A APPENDIX

A.1 DATA AND MUJUCO ENVIRONMENT

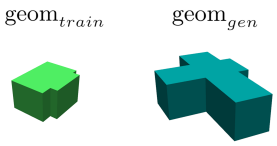

Figure A.1.1: Example of object geometries used for training and generalization.

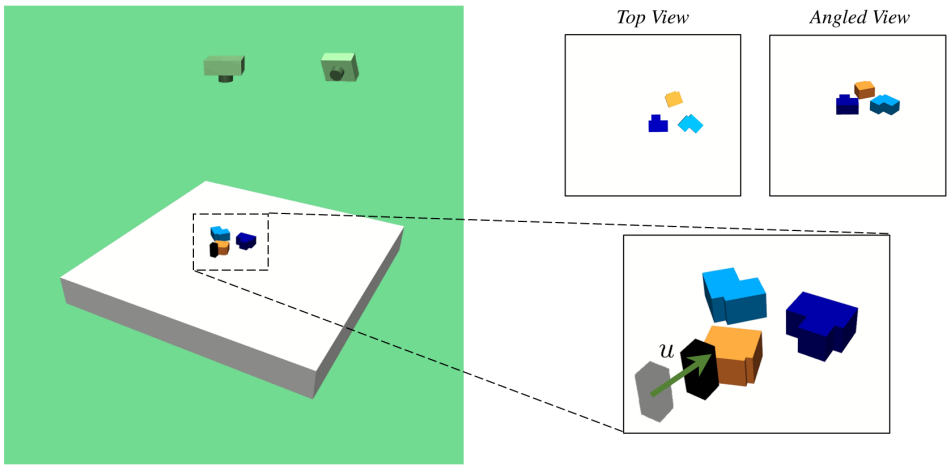

Figure A.1.2: Experiment setup details in MuJoCo.

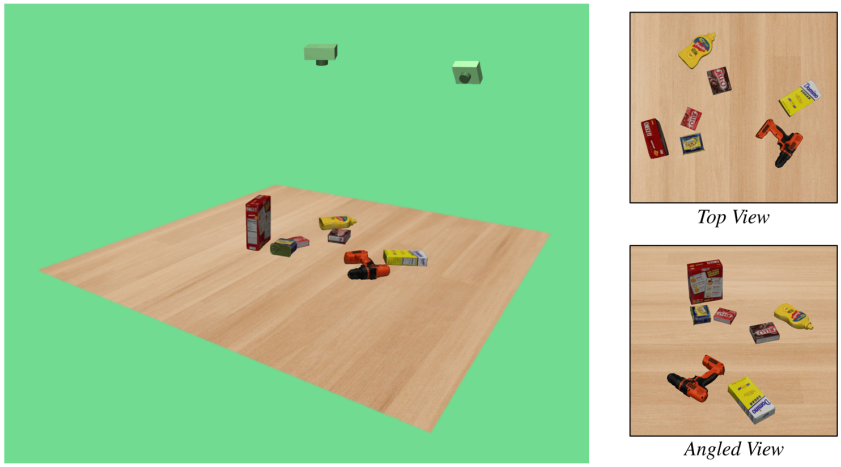

Figure A.1.3: Experiment setup details in MuJoCo.

BlockObjs. We simulate robotic pushing manipulation of $N \in \{1, \ldots, 5\}$ objects in a 3D environment. To generate the simulated dataset, we construct each object with two cuboid-shaped geoms in MuJuCo. We initialize each simulation episode by randomizing the geom size, color, and pose. We define the training geom dimensions geom_{train} with a width in range $[0.02, 0.04]$ and a length in

range $[0.02, 0.06]$. We also generate a dataset with out-of-distribution geometries geom_{gen} for generalization experiments with length in range $[0.06, 0.18]$. All geom heights are set to 0.03. Figure A.1.1 shows examples of these objects.

YCBObjs. To create a dataset with challenging and more realistic objects we use a subset of YCB objects with diverse size, mass, texture, geometry. We simulate pushing manipulation of $N \in \{1, \ldots, 5\}$ YCB objects in a 3D environment (see Figure A.1.3). Each manipulation episode starts by randomizing the number of objects in the scene and the initial pose of the object that includes position and orientation of the objects.

In both *BlockObjs* and *YCBObjs* simulation environments , A single step point-to-point pushing actions is randomly generated in range $[0.01, 0.05]$ with a random initial position in proximity of the objects $[0.01, 0.03]$ towards a uniformly sampled direction $[0, 2\pi]$. We deliberately sampled the action initial position near objects to increase the likelihood of interaction. An additional cuboid replicates the robot's end effector in the environment (Fig A.1.2, black cuboid). The action vector u is applied to this end effector as a displacement at a constant speed. Two cameras are added to the environment (Fig A.1.2 and Fig A.1.3, *Top* and *Angled View*) to capture an image of the scene before and after the action is applied. The end effector is lifted and removed before capturing the images.

A.2 Learned Graph Representation

We analyze the learned graph representation in our model to demonstrate the effectiveness of our approach for learning meaningful object-centric representation of the system. To do so, we compute 2D t-SNE embeddings of the learned node features in our model $\{\tilde{\mathbf{n}}^k_{0:t}\}$. The learned graph representation is demonstrated for *Top View* and *Angled View* observations in Figure A.2.1. In these examples, the learned node representation of keypoints that belong to the same object form distinct clusters. This further indicates that keypoint factorization of the scene captures a rich object-centric representation.

Additionally, we show examples of the inferred graph adjacency matrix $\tilde{\mathbf{A}}_t$ (see Fig. A.2.1). Although it is not trivial how the structure of the scene is reflected in the graph connectivity, our generalization results shows that a probabilistic graph representation enhances the message passing in the model and results in better generalize to unseen geometries. Note that the adjacency matrix in our model is probabilistic but here we only show the edges with $p \geq 0.5$.

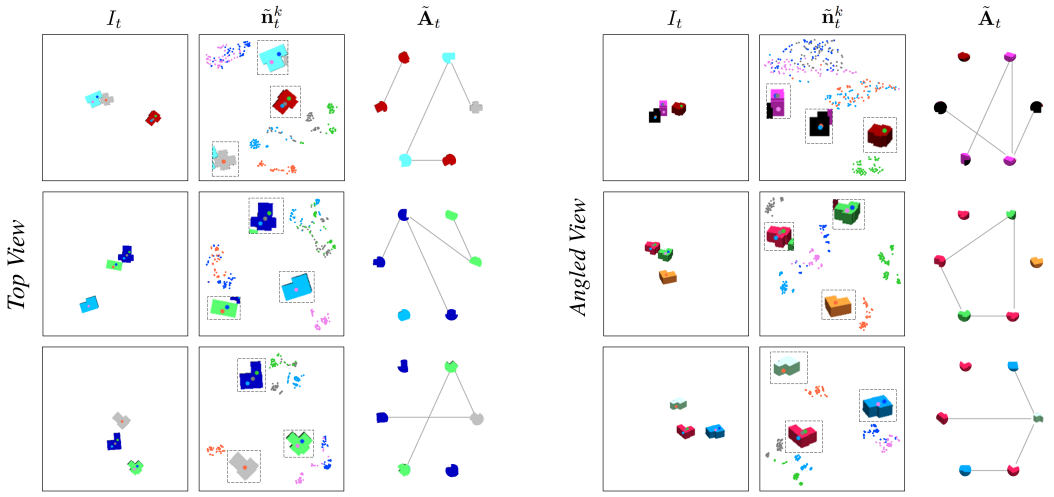

Figure A.2.1: Graph representations learned from *Top* and *Angled View* observations.

A.3 EFFECT OF VARYING THE NUMBER OF KEYPOINTS

We show the effect of varying the number of keypoint on the performance of the KINet for the *YCBObjs* dataset. We separately trained six variations of KINet for 2 epochs by changing the number of keypoints ($K = 3, 6, 9, 12, 15, 18$). All other hyperparameters except for the number of keypoints are the same for all models. We measured the prediction accuracy for 3 YCB objects and noticed the performance is best with $K = 12$. The model is least accurate with 3 keypoints which is expected since the object geometry and textures are complicated to capture with only three keypoints. For $K = 15, 18$ we also noticed a performance drop which indicates other hyperparameters such as the keypoint feature map dimensions need to be tuned accordingly for optimal performance of the model.

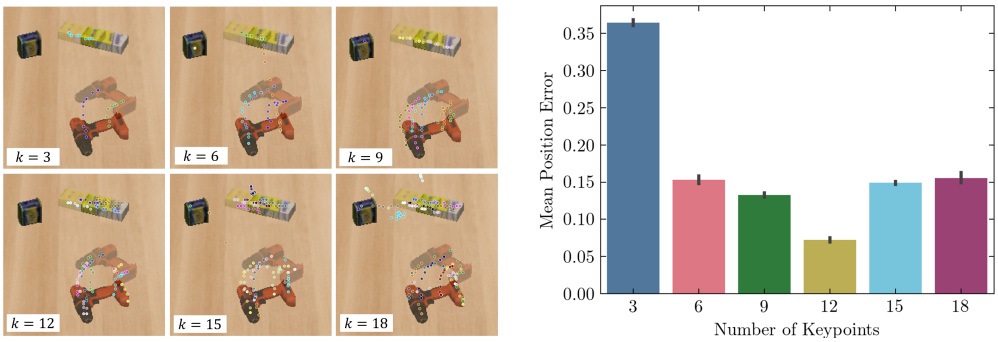

Figure A.3.1: Results for varying the number of keypoints in the *YCBObjs*.

A.4 REAL-ROBOT EXAMPLES

We include more examples of testing our approach on real-robot dataset from Yan et al. (2020). This dataset is collected by performing random point-to-point pushing actions on 7 simple blocks using a Sawyer robot. Specifically, the dataset includes RGB image pairs of the scene before and after applying the action along with the action vector in the image space. Our model factorizes the observation into a keypoints and accurately predicts the future keypoint locations conditioned on an external action. It then uses the predicted keypoint locations to reconstruct the future appearance of the scene. These qualitative examples shows that our model learns to predict the effect of action on objects as well as object to object interactions.

A.5 DETAILED GENERALIZATION MPC RESULTS

We include more detailed qualitative results of the MPC planning steps using our model for *Top View* and *Angled View* (Fig A.5.1) observations. In particular, we use the KINet model trained on $N = 3$ objects in our proposed GraphMPC algorithm to bring the scene into a goal configuration. We set the planning horizon to $T = 80$. Examples shown here are zero-shot generalization cases to a different number of objects and out-of-distribution geometries. Our model, trained on $N = 3$ objects, is able to repurpose the learned forward model and generalize to these unseen circumstances. Since our model learns to perform forward modeling in the keypoint space, with zero-shot generalization, it reassigns the expected keypoints ($K = 6$) to unseen objects and then makes forward predictions. These examples also show how GraphMPC, unlike conventional MPC only with respect to positional states, accurately brings the system to a goal state both explicitly (i.e, position) and implicitly (i.e, pose, orientation, and visual appearance).

We demonstrate detailed MPC results for zero-shot generalization to randomized background textures. The model, trained on a fixed white background, successfully generalizes to unseen randomized backgrounds. Figure A.5.2) shows qualitative results of the control task performed in the unseen background textures. Since the keypoint extraction step relies on visual features of salient objects, our model successfully performs the control tasks by ignoring the background and appropriately assigning keypoints to the objects. In extreme out-of-distribution cases such as the dirt texture (Fig A.5.2, last row), a few keypoints are assigned to specific parts of the background. We speculate that this is because the unseen texture, unlike the training set, has rich visual features (see Fig. A.6.2 for failed cases).

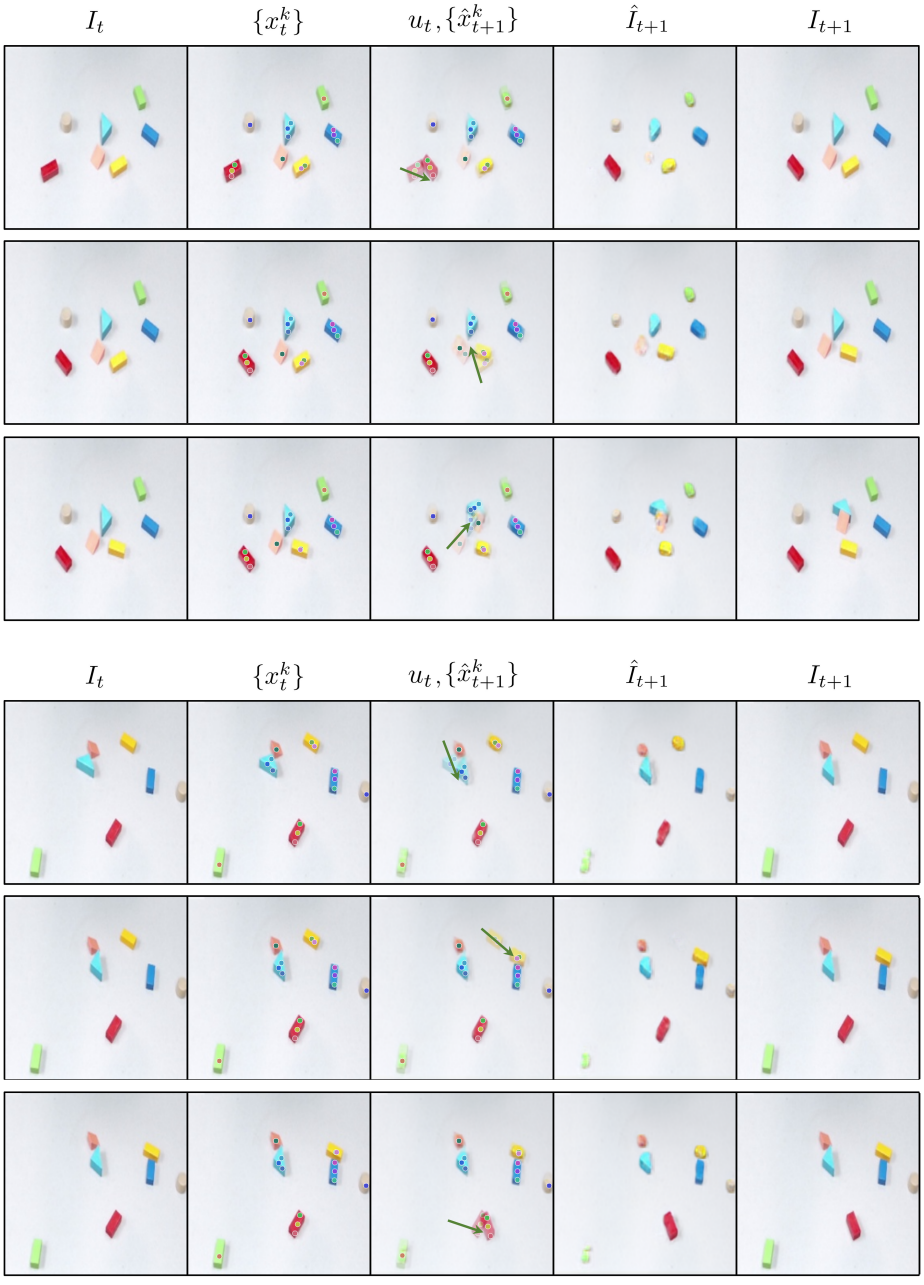

Figure A.4.1: Qualitative examples of real robot dataset (Ye et al., 2020). Given an image (I_t), our model factorizes the scene into keypoints (x_t^k) and conditioned on the action (u_t, green arrows) estimates the next keypoint coordinates ($\hat{x}_{t+1}^k$) and appearance ($\hat{I}_{t+1}$) of the scene.

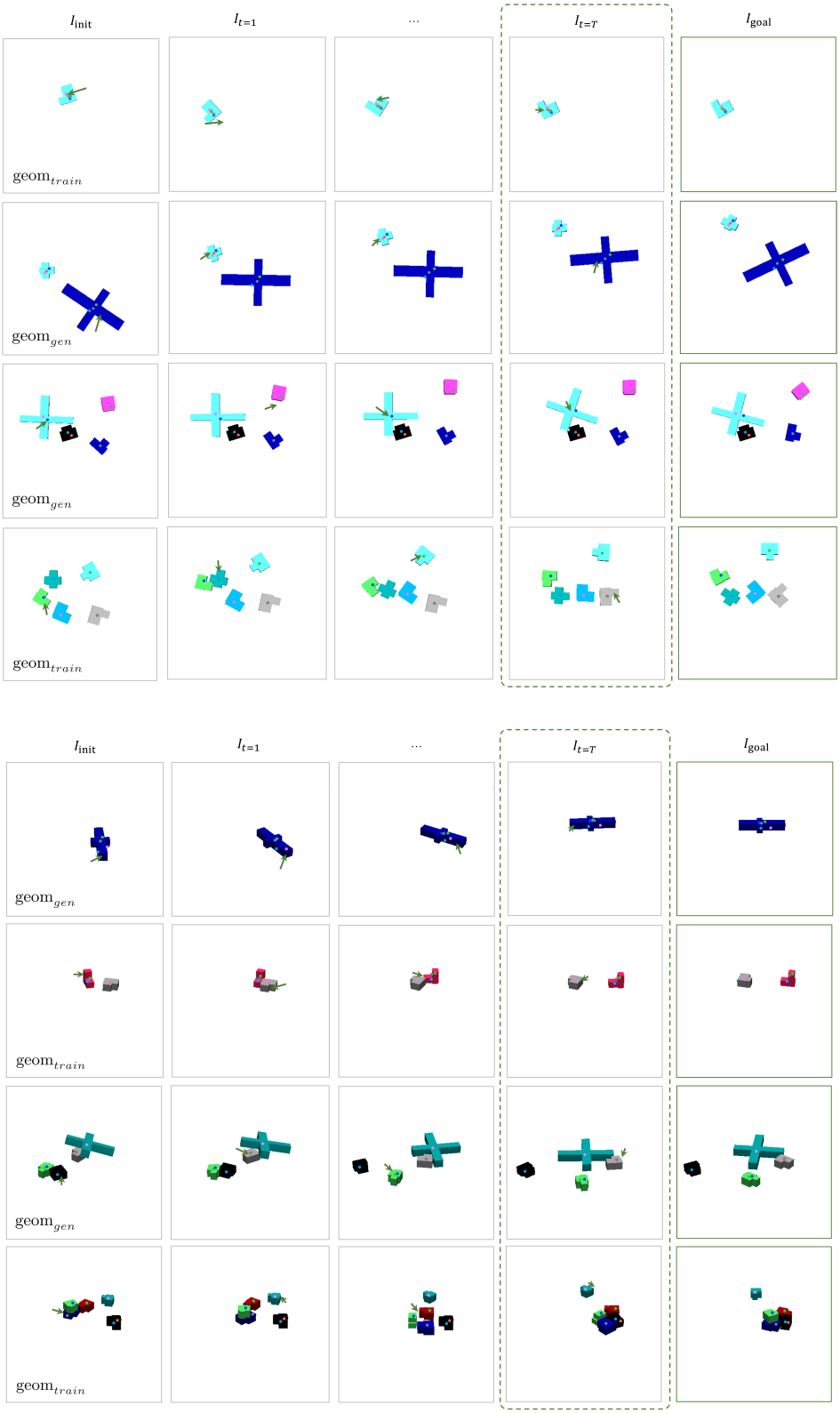

Figure A.5.1: MPC results steps for *Top View* and *Angled View* observations.

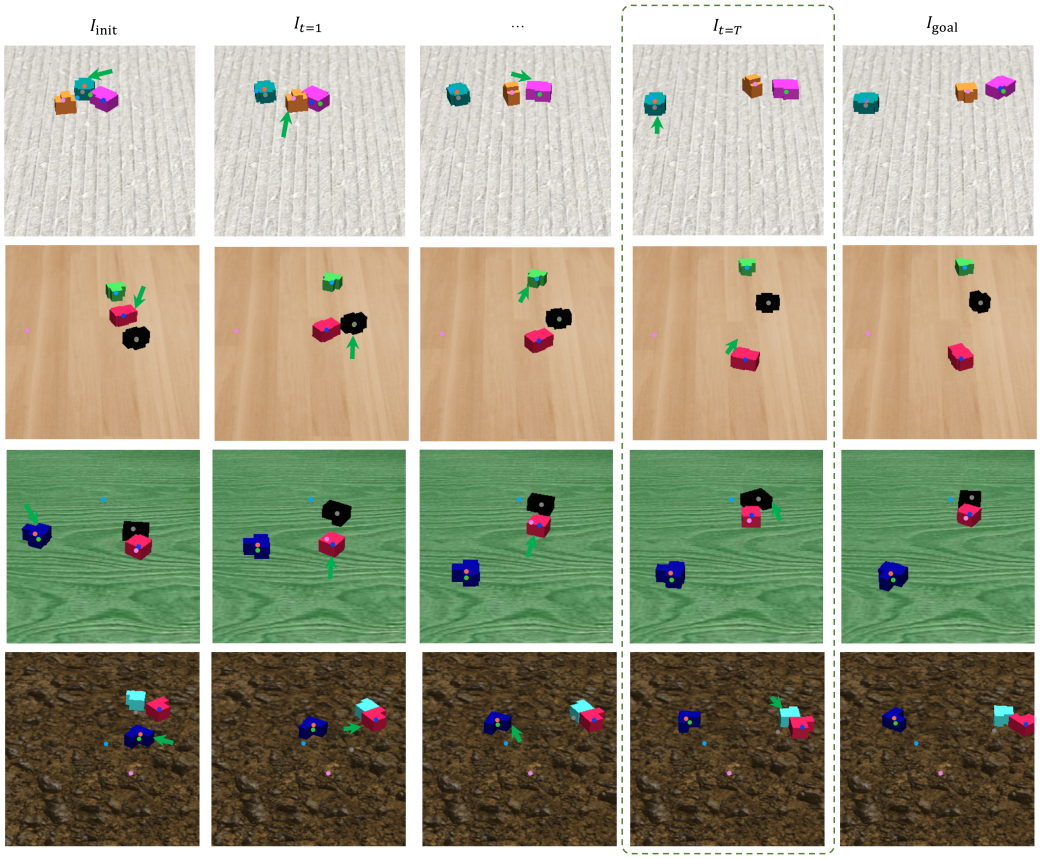

Figure A.5.2: Qualitative results of zero-shot generalization to unseen backgrounds. The green arrows are the optimal actions.

A.6 Limitations and Failure Cases

One limitation that remains in our formulation is the inflexibility with respect to the number of keypoints. Unlike hard-coding the number of objects as in prior work, we show that a fixed number of keypoints is a better design choice in terms of generalizability (see Table 2). We also highlight failure cases in the keypoint extraction (see Figure A.6.1). In real-robot, few cases where multiple objects were pushed out of the image frame had inconsistent keypoints that were not attached to objects. This is because the subset of keypoints that are sufficient for the keypoint module to reconstruct the image. Therefore, the remaining keypoints randomly bind to the background. Also, in simulation we observed a few cases with 5 objects where an extremely out-of-distribution geometry caused issue in the keypoint assignment. However, note that these are based on zero-shot generalization and we expect the framework to perform better if the keypoint model is trained on these cases. Additionally, we include the failure cases for generalization to unseen backgrounds. We noticed that in cases where the unseen background color matches the color of one of the objects, the keypoint detection confuses the object with the background. This results in either missing that object in the keypoint assignment or inconsistency in the position of the keypoints. (see Figure A.6.2)

We demonstrate the performance of our framework for cases with inconsistent keypoints. We showcase the real robot examples from Fig. A.6.1. In both of these cases, the majority of objects are pushed out of the image frame. Therefore, some of the keypoints are not attached to any object and are randomly registered to the background. When predicting the future keypoint locations, our model is able to correctly predict the future state of the keypoints that are attached to the objects (see Fig. A.6 yellow object in the top row). However, the model also predicts translation in some of the keypoints that were randomly attached to the background. Regardless, the reconstruction module is able to estimate the future image of the scene $\hat{I}_{t+1}$ through keypoints assigned to the objects.

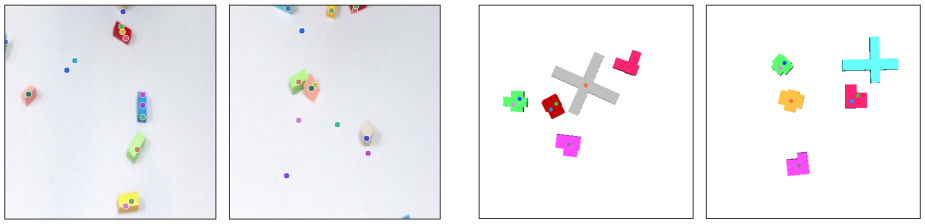

Figure A.6.1: Failure cases for keypoint detection.

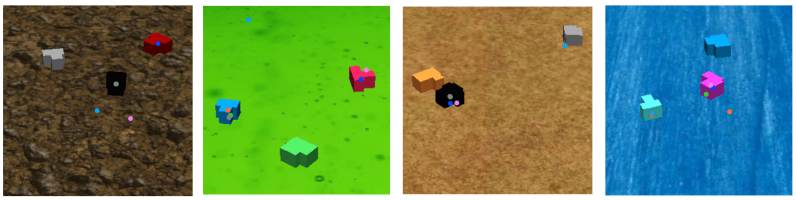

Figure A.6.2: Failure cases for keypoint detection in generalization to unseen background textures.

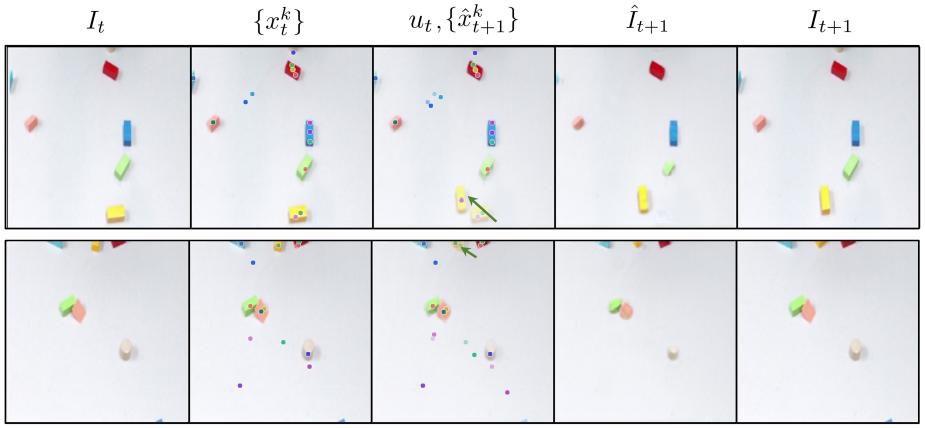

Figure A.6.3: Model performance for cases with inconsistent keypoint detection. Although some keypoints were assigned to the background the final reconstruction of the scene accurately predicts the future appearance after applying the action $\hat{I}_{t+1}$.