Object and Relation Centric Representations for Push Effect Prediction

Ahmet E. Tekden[1], Aykut Erdem[2], Erkut Erdem[3], Tamim Asfour[4], and Emre Ugur[1]

***Abstract*—Pushing is an essential non-prehensile manipulation skill used for tasks ranging from pre-grasp manipulation to scene rearrangement, reasoning about object relations in the scene, and thus pushing actions have been widely studied in robotics. The effective use of pushing actions often requires an understanding of the dynamics of the manipulated objects and adaptation to the discrepancies between prediction and reality. For this reason, effect prediction and parameter estimation with pushing actions have been heavily investigated in the literature. However, current approaches are limited because they either model systems with a fixed number of objects or use image-based representations whose outputs are not very interpretable and quickly accumulate errors. In this paper, we propose a graph neural network based framework for effect prediction and parameter estimation of pushing actions by modeling object relations based on contacts or articulations. Our framework is validated both in real and simulated environments containing different shaped multi-part objects connected via different types of joints and objects with different masses. Our approach enables the robot to predict and adapt the effect of a pushing action as it observes the scene. Further, we demonstrate 6D effect prediction in the lever-up action in the context of robot-based hard-disk disassembly.**

***Index Terms*—Push Manipulation, Effect Prediction, Parameter Estimation, Graph Neural Networks, Interactive Perception, Articulation Prediction**

I. Introduction

Pushing is a fundamental non-prehensile (manipulation without grasping) motion primitive that gives robots great flexibility in manipulating objects [1], [2]. Using push actions, a robot can navigate objects to goal configurations even when objects are not graspable [3]; it can manipulate objects under uncertainty [4], or bring an object to the graspable area [5]. Compared to grasping actions, it is not as restrictive; however, the issue is that the robot does not have direct control over the state of the manipulated objects. This results in greater complexity in planning and control as the dynamics of the manipulated objects are often required to be taken into consideration [1]. Effect prediction of pushing action has many applications [2], [6], including scene rearrangement [7], object segmentation [8], object singulation [9], [10], pre-grasp manipulation [10]–[13]. However, action-effect prediction of pushing actions depends on many factors [14] and requires adaptation when mispredictions occurs. Figure 1 shows an example illustration. The initial prediction of the robot will be objects getting scattered. However, after seeing some of the objects moving together, the robot will understand that their future motion will continue reflecting this dynamic.

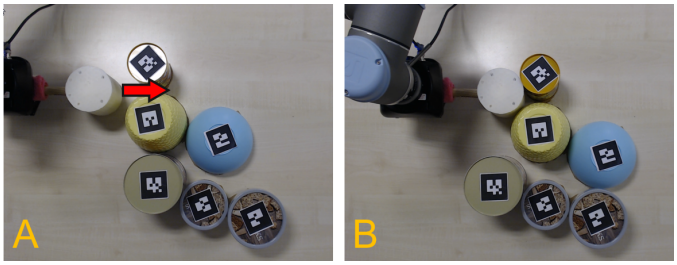


Fig. 1. We will normally expect the action of the robot on the left image to scatter contacted objects. However, seeing the contacted objects moving together, the robot should correct its belief to enable this dynamic.

In many environments, robots work with object clutters containing different shaped and weighted objects with possible articulations between them. A robot should be able to reason about the influence of shape and mass of objects, physical connections like contacts or different types of articulations between objects, propagation of motions between objects, and correction of unknown or partially known objects or object parts in the environment. Current approaches model environments with a fixed number of objects or use image data, an object-independent representation. While there has been great progress on effect prediction using raw sensory data [15]–[18], using them on decision making level has been difficult and required tasks to be generated on pixel level. While there are certain advantages of such approaches, many tasks often require more interpretable representations for the task to be defined. Humans decompose environments into objects and use their interactions for physical reasoning [19]–[21], so there is certainly value in using such representations in effect prediction. We propose using graph neural networks (GNNs) for push effect prediction. Graph neural networks [22] can exploit the graph structure of multi-objects systems by exploiting and using object- and relation-centric representations and they are heavily used in modelling physics [21], [23]–[29].

In this paper, we propose a general-purpose learnable physics engine in which object- and relation-centric representations are learned via a shared propagation network and used for physics prediction and parameter estimation in push manipulation tasks[1]. We use articulation based graph represen-

[1]Ahmet E. Tekden and Emre Ugur are with Computer Engineering Department, Bogazici University, Turkey.
[2]Aykut Erdem is with Computer Engineering Department, Koç University, Turkey
[3]Erkut Erdem is with Computer Engineering Department, Hacettepe University, Turkey
[4]Tamim Asfour is with with the Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology, Germany
*Corresponding author: Ahmet E. Tekden, email: ercan.tekden@boun.edu.tr

[1]Project page: https://fzaero.github.io/push_learning/

tations that use cylinder- and cuboid-shaped objects and their possible interactions via contacts or joints for modeling multi-part object systems. We resort to a two-step training scheme where our framework is first trained for effect prediction, then using learned object and relation representations, it is trained for parameter estimation. Our framework can predict low-level trajectories of groups of articulated objects given robot actions and estimate the mass of observed objects and joint relations between them based on their interaction history. Using articulation based representation, novel tools that are not encountered during training can be built by connecting multiple cuboids via fixed joints, and they can be used in planning in tool manipulation tasks.

An early version of this work was published in [30]. However, this paper significantly extend the work in several important directions. In [30], for physics prediction and parameter estimation, two independents networks were required. By employing a new weight-sharing mechanism that allows these tasks to share object- and relation-centric representations, the number of learnable parameters is decreased by about thirty percent. Previously, our framework was only able to model cylindrical objects. We extend the input representations of objects and their relations, allowing our network to handle objects with different shapes, predict the mass of objects, and represent complex shaped objects that are built by connecting multiple cuboids and cylinders, which even allow our framework to work with tools that are not previously encountered. In addition, we have shown that our framework can make 6D effect predictions. Furthermore, the training of the network has been improved by the use of scheduled sampling [31] and greater data distribution. These novel contributions decrease the errors for long-horizon prediction tasks, and in Section V-E, our new results have been shown to surpass the ones in [30]. More specifically, the general contributions of our framework can be listed as follows:

- We develop a graph neural network based framework for parameter estimation and physics prediction in push manipulation tasks.
- We utilize a weight-sharing mechanism to transfer learned representations to be used in new tasks.
- We show the feasibility of articulation based graph representations for modeling multi-part objects.
- We design a novel 6-D action-effect prediction in lever-up task in the context of hard-disk drive disassembly.
- Through simulated and real-world experiments, we verify our framework in joint relation and mass prediction, physics prediction, and tool manipulation and planning tasks.

II. Related Work

a) Learning Dynamics / Modelling Physics: Modeling intuitive physics has attracted considerable interest in recent years [32]. For instance, Battaglia *et al.* [33] proposed a Bayesian model called Intuitive Physics Engine and showed that the physics of stacked cuboids could be modeled with this model. Similarly, Hamrick *et al.* [34] showed that humans could reason about object masses from their interactions and modeled it with Bayesian models. Smith *et al.* [35] have modelled expectation violation in intuitive physics. They discuss how humans surprise when their physical expectations mismatch with reality, and they modeled this with deep learning methods. Deisenroth *et al.* [36] suggested a probabilistic dynamic model that depends on Gaussian Processes and that is capable of predicting the next state of a robot given the current state and the action. Recently, these studies have been extended through the use of deep learning methods. Lerer *et al.* [37] trained a deep network to predict the stability of the block towers given their raw images obtained from a simulator. Groth *et al.* [38] extended this idea by allowing stacking of objects with different geometries. They showed that their proposed network could predict the stability of given towers in this more difficult setup. The tower stacking task has continued to be an important environment for intuitive physics problems [39].

A specific topic of interest within modeling physics with deep learning is motion prediction from images, which has gained increasing attention over the last few years. Mottaghi *et al.* [40] trained a Convolutional Neural Networks (CNN) for motion prediction on static images by casting this problem as a classification problem. Mottaghi *et al.* [41] employed CNNs to predict movements of objects in static images in response to applied external forces. Fragkiadaki *et al.* [42] suggested a deep architecture in which the outputs of a CNN are used as inputs to Long Short Term Memory (LSTM) cells [43] to predict movements of balls in simulated environments.

b) Graph Neural Networks (GNNs) for Learning Physics: As deep structured models, GNNs allow learning useful representations of entities and relations among them, providing a reasoning tool for solving structured learning problems. Hence, it has found extensive use in physics prediction. Interaction network by Battaglia *et al.* [23] and Neural Physics Engine by Chang *et al.* [24] are the earliest examples of general-purpose physics engines that depend on GNNs. These models do object-centric and relation-centric reasoning to predict the movements of objects in a scene. While they were successful in modeling dynamics of several systems such as n-body simulation and billiard balls, their models had certain shortcomings, especially when movements of objects have a chain effect on other objects (e.g., a pushed object pushes a group/sequence of objects it is contacting with) or when the objects are composed of complex shapes. These shortcomings can be partly handled by including a message passing structure within GNNs as done in the recent works such as [21], [25], [26]. Most of these networks used simple neural networks for encoding object and relation information. Kipf *et al.* [44] showed that variational autoencoders could be used in encoding object and relation information, where their network was shown to encode object information directly from trajectories of the objects in an unsupervised way.

Another approach was acquiring object information directly from images. Ye *et al.* [45] used image and detected the location of objects to predict the latent representation of the next time step. This latent representation was then decoded to create the image expected to be observed in the next time step. Watters *et al.* [27] and van Steenkiste *et al.* [28] proposed hybrid network models which encode object infor-

mation directly from images via CNNs and predict the next states of the objects via GNNs. Lately, these networks have been extended to handle even more complex environments. Sanchez-Gonzales *et al.* [29] showed that GNNs could be used for learning particle-based simulations that consist of more than 1000 particles.

c) Effect Prediction in Robotics: Action-effect prediction has been investigated using model-based approaches that use analytical models [14], [46], data-driven methods that use machine learning methods and hybrid methods that incorporate machine learning into analytical modeling [47], [48]. The effect prediction methods can be further divided into two categories depending on the number of involved objects. In order to deal with predicting action effects on single objects, object masks have been heavily used [11], [49]–[51]. Recently, Kopicki *et al.* [52] proposed learning multiple motion predictor models for different shaped single objects, where a vision system selects a predictor depending on the context. Seker *et al.* [53] investigated how changing object shapes affects low-level object motion trajectories and modeled it using CNNs and LSTMs.

In the context of end-to-end learning, Agrawal *et al.* [54] trained forward and inverse models for learning how to poke an object to move it into a target position. This network uses latent vectors of CNN to train predictive models. The forward model tries to predict the latent representation of the final image using the current image, and the inverse model took latent representations of both final and initial images to find the parameters of the poke action. Finn *et al.* [15] proposed a convolutional recurrent neural network [55] to predict the future image frames using only the current image frame and actions of the robot. Byravan *et al.* [17] presented an encoder-decoder like architecture to predict SE(3) motions of rigid bodies in depth data. However, the output images get blurry over time, or their predictions tend to drift away from the actual data due to the accumulated errors, making it not straightforward to use for long-term predictions in robotics.

The previous data-driven methods that directly used object-centric representations cannot deal with multiple (any number of) objects and relations as the predictors have generally fixed input and output dimensions. End-to-end approaches can handle multiple objects as their inputs and outputs are images, however, the pixel-based prediction quickly accumulated, resulting in blurry long-term predictions. Recently, GNNs that can represent multiple objects in an object-centric way have started being employed in robotics research as well. Janner *et al.* [56] used GNNs to learn object representations from perception and physics prediction jointly. Ye *et al.* [57] learned object-centric forward models for planning and control. Their model takes object bounding boxes as input and learns future state prediction from object embeddings generated by CNNs. Tung *et al.* [58] similarly use object bounding boxes with GNNs for effect prediction and control. Paus *et al.* [6] used GNNs for action-effect prediction. Sanchez-Gonzales *et al.* [59] have used graph networks as learnable physics engines in robotic setups. While previous GNN based robotic effect prediction models were successful in modeling physics, they largely overlook unknown or partial information. Our model can also handle more complex shaped objects by modeling them as a group of articulated simple shaped objects.

d) Parameter Estimation: Wu *et al.* [60] proposed a deep approach for finding the parameters of a simulation engine that predicts the future positions of the objects that slide on various tilted surfaces. Zheng *et al.* [61] used perception prediction networks, a type of graph neural network, for learning latent object properties from interaction experience to simulate system dynamics.

In many scenarios, simply observing the scene may not yield enough information, and the robot may need to actively act on the environment to perceive more. In these cases, the robot can improve its perception by actions [62]. Li *et al.* [63] used recurrent neural networks to predict the center of mass from object mask and interaction experience. Xu *et al.* [64] used a deep learning architecture for learning object properties. In their settings, a robot slides an object from an inclined surface and cause it to collide with another. Using a sequence of dynamic interactions, they showed that their model could learn to predict object representations. Kumar *et al.* [65] trained policy and predictor networks to estimate the mass distribution of articulated objects. They showed that their policy network improves the mass prediction capacity of the predictor network compared to the random policy. However, their approach was limited to articulated objects with a fixed number of parts.

In [66]–[69], researchers also studied estimating the joint relations between objects for real-time tracking and prediction of the articulated motions in challenging interactive perceptual settings. These works, however, assume expert knowledge about the joint types and hard-code the corresponding transformation matrices [67], candidate template models [66], specific measurement models [68], [69] to detect kinematic structures. Our system assumes no prior knowledge about joint dynamics, and the robot learns the dynamics of categories purely from observations. Therefore, the learning dynamics of completely novel relation types is possible with our system. Exceptionally, in [66], Sturm *et al.* proposed to learn articulation dynamics from data; however, it was only realized on a single-pair of objects from a single articulation observation (garage door motion). Furthermore, these studies do not learn or predict how the pairs or chains of non-articulated touching objects would propagate the applied forces along the cluster/chain. In contrast, our system can predict the propagated effect on groups of touching non-articulated objects.

In our work, we verified the prediction and reasoning capability of the robot in use of tools that are composed of basic primitive shapes. While our main focus is not on decomposing objects into primitives, it should be noted that this topic has been studied in the literature. For example, Deng *et al.* [70] showed that from input images, objects can be decomposed into convex hulls. In addition, they showed that these convex hulls could be used for physics simulation. Similarly, Pashevich *et al.* [71] proposed a framework that can propose different part sets where objects can be divided into, and then reconstruct the divided object in the real world with a robot using the available primitives in the workspace.

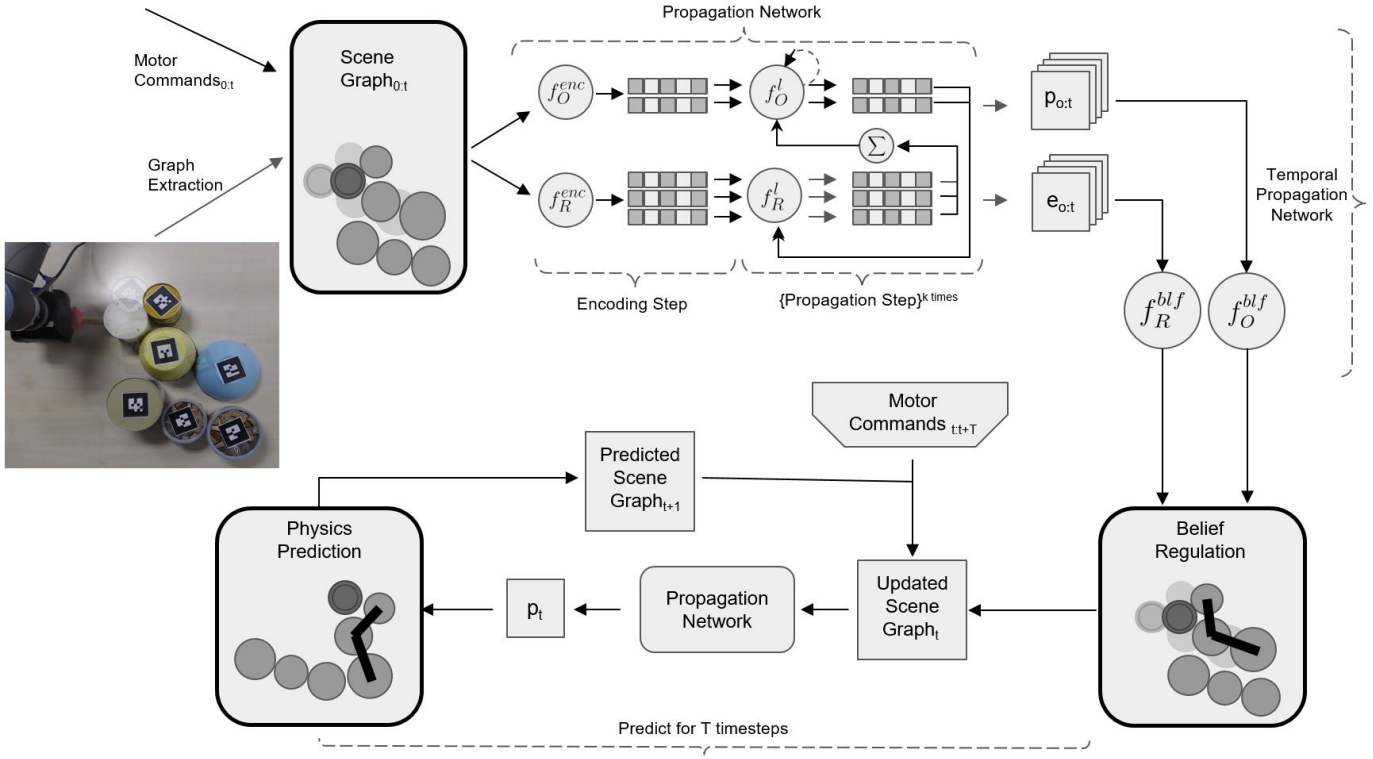


Fig. 2. Our framework extracts object- and relation-centric latent representations from the current physical scene. The latent representations are initially used to update unknown parameters of the scene graph, then with the planned motor commands, they are used for predicting future motion of the manipulated objects.

III. Proposed Framework

we propose methods and framework that are capable of learning object- and relation-centric representations for different physical scenes. These representations can be used in a set of various tasks. In this work, we designed our framework around solving two complementary tasks, namely *belief regulation* and *physics prediction*. Figure 2 shows a graphical illustration of our framework. First, object- and relation-centric representation for each object and their object-object relations are learned using propagation network. By giving these representations to RNN networks, our framework finds unknown object and relation parameters and acquires an updated graph of the scene. By passing the updated scene graph and future robot actions to the same propagation network, our framework predicts the future motion of the manipulated objects by chaining the effect predictions. In the rest of the section, more technical details will be provided.

A. Preliminaries

Physical System as a Graph: From a physical system with multiple interacting objects, we form a graph $G = \langle O, R \rangle$ where each object O is represented by the nodes (of cardinality N^o) $O = \{o_i\}_{i=1:N^o}$ and the relations R between objects such as a contact or a joint are represented by the edges (of cardinality N^r) $R = \{r_k\}_{k=1:N^r}$ of the graph.

Representing Push Manipulation Tasks: We are interested in representing the push manipulation task as a robot interacting with an object clutter. The clutter could contain many objects that may have different parts with different mass distributions, objects with possible articulations, etc. We plan to represent such a system with the aforementioned graphs $G = \langle O, R \rangle$.

Each node $o_i = \langle x_i, a_i^o \rangle$ store object or part vectors, where $x_i = \langle q_i, \dot{q}_i \rangle$ is the state of the object i, with its pose q_i and velocity $\dot{q}_i$. a_i^o stands for object properties such as shape or mass. Between each i, j node pair, there is an edge $r_k = \langle d_k, s_k, a_k^r \rangle$ that represents object-object relations where $d_k = q_i - q_j$ stands for displacement vector, $s_k = \dot{q}_i - \dot{q}_j$ stands for velocity difference, and a_k^r corresponds to properties of relation k between objects i and j.

Representation of Robot: We propose representing the end-effector of the robot as a part of the graph. For this, a robot flag and a control vector that shows how the end-effector will move in the next step are used.

Leveraging Graph Representation: For this work, our representation covers cylinders, cuboids, and objects that can be represented with the combination of two. Objects in the scene are represented with their shape, state, and other object features such as mass. Shape of objects are represented with their dimensions (the radius for cylinder and edge lengths for cuboid) and their orientations. Orientations of objects are represented with vector $[cos(\theta), sin(\theta)]$ for 2D cases, and with quaternions for 3D cases. Unlike previous work [66]–[69], the system has no prior information about how joints behave, and the articulation dynamics are left for the network to learn.

B. Physics Prediction

Propagation Network: We used propagation network as a base for learning object- and relation-centric representations. In this network, first, the state of each object and the relations between them are encoded separately. This step is shown in Figure 2 (Encoding-Step). The encoding process is achieved by use of f_R^{enc} and f_O^{enc} encoders where former process relation features $r_{k,t}$, while the latter process the object features $o_{i,t}$. $c_{k,t}^r$ and $c_{i,t}^o$ are the latent encodings of the objects and the relations.

$$c_{k,t}^r = f_R^{enc}\left(r_{k,t}\right), \quad k = 1 \dots N^r \tag{1}$$

$$c_{i,t}^o = f_O^{enc}\left(o_{i,t}\right), \quad i = 1 \dots N^o \tag{2}$$

Next, the network incorporates interactions between objects and propagations of these interactions between non-neighbor objects (e.g., force transmission between non-contacting objects) into object and relation latent vectors. This step in shown in Figure 2 (Propagation Step). For this, $c_{k,t}^r$ and $c_{i,t}^o$ are passed to propagator functions f_R^l and f_O^l respectively for estimating propagation latent vectors $e_{k,t}^l$ for relation k and $p_{i,t}^l$ for object i, for each propagation step l at time t. Using these functions in subsequent propagation steps allow for nodes and edges to accumulate propagated information from nodes and edges connected to them in $e_{k,t}^l$ and $p_{i,t}^l$.

$$e_{k,t}^l = f_R^l\left(c_{k,t}^r, p_{i,t}^{l-1}, p_{j,t}^{l-1}\right), \quad k = 1 \dots N^r \tag{3}$$

$$p_{i,t}^l = f_O^l\left(c_{i,t}^o, p_{i,t}^{l-1}, \sum_{k \in \mathcal{N}_i} e_{k,t}^{l-1}\right), \quad i = 1 \dots N^o \tag{4}$$

where $\mathcal{N}_i$ stands for set of relations object i is part of.

Effect propagation allows network to pass information between non-connected objects, and it benefits our framework in two important ways. Firstly, it allows force transmission when the robot pushes objects towards another one, effectively pushing both objects while contacting only one of them. Secondly, it allows mass and friction feedback between objects (e.g., when a light object is pushed towards a heavy object, the light object will not move the heavy objects in the push direction, but instead its motion will be shifted toward light or left side.). Figure 3 shows a simple illustration of how the robot initiates a chain of interaction and how force applied by the robot end-effector propagates. In initial propagation step, force that emerge from motion of robot is passed to contacted objects and in second propagation step, this force propagates to non-directly interacted objects. How many subsequent propagation steps to apply can be chosen based on the difficulty of the task.

Resulting $e_{k,t}^l$ and $p_{i,t}^l$ well represent the objects and their relations in the graph and can be further passed to other networks for physics prediction and belief regulation.

Physics Prediction: For each object, the latent vector $p_{i,t}^l$ can be used to predict the next state of object $x_{i,t+1}$. Given states of the objects in time t, our framework can be used for predicting the trajectory rollout of objects between time t and $t+T$ by chaining its estimates, using the predictions as an input for estimating subsequent states of objects.

C. Belief Regulation

Temporal propagation network: We propose a temporal propagation network to estimate and correct object and relation properties over time. The propagation network is augmented with long short-term memory (LSTM) networks to regulate object and relation beliefs. Network illustration is shown in Figure 2 (Temporal Propagation Network). In temporal propagation network, sequence of propagation latent vectors $e_{k,t}^l$ and $p_{i,t}^l$ are passed to LSTM-based encoder functions f_R^{blf} and f_O^{blf}. In this way, the temporal propagation network estimates and corrects object and relation properties by considering their overall state history during the robot execution.

$$o'_{i,t} = f_R^{blf}\left(p_{i,t}^L, o'_{i,t-1}\right), \quad i = 1 \dots N^o \tag{5}$$

$$r'_{k,t} = f_O^{blf}\left(e_{k,t}^L, r'_{k,t-1}\right), \quad k = 1 \dots N^r \tag{6}$$

Belief Regulation: Belief Regulation module can continuously regulate beliefs regarding objects and relations states ($o_{i,t}$ and $r_{k,t}$). These beliefs can then be used in physics prediction to compensate for errors that arise from unknown or partial information regarding the scene. This will allow our network to close the gap between its physics predictions and reality.

Weight Sharing: After training the propagation network for physics prediction, learned weights can be reused in belief regulation, preventing the framework from having to learn two separate networks. This decreases the number of parameters by about thirty percent. As we show in our experimental analysis, the representation used with physics prediction well represents the environment and can be used in transfer learning[2], without affecting the system performance.

IV. Experimental Setups

In this section, we explain the details of the experimental setups that are designed to evaluate how our model can be used for predicting object properties, relations between objects, and future object trajectories.

A. Robotic Setup

Experiments are conducted with a 6 DoF UR10 robot arm with a cylinder shaped object attached to its end-effector both in simulation and real-world. For simulation experiments, CoppeliaSim [72] with Pyrep toolkit [73] is used. For demonstrating prediction capacity of our framework, two different object setups, namely *Multiple Parts Setup* and *Different Masses Setup*, are defined. The former setup includes a diverse set of interactions in the form of joints and is designed with the aim of showing the full capacity of our framework. As the physical effects of object parameters are limited in the former setup, the latter setup is designed with the aim of showing the performance of our framework in setups where effect variation result from object parameters. In these setups, edges between objects are dynamically created as objects approach each other.

[2]In more complex tasks, fine-tuning the propagation network may be required, but physics prediction pre-training will still hasten the learning process.

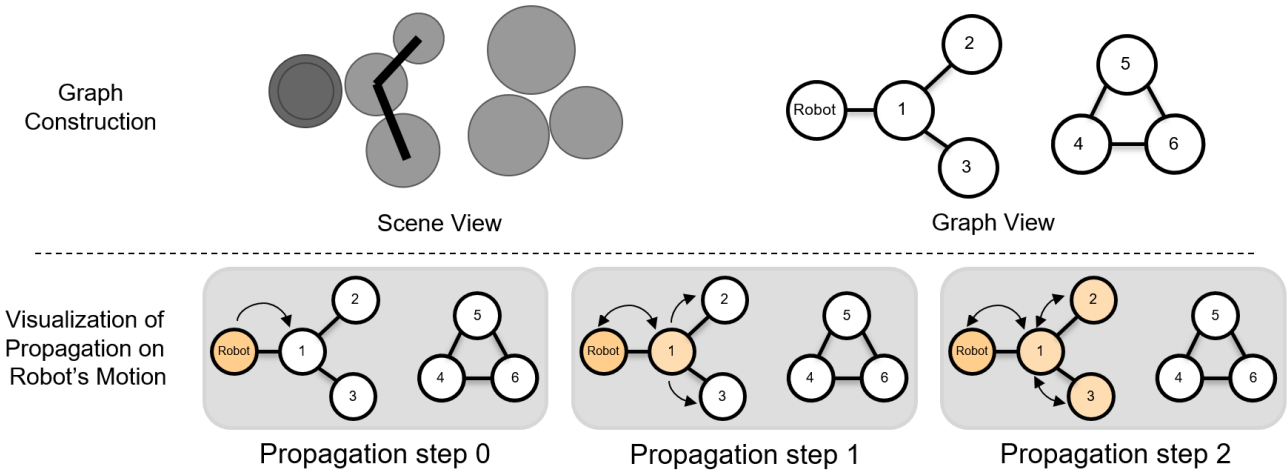
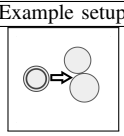
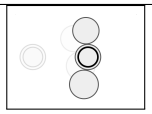
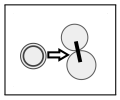
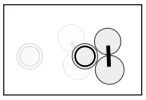
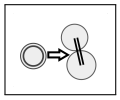
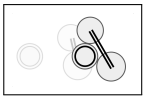
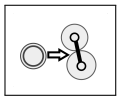
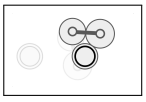


Fig. 3. This illustration shows how the graph of the scene is constructed and how the force emerging from robot end-effector motion is passed to the faraway objects. After graph construction, each node holds state information of their corresponding objects, including the robot. Considering how state information of robot is passed, in the first propagation step, it is passed to nodes of objects that contact the robot end-effector. In the second propagation step, via nodes of objects that the robot initially contact, this state information is passed to nodes of non-contacted objects.

TABLE I
EXPLANATIONS OF THE JOINT TYPES AND THEIR EFFECTS.

Example setup	Effect of action	Outcome Explanation
		*No joint:* The objects would move independent of each other as they are separated by the gripper.
		*Fixed joint:* The objects would move together with the end-effector of the robot.
		*Prismatic joint:* The object below would move in linear line along the direction between the above object to below object.
		*Revolute joint:* Both of the objects would move, but as the end-effector mainly contacts object below, the robot will rotate the object below around the object above.

Note: Objects and the robot are shown with single-edged and double-edged circles respectively, and the lines between objects represent different joint types. The arrow shows how the robot end-effector will move.

As the robot interacts with the objects in the environment, only a certain subset of objects will be in the same sub-graph of the robot (This can be seen in Figure 3 graph view.), and accordingly, this allows the system to encounter sub-graphs with a different number of objects and relations.

Multiple Parts Setup: This setup consists of a group of articulated objects where our framework should learn dynamics of objects, including cylinders and cuboids, with complex spatial relations between them. The objects may be connected to each other through three different joint relation types, namely *fixed*, *revolute* and *prismatic* joints, or they may have no joint connections between them (*no-joint*). The Illustration of these joint relations and their explanations are shown in Table I.

Different Masses Setup: This setup consists of differently massed cylindrical objects where masses of objects have an effect on their future motion. From the motion trajectories of the objects, our framework should be able to predict their masses. The masses are sampled from three intervals: $0.2 - 0.5\ kg$, $1.0 - 2.0\ kg$, $8.0 - 10.0\ kg$, representing light, normal and heavy objects, respectively.

For both of these setups, we generated datasets containing 30,000 training and 1000 validation trajectories with 9 objects. Since it is hard to exactly tune end-effector velocity to match real-world, end-effector velocity of the robot is changed between different trajectories so that it can generalize to different values. For testing the generalization capacity of the network to changing number of objects, we used trajectories consisting of 9, 6, and 12 objects, each with 1000 trajectories.

B. Implementation Details

Generation of Graph: For each object in the scene and close-by object pair, a node and two directed edges (a receiver and a sender) are created. To make the system position and orientation invariant, object position and orientations are not included in the node features. Instead, for each object-object relation, the pose of the object on the sender side of the relation is encoded with respect to frame of the object on the receiver side of the relation. After the motion of an object on its own frame is predicted, it is transformed back to the global frame.

Network information: f_O^{enc} is a two 256-dim hiddenlayer MLP, and f_R^{enc} is a three 256-dim hidden layer MLP. f_O^l and f_R^l are MLPs with 256-dim single hidden layer. f_O^l and f_R^l are chosen to have a low number of layers since these network called multiple times successively and therefore more costly to use than f_O^{enc} and f_R^{enc}. Finally, f_O^{blf} and f_R^{blf} are LSTM with 256 neurons. For physics prediction, outputs of f_O^l is given to an MLP with one hidden layer and one linear layer to predict velocity ($\dot{q}_i$) of each object; and for belief regulation, outputs of f_O^{blf} and f_R^{blf} are given to an MLP with single linear layer to predict object masses and joint relations.

In the belief regulation module, as more interaction experience is acquired, the framework is expected to have higher

accuracy in identifying initially unknown parameters of the environment. For this reason, the loss function is scaled in a way that further time-steps have a higher loss value compared to earlier time-steps. Besides, to make networks predictions smooth and preventing them from oscillating between different outcomes, outputs of f_O^{blf} and f_R^{blf} are regularized by applying MSE loss between latent vectors of successive time-steps.

The network is trained with 16 batch-size and 3e-4 learning rate using Adam optimizer [74] with AMSgrad [75]. The learning rate is reduced by 0.8 when the validation error stopped decreasing for a window of 20 epochs. Networks are trained for 1000 epochs. The physics prediction module is trained with epochs of 10,000 batches of randomly sampled time-steps, and for the training belief regulation, 200 batches of randomly sampled trajectories from the training scenes are used.

First, our network is trained on physics prediction. After the training is complete, the weights of the shared part of the network are frozen, and then the belief regulation module is trained. To increase the performance of physics prediction, we used scheduled sampling [31]. Using Nvidia P100 GPU, the physics prediction and belief regulation modules are trained for two and one days respectively.

V. Results

For quantitative analysis, our framework is evaluated in joint prediction and mass prediction tasks. For the relation prediction case, our results are compared with PropNets with three different relation assignment strategies.

1) **Oracle** This relation assignment strategy utilizes ground-truth relations. In the ideal case, as more interactions are observed, the performance of our framework should approach to oracle.
2) **No-Joint** This relation assignment strategy assumes there are no joints in the scene.
3) **All-Fixed** This relation assignment strategy assumes a fixed joint between every contacting object pairs.

A. Quantitative Analysis in Multiple Parts Setup

For evaluating the physics prediction module, our framework is tested with the oracle relation assignment strategy in *multiple parts setup*. In this setup, while collecting each trajectory, the robot executes 9 linear pushes of 30 cm, contacting with a most diverse set of objects. In this setup, outputs of physics prediction module are chained to predict multiple time-step trajectory roll-outs (i.e., essentially simulating the environment with network predictions). These trajectory roll-outs are used in evaluation. Figure 4 presents the performance in scenarios with different number of objects. As the length of predicted trajectory roll-outs increase, the errors in higher number of trajectories accumulate. This results in trajectories to drift away from the ground truth. In Figure 4, on the left, as the roll-out length is shorter in each environment setup, more than 600 trajectories have lower mean error than $0.1\ cm$, and most of the remaining trajectories have a lower mean error than $0.4\ cm$. On the right, the roll-out length is longer, and less than 400 trajectories have a lower mean error than $0.1\ cm$. Besides

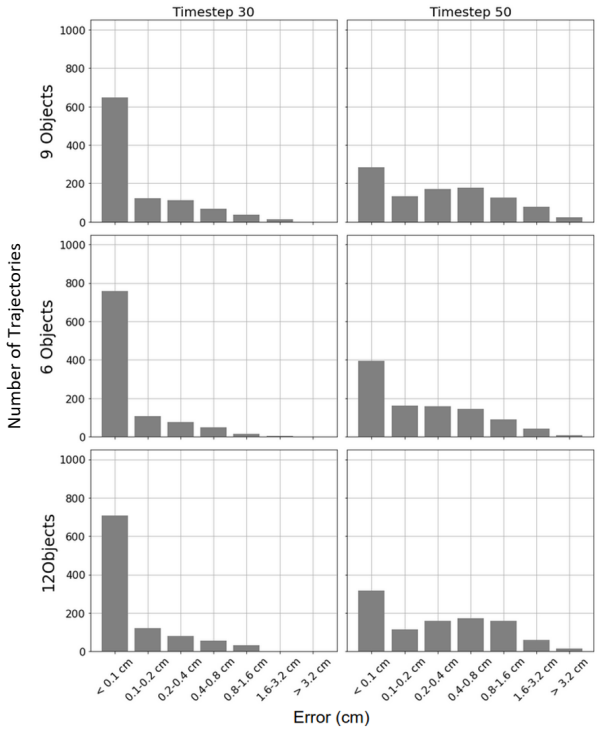


Fig. 4. Physics prediction results on articulated object environments.

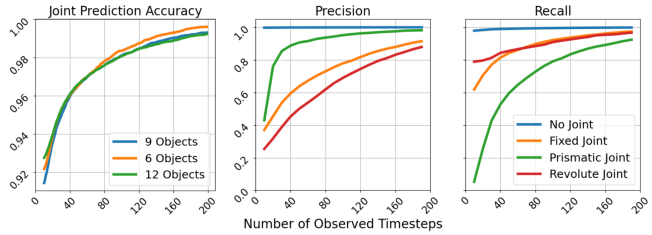


Fig. 5. Belief regulation results on articulated object environments.

for the rest of the trajectories, there are more trajectories in high mean error bins.

Next, the belief regulation module is evaluated on prediction of joint relations. As shown in Figure 5, as the robot interacts with the objects and higher number of observation data is acquired, our network becomes better at predicting the joint relation types more accurately. The joint prediction plot in Figure 5 shows that our method performs similarly independent of the number of objects used due to the underlying graph structure. In the same figure, on precision plot, no joint (blue) and prismatic joint(green) lines show that networks are good at identifying whether there is a joint between two objects and whether this joint is prismatic. Compared to the prismatic joint, the model is more likely to make erroneous predictions on whether a joint is fixed or revolute. This is likely because without interaction experience, it is easier for network to mix these two joints. Nonetheless, from the recall plot, we can see that the model can correct its predictions on fixed and revolute as it observes more robot interactions. From both precision and recall plots, the model abstains from predicting a joint prismatic unless it is certain. This may be because

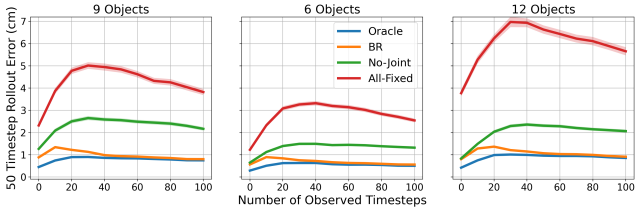


Fig. 6. Results of coupled system on articulated object environments.

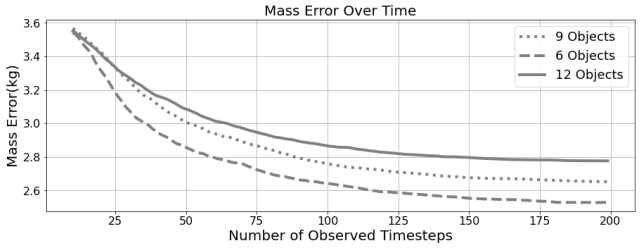


Fig. 7. Belief Regulation results on mass prediction. As more motion is observed in the scene, mass prediction error decreases, but eventually converges to about $2.7\ kg$ mean error.

prismatic joint dynamics are similar to no-joint dynamics unless robot gains enough observations about objects the joint are connected to.

Finally, the coupled results of the physics prediction and the belief regulation modules can be seen in Figure 6. The lines show the mean errors, and the shaded regions show the standard error. As expected, physics prediction done with no-joint and all-fixed relation assignment strategies performed poorly. This is because these relation assignment strategies do not learn from interactions. As the number of observed time-steps increases, the mean error of the coupled modules decreases and eventually in 40 time-steps, it reaches to the mean error of the physics prediction of the oracle system that has access to ground-truth joint relations.

B. Quantitative Analysis of Belief Regulation for Mass Prediction

We design *different masses* experimental setup for further testing the object-centric prediction capacity of our framework. In this setup, in each trajectory, the robot executes a total of 3 linear pushes of 30 cm, scattering objects as much as possible. In this experiment, our framework should predict object masses, and as the robot acquires more observations, it should improve its mass prediction accuracy further. Mean errors for mass prediction is shown in Figure 7. Considering the distribution masses, our model manages to decrease mass errors over time as it acquires more observations. However, the predictions seem to not go below a certain value. This may be because the robot has limited interaction with the objects in the scene, and this limits the capacity of the model to predict masses of objects correctly.

To further analyze the performance of our system in mass prediction, we prepared two controlled environment test setups to examine why mass error does not decrease below a certain value. These setups can be seen in Figure 8. In these setups,

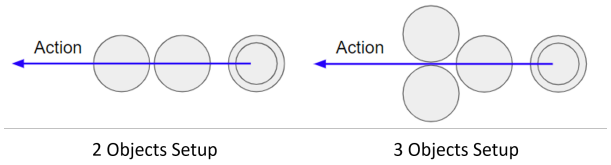


Fig. 8. Visualization of controlled environment setups for mass prediction. In these configurations, object masses are changed between different runs while keeping robot motion and object shapes the same.

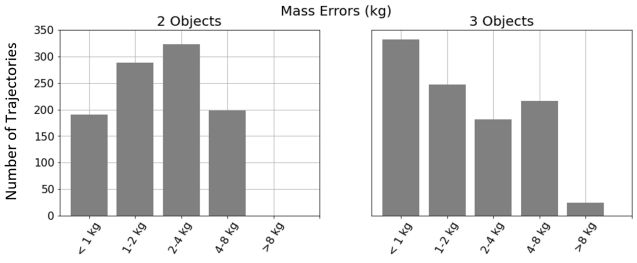


Fig. 9. Mass prediction results in controlled environments. In many cases, our model acquires low error, however there are still many cases that have high error.

we only change the mass of objects while setting robot action, initial positions of objects, and shapes of objects same. The robot manipulates each object, so it should be possible for the network to predict mass if it is predictable. The results obtained in the these controlled settings are provided in Figure 9. Considering the mass distribution of the objects, the first two bars of both plots show that our framework predicts light and medium within their cluster correctly half of the time. The third and fourth bin shows that our framework sometimes mixes light and medium objects and medium and heavy objects. For three objects, the fifth bin shows that our framework mixes light and heavy objects in rare cases[3] A number of representative correct and incorrect predictions are provided in Figure 10. We investigated setups where the network made high-error in mass predictions and observe that there are cases where different objects mass configurations having same object motions. Figure 10C and Figure 10D, the robot observes very similar trajectories with 0.15 cm difference between them, despite the interacted objects having very different masses. In these scenes, the network makes very similar predictions. However, only in the former scene, it is correct.

C. Qualitative Analysis - Tool Usage

We design a tool manipulation and planning experiment. Given a goal position, the aim is to select the best tool and action sequence to bring a given object to the goal position using the corresponding tool. In addition, this experiment aims to show generalization capacity of our framework by transferring representation and the network trained in *multiple parts setup* for modelling novel tools that are not encountered in the training distribution.

[3]Videos of the results are available at project page.

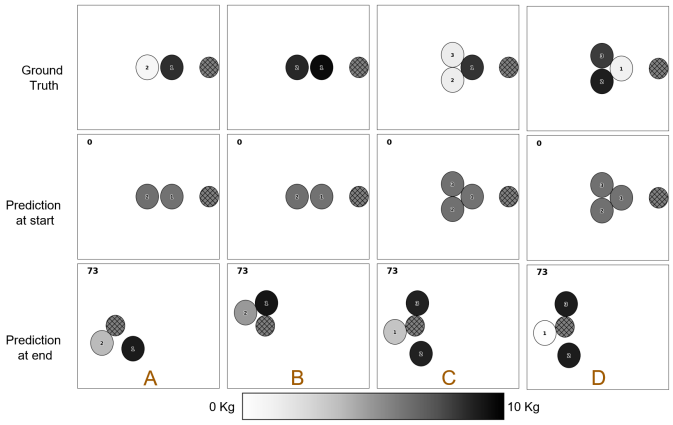


Fig. 10. Mass predictions for two very close observations. The same observations are acquired from scenes with two different mass configurations, and our framework could not differentiate between the two. Our framework makes the same mass prediction for both; one of them is correctly predicted, while the other is not.

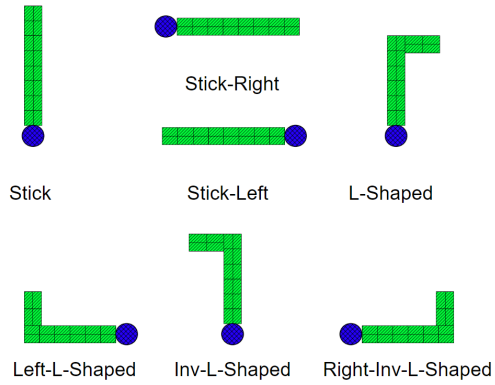


Fig. 11. Tools used In tool selection and planning experiments.

In this experiment, stick, L-shaped tool, inv-L-shaped tool, and their various configurations are used as shown in Figure 11. These tools are represented as multi-part objects composed of cuboids and fixed joints, and are attached to robot end-effector. The robot uses linear pushes in principal directions to manipulate the object on the table. In these actions, tool motion is modeled kinematically and not updated from the network prediction. Please note that a new network is not trained and the results obtained by the previously trained network are reported.

In each test case, the robot should select one of the available tools and apply three pushes of 20 cm in principle directions to move an object to a given goal position. To make all test cases feasible, goal points are generated through simulation. More specifically, 24 uniform initial positions are generated from $-0.7 \leq x \leq -0.1$ and $-0.5 \leq y \leq 0.5$ for. Then, on each initial position, a cylindrical object is generated, and all possible action sequences are applied using each of the tools. The final positions of objects are recorded. These final positions are filtered where if a final position of object is less than 5 cm away from its initial position, it is removed. Besides, if the difference between any two initial and final position pair is lower than 5 cm, one of them is removed as well. In this

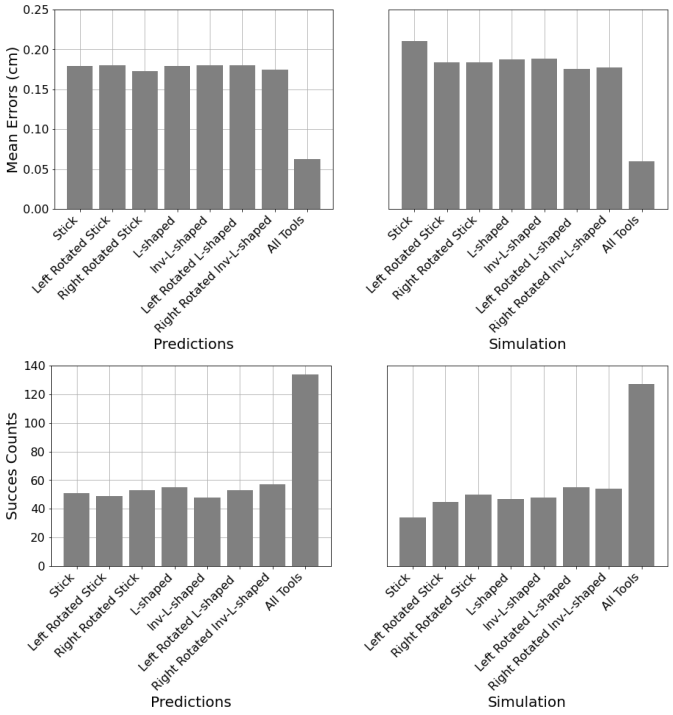


Fig. 12. Tool results. As robot is allowed to use wider variety of tools, success rate increases and error amount decreases.

way, a dataset for tool and action selection that contains 166 completely diverse solvable initial and final position pairs are generated.

The task is defined as the selection of the best tool and best action sequence from all possible tools and action sequences. The network is run for all the initial-target position pairs for each possible tool and action sequences. For each of these pairs, the tool and action sequence that gives the lowest mean error is selected. Besides, for comparison, to see whether our framework can utilize each of the tools, the best action sequences for each tool are found as well. Then, each solution is transferred to simulation to testing their correctness.

The results can be seen in Figure 12. The left column shows the prediction errors of selected action sequences, and the right column shows actual errors of selected actions when they are run on simulation. The first row shows the mean error between the final positions of manipulated objects and the goal positions. The second row shows the number of successful action sequences (i.e., action sequences where the final position of the object is less than 5 cm away from its target position.). Each bar corresponds to the result for action selection with a particular tool, and with the last one, the tool can be selected as well. From the figure, it can be seen that our framework managed to utilize all tools for solving about 40 of the tasks, and when all tools are allowed to be used, about 130 of the tasks are solvable. Comparing prediction and simulation results shows that predictions made by our framework are plausible, and there is just marginal loss of performance when found action sequences are transferred to simulation. Our framework is successful in tool manipulation and action selection despite its not being designed for such a

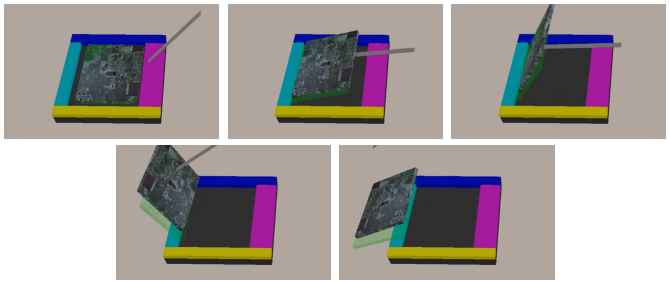


Fig. 13. Snapshots of 6D Effect Prediction. Ground truth pose of object is shown with transparent cuboid. As can be seen, prediction is very close to ground truth.

Fig. 14. Snapshots of a robot interaction in real-world. Our framework continuously updates its joint predictions as it observes the motion of objects and predicts their future positions.

task.

D. Qualitative Analysis in Simulation - 6D Motion Prediction

Finally, we designed an experimental setup where we can test our framework on 6D rigid body motion prediction. In this setup, the robot is tasked to lever up a printed circuit board (PCB) from a hard drive disk (HDD) with a screwdriver tool. PCB is on top of the HDD, and at each side of the HDD, there may be a ledge that PCB may contact while being levered up. PCB and HDD are represented as a set of boxes, and their sizes change between runs. Note that some sides of the HDD may have no ledge in different scenes, and therefore, while representing a scene in a graph, the number of nodes changes between runs.

For scene generation, lengths of both sides of HDD are set to 20 cm. There is either a ledge of size between 0 to 8 cm, or no ledge at each side of HDD. In the middle of HDD, a PCB with its side lengths between 10 to 20 cm are generated. The network is trained using 500 lever-up interactions on scenes with 125 different procedurally generated hard-disks (One lever-up action from each side of HDD).

A sample prediction can be seen in Figure 13. Our further results on this setup can be found on project page. In this setup, our network make plausible predictions that match well with the ground truth.

E. Analysis of our framework in real world

In this section, our framework is evaluated with a real-world dataset, presented in [30]. In this dataset, a UR10 robot arm holds a hammer and use it for pushing objects. The dataset contains cylinder-shaped objects and possible fixed joints between them. The effect of a fixed joint between objects is mimicked by placing customized card-boards under them. A sample created scene and how the robot makes its manipulation on objects can be found in Figure 14. As the dataset does

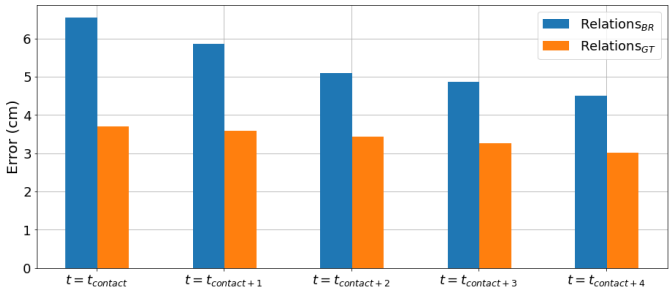


Fig. 15. Average errors (in cm) change in real world as robot makes its first contact with the objects.

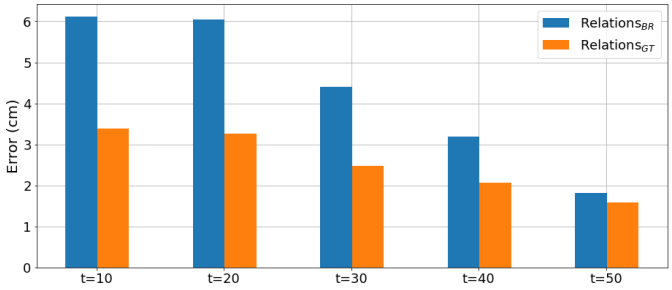


Fig. 16. Average errors (in cm) change in real-world as our framework acquires more object tracking information.

not have angle information, our network is retrained with the angles of cylinders removed. Since it is also possible for our network to predict revolute or prismatic joint, prediction are limited only to no-joint and fixed joint relations[4] (By selecting the joint relation with the max probability between no-joint and fixed joint relations.).

The dataset contains scenes with 2 to 5 cylindrical objects and 1 to 3 fixed joint relations between them. In total, there are 102 different test setups in the dataset. On average, objects move 19.5 cm, and our physics prediction network achieves 3.5 cm in predicting final object positions where [30] achieved 6.6 cm in the same test. Our coupled framework is further analyzed with the same dataset in Figure 15. Similar to [30], we tested our framework on exact timesteps where the first contact between robot and objects occurs. Our network manages to acquire better results than the one in [30] for both physics prediction with ground truth and with predicted relations (In [30], prediction with ground truth and predicted relations acquires 6.5 cm and 8.5 cm at time t and 4 cm and 6.5 cm at time $t + 4$.). In Figure 16, performance of our framework on different time-steps is shown where predictions of our framework catch up to the ground truth as more observations are acquired.

VI. Conclusion

We presented methods and a framework for learning action-effects in object and relation-centric push manipulation tasks. Our framework allows the robot to correct its belief about object and relation parameters as it interacts with the scene

[4]Unlike [30], we do not retrain our network with only cylindrical objects and fixed joints; we only remove angle information of cylindrical objects.

and observe the effects of its actions. It then can continuously predict the future dynamics of objects. We have tested belief regulation and physics prediction performance on multiple experiments, including a real-world one. We have shown that our framework can predict joint types in articulated object settings with different object and relation types, masses of objects, and their future motion. We have shown that our framework can be extended for 6D trajectory prediction. Furthermore, we also validated our framework on action selection in a tool manipulation task. Although we do not train a new network that includes situations that are not present in our articulated object setting, our network was successfully transferred to this new domain and succeeded in finding action sequences that complete the given tasks.

As our framework is very generic, we believe it can be further refined and extended. First, our framework can benefit from intelligent exploration strategies that can generalize to a changing number of objects. In addition, learning of unsupervised representations for objects via interactions can be very powerful for the visual grounding of objects. In future work, we are planning to extend our framework for these adaptations.

REFERENCES

[1] F. Ruggiero, V. Lippiello, and B. Siciliano, "Nonprehensile dynamic manipulation: A survey," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1711–1718, 2018.

[2] J. Stüber, C. Zito, and R. Stolkin, "Let's push things forward: A survey on robot pushing," *Frontiers in Robotics and AI*, vol. 7, p. 8, 2020.

[3] J. Stüber, M. Kopicki, and C. Zito, "Feature-based transfer learning for robotic push manipulation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–5.

[4] M. R. Dogar and S. S. Srinivasa, "Push-grasping with dexterous hands: Mechanics and a method," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 2123–2130.

[5] J. E. King, M. Klingensmith, C. M. Dellin, M. R. Dogar, P. Velagapudi, N. S. Pollard, and S. S. Srinivasa, "Pregrasp manipulation as trajectory optimization." in *Robotics: Science and Systems*. Berlin, 2013.

[6] F. Paus, T. Huang, and T. Asfour, "Predicting pushing action effects on spatial object relations by learning internal prediction models," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 10 584–10 590.

[7] T. Meriçli, M. Veloso, and H. L. Akın, "Push-manipulation of complex passive mobile objects using experimentally acquired motion models," *Autonomous Robots*, vol. 38, no. 3, pp. 317–329, 2015.

[8] H. Van Hoof, O. Kroemer, H. B. Amor, and J. Peters, "Maximally informative interaction learning for scene exploration," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5152–5158.

[9] A. Eitel, N. Hauff, and W. Burgard, "Learning to singulate objects using a push proposal network," in *Robotics Research*. Springer, 2020, pp. 405–419.

[10] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, "Learning synergies between pushing and grasping with self-supervised deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4238–4245.

[11] D. Omrčen, C. Böge, T. Asfour, A. Ude, and R. Dillmann, "Autonomous acquisition of pushing actions to support object grasping with a humanoid robot," in *2009 9th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2009, pp. 277–283.

[12] D. Kappler, L. Y. Chang, N. S. Pollard, T. Asfour, and R. Dillmann, "Templates for pre-grasp sliding interactions," *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 411–423, 2012.

[13] S. Elliott, M. Valente, and M. Cakmak, "Making objects graspable in confined environments through push and pull manipulation with a tool," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 4851–4858.

[14] K.-T. Yu, M. Bauza, N. Fazeli, and A. Rodriguez, "More than a million ways to be pushed. a high-fidelity experimental dataset of planar pushing," in *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2016, pp. 30–37.

[15] C. Finn, I. Goodfellow, and S. Levine, "Unsupervised learning for physical interaction through video prediction," in *Advances in neural information processing systems*, 2016, pp. 64–72.

[16] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 2786–2793.

[17] A. Byravan and D. Fox, "SE3-Nets: Learning rigid body motion using deep neural networks," in *International Conference on Robotics and Automation*, 2017, pp. 173–180.

[18] I. Nematollahi, O. Mees, L. Hermann, and W. Burgard, "Hindsight for foresight: Unsupervised structured dynamics models from physical interaction," *arXiv preprint arXiv:2008.00456*, 2020.

[19] E. S. Spelke, K. Breinlinger, J. Macomber, and K. Jacobson, "Origins of knowledge." *Psychological review*, vol. 99, no. 4, p. 605, 1992.

[20] J. B. Tenenbaum, C. Kemp, T. L. Griffiths, and N. D. Goodman, "How to grow a mind: Statistics, structure, and abstraction," *science*, vol. 331, no. 6022, pp. 1279–1285, 2011.

[21] D. Mrowca, C. Zhuang, E. Wang, N. Haber, L. F. Fei-Fei, J. Tenenbaum, and D. L. Yamins, "Flexible neural representation for physics prediction," in *Advances in neural information processing systems*, 2018, pp. 8813–8824.

[22] P. Battaglia, J. B. C. Hamrick, V. Bapst, A. Sanchez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. E. Dahl, A. Vaswani, K. Allen, C. Nash, V. J. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, "Relational inductive biases, deep learning, and graph networks," *arXiv*, 2018. [Online]. Available: https://arxiv.org/pdf/1806.01261.pdf

[23] P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende *et al.*, "Interaction networks for learning about objects, relations and physics," in *Advances in neural information processing systems*, 2016, pp. 4502–4510.

[24] M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum, "A compositional object-based approach to learning physical dynamics," *arXiv preprint arXiv:1612.00341*, 2016.

[25] 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 *International Conference on Robotics and Automation*, 2019, pp. 1205–1211.

[26] Y. Li, J. Wu, R. Tedrake, J. B. Tenenbaum, and A. Torralba, "Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids," in *International Conference on Learning Representations*, 2019.

[27] N. Watters, D. Zoran, T. Weber, P. Battaglia, R. Pascanu, and A. Tacchetti, "Visual interaction networks: Learning a physics simulator from video," in *Advances in neural information processing systems*, 2017, pp. 4539–4547.

[28] S. van Steenkiste, M. Chang, K. Greff, and J. Schmidhuber, "Relational neural expectation maximization: Unsupervised discovery of objects and their interactions," in *International Conference on Learning Representations*, 2018.

[29] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia, "Learning to simulate complex physics with graph networks," in *International Conference on Machine Learning*. PMLR, 2020, pp. 8459–8468.

[30] A. E. Tekden, A. Erdem, E. Erdem, M. Imre, M. Y. Seker, and E. Ugur, "Belief regulated dual propagation nets for learning action effects on groups of articulated objects," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 10 556–10 562.

[31] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 1171–1179.

[32] J. R. Kubricht, K. J. Holyoak, and H. Lu, "Intuitive physics: Current research and controversies," *Trends in cognitive sciences*, vol. 21, no. 10, pp. 749–759, 2017.

[33] P. W. Battaglia, J. B. Hamrick, and J. B. Tenenbaum, "Simulation as an engine of physical scene understanding," *Proceedings of the National Academy of Sciences*, vol. 110, no. 45, pp. 18 327–18 332, 2013.

[34] J. B. Hamrick, P. W. Battaglia, T. L. Griffiths, and J. B. Tenenbaum, "Inferring mass in complex scenes by mental simulation," *Cognition*, vol. 157, pp. 61–76, 2016.

[35] K. Smith, L. Mei, S. Yao, J. Wu, E. Spelke, J. Tenenbaum, and T. Ullman, "Modeling expectation violation in intuitive physics with coarse probabilistic object representations," in *Advances in Neural Information Processing Systems*, 2019, pp. 8983–8993.

[36] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on machine learning (International Conference on Machine Learning)*, 2011, pp. 465–472.

[37] A. Lerer, S. Gross, and R. Fergus, "Learning physical intuition of block towers by example," in *International Conference on Machine Learning*, 2016, pp. 430–438.

[38] O. Groth, F. B. Fuchs, I. Posner, and A. Vedaldi, "Shapestacks: Learning vision-based physical intuition for generalised object stacking," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 702–717.

[39] W. Li, S. Azimi, A. Leonardis, and M. Fritz, "To fall or not to fall: A visual approach to physical stability prediction," *arXiv preprint arXiv:1604.00066*, 2016.

[40] R. Mottaghi, H. Bagherinezhad, M. Rastegari, and A. Farhadi, "Newtonian scene understanding: Unfolding the dynamics of objects in static images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3521–3529.

[41] R. Mottaghi, M. Rastegari, A. Gupta, and A. Farhadi, ""what happens if..." learning to predict the effect of forces in images," in *European Conference on Computer Vision*. Springer, 2016, pp. 269–285.

[42] K. Fragkiadaki, P. Agrawal, S. Levine, and J. Malik, "Learning visual predictive models of physics for playing billiards," in *International Conference on Learning Representations*, 2016.

[43] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[44] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel, "Neural relational inference for interacting systems," in *International Conference on Machine Learning*. PMLR, 2018, pp. 2688–2697.

[45] Y. Ye, M. Singh, A. Gupta, and S. Tulsiani, "Compositional video prediction," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 10 353–10 362.

[46] F. R. Hogan and A. Rodriguez, "Feedback control of the pusher-slider system: A story of hybrid and underactuated contact dynamics," in *Algorithmic Foundations of Robotics XII*. Springer, 2020, pp. 800–815.

[47] J. Zhou, M. T. Mason, R. Paolini, and D. Bagnell, "A convex polynomial model for planar sliding mechanics: theory, application, and experimental validation," *The International Journal of Robotics Research*, vol. 37, no. 2-3, pp. 249–265, 2018.

[48] A. Kloss, S. Schaal, and J. Bohg, "Combining learned and analytical models for predicting action effects," *arXiv preprint arXiv:1710.04102*, 2017.

[49] J. King, J. A. Haustein, S. S. Srinivasa, and T. Asfour, "Nonprehensile whole arm rearrangement planning with physics manifolds," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 2508–2515.

[50] J. A. Haustein, J. King, S. S. Srinivasa, and T. Asfour, "Kinodynamic randomized rearrangement planning via dynamic transitions between statically stable states," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 3075–3082.

[51] M. Kopicki, S. Zurek, R. Stolkin, T. Mörwald, and J. Wyatt, "Learning to predict how rigid objects behave under simple manipulation," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 5722–5729.

[52] M. Kopicki, S. Zurek, R. Stolkin, T. Moerwald, and J. L. Wyatt, "Learning modular and transferable forward models of the motions of push manipulated objects," *Autonomous Robots*, vol. 41, no. 5, pp. 1061–1082, 2017.

[53] M. Y. Seker, A. E. Tekden, and E. Ugur, "Deep effect trajectory prediction in robot manipulation," *Robotics and Autonomous Systems*, vol. 119, pp. 173 – 184, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889019300740

[54] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine, "Learning to poke by poking: Experiential learning of intuitive physics," in *Advances in Neural Information Processing Systems*, 2016, pp. 5074–5082.

[55] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in neural information processing systems*, 2015, pp. 802–810.

[56] M. Janner, S. Levine, W. T. Freeman, J. B. Tenenbaum, C. Finn, and J. Wu, "Reasoning about physical interactions with object-centric models," in *International Conference on Learning Representations*, 2019.

[57] Y. Ye, D. Gandhi, A. Gupta, and S. Tulsiani, "Object-centric forward modeling for model predictive control," in *Conference on Robot Learning*. PMLR, 2020, pp. 100–109.

[58] H.-Y. F. Tung, Z. Xian, M. Prabhudesai, S. Lal, and K. Fragkiadaki, "3d-oes: Viewpoint-invariant object-factorized environment simulators," *arXiv preprint arXiv:2011.06464*, 2020.

[59] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia, "Graph networks as learnable physics engines for inference and control," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4470–4479.

[60] J. Wu, I. Yildirim, J. J. Lim, B. Freeman, and J. Tenenbaum, "Galileo: Perceiving physical object properties by integrating a physics engine with deep learning," in *Advances in neural information processing systems*, 2015, pp. 127–135.

[61] D. Zheng, V. Luo, J. Wu, and J. B. Tenenbaum, "Unsupervised learning of latent physical properties using perception-prediction networks," *arXiv preprint arXiv:1807.09244*, 2018.

[62] J. Bohg, K. Hausman, B. Sankaran, O. Brock, D. Kragic, S. Schaal, and G. S. Sukhatme, "Interactive perception: Leveraging action in perception and perception in action," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1273–1291, 2017.

[63] J. K. Li, W. S. Lee, and D. Hsu, "Push-net: Deep planar pushing for objects with unknown physical properties." in *Robotics: Science and Systems*, vol. 14, Pittsburgh, Pennsylvania, June 2018.

[64] Z. Xu, J. Wu, A. Zeng, J. B. Tenenbaum, and S. Song, "Densephysnet: Learning dense physical object representations via multi-step dynamic interactions," in *Robotics: Science and Systems (RSS)*, 2019.

[65] N. K. Kannabiran, I. Essa, and C. K. Liu, "Estimating mass distribution of articulated objects through physical interaction," *arXiv preprint arXiv:1907.03964*, 2019.

[66] J. Sturm, V. Pradeep, C. Stachniss, C. Plagemann, K. Konolige, and W. Burgard, "Learning kinematic models for articulated objects," in *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

[67] T. Schmidt, R. A. Newcombe, and D. Fox, "Dart: Dense articulated real-time tracking." in *Robotics: Science and Systems*, vol. 2, no. 1. Berkeley, CA, 2014.

[68] R. Martín-Martín, S. Höfer, and O. Brock, "An integrated approach to visual perception of articulated objects," in *International Conference on Robotics and Automation*. IEEE, 2016, pp. 5091–5097.

[69] R. Martín-Martín and O. Brock, "Coupled recursive estimation for online interactive perception of articulated objects," *The International Journal of Robotics Research*, 2019.

[70] B. Deng, K. Genova, S. Yazdani, S. Bouaziz, G. Hinton, and A. Tagliasacchi, "Cvxnet: Learnable convex decomposition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 31–44.

[71] A. Pashevich, I. Kalevatykh, I. Laptev, and C. Schmid, "Learning visual policies for building 3d shape categories," *arXiv preprint arXiv:2004.07950*, 2020.

[72] E. Rohmer, S. P. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 1321–1326.

[73] S. James, M. Freese, and A. J. Davison, "Pyrep: Bringing v-rep to deep robot learning," *arXiv preprint arXiv:1906.11176*, 2019.

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

[75] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," in *International Conference on Learning Representations*, 2018.

ACKNOWLEDGMENT

This research has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 731761, IMAGINE; supported by a TUBA GEBIP fellowship awarded to E. Erdem; and supported by a Tubitak 2210-A scholarship awarded to A.E. Tekden.

The numerical calculations reported in this work were partially performed at TUBITAK ULAKBIM, High Performance and Grid Computing Center (TRUBA resources).