Supplementary Material
AGENT: A Benchmark for Core Psychological Reasoning

Tianmin Shu[1] Abhishek Bhandwaldar[2] Chuang Gan[2] Kevin A. Smith[1] Shari Liu[1] Dan Gutfreund[2]
Elizabeth Spelke[3] Joshua B. Tenenbaum[1] Tomer D. Ullman[3]

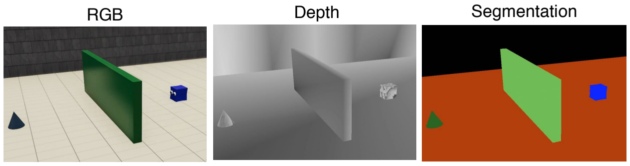


Figure 1. Example RGB frame, depth map, instance segmentation map provided in AGENT.

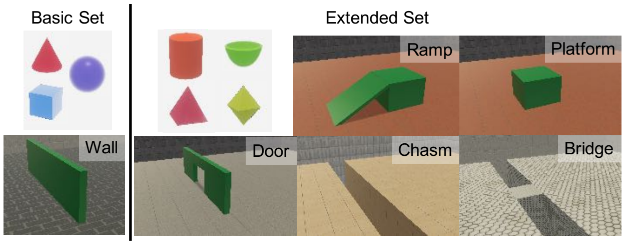


Figure 2. The basic set and the extended set of object shapes and obstacles in AGENT.

A. Dataset Details

A.1. Multi-modal Input

In additional to the ground-truth states and object labels, we also provide RGB frames, depth maps, and instance segmentation maps for all videos in AGENT as shown in Figure 1.

A.2. Statistics

There are two sets of the object shapes and obstacles in AGENT – a basic and an extended set (as depicted in Figure 2). In AGENT, there are 1100 trials generated with the basic set, and the remaining ones are generated with the extended. This split enables us to conduct evaluation of generalization to novel shapes and obstacles unseen in training (see Section D.2).

Videos in AGENT are rendered with 3 background wall textures and 7 floor textures (as shown in Figure 3).

Table 1 summarizes the number of trials in each type in AGENT for the training, validation, and testing sets.

B. Model Implementation Details

B.1. BIPaCK

Planner. We devise an RRT*-based planner, which first searches for the most efficient path from the initial position to the target position, then computes the force needed to move the agent along the path. For computing the cost between two positions, s_a, s'_a, we first compute the force needed to move from s_a, s'_a. When s_a and s'_a are on the same surface (on the ground, a ramp, or a platform), we compute the force needed to move in the direction from s_a to s'_a at a constant speed. In practice, we set the constant speed to be the average speed of agents in the training trials. When moving from s_a to s'_a requires an elevation in the vertical direction, such as jumping over a wall or a chasm, or jump onto a platform, we derive the minimum vertical force needed to reach s'_a without colliding into the wall / platform, or fall into a chasm, while maintaining a constant speed for the horizontal motion $v_h^{\text{jump}} = 1.3$. We define a collision check condition for the expansion in RRT*, as illustrated in Figure 4, so that it is allowed to go from one side of a wall or chasm and reach to the other side, or go from the ground and land onto a platform, but it is not allowed to land onto a wall or in a chasm.

Physics parameters. For the physics parameters, we consider coordinate transformation, gravity, friction, densities of entities, and time unit (how long a step in the video correspond to one simulation step in PyBullet). Most of these parameters have little effect on the final accuracy as the remaining parameters can compensate their effects (e.g., the large gravity could be offset by a lower cost weight for the vertical forces), so we set them to be constant in all scenarios. In particular, we set the gravity to be 9.81, densities

[1] Massachusetts Institute of Technology [2] MIT-IBM Watson AI Lab [3] Harvard University. Correspondence to: Tianmin Shu <tshu@mit.edu>.

Table 1. The number of trials for each type in AGENT.

Set	Goal Preferences					Action Efficiency						Unobs.			Cost-Reward			All
	1.1	1.2	1.3	1.4	All	2.1	2.2	2.3	2.4	2.5	All	3.1	3.2	All	4.1	4.2	All	
Train	120	160	120	160	560	80	80	80	160	80	480	160	240	400	240	240	480	1920
Val	30	40	30	40	140	20	20	20	40	20	120	40	60	100	60	60	120	480
Test	60	80	60	80	280	40	40	40	80	40	240	80	120	200	120	120	240	960

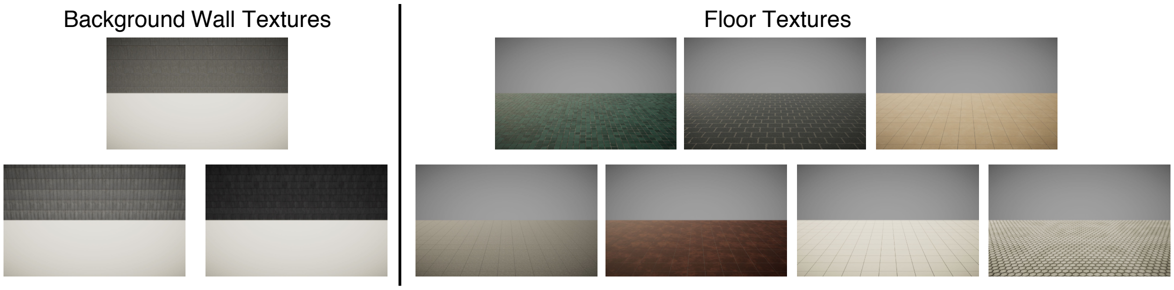


Figure 3. Background wall and floor textures in AGENT.

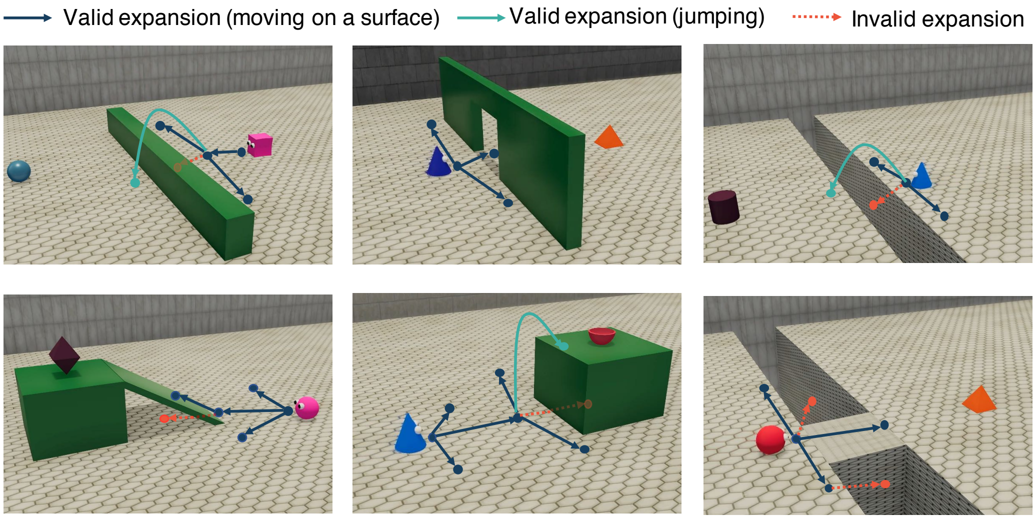

Valid expansion (moving on a surface)
Valid expansion (jumping)
Invalid expansion

Figure 4. Illustration of valid and invalid expansions in our RRT*-based planner.

of all moveable entities to be 1 (note that we also assume that obstacle blocks are always immobile), and time unit to be 5 ms per simulation step. In addition, the 3D states in the original videos have a different coordinate system than the one in PyBullet. For this, we have to determine which axis in the original coordinate system corresponds to the vertical axis in PyBullet (the mapping of the two horizontal axes do not matter in our approach). We achieve this by setting the axis which has the least amount of change in the agents' trajectories (since they mainly travel horizontally). Finally, we need to search for the friction from a fixed range $\{0, 0.05, 0.1, 0.15, 0.2\}$ (we define the friction as a constant force applied to an agent moving on any surface in PyBullet). In our experiments, a friction between 0 and 0.1 result in a similar performance.

Agent parameters. As discussed in the main paper, we search for the best reward and cost function for an agent. In this work, we set the cost function to be $C(s_a, s'_a) = \mathbf{w}^\top \mathbf{f} = w_h f_h + w_v f_v$, where f_h and f_v are the horizontal and vertical force the agent needs to move from s_a to s'_a. We always set $w_h = 1$, and consider a finite set for w_v, i.e., $\{0.1, 0.2, 0.3, 0.5, 1.0, 10.0\}$. We assume a continuous range for the rewards of goal objects, i.e., $r_g \in (0, 100.0]$, $\forall g \in \mathcal{G}$.

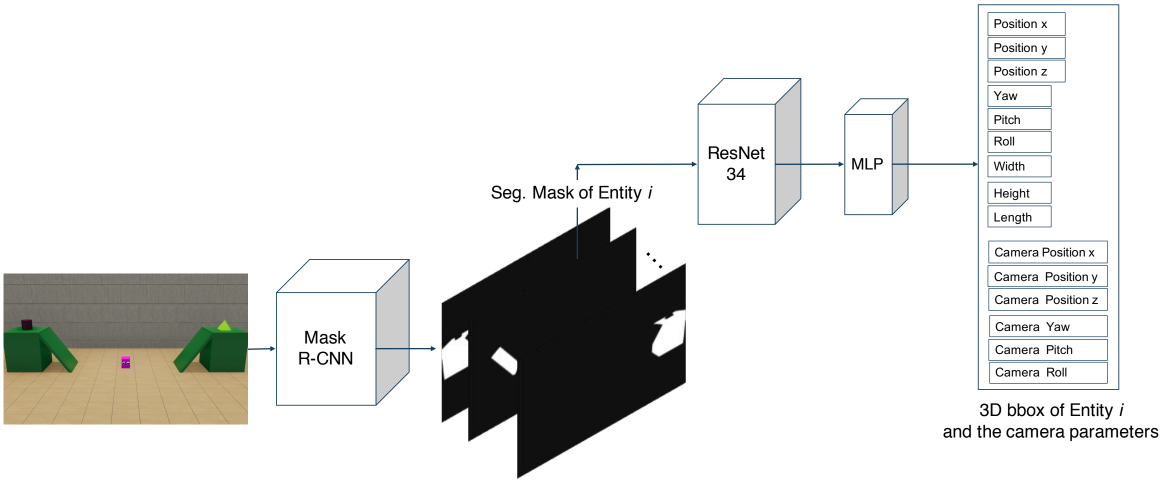


Figure 5. Network architecture of the derender.

Sampling. For each test video, we sample 10 trajectories to compute score. Using 10 parallel processes, on a 16-core CPU with 64 GB RAM, it takes 1-3 s to compute the surprising score for a test video.

B.2. ToMnet-G

Network architecture. Each GNN consists of N nodes; each node has an input, $v_i = (x_i, a_i)$, where x_i is the center of the 3D bounding box x_i, and a_i is the appearance information, including the object label, the width, length, and height of the 3D bounding box, and the orientation of the bounding box. The node input is encoded by $\phi_v(v_i) = [\phi_v^x(x_i), \phi_v^a(a_i)]$, where both ϕ_v^x and ϕ_v^a are a 32-dim fully-connected (FC) layer. The edge between node i and j is encoded by a 64-dim FC layer, $\phi_e(v_i, v_j)$. For the agent node, we connect all other entities to it and aggregate the edge embeddings by a sum-pool. Concatenated with the embedding of the agent node itself, we get $[\phi_v(v_i), \sum_{j\neq i} \phi_e(v_i, v_j)]$, which is passed to a 64-dim FC layer to get the final agent node embedding $\phi_{\text{agent}}([\phi_v(v_i), \sum_{j\neq i} \phi_e(v_i, v_j)])$. For a familiarization video k, this is then passed to an LSTM with 64 hidden units to get a encoding of the video e_{fam}^k (i.e., the last latent state of the LSTM). By a sum-pool over the encoding of all familiarization videos, we get $e_{\text{char}} = \sum_k e_{\text{fam}}^k$. For a test video, the agent node embedding is concatenated with e_{char} and passed to two 64-dim FC layers, followed by an LSTM with 64 hidden units. The hidden state of this LSTM becomes e_{mental}. Concatenating e_{mental} and e_{char}, we predict the displacement of the agent in one step $\delta\mathbf{x}$ with an FC layer. $\delta\mathbf{x}$ is then used to update x_i in the agent node's input. We predict the movement of the agent until the end of the test video.

Training. We train the network using Adam (Kingma & Ba, 2014), with a batch size of 16, and a learning rate of 0.001.

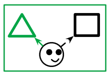
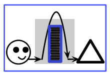
C. Derendering for Visual Perception

In order to test the performance of baseline models that only have access to the videos, we introduce a visual perception front-end. This derenderer model extracts the 3D states of each entity in a video over time, which are then used as the inputs to BIPaCK and ToMnet-G.

Network architecture. Figure 5 shows the architecture of the derender, which first obtains instance segmentation from a video frame using Mask R-CNN (He et al., 2017), and then recognizes the 3D bounding of each entity and the camera parameters based the mask of each entity, using a ResNet-34 (He et al., 2016) and a two fully-connected layers (which are 256-dim and 15-dim respectively). In particular, we have 11 object labels for the instance segmentation: agent, obstacle, 8 goal object color codes (each represents a unique object identity defined by its color code), and occluder. This design ensures that we can reconstruct the rough 3D scene without overfitting to certain object shapes in the training data, and the approximated scenes should be sufficient for the downstream psychological reasoning tasks.

Training. We adopt a two-phase training procedure: we first finetune Mask R-CNN (which is pretrained on ImageNet; Deng et al. 2009) using the ground-truth segmentation masks, and then train the remaining part based on segmentation masks from the fixed Mask R-CNN. We sample 36,000 frames from the training trials of all types as the training set, and 14,000 frames from the validation trials as the validation set. For both training stages, we use Adam with a learning rate of 0.000067, and a batch size of 160.

Table 2. Model performance based on derendering results. ‘All’ indicates performance of models trained on all concepts.

Condition	Method	Goal Preferences					Action Efficiency						Unobs.			Cost-Reward			All
																			
		1.1	1.2	1.3	1.4	All	2.1	2.2	2.3	2.4	2.5	All	3.1	3.2	All	4.1	4.2	All	
All	ToMnet-G	.53	.56	.47	.56	.54	.45	.70	.45	.73	.15	.52	.55	.35	.43	.50	.57	.53	.51
	BIPaCK	.77	.73	.80	.65	.73	.60	.65	.85	.50	.70	.63	.45	.62	.55	.63	.75	.69	.65

	Goal Preferences					Action Efficiency						Unobs.			Cost-Reward			All
	1.1	1.2	1.3	1.4	All	2.1	2.2	2.3	2.4	2.5	All	3.1	3.2	All	4.1	4.2	All	
SD	.05	.05	.11	.03	.06	.08	.05	.09	.06	.11	.09	.10	.06	.08	.09	.10	.10	.08

D. Additional Results

D.1. The Variance of Human Performance

We report the standard deviations of human performance as shown in Table D.1. The results suggest that the variance of the human performance is relatively small and thus validates the data as well as the experimental procedure.

D.2. Generalization to Unseen Shapes and Obstacles

In addition to the four generalization tests, we also evaluate the models’ performance on trials generated with the extended set shown in Figure 2 when only trained on trials synthesized from the basic set. Even when we provide the ground-truth bounding boxes of all entities, ToMnet-G’s averaged accuracy still drops to 0.57, whereas BIPaCK-achieves an accuracy of 0.95. Although ToMnet-G did not have the opportunity to encode these objects during its training (as opposed to BIPaCK, for which the encoding into simple shapes is fixed), this nonetheless highlights the need for generalizable object representations when faced with novel physical environments.

D.3. Results Based on Derendering

The averaged IoU (Intersection over Union) between the 3D bounding boxes generated by the derender and the ground-truth is 0.07 (SD=0.11). This low IoU is mainly caused by inaccurate center position and orientation estimations. Given the noisy derendering results, we evaluate the ToMnet-G and BIPaCK trained on all types and all scenarios, as reported in Table 2. Without the ground-truth 3D states, the performance of both models drops significantly. However, more advanced derenderers with better position and orientation estimates could ameliorate this drop.

D.4. Qualitative Results

We visualize some typical failures examples of both ToMnet-G and BIPaCK in different generalization tests in Figure 6, where we show the predicted agent trajectories and the ground-truth agent trajectories in the expected test videos, indicating failure modes of both models. Note that both models have access to the ground-truth 3D states of the objects and obstacles.

ToMnet-G: violations of physics. While ToMnet-G can predict that an agent will travel towards its goal as efficiently as possible, it can ignore constraints and predict that the object will take non-physical paths (e.g., passing through solid obstacles; Figure 6ABDH).

ToMnet-G: inefficient paths. ToMnet-G sometimes predicts inefficient paths (e.g., jumping unnecessarily when the obstacle is out of the way; Figure 6G).

ToMnet-G: incorrect goal selections. ToMnet-G can make erroneous goal predictions, e.g., selecting no clear goal for the agent (Figure 6CE), or predicting the wrong goal (Figure 6DF).

BIPaCK: mis-estimates of cost. In some situations, BIPaCK will incorrectly estimate the costs of moving horizontally vs. jumping, and thus will, for instance, select a longer path around an obstacle rather than a short hop over it (Figure 6H).

References

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

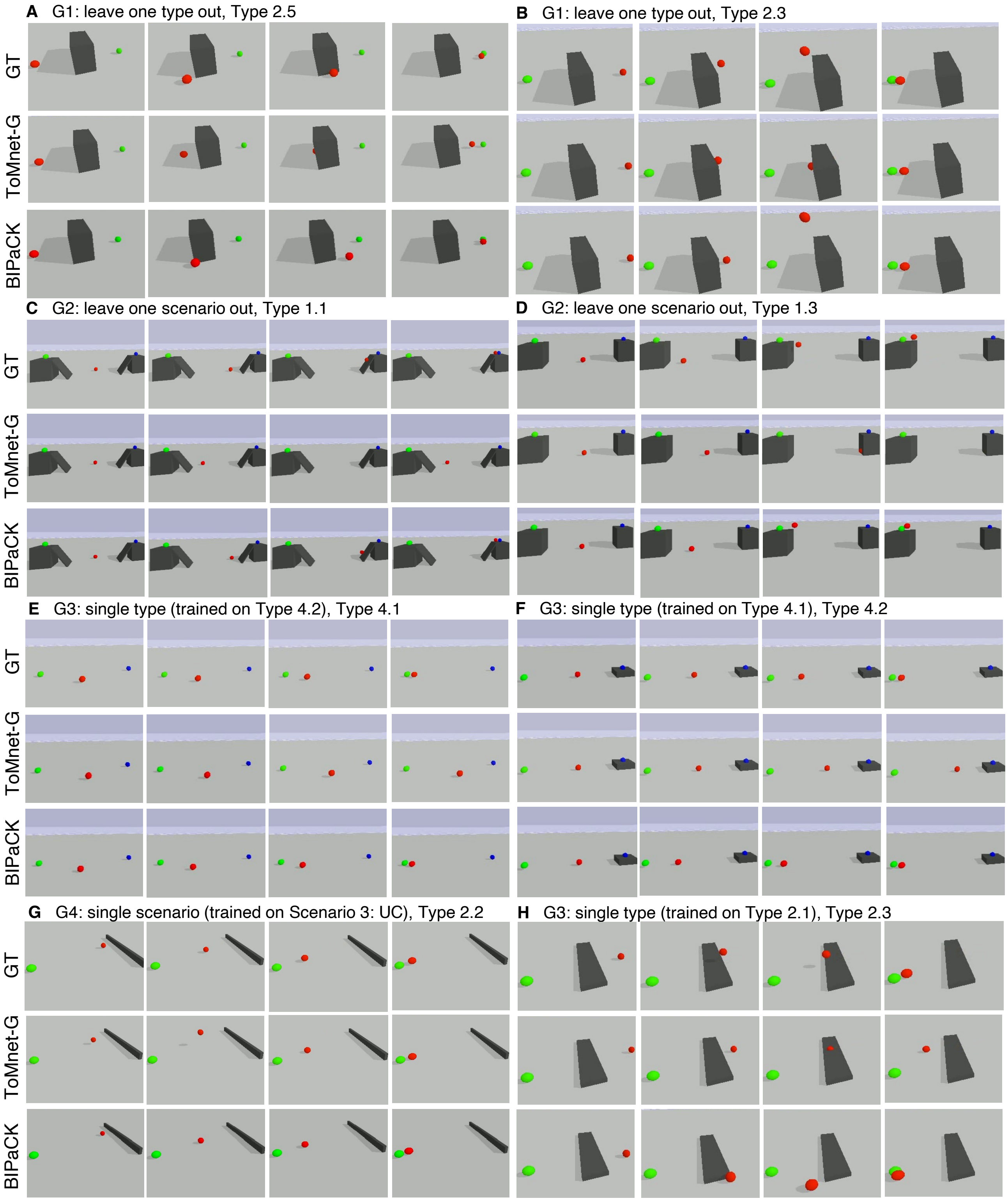

A G1: leave one type out, Type 2.5
B G1: leave one type out, Type 2.3
C G2: leave one scenario out, Type 1.1
D G2: leave one scenario out, Type 1.3
E G3: single type (trained on Type 4.2), Type 4.1
F G3: single type (trained on Type 4.1), Type 4.2
G G4: single scenario (trained on Scenario 3: UC), Type 2.2
H G3: single type (trained on Type 2.1), Type 2.3
GT
ToMnet-G
BIPaCK

Figure 6. Qualitative results of failure examples in different generalization tests. Here we show approximated scenes where objects and agents are represented by spheres and the obstacles are recreated with cubes. The agent is always represented by the red sphere. All examples are the models' prediction or the ground-truth (GT) agent behaviors in the expected test videos.