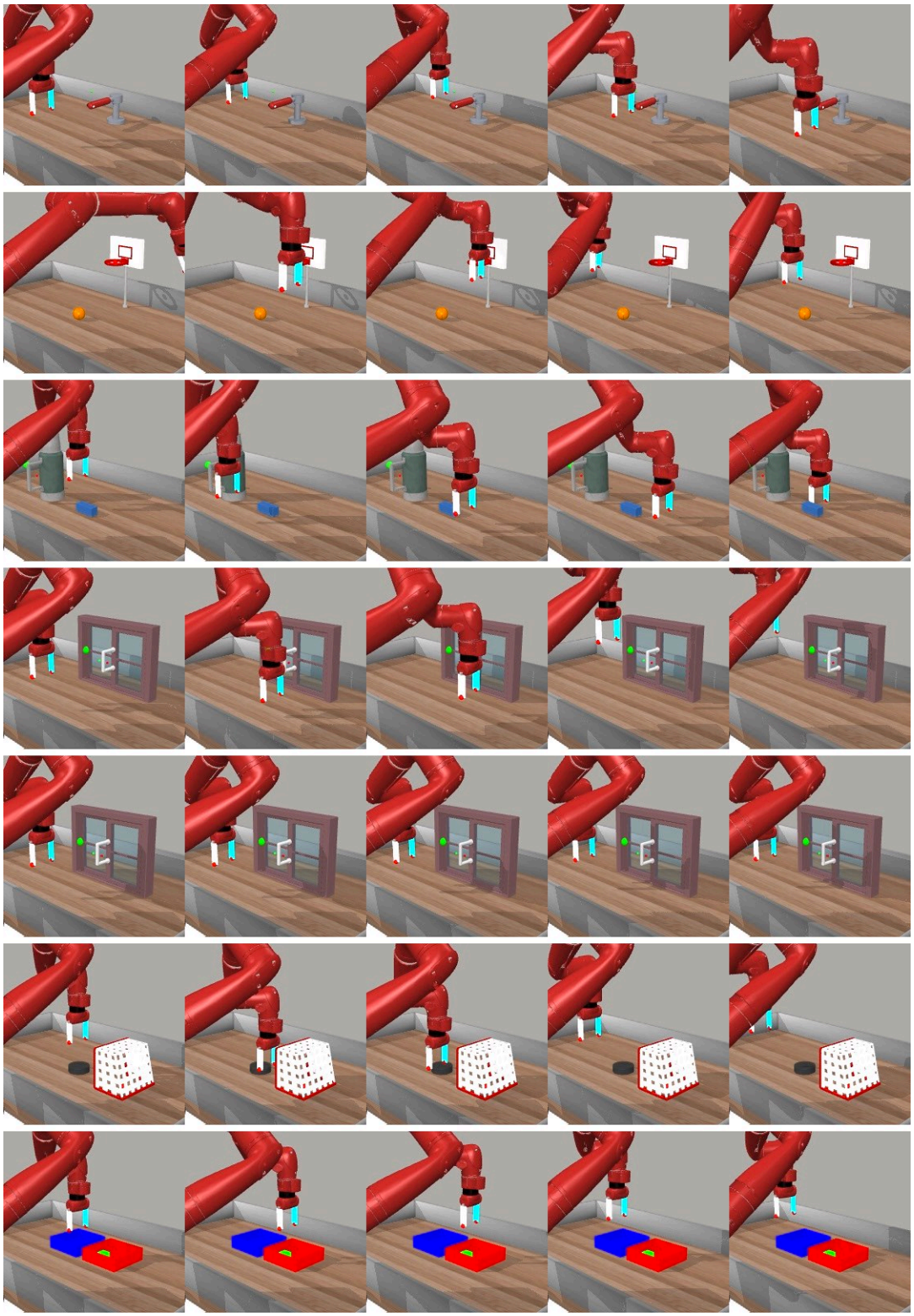

Fig. S6. We visualize example trajectories generated by our proposed trajectory synthesis method (Section 4.2). Trajectories are synthesized by driving the end-effector through randomly sampled waypoints in the environments used for training. Introducing this data into the training dataset helps decorrelate tasks from task contexts.

REFERENCES

[1] S. Dasari and A. Gupta, "Transformers for one-shot visual imitation," in *Proceedings of the Conference on Robot Learning (CoRL)*, 2020.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[3] Z. Mandi, F. Liu, K. Lee, and P. Abbeel, "Towards more generalizable one-shot visual imitation learning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2022.

[4] T. Yu, C. Finn, A. Xie, S. Dasari, T. Zhang, P. Abbeel, and S. Levine, "One-shot imitation from observing humans via domain-adaptive meta-learning," *arXiv preprint arXiv:1802.01557*, 2018.