KINet: Unsupervised Forward Models for Robotic Pushing Manipulation

Alireza Rezazadeh and Changhyun Choi, *Member, IEEE*

***Abstract*—Object-centric representation is an essential abstraction for forward prediction. Most existing forward models learn this representation through extensive supervision (e.g., object class and bounding box) although such ground-truth information is not readily accessible in reality. To address this, we introduce KINet (Keypoint Interaction Network)—an end-to-end unsupervised framework to reason about object interactions based on a keypoint representation. Using visual observations, our model learns to associate objects with keypoint coordinates and discovers a graph representation of the system as a set of keypoint embeddings and their relations. It then learns an action-conditioned forward model using contrastive estimation to predict future keypoint states. By learning to perform physical reasoning in the keypoint space, our model automatically generalizes to scenarios with a different number of objects, novel backgrounds, and unseen object geometries. Experiments demonstrate the effectiveness of our model in accurately performing forward prediction and learning plannable object-centric representations for downstream robotic pushing manipulation tasks.**

***Index Terms*—Representation learning, deep learning methods, manipulation planning.**

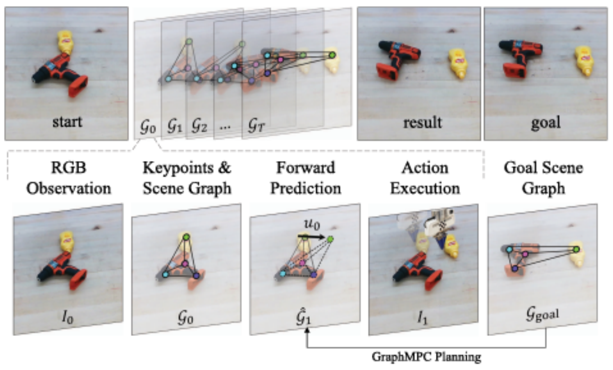

Fig. 1. Existing forward modeling methods rely on the ground-truth object states, limiting their application to the real setting where such data is hard to obtain. Our framework (KINet) learns unsupervised forward models from raw RGB images. At each timestep, using the image observation (I_t), keypoints are extracted to construct the scene graph ($\mathcal{G}_t$). The graph-based forward model estimates the future keypoints ($\hat{\mathcal{G}}_{t+1}$) conditioned on the action u_t. Using this forward model, we perform model-based planning and find a sequence of actions to achieve a goal configuration based on similarity to the goal graph.

I. Introduction

DISCOVERING a structured representation of the world allows humans to perform a wide repertoire of motor tasks such as interacting with objects. The core of this process is learning to predict the response of the environment to applying an action [20]. The internal models, often referred to as the forward models, come up with an estimation of the future states of the world given its current state and the action. By cascading the predictions of a forward model, it is possible to plan a sequence of actions that would bring the world from an initial state to a desired goal state.

Recently, deep learning architectures have been proposed to perform forward prediction using an object-centric representation of the system [5], [17], [24], [32]. This representation is learned from the visual observation by factorizing the scene into the underlying object instances using ground-truth object states (e.g., object class, position, and bounding box). We identified two major limitations in the existing work: First, they either assume access to the ground-truth object states [2], [17] or predict them using pre-trained object detection or instance segmentation models [24], [32]. However, obtaining the ground truth object states is not always feasible in practice, especially in real-world settings. Relying on pre-trained object detection and segmentation models further makes the forward models fragile as the perception models do not work on novel objects, significantly limiting their generalization capability. Second, factorizing the scene into a fixed number of object instances limits the generalization of the model models to scenarios with a different number of objects.

In this letter, we address both of these limitations by proposing to learn forward models using a keypoint representation. Keypoints represent a set of salient locations of moving entities. Our model KINet (Keypoint Interaction Network) learns an unsupervised forward model in three steps (Fig 1): 1) A keypoint extractor factorizes the scene into keypoints with no supervision other than raw visual observations. 2) A graph representation of the scene is learned, where each node corresponds to a keypoint and edges are keypoints relations. Node features carry implicit object-centric features as well as explicit keypoint state information. 3) With probabilistic message passing, our model learns an action-conditional forward model to predict the future location of keypoints and reconstruct the future appearances of

Manuscript received 27 April 2023; accepted 2 August 2023. Date of publication 9 August 2023; date of current version 21 August 2023. This letter was recommended for publication by Associate Editor Z. Xiong and Editor H. Liu upon evaluation of the reviewers' comments. This work was supported in part by the Sony Research Award Program and NSF under Grant 2143730. *(Corresponding author: Alireza Rezazadeh.)*

The authors are with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55414 USA (e-mail: rezaz003@umn.edu; cchoi@umn.edu).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2023.3303829, provided by the authors.

Digital Object Identifier 10.1109/LRA.2023.3303829

the scene. We evaluate KINet's forward prediction accuracy and demonstrate that, by learning forward prediction in a keypoint coordinate, our model effectively re-purposes this knowledge and generalizes it to complex unseen circumstances.

Our key contributions are: 1) We introduce KINet, a graph-based and end-to-end method for learning unsupervised action-conditional forward models from visual observations 2) We introduce probabilistic message-passing operation for efficient aggregation of relevant information in the graph. 3) We introduce GraphMPC for accurate action planning using graph similarity. 4) We demonstrate learning forward models with keypoints enables generalization to complex unseen scenarios.

II. Related Work

Unsupervised keypoint extraction. Keypoints have been widely used in pose tracking [31], [33] and video prediction [19], [21], [29], [33]. Recent work explored keypoints for reinforcement learning in a keypoint space [4], [9], [14].

Forward models. The most fundamentally relevant work to ours is Interaction Networks (IN) [2], [25] and follow-up work using graph networks for forward simulation [11], [16], [22]. These methods rely on the ground-truth state of objects to build graphs where each node represents an object. Several approaches extended IN by combining explicit states with implicit visual features [24], [28], [32]. However, two main concerns remain unaddressed. First, object features in the graph are often obtained from extensive ground-truth information such as position, bounding box, and mask [11], [15], [24], [32]. Second, these approaches lack generalization to varying number of objects as they are formulated on fixed-size graphs where each node has to correspond to one of the objects. To address these limitations, our framework extracts unsupervised keypoints to build a scene graph and performs forward prediction in the keypoint space. This allows performing forward modeling without any supervision other than RGB images. Also, factorizing the scene to keypoints instead of objects allows for automatic generalization to varying number of objects.

Action-conditional forward models. Battaglia et al. [2] augments the action to the node embeddings. Ye et al. [32] included the action as an additional node in a fully connected graph while other nodes represent objects. For probabilistic forward models, Henaff et al. [8] suggests using a latent variable dropout to condition the model on action [6]. Minderer et al. [30] showed the effectiveness of contrastive estimation [23] to learn actionable representations. Minderer et al. [21] uses keypoints for video prediction from a history of frames. However, their dynamics model is not conditioned on action and cannot be used for action planning. In our work, we ensure conditioning on the action by incorporating the action vector in the graph message-passing operation. In addition, we extend the contrastive estimation method for graph embeddings to further enhance the learned representations.

Unsupervised Forward Models. Kipf et al. [12] uses a contrastive loss to learn object-centric abstractions in multibody environments of fixed objects with minimal visual features such as 2D shapes. In our work, we randomize the properties of the objects and examine challenging realistic objects. Kossen et al. [13] uses images to infer a set of object states (e.g., position and velocity) to predict the future state of each object using graph networks. Although learning unsupervised states, this method is formulated on a fixed number of objects and only tested on environments with simple 2D geometries. Li et al. [18] infers the causal graph representation and makes future predictions on a fixed dynamic system in simulation using a pretrained keypoint extractor from topview images. In our work, we experiment with other camera angles and real-world 3D objects with various properties such as geometry and texture.

III. Keypoint Interaction Networks (KINet)

We assume access to RGB image, action vector, and the image after applying the action: $\mathcal{D} = \{(I_t, u_t, I_{t+1})\}$. Our goal is to learn a forward model that predicts the future states of the objects. We describe our approach in two main steps (see Fig. 2): learning to encode visual observations into keypoints and learning an action-conditioned forward model in the keypoint space.

A. *Unsupervised Keypoint Detection*

The keypoint detector (f_{kp}, Fig. 2) is a mapping from RGB visual observations of the scene to a lower-dimensional set of K keypoint coordinates $\{x_t^k\}_{k=1\ldots K} = f_{\text{kp}}(I_t)$. The keypoint coordinates are learned by capturing the spatial appearance of the objects in an unsupervised manner. Specifically, the keypoint detector receives a pair of initial and current images (I_0, I_t) and uses a convolutional encoder to compute a K-dimensional feature map for each image $\Phi(I_0), \Phi(I_t) \in \mathbb{R}^{H' \times W' \times K}$. The expected number of keypoints is set by the dimension K. Next, the feature maps are marginalized into a 2D keypoint coordinate $\{x_0^k, x_t^k\}_{k=1\ldots K} \in \mathbb{R}^2$. We use a convolutional image reconstruction model (f_{rec}, Fig. 2) with skip connections to inpaint the current image frame using the initial image and the predicted keypoint coordinates $\hat{I}_t = f_{\text{rec}}(I_0, \{x_0^k, x_t^k\}_{k=1\ldots K})$. With this formulation, f_{kp} and f_{rec} create a bottleneck to encode the visual observation in a temporally consistent lower-dimensional set of keypoints [14].

B. *Graph Representation of Scene*

After factorizing the scene into K keypoints, we build a graph $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t, \mathbf{Z})$ (undirected, no self-loop) where keypoints and their pairwise relations are the graph nodes and edges. Keypoint positional and visual information are encoded into embedding of nodes $\{\mathrm{n}_t^k\}_{k=1\ldots K} \in \mathcal{V}_t$ and edges $\{\mathrm{e}_t^{ij}\} \in \mathcal{E}_t$. We define the adjacency matrix to specify the graph connectivity as $\mathbf{Z} \in \mathbb{R}^{K \times K}$ where $z_{ij} \in [0, 1]$ is the probability of the edge $\mathrm{e}^{ij} \in \mathcal{E}$. At timestep t, node embeddings encode keypoint positional and visual features $\mathrm{n}_t^k = [\Phi_t^k, x_t^k]$. Edge embeddings contain relative positional information of each node pair $\mathrm{e}_t^{ij} = [x_t^i - x_t^j, \|x_t^i - x_t^j\|_2^2]$.

Existing graph-based forward modeling methods construct the scene graph such that each node represents an object. The node embeddings are positional and visual features extracted

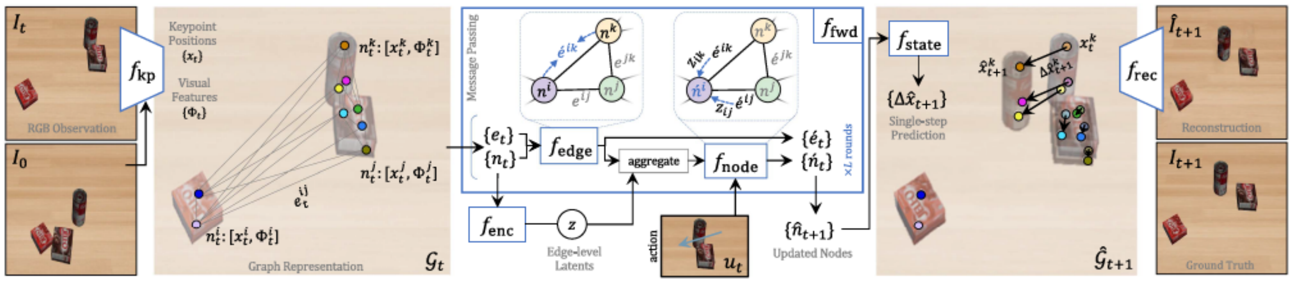

Fig. 2. Keypoint Interaction Network (KINet): a keypoint detector f_{kp} utilizes the current RGB image I_t and the initial frame I_0 to identify a set of unsupervised keypoints $\{x_t\}$ and their corresponding visual features $\{\Phi_t\}$. A scene graph $\mathcal{G}t$ is constructed, where the keypoints are represented as nodes. The forward model f_{forw} processes the graph and action u_t through L rounds of probabilistic message-passing operations using the edge function f_{edge} and the node function f_{node}, with the edge probabilities generated by an encoder f_{enc}. Subsequently, a state decoder estimates the future positions of the keypoints $\hat{x}_{t+1}$. Finally, the appearance of the future scene is reconstructed by f_{rec}.

using ground-truth object state information [11], [24], [32]. Our framework, however, does not rely on any ground-truth information supervision and only uses RGB images.

C. Probabilistic Graph-Based Forward Model

We extend the existing graph-based approaches [2], [25] and propose probabilistic graph-based forward models. The core of our forward model is probabilistic message-passing using edge-level latent variables $z_{ij} \in \mathbb{R}^d$ that represent the edge probabilities. A posterior network p_θ infers the elements of the adjacency matrix given the scene graph representation. In particular, we model the posterior as $p_\theta(z_{ij}|\mathcal{G}_t) = \sigma(f_{\text{enc}}([\mathrm{n}_t^i, \mathrm{n}_t^j]))$ where $\sigma(.)$ is the sigmoid function.

The forward model $\hat{\mathcal{G}}_{t+1} = f_{\text{fwd}}(\mathcal{G}_t, u_t, \mathbf{Z})$ predicts the graph representation at the next timestep by taking as input the current graph representation and the action vector (f_{fwd}, Fig. 2) and performing L rounds of probabilistic message-passing. Fig. 2-f_{fwd} illustrates the probabilistic message-passing operation. A single round of message-passing operation in the forward model can be described as,

$$\text{Edge update}: \mathrm{e}'^{ij} \leftarrow f_{\text{edge}}(\mathrm{n}^i, \mathrm{n}^j, \mathrm{e}^{ij})$$

$$\text{Aggregation}: \bar{\mathrm{e}}^k \leftarrow \Sigma_{i \in N(k)} z_{ik} \mathrm{e}'^{ik}$$

$$\text{Node update}: \mathrm{n}'^k \leftarrow f_{\text{node}}(\mathrm{n}^k, \bar{\mathrm{e}}^k, u_t)$$

where the edge-specific function f_{edge} first updates edge embeddings, then the node-specific function f_{node} updates each node embedding n'^k by aggregating its neighboring nodes $N(k)$ information. Specifically, the updated edges are aggregated using the edge probabilities from the inferred adjacency matrix $\mathbf{Z}$ as weights. The action u_t is also an input to the aggregation step. After performing the L rounds of message-passing, the resulting updated node embeddings estimate the graph's nodes at the next timestep $\{\hat{\mathrm{n}}_{t+1}\}$.

Recent models for forward prediction rely on fully connected graphs for message passing [15], [24], [32]. Our model, however, learns to probabilistically sample the neighbor information. Intuitively, this adaptive sampling allows the network to efficiently aggregate the most relevant neighboring information. This is specifically essential in our model as keypoints could provide redundant information if they are in close proximity.

D. Forward Prediction

The state decoder (f_{state}, Fig. 2) transforms the predicted node embeddings of the updated graph $\hat{\mathcal{G}}_{t+1}$ to a first-order difference $\{\Delta\hat{x}_{t+1}^k\} = f_{\text{state}}(\{\hat{\mathrm{n}}_{t+1}^k\})$, which is integrated once to predict the position of the keypoints in the next timestep $\{\hat{x}_{t+1}^k\} = \{x_t^k\} + \{\Delta\hat{x}_{t+1}^k\}$. To reconstruct the image at the next timestep, we borrow the reconstruction model f_{rec} from the keypoint detector $\hat{I}_{t+1} = f_{\text{rec}}(I_0, \{x_0^k, \hat{x}_{t+1}^k\})$.

E. Learning KINet

Reconstruction loss. The keypoint detector is trained to reconstruct the image at each timestep $\mathcal{L}_{\text{rec}} = \|\hat{I}_t - I_t\|_2^2$. As suggested by [21], errors from the keypoint detector were not backpropagated to other modules of the model. This ensures that the model does not conflate errors from keypoint extraction and forward prediction functions.

Forward loss. The model is optimized to predict the next state of the keypoints. A forward loss penalizes the distance between the estimated future keypoint locations using first-order state decoder and the keypoint extractor predictions: $\mathcal{L}_{\text{fwd}} = \sum_K \|\hat{x}_{t+1}^k - f_{\text{kp}}(I_{t+1})^k\|_2^2$.

Inference loss. Our model is trained to minimize the KL-divergence between the posterior and prior: $\mathcal{L}_{\text{infer}} = D_{\text{KL}}(p_\phi(\mathbf{Z}|\mathcal{G}) \| p(\mathbf{Z}))$. We use independent Gaussian prior $p(\mathbf{Z}) = \prod_i \mathcal{N}(\mathrm{z}_i)$ and reparameterization for training [10].

Contrastive loss. We use the contrastive estimation method to further enhance the learned graph representations. We add a contrastive loss [23], [30] and reframe it for graph embeddings as $\mathcal{L}_{\text{ctr}} = -\mathbb{E}_{\mathcal{D}}[\log(\mathcal{S}(\hat{\mathcal{G}}_{t+1}, \mathcal{G}_{t+1}^+)/\sum \mathcal{S}(\hat{\mathcal{G}}_{t+1}, \mathcal{G}_{t+1}^-))]$ such that the predicted graph representations $\hat{\mathcal{G}}_{t+1}$ are maximally similar to their corresponding positive sample pair $\mathcal{G}_{t+1}^+ := \mathcal{G}_{t+1}$ and maximally distant from the negative sample pairs $\mathcal{G}_{t+1}^- := G_\tau \quad \forall \tau \neq t+1$. We use a simple node embedding similarity as the graph matching algorithm $\mathcal{S}(\mathcal{G}_1, \mathcal{G}_2) = \sum_K \{n_1^k\}.\{n_2^k\}$. The motivation behind adding a contrastive loss is aligning the graph representation of similar object configurations while pushing apart those of dissimilar configurations in the embedding space to enhance the learned graphs. Finally, the combined loss is: $\mathcal{L} = \lambda_{\text{rec}} \ \mathcal{L}_{\text{rec}} + \lambda_{\text{fwd}} \ \mathcal{L}_{\text{fwd}} + \lambda_{\text{infer}} \ \mathcal{L}_{\text{infer}} + \lambda_{\text{ctr}} \, \mathcal{L}_{\text{ctr}}$.

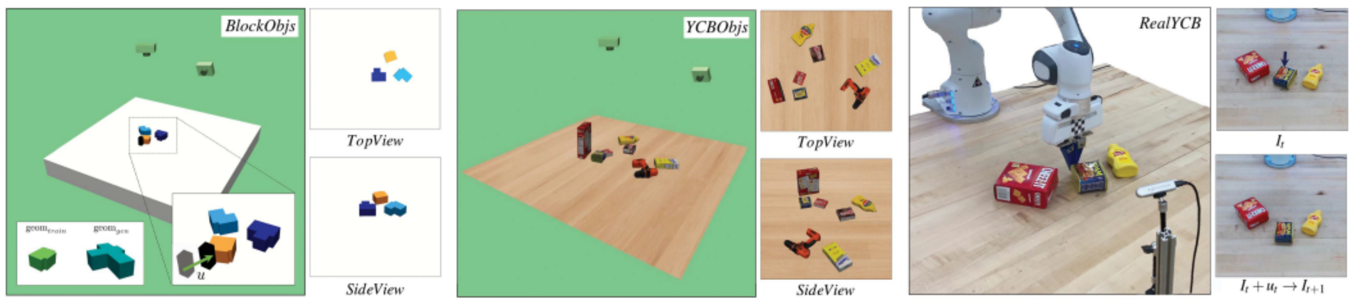

Fig. 3. Experiment setups including two simulated environments with block objects and YCB objects, and a real robot environment with YCB objects. We trained our model in simulation on 3 objects and then tested for their generalization to unseen geometries and an unseen number of objects.

F. GraphMPC Planning With KINet

A learned KINet model can be used to perform model-based planning. We extend the Model Predictive Control (MPC) method [7] and propose the GraphMPC algorithm based on graph embeddings. The graph representation of the next timestep is estimated for multiple sampled actions using f_{fwd}. The optimal action is selected such that it produces the most similar graph representation to the goal graph representation $\mathcal{G}^{\text{goal}}$. We describe our GraphMPC algorithm with a horizon of T as: $u_t^* = \arg\max\{\mathcal{S}(\mathcal{G}^{\text{goal}}, f_{\text{fwd}}(\mathcal{G}^t, \{u_{t:T}\}))\};\ t \in [0, T]$. Unlike performing conventional MPC only with respect to positional states, GraphMPC allows for accurately bringing the objects to a goal configuration both explicitly (i.e., position) and implicitly (i.e., pose, orientation, and visual appearance).

IV. Experimental Setups

Our experiments address the following: 1) How accurate is our forward model? 2) Can the model be used for action planning? 3) Does the model generalize to unseen scenarios?

A. Dataset

We apply our approach to learn a forward model for multi-object manipulation tasks. The task involves rearranging the objects to achieve a goal configuration using pushing actions. We use MuJoCo 2.0 [27] to generate two simulated environments. We also test on a real robot environment. Fig. 3 demonstrates these three experiment setups.

BlockObjs: Each object is constructed with two cuboid geoms. We sample the geom dimensions from a continuous range denoted as geom_{train} for training and geom_{gen} for generalization. Unseen geometries are designed to have elongated shapes to create extreme out-of-distribution cases (see Fig. 3).

YCBObjs: A subset of YCB objects placed on a wooden table that includes objects of daily life with diverse properties such as shape, size, and texture [3] (see Fig. 3).

In both of the simulated environments, we generate 10 K episodes of random pushing on multiple objects (1–5 objects) where a simplified robot end-effector applies randomized pushing for 60 timesteps per episode. We collect the 4-dimensional action vector (pushing start and end location) and RGB images before and after each action is applied. Images are obtained using an overhead (*TopView*) and an angled camera (*SideView*).

RealYCB: We directly transfer the learned model from the *YCBObjs* simulation to the real environment with YCB objects. The real robot setup includes a Franka Emika Panda robot with Festo soft fingers and an Intel RealSense camera on the side of the tabletop (see Fig. 3).

B. Baselines

We compare our approach with existing methods for learning object-centric forward models:

Forward Model *(Forw):* We train a convolutional encoder to extract visual features of the scene image (*Img*) and learn a forward model in the feature space.

Forward-Inverse Model [1] *(ForwInv):* We train a convolutional encoder to extract visual features of the scene image (*Img*) and jointly learn forward and inverse models.

Interaction Network [2], [25] *(IN):* We build an Interaction Network based on the ground truth location of the objects available in simulation. Each object representation contains the ground-truth position and velocity of the objects (*GT state*). Note that this approach is only applicable to scenarios where the number of objects in the scene is known and fixed.

Visual Interaction Network [28], [32] *(VisIN):* We train a convolutional encoder to extract visual features of fixed-size bounding boxes centered on ground-truth object locations (*GT state + Img*). We use the extracted visual features as object representations in the Interaction Network. Note that this approach requires prior knowledge of the number of objects.

Causal Discovery from Videos [18] *(V-CDN):* A pretrained perception module extracts keypoints that are used in an inference module to predict a causal structure of the visual observation which is then used in a dynamics module to predict the future location of the keypoints.

We also compare KINet with two variants: 1) *KINet - determ*, which employs a fully connected graph instead of probabilistic edges, and 2) *KINet - no ctr*, which excludes the contrastive loss.

C. Training and Evaluation Setting

All models are trained on a subset of the simulated data (*BlockObjs* and *YCBObjs*) with 3 objects (8 K episodes: 80% training, 10% validation, and 10% testing sets). To evaluate generalization to a different number of objects, we use other subsets of data with 1, 2, 4, and 5 objects (∼ 400 episodes for each case). We train

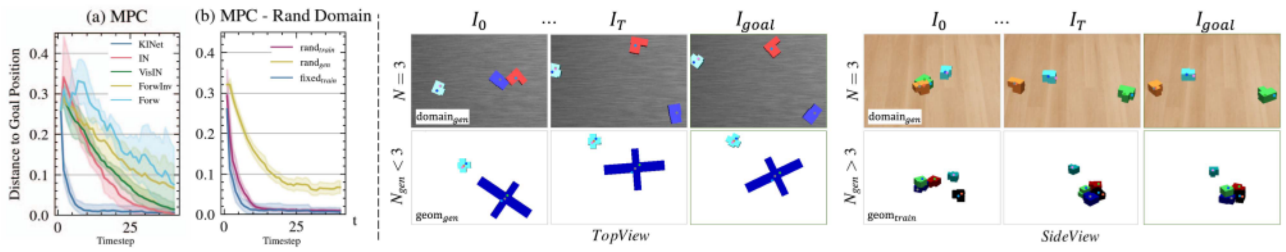

Fig. 4. MPC on *BlockObjs*. Left panel: MPC result measured as the distance to goal configuration. (a) Comparison with baselines. (b) MPC for KINet trained on a fixed white background (fixed$_{train}$), generalization to random backgrounds (rand$_{gen}$), and trained on random backgrounds (rand$_{train}$). Right panel: qualitative examples of generalization to unseen geometries, unseen number of objects, and unseen table textures.

TABLE I
FORWARD PREDICTION PERFORMANCE ON *BLOCKOBJS* MEASURED AS SINGLE-STEP MEAN PREDICTIONS ERROR $\times 10^{-3}$

Model	*SideView*	*TopView*	Img	GT state
IN	0.109 ± 0.01	0.112 ± 0.01	-	✓
VisIN	0.121 ± 0.03	0.107 ± 0.01	✓	✓
Forw	0.317 ± 0.09	0.309 ± 0.12	✓	-
ForwInv	0.266 ± 0.02	0.293 ± 0.08	✓	-
KINet (ours)	$\mathbf{0.129} \pm 0.02$	$\mathbf{0.122} \pm 0.01$	✓	-
KINet - determ	0.133 ± 0.01	0.127 ± 0.03	✓	-
KINet - no ctr	0.169 ± 0.05	0.173 ± 0.02	✓	-

TABLE II
GENERALIZATION RESULTS MEASURED AS THE AVERAGE DISTANCE TO THE GOAL POSITION $\times 10^{-3}$

	geom$_{train}$		geom$_{gen}$	
N	KINet	KINet - determ	KINet	KINet - determ
1	0.21 ± 0.04	0.28 ± 0.01	0.35 ± 0.04	0.36 ± 0.08
2	0.21 ± 0.03	0.22 ± 0.05	0.53 ± 0.06	0.59 ± 0.07
3	0.19 ± 0.02	0.19 ± 0.03	0.20 ± 0.05	0.31 ± 0.08
4	0.51 ± 0.02	0.57 ± 0.12	0.65 ± 0.07	0.96 ± 0.09
5	0.89 ± 0.13	1.05 ± 0.16	1.64 ± 0.11	2.17 ± 0.10

our model separately on images obtained from the overhead camera (*TopView*) and the angled camera (*SideView*). We set the expected number of keypoints $K = 6$ for *BlockObjs*, $K = 9$ for *YCBObjs* and *RealYCB*. For the message-passing operation, the number of rounds is set to $L = 3$. Note that f_{enc}, f_{edge}, and f_{node} are MLPs with three 128-dim hidden layers.

V. RESULTS

This section is organized to thoroughly evaluate our model in simulated and real environments.

A. *Does the Model Accurately Learn a Forward Model?*

First, we evaluate the forward prediction accuracy. Our model factorizes the observation into a set of keypoint representations and accurately estimates the future appearance of the scene conditioned on external action.

Using *BlockObjs* data, we quantify the effectiveness of our model in comparison with Forw, ForwInv, IN, and VisIN baselines (Table I). We separately train and examine each model on *TopView* and *SideView* images. The prediction error is computed as the average distance between the predicted and ground-truth positional states. VisIN performs the best among baseline models as it builds object representations with explicit ground-truth object positions and their visual features. Our model, on the other hand, achieves a comparable performance to VisIN while it does not rely on any supervision beyond the scene images. Forw and ForwInv baselines have similar supervision to ours but are significantly less accurate. This emphasizes the capability of our approach in learning a rich graph representation of the scene and an accurate forward model, while relaxing prevailing assumptions of the prior work on the structure of the environment and the availability of ground-truth state information. KINet - determ & no ctr will be discussed in Section V-D.

B. *Can We Use the Model in Control Tasks?*

We design a robotic manipulation task of rearranging a set of objects to the desired goal state using MPC with pushing actions. For all models, we run 1 K episodes with randomized object geometries and initial poses and a random goal configuration of objects. The planning horizon is set to $T = 40$ timesteps in each episode. For our model, we perform GraphMPC based on graph embedding similarity as described in Section III-F. For all baseline models, we perform MPC directly on the distance to the goal. Fig. 4(a) shows MPC results of *BlockObjs* based on *Top View* observations. Our approach is consistently more accurate and faster in reaching the goal configuration compared to the baseline models.

C. *Does the Model Generalize to Unseen Circumstances?*

One of our main motivations to learn a forward model in the keypoint space is to eliminate the dependency of model formulation on the number of objects in the scene which allows for generalization to an unseen number of objects, object geometries, background textures, etc.

BlockObjs. We train KINet on 3 randomized blocks with (geom$_{train}$) and test for generalization to an unseen number of objects (1, 2, 4, 5), unseen object geometries (geom$_{gen}$). Fig. 4 shows qualitative generalization results. We separately train and examine for generalization on *TopView* and *SideView* images. Since our model learns to perform forward modeling in the keypoint space, it reassigns the keypoints to unseen objects and makes forward predictions. Table II summarizes the generalization performance to unseen number of objects and unseen geometries. As expected, by increasing the number

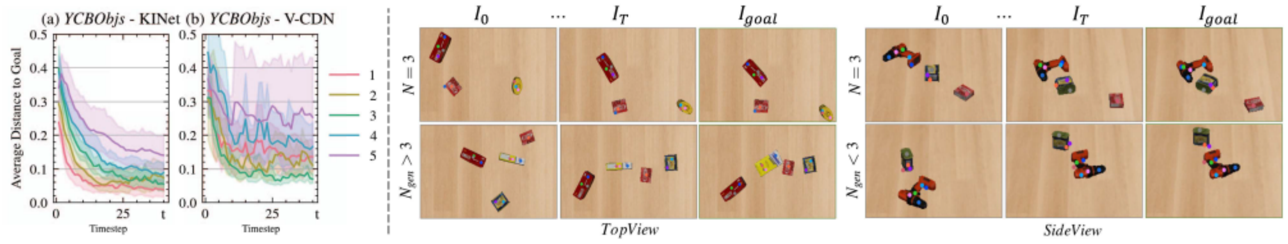

Fig. 5. MPC for generalization to unseen number of objects on *YCBObjs*. Left panel: baseline comparison. Right panel: results for *Top* and *SideView*.

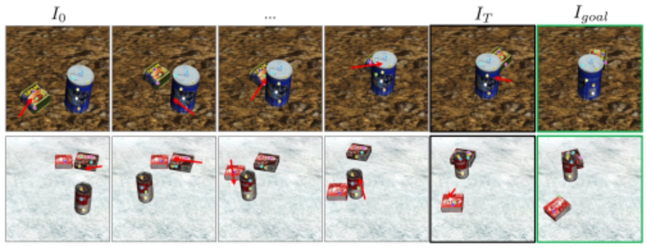

Fig. 6. KINet performance with challenging background textures and occlusion in the goal configurations.

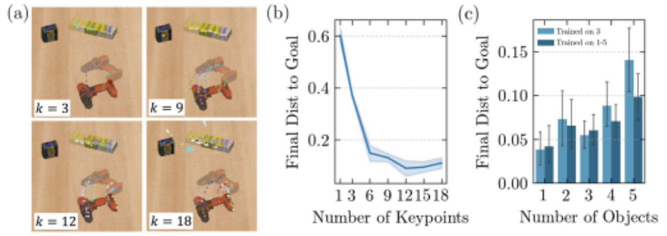

Fig. 7. (a) Qualitative examples showcasing the predicted keypoints for varying K. (b) Distance to the goal configuration after 30 steps (T = 30) for KINets trained with varying numbers of keypoints. (c) Distance to goal after 30 steps (T = 30) for KINet trained on 3 objects compared to 1–5 objects.

of objects the average distance to the goal position increases. Also, objects with out-of-distribution geometries (geom$_{gen}$) have more distance to the goal position. KINet - determ will be discussed in Section V-D.

Further, we test the performance of the model on unseen background textures (i.e., the table texture). Since the keypoint extraction relies on visual features of salient objects, our model is able to perform the control tasks by ignoring the background and assigning keypoints to the moving objects. Fig. 4(b) compares the MPC results for the KINet trained on a fixed white background (fixed$_{train}$), zero-shot generalization of KINet trained on the fixed background to randomized backgrounds (rand$_{gen}$), and KINet trained directly on randomized backgrounds (rand$_{train}$). As expected, although the MPC converges, the final distance to the goal configuration is larger for rand$_{gen}$. This final error is statistically at the same level of accuracy for fixed$_{train}$ and rand$_{train}$. Qualitative examples are included in Fig. 4.

YCBObjs. We train our model on a random subset of 3 YCB objects and test for generalization to an unseen number of objects (1, 2, 4, 5). As shown in Fig. 5, our method generalizes well to an unseen number of objects and performs the control task accurately. Importantly, assigning multiple keypoints to each object allows our framework to implicitly capture the orientation of each object, as well as their position without any supervision on the object pose (e.g., compare the power drill pose in Fig. 5). Additional qualitative results for two challenging scenarios are included in Fig. 6. In these two examples, the tabletop is highly textured which shows the robustness of the keypoint factorization in binding to the moving object and ignoring the background. Moreover, objects in the goal configuration in these two examples are partially occluded. This means that even if an object is heavily occluded, our framework should still be able to find a sequence of actions to recover from the occluded configuration by leveraging the keypoints assigned to the non-occluded objects. We believe this is a significant advantage, as it allows our framework to handle challenging scenarios and achieve accurate performance.

We compare the performance of our framework with V-CDN [18] baseline which is also a keypoint-based model to learn the structure of physical systems and perform future predictions and potentially generalize to an unseen number of objects. Although V-CDN is formulated to extract the causal structure of a fixed system through visual observations, we pretrain its perception module on our randomized *YCBObjs* dataset for a direct comparison to our model. To perform a control task, we condition the V-CDN on the action by adding an encoding of the action vector to each keypoint embedding. Fig. 5 compares the MPC results (normalized to the number of objects) of our method and V-CDN both trained on a subset of 3 YCB objects. The MPC performance of both models is comparable for rearranging 3 objects (green lines). However, V-CDN is most accurate for the number of objects it has been trained on and significantly less accurate when generalizing to an unseen number of objects. We attribute this to two reasons: 1) Although using keypoints, V-CDN is formulated to infer explicit causal structure and physical attributes for the graph representation of fixed environments which does not necessarily carry over to unseen circumstances. 2) Unlike our method, V-CDN does not take into account the visual features in the model and only uses the keypoint positions.

D. Analysis and Ablation

We justify the major choices we made to formulate the model with ablation studies: the probabilistic graph representation (*KINet - determ*), and the contrastive loss (*KINet - no ctr*). As shown in Table I, the best forward prediction for both *TopView* and *SideView* images is achieved when the model is probabilistic

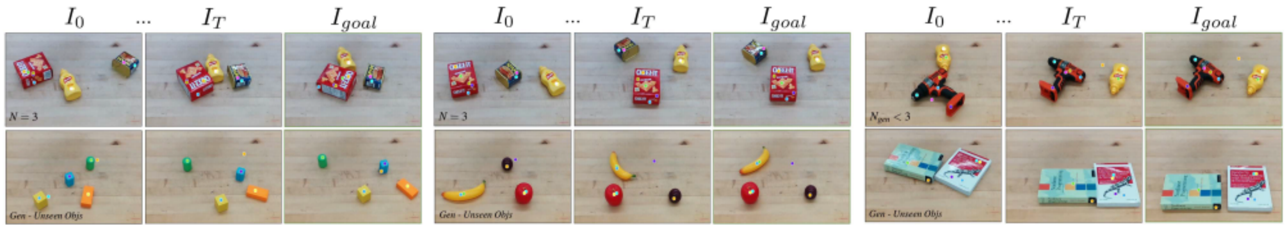

Fig. 8. Qualitative results in the real setting. Top row: Performance of the model on *RealYCB* objects. Bottom row: Generalization to unseen objects.

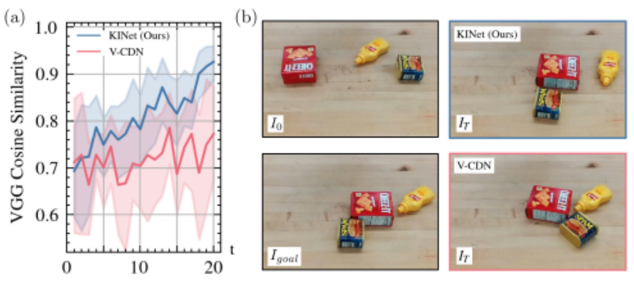

Fig. 9. Baseline comparison in the real setting. (a) Similarity to the goal image (b) Our model more accurately rearranges *YCB* Objects to match the goal image.

and trained with a contrastive loss. The contrastive loss is an essential element in our approach to ensure the learned forward model is accurately action conditional. Also, with a probabilistic graph representation, our model achieves better generalization compared to the deterministic variant. This performance gap is more evident when generalizing to unseen geometries (Table II).

We ran additional experiments to examine the effect of the number of keypoints. Fig. 7(b) shows the distance to the goal configuration after executing 30 planning steps with KINet, trained with different numbers of keypoints. Notably, for object rearrangement involving 3 YCB objects, KINet with fewer than 6 keypoints exhibited considerably larger errors. This indicates that in environments with complex visual features, optimal performance is achieved by a larger number of keypoints to ensure proper keypoint assignment (i.e., with $K = 3$, Fig. 7(a) shows one object remains unassigned).

Since the number of objects is not hard-coded into our formulation, KINet allows for training on a variable number of objects. We experimented with training on varying numbers of *YCBObjs* ($N = 1$ to 5), as opposed to a fixed number ($N = 3$). The results, shown in Fig. 7(c), demonstrate that training KINet on a variable number of objects leads to a more balanced performance. As expected, for $N > 3$, the final distance to the goal improved since the model was trained on these scenarios instead of generalizing. However, we did not observe a statistically significant change for $N \leq 3$.

E. Real Robot Results

After training the model in *YCBObjs* simulation, we test for the real robot performance (see Fig. 3). Fig. 8 (top row) shows examples of the control task. The framework successfully transferred to the real setting and was able to perform the *RealYCB* object rearrangement tasks. We observed that some of the detected keypoints were slightly less consistent compared to the simulation. We attribute this to the sim2real domain gap (e.g., the tabletop texture and camera noise).

We also tested for generalization to unseen objects (Fig. 8, bottom row). Our model generalized to unseen objects by appropriately distributing the keypoints across the objects. We noticed that in some instances a keypoint is attached to the tabletop due to its visual feature. Regardless, the control task is achieved as the majority of keypoints are attached to the objects. The GraphMPC algorithm selects the action based on maximizing graph similarity to the goal scene across all nodes which reduces the effect of a single inconsistent keypoint on the rearrangement task.

We also compare with V-CDN baseline in the real setting. Since the ground-truth positions are unknown in the real world, we use image similarity to quantify the accuracy of the rearrangement task. Specifically, we used the pretrained VGG [26] to measure the cosine similarity between the scene image and the goal image in the feature space to obtain MPC results for the real-world setting. As shown in Fig. 9, our model is consistently more accurate in rearranging the real scene and archives a significantly higher similarity to the goal image.

VI. Conclusion

In this letter, we propose a method for learning action-conditioned forward models based only on RGB images. We showed that our approach effectively makes forward predictions with keypoint factorization of the scene image. Also, we showed that a keypoint-based forward model, unlike prior work, does not make assumptions about the number of objects which allows for automatic generalization to a variety of unseen circumstances. Importantly, our model is trained without any explicit supervision on ground-truth object states. Our framework has a limitation regarding the fixed number of expected keypoints. Although we demonstrate that this approach enhances generalizability compared to fixing the number of objects, determining the optimal number of keypoints for a specific environment requires experimenting with various numbers. Furthermore, we observed slight inconsistencies in keypoints for real-robot. As a future direction, enhancing the keypoint extraction module to better handle real settings would be an interesting area of investigation.

References

[1] P. Agrawal, A. Nair, P. Abbeel, J. Malik, and S. Levine, "Learning to poke by poking: Experiential learning of intuitive physics," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 5092–5100.

[2] P. Battaglia, R. Pascanu, M. Lai, D. Jimenez Rezende, and K. Kavukcuoglu, “Interaction networks for learning about objects, relations and physics,” in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 4509–4517 .

[3] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and Aaron M. Dollar, “The YCB object and model set: Towards common benchmarks for manipulation research,” in *Proc. Int. Conf. Adv. Robot.*, 2015, pp. 510–517.

[4] B. Chen, P. Abbeel, and D. Pathak, “Unsupervised learning of visual 3D keypoints for control,” in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 1539–1549 .

[5] Z. Chen, J. Mao, J. Wu, K.-Y. K. Wong, J. B. Tenenbaum, and C. Gan, “Grounding physical concepts of objects and events through dynamic visual reasoning,” in *Proc. Int. Conf. Learn. Representations*, May, 2021.

[6] Y. Gal and Z. Ghahramani, “Dropout as a Bayesian approximation: Representing model uncertainty in deep learning,” in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1050–1059.

[7] C. E. Garcia, D. M. Prett, and M. Morari, “Model predictive control: Theory and practice–a survey,” *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[8] M. Henaff, A. Canziani, and Y. LeCun, “Model-predictive policy learning with uncertainty regularization for driving in dense traffic,” in *Proc. Int. Conf. Learn. Representations*, 2019.

[9] T. Jakab, A. Gupta, H. Bilen, and A. Vedaldi, “Unsupervised learning of object landmarks through conditional image generation,” in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 4020–4031.

[10] P. D. Kingma and M. Welling, “Auto-encoding variational Bayes,” *stat*, 2014, Art. no. 1050.

[11] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel, “Neural relational inference for interacting systems,” in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 2688–2697.

[12] T. Kipf, E. Van der Pol, and M. Welling, “Contrastive learning of structured world models,” in *Proc. Int. Conf. Learn. Representations*, 2019.

[13] J. Kossen, K. Stelzner, M. Hussing, C. Voelcker, and K. Kersting, “Structured object-aware physics prediction for video modeling and planning,” in *Proc. Int. Conf. Learn. Representations*, 2019.

[14] T. Kulkarni et al., “Unsupervised learning of object keypoints for perception and control,” in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 10724–10734 .

[15] Y. Li, J. Wu, J.-Y. Zhu, J. B. Tenenbaum, A. Torralba, and R. Tedrake, “Propagation networks for model-based control under partial observation,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 1205–1211.

[16] Y. Li, J. Wu, J.-Y. Zhu, J. B. Tenenbaum, A. Torralba, and R. Tedrake, “Propagation networks for model-based control under partial observation,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2019, pp. 1205–1211.

[17] Y. Li et al., “Visual grounding of learned physical models,” in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 5927–5936.

[18] Y. Li, A. Torralba, A. Anandkumar, D. Fox, and A. Garg, “Causal discovery in physical systems from videos,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 9180–9192.

[19] L. Manuelli, Y. Li, P. Florence, and R. Tedrake, “Keypoints into the future: Self-supervised correspondence in model-based reinforcement learning,” in *Proc. Conf. Robot Learn.*, 2021, pp. 693–710 .

[20] R. C. Miall and D. M. Wolpert, “Forward models for physiological motor control,” *Neural Netw.*, vol. 9, no. 8, pp. 1265–1279, 1996.

[21] M. Minderer, C. Sun, R. Villegas, F. Cole, K. Murphy, and H. Lee, “Unsupervised learning of object structure and dynamics from videos,” in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 92–102 .

[22] D. Mrowca et al., “Flexible neural representation for physics prediction,” in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 8813–8824 .

[23] A. V. D. Oord, Y. Li, and O. Vinyals, “Representation learning with contrastive predictive coding,” 2018, *arXiv:1807.03748*.

[24] H. Qi, X. Wang, D. Pathak, Y. Ma, and J. Malik, “Learning long-term visual dynamics with region proposal interaction networks,” in *Proc. Int. Conf. Learn. Representations*, 2020.

[25] A. Sanchez-Gonzalez et al., “Graph networks as learnable physics engines for inference and control,” in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4470–4479.

[26] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *Proc. Int. Conf. Learn. Representations*, 2014.

[27] E. Todorov, T. Erez, and Y. Tassa, “MuJoCo: A physics engine for model-based control,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 5026–5033.

[28] N. Watters, D. Zoran, T. Weber, P. Battaglia, R. Pascanu, and A. Tacchetti, “Visual interaction networks: Learning a physics simulator from video,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4539–4547.

[29] T. Xue, J. Wu, K. L. Bouman, and W. T. Freeman, “Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks,” in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 91–99 .

[30] W. Yan, A. Vangipuram, P. Abbeel, and L. Pinto, “Learning predictive representations for deformable objects using contrastive estimation,” in *Proc. Conf. Robot Learn.*, 2021, pp. 564–574 .

[31] Y. Yao, Y. Jafarian, and Hyun Soo Park, “MONET: Multiview semi-supervised keypoint detection via epipolar divergence,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*2019, pp. 753–762.

[32] Y. Ye, D. Gandhi, A. Gupta, and S. Tulsiani, “Object-centric forward modeling for model predictive control,” in *Proc. Int. Conf. Robot Learn.*, 2020, pp. 100–109.

[33] Y. Zhang, Y. Guo, Y. Jin, Y. Luo, Z. He, and H. Lee, “Unsupervised discovery of object landmarks as structural representations,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2694–2703.