CHALMERS
UNIVERSITY OF TECHNOLOGY

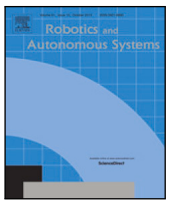
Object and relation centric representations for push effect prediction

Robotics and Autonomous Systems 174 (2024) 104632

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot



Object and relation centric representations for push effect prediction

Ahmet E. Tekden [a,b,*], Aykut Erdem [c], Erkut Erdem [d], Tamim Asfour [e], Emre Ugur [a]

[a] Computer Engineering Department, Bogazici University, Turkey
[b] Electrical Engineering Department, Chalmers University of Technology, Sweden
[c] Computer Engineering Department, Koç University, Turkey
[d] Computer Engineering Department, Hacettepe University, Turkey
[e] Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology, Germany

ARTICLE INFO

Keywords:
Push manipulation
Effect prediction
Parameter estimation
Graph neural networks
Interactive perception
Articulation prediction

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, and it outperforms image-based representations on physics prediction. Our approach enables the robot to predict and adapt the effect of a pushing action as it observes the scene. It can also be used for tool manipulation with never-seen tools. Further, we demonstrate 6D effect prediction in the lever-up action in the context of robot-based hard-disk disassembly.

1. 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 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. Fig. 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.

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 at the 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

* Corresponding author at: Electrical Engineering Department, Chalmers University of Technology, Sweden.
E-mail address: tekden@chalmers.se (A.E. Tekden).

https://doi.org/10.1016/j.robot.2024.104632
Received 22 February 2023; Received in revised form 14 September 2023; Accepted 8 January 2024
Available online 17 January 2024

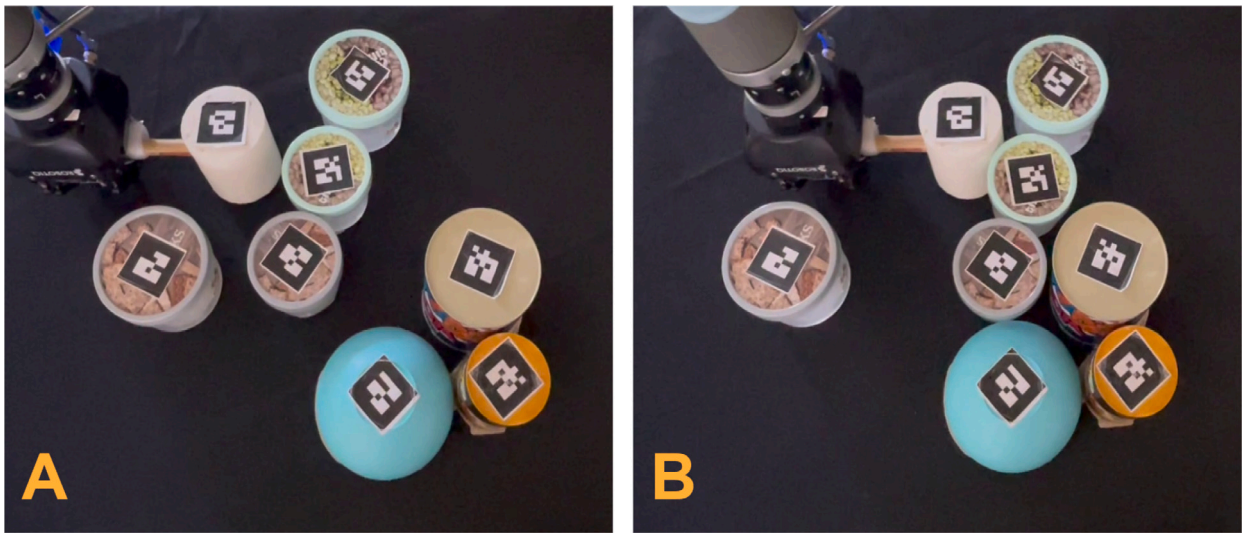


Fig. 1. We will normally expect the action of the robot on the left image to scatter contacted objects. However, seeing the contacted objects moving together, the robot should correct its belief to enable this dynamic.

relation-centric representations and they are heavily used in modeling 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 representations 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 an 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. In addition, we have shown that our framework can make 6D effect predictions.

In our previous work [30], we introduced a GNN based network for effect prediction and parameter estimation. In this current work, we have significantly extended and improved upon our previous approach in several key aspects. Initially, our model in [30] relied on two independent networks: one for physics prediction and one for parameter estimation. However, we have now devised a novel weight-sharing mechanism that enables the fusion of object- and relation-centric representations. This has led to a significant reduction of approximately thirty percent in the number of learnable parameters. Additionally, while our earlier work [30] was limited to modeling cylindrical objects, our current approach extends the input representations to handle diverse object shapes. We can now represent complex-shaped objects constructed from multiple cuboids and cylinders, and predict object masses. This significant extension empowers our framework to handle a wide range of tools and objects not encountered previously, making it more versatile and applicable to real world scenarios. In addition to these advancements, we have also improved the training process of the underlying networks by incorporating scheduled sampling [31], enhancing data distribution, and refining parameter estimation supervision. These comprehensive enhancements collectively elevate the capabilities and performance of our approach in both physics prediction and parameter estimation tasks. Overall, our extended GNN-based network not only outperforms our previous work [30] but also demonstrates greater adaptability and robustness, making it a promising tool for effect prediction in complex robotic systems. 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 that are usable in new tasks.
- We show the feasibility of articulation based graph representations for modeling multi-part objects and show that it outperforms object-centric image based representation in physics prediction task.
- We show that the proposed network outperforms a coarse-physics based baseline on model predictive control.
- We design a novel 6D 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.

2. Related work

Learning dynamics/modeling 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 modeled 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

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

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.

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 information 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.

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 [56]. Janner et al. [57] used GNNs to learn object representations from perception and physics prediction jointly. Ye et al. [58] 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. [59] similarly use object bounding boxes with GNNs for effect prediction and control. Lin and Weng et al. [60] employed GNNs for learning cloth dynamics. Paus et al. [6] used GNNs for action-effect prediction. Sanchez-Gonzales et al. [61] 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.

Parameter estimation. Wu et al. [62] 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. [63] 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 engage with the environment to perceive more. In these cases, the robot can improve its perception by actions [64]. Li et al. [65] used recurrent neural networks to predict the center of mass from object mask and interaction experience. Xu et al. [66] 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. [67] 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 [68–71], 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 [69], candidate template models [68], specific measurement models [70,71] 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 [68], 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

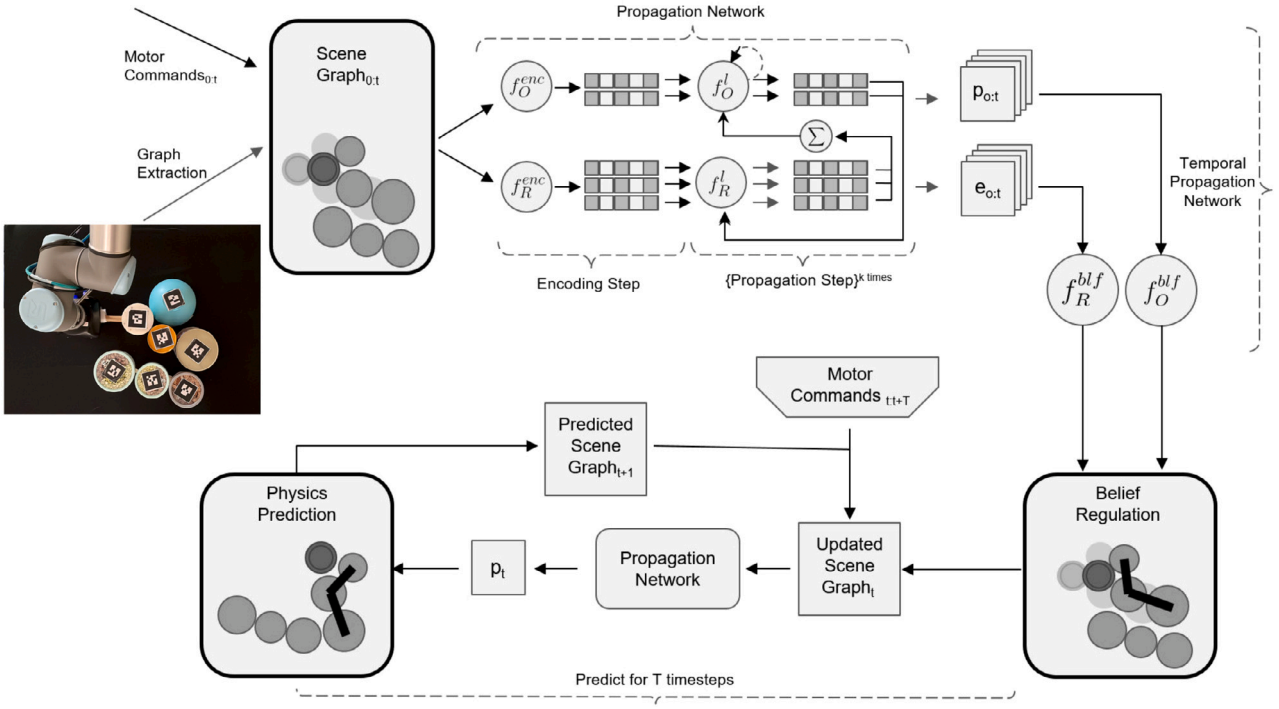


Fig. 2. Our framework extracts object- and relation-centric latent representations from the current physical scene. The latent representations are initially used to update unknown parameters of the scene graph, then with the planned motor commands, they are used for predicting future motion of the manipulated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

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. [72] 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. [73] 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.

3. 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 across various tasks. In this work, we designed our framework around solving two complementary tasks, namely *belief regulation* and *physics prediction*. Fig. 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.

3.1. Preliminaries

In this work, we employ graph neural networks to capture the dynamics of multi-object systems, elucidating how each object responds to the fundamental principles of physics. Our approach encompasses two complementary tasks: physics prediction and belief regulation. Physics prediction involves the forecasting of physical system behavior, offering insights into how objects interact within dynamic environments. Concurrently, belief regulation entails the estimation of essential parameters required for constructing an accurate model of the physical system, leveraging available data. For this, we concentrate on push manipulation scenarios featuring multiple objects, where the robot's primary action involves linearly pushing an object situated on a table. In these scenarios, the physics prediction task corresponds to modeling how objects move with respect to the motion of the robot. In parallel, belief regulation focuses on deducing crucial object- and relation-specific parameters, notably encompassing joint types and object masses. Throughout this work, we operate under the assumption of having access to object shape and tracking information up to the current timestep.

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 $o_i = \left\langle x_i, a_i^o \right\rangle$ store object or part vectors where for object i, $x_i = \langle q_i, \dot{q}_i \rangle$ is the state of the object, with its pose q_i and velocity $\dot{q}_i$, and a_i^o stands for object properties such as shape or mass. Between

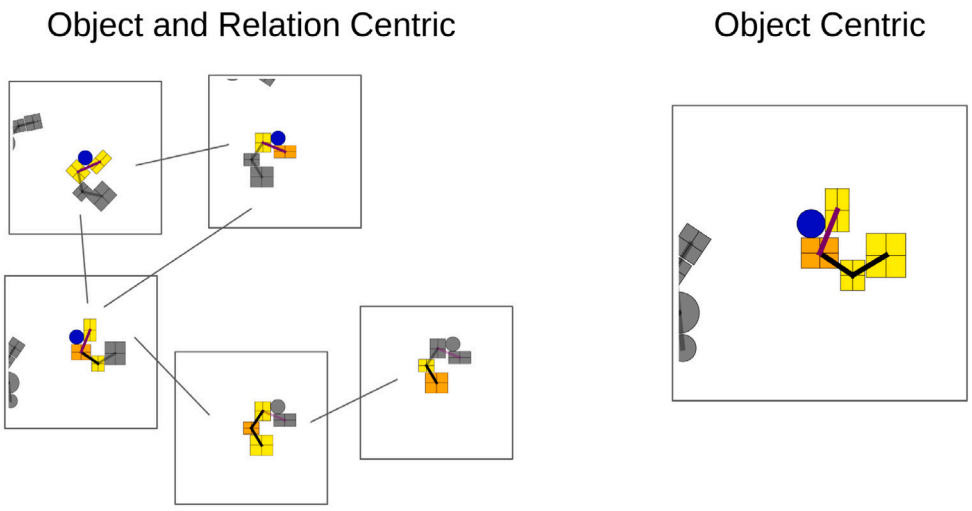


Fig. 3. Comparison between object-centric vs. object- and relation-centric representations. On the left, the process of handling graph-based input is illustrated, with boxes representing nodes and connecting lines representing the edges within the graph. On the right, we depict the procedure for processing image-based input. For each box, the latent representation of the orange object is estimated. Blue and yellow colored objects correspond to the robot, and the other objects that are used in this estimation, respectively. Colors between objects correspond to their articulation type. The representation on the left allows the network to capture object details in a more compositional way, allowing it to propagate action-effects between objects and predicting action-effects of each object more accurately. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

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.

Previously, push-effect prediction tasks were often represented with object-centric image-based representations. Compared to these representations, our graph based representations can represent scene in both object- and relation-centric way. The visualization of both representations are shown in Fig. 3. This inductive bias allows our network to capture scene dynamics more accurately and efficiently. For object-centric representations, action-effect for each object has to be calculated separately which may more often result in inconsistent predictions.

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 u 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. To make the system position and orientation invariant, object position and orientations are not included in node representation. Only relative position and orientations are provided in the edge representations. This essentially leads to representing each object in its canonical pose and the acquiring object- and relation-centric representations shown in Fig. 3. Unlike previous work [68–71], the system has no prior information about how joints behave, and the articulation dynamics are left for the network to learn. That is, our framework only has access to labels of the joints during training.

3.2. Physics prediction

Propagation network. We use 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 Fig. 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, where t corresponds to timestep.

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

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

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 Fig. 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 the network to pass information between not connected objects. This allows force transmission when the robot pushes objects towards another one, effectively pushing both objects while contacting only one of them. Fig. 4 shows a simple illustration of how the robot initiates a chain of interaction and how force applied by the robot end-effector propagates. In the initial propagation step, how the force that emerges from the motion of the robot is passed to contacted objects is shown. In the second propagation step, this force propagates to non-directly interacted objects. The number of subsequent propagation steps to apply can be chosen based on the difficulty of the task.

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

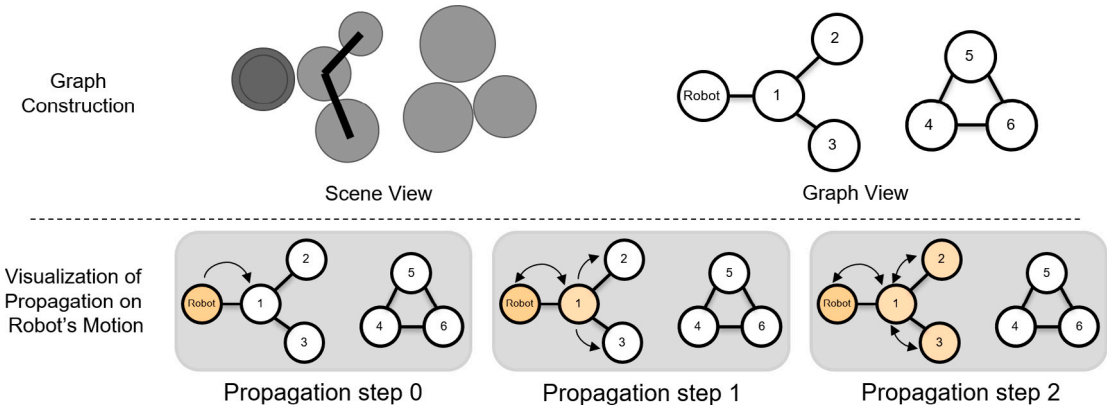


Fig. 4. This illustration shows how the graph of the scene is constructed and how the force emerging from robot end-effector motion is passed to the faraway objects. After graph construction, each node holds state information of their corresponding objects, including the robot. Considering how state 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.

Effect prediction. Effect prediction refers to predicting the future states of all objects one timestep at a time from their present states and the ongoing robot actions. The object embeddings acquired from the propagation network can be used for this task. For each object, the latent vector $p^l_{i,t}$ is used to estimate the next state of the object $x_{i,t+1}$. For this, object velocity $\dot{q}_{i,t+1}$ at time $t+1$ is first predicted, then utilized to estimate the object pose $q_{i,t+1}$ at time $t+1$.

$$\dot{q}_{i,t+1} = f_O^{phy}\left(p^L_{i,t}\right), \quad i = 1 \dots N^o \tag{5}$$

$$q_{i,t+1} = q_{i,t} + \Delta t\ \dot{q}_{i,t+1} \quad i = 1 \dots N^o \tag{6}$$

For simplicity, in our work, we assume $\Delta t = 1$. As the underlying object representation is orientation invariant, the velocity of each object is estimated in terms of the object's own frame from the poses of objects. Before estimating the new pose of the object, it is transformed back into the global frame in which each object has its own orientation. Then, the predicted state information is fed back to the physics prediction network to estimate future object states. Given the 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.

3.3. 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. The illustration of the temporal propagation network is shown in Fig. 2. Node and edge embeddings estimated through the propagation network can be used independently of the graph topology with LSTM. With this, sequences of propagation latent vectors $e^L_{k,t}$ and $p^L_{i,t}$ are passed to LSTM-based encoder functions f_R^{tmp} and f_O^{tmp}.

$$P_{i,t} = f_O^{tmp}\left(p^L_{i,t}, P_{i,t-1}\right), \quad i = 1 \dots N^o \tag{7}$$

$$E_{k,t} = f_R^{tmp}\left(e^L_{k,t}, E_{k,t-1}\right), \quad k = 1 \dots N^r \tag{8}$$

Resulting encodings $P_{i,t}$ and $E_{k,t}$ can be used to estimate missing object and relation parameters, such as joint types and mass. With each newly observed timestep, the encodings $P_{i,t}$ and $E_{k,t}$ are updated. Note that graph topology is updated for each timestep based on the current observed or estimated states of the objects. Propagations are not performed across time. Once embeddings $e^L_{k,t}$ and $p^L_{i,t}$ are estimated for the nodes and edges for a given graph at the current timestep, these embeddings remain independent of the graph's topology. In this way, the temporal propagation network estimates and corrects object and relation properties by considering their overall state history during the robot execution.

Belief regulation. Belief regulation refers to the estimation of object- and relation-related parameters and their correction over time using motion trajectories of all objects until the present state. The belief regulation module can continuously regulate beliefs regarding objects and relations states ($o_{i,t}$ and $r_{k,t}$) using the encodings $P_{i,t}$ and $E_{k,t}$.

$$\tilde{a}^o_{i,t} = f_O^{blf}\left(P_{i,t}\right), \quad i = 1 \dots N^o \tag{9}$$

$$\tilde{a}^r_{k,t} = f_R^{blf}\left(E_{k,t}\right), \quad k = 1 \dots N^r \tag{10}$$

$\tilde{a}^o_{i,t}$ and $\tilde{a}^r_{k,t}$ are used to estimate object and relation parameters and update them over time as more observations are acquired. Specifically, $\tilde{a}^o_{i,t}$ and $\tilde{a}^r_{k,t}$ corresponds to object masses and joint types in this work, and we assume we have access to object shape information. The updated object and relation parameters allows physics prediction to compensate for errors that arise from unknown or partial information regarding the scene. This allows our network to close the gap between its physics predictions and reality.

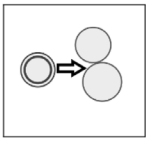
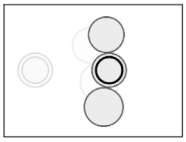
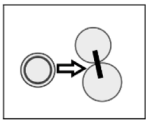
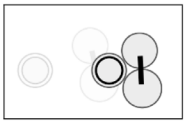
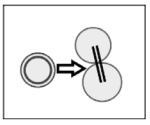
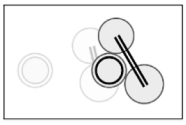
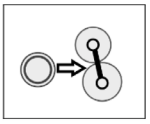
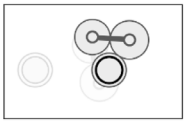
Weight sharing. After training the propagation network for physics prediction, learned weights can be reused in belief regulation, preventing the framework from having to learn two separate networks. This decreases the number of parameters by about thirty percent.[2] The weight sharing significantly lowers the training time. In addition, it allows easier inference as it is easier to load a single model to the employed system. As we show in our experimental analysis, the representation used with physics prediction well represents the environment and can be used in transfer learning without affecting the system performance.

4. 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.

[2] The multiple part setup has 1,653,509 parameters with weight-sharing and 2,253,061 without weight sharing which gives around 27% decrease. However, since this setup does not require f_O^{blf} and f_O^{temp}, the actual decrease is even higher (~35%, from 1,726,468 to 1,126,916).

Table 1
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 contacts the lower object, 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.

4.1. Robotic setup

Experiments are conducted with a 6 DoF UR10 robot arm with a cylindrical shaped end-effector both in simulation and real world. For simulation experiments, we use CoppeliaSim [74] with Pyrep toolkit [75]. For demonstrating prediction capacity of our framework, we define two different object setups, namely *Multiple Parts Setup* and *Different Masses Setup*. 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 the effect variation can occur from object parameters. In these setups, edges between objects are dynamically created as objects approach to each other. 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 the graph shown in Fig. 4), 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*). An illustration of these joint relations and their explanations are shown in Table 1.

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 the physics of the realworld, 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.

4.2. Implementation details

Generation of graph. For each object in the scene and close-by object pair, i.e., objects with Euclidean distance less than 35 cm; a node and two directed edges, the receiver and the sender (corresponding to incoming and outgoing edges, respectively), are created. Each object propagates forces from its sender edges, which correspond to receiver edges for the other close-by objects, to the other objects. 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 hidden layer 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 networks are called multiple times successively and therefore are more costly to use than f_O^{enc} and f_R^{enc}. Finally, f_O^{tmp} and f_R^{tmp} are LSTM with 256 neurons. For physics prediction, the output of f_O^l is given to f_O^{phy}, an MLP with one 256-dim hidden layer and one linear layer, to predict velocity ($\dot{q}_i$) of each object; and for belief regulation, outputs of f_O^{tmp} and f_R^{tmp} are given to f_O^{blf} and f_R^{blf} with a single linear layer to predict object masses and joint relations. During the belief regulation step, the shared propagation network requires the same input as the physics prediction. For this reason, for parameters that are unknown, we use values that do not have any correlation with the actual values and remain constant at training and inference time to not add any bias to the system.

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. This incentives the network to improve its accuracy over time, while not penalizing errors in initial steps significantly, where the robot has limited interaction experience with the objects. Besides, to make networks predictions

smooth and preventing them from oscillating between different outcomes, the outputs of f_O^{tmp} and f_R^{tmp} are regularized by applying MSE loss between latent vectors of successive time-steps.

The network is trained with batch-size of 16 and 3e−4 learning rate using Adam optimizer [76] with AMSgrad [77]. The learning rate is reduced by 0.8 when the validation error stops 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]. This enables the network to recover more easily from predictions that are slightly off, *e.g.,* the effect predictions in which objects slightly penetrate each other. Using Nvidia P100 GPU, the physics prediction and belief regulation modules are trained for two and one days respectively.

5. Results

For quantitative analysis, our framework is evaluated on 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 be similar to that of 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 pair.

5.1. Quantitative analysis in multiple parts setup

For evaluating the physics prediction module, we compared our graph-based architecture with an CNN-based physics prediction architecture that takes rendered object-centric images of pushed objects and predicts the displacement vector. Both architectures are 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, making contact with a most diverse set of objects. For both architectures, outputs of physics predictions 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. Specifically, we estimate the Root Mean Squared Error (RMSE) between each predicted and corresponding ground truth trajectory rollout for all timesteps and take their averages. These trajectory rollouts are 30 and 50 timestep length. Fig. 5 presents the performance in scenarios with different numbers of objects. Comparing error values, it can be seen that both architectures are able to generalize to different number of objects. However, our architecture performs significantly better than then CNN-based one as errors for our network are skewed towards low values while CNN-based architecture errors are skewed towards high values. Regardless, as the length of predicted trajectory roll-outs increase, the errors in a higher number of trajectories accumulate. This is because the propagated errors from earlier predictions affect the accuracy of physics prediction for each subsequent time-step. This causes the trajectories to drift away from the ground truth. However, our method has significantly lower compounding error than the CNN-based one: for our architecture, in Fig. 5 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. Additionally, for the rest of the trajectories, there are more trajectories in high mean error bins.

Next, the belief regulation[3] module is evaluated on the prediction of joint relations. As shown in Fig. 6, as the robot interacts with the objects and more observation data is acquired, our network becomes better at predicting the joint relation types more accurately. The number of observed timesteps used with belief regulation for joint prediction is shown in the x axis of the figure. Fig. 6A shows that our method performs similarly independent of the number of objects used due to the underlying graph structure. In Fig. 6B, 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 the network to mix these two joints. Nonetheless, from Fig. 6C, we can see that the model can correct its predictions on fixed and revolute joints as it observes more robot interactions. The Fig. 6B and C shows that the model abstains from predicting a joint as prismatic unless it is certain. This may be because prismatic joint dynamics are similar to no-joint dynamics unless the robot gains enough observations about objects that are connected to the joint.

Finally, the coupled results of the physics prediction and the belief regulation modules can be seen in Fig. 7. In these results, we predict joint relations with the belief regulation module using the observed timesteps. The number of observed timesteps are shown in the x axis of the plots. The estimated joint relations are used in predicting 50-timestep trajectory rollouts. Then, we estimate the error the same way as we do for physics prediction results. We compare our network with other alternative relation assignment strategies. In Fig. 7, the lines show the mean errors, and the shaded regions show the standard deviation. 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.

5.2. 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. We predict the object masses using the observed timesteps for each trajectory. Using these masses, RMSE between predicted and ground truth masses are estimated. We take the average of RMSE corresponding to each trajectory. This is performed on a timestep level to show how mass prediction improves over time. The results for mass prediction are shown in Fig. 8. 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

[3] Referred to as BR in some of the figures due to space limitations.

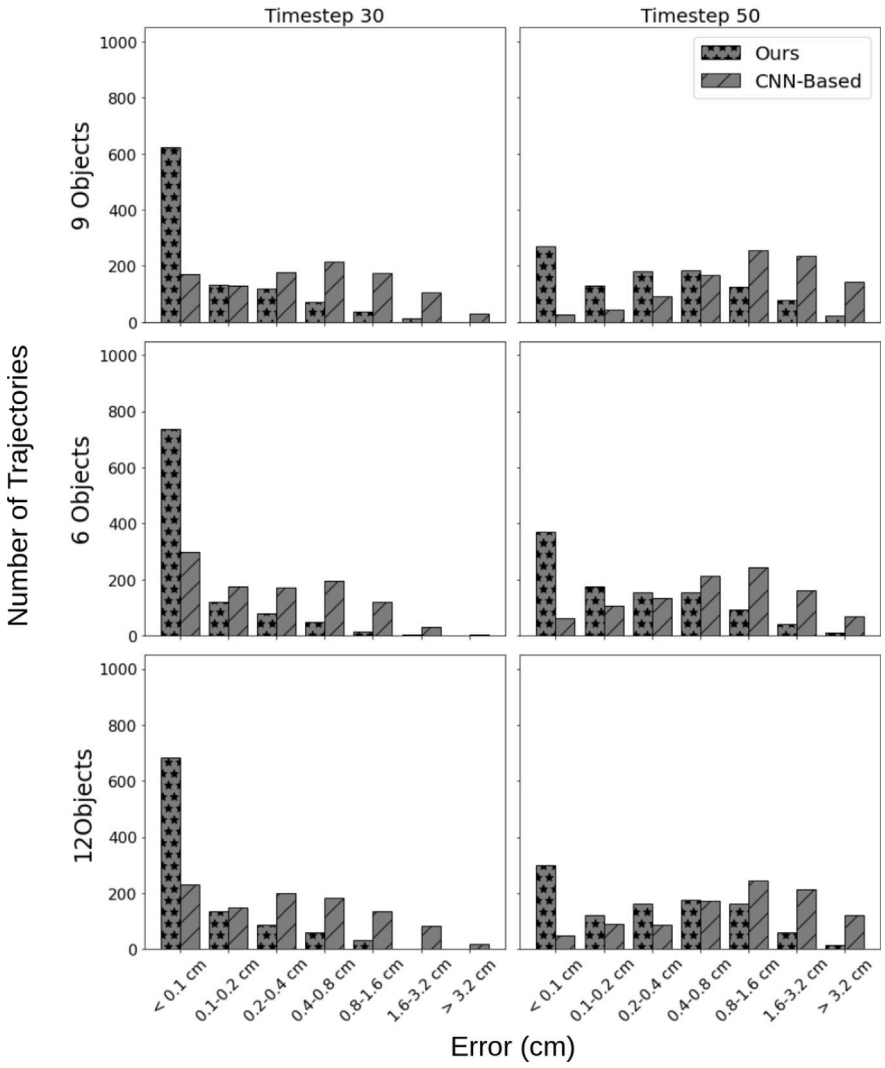


Fig. 5. Physics prediction results on articulated object environments. Error distribution of our network is skewed towards lower error, while CNN-based architecture that uses object-centric images has error distribution skewed towards higher error.

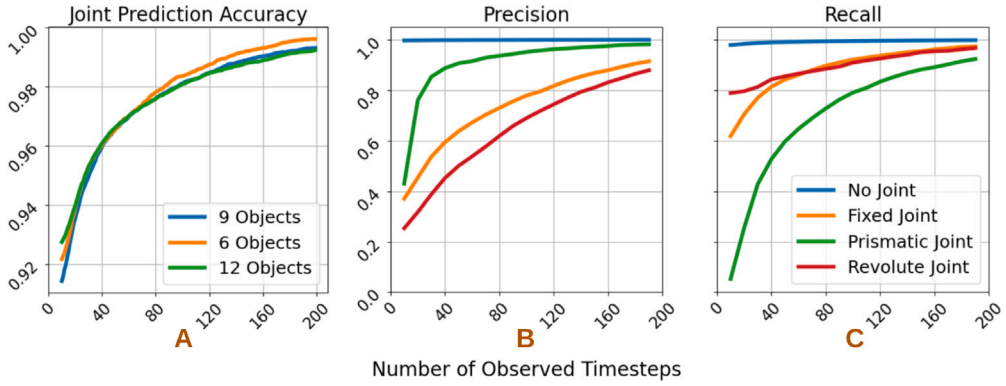


Fig. 6. Belief regulation results on articulated object environments. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

can be seen in Fig. 9. In these setups, we only change the mass of objects while keeping the same 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. In total, we collect 1000 trajectory rollouts for this analysis. In this experiment, we estimate the mass using all timesteps of the trajectories.

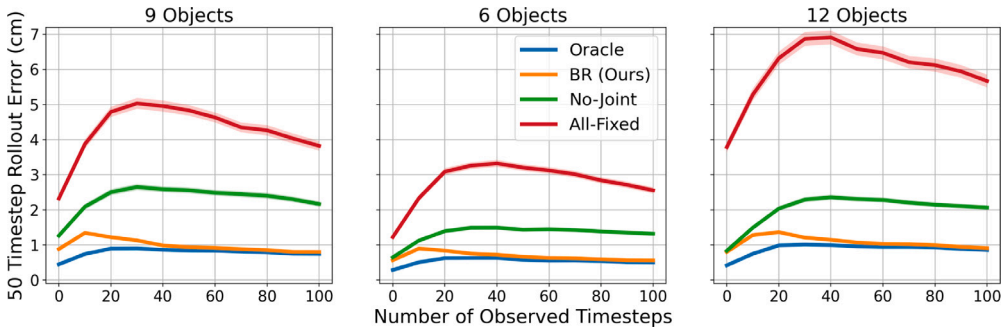


Fig. 7. Results of coupled system on articulated object environments.

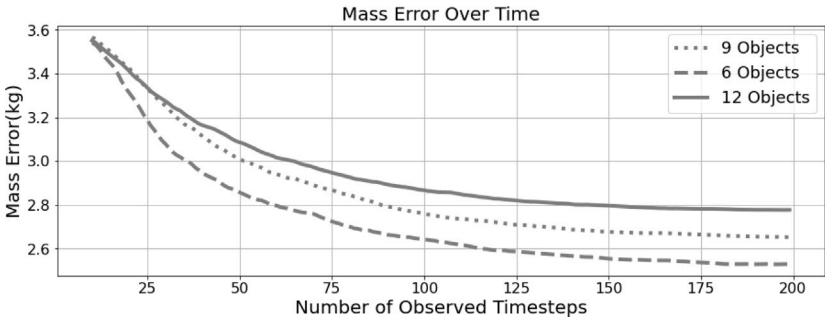


Fig. 8. Belief regulation results on mass prediction. As more observations of motion are gathered from the scene, mass prediction error decreases, but eventually converges to about 2.7 kg mean error.

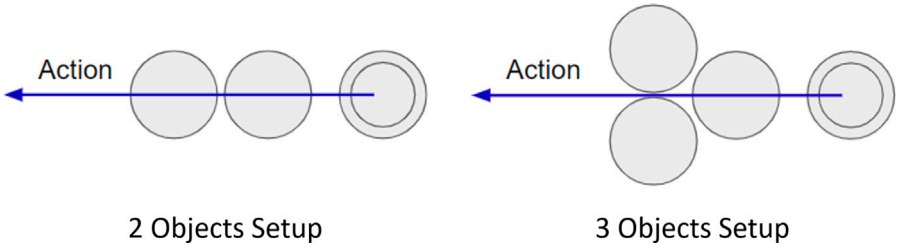


Fig. 9. Visualization of controlled environment setups for mass prediction. In these configurations, object masses are changed between different runs while keeping robot motion and object shapes the same.

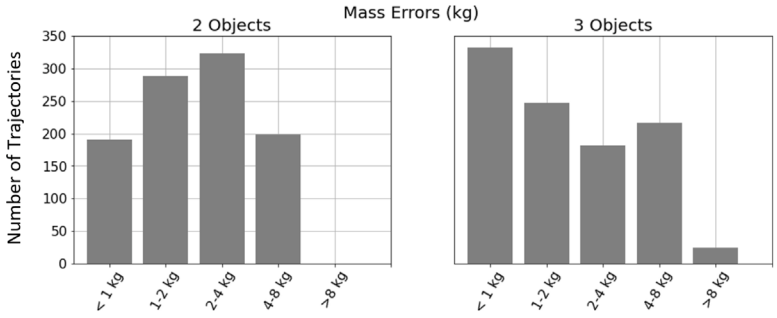


Fig. 10. Mass prediction results in controlled environments. In many cases, our model acquires low error, however there are still many cases that have high error.

The results obtained in these controlled settings are provided in Fig. 10. 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 confuses light and medium objects and medium and heavy objects. For three objects, the fifth bin

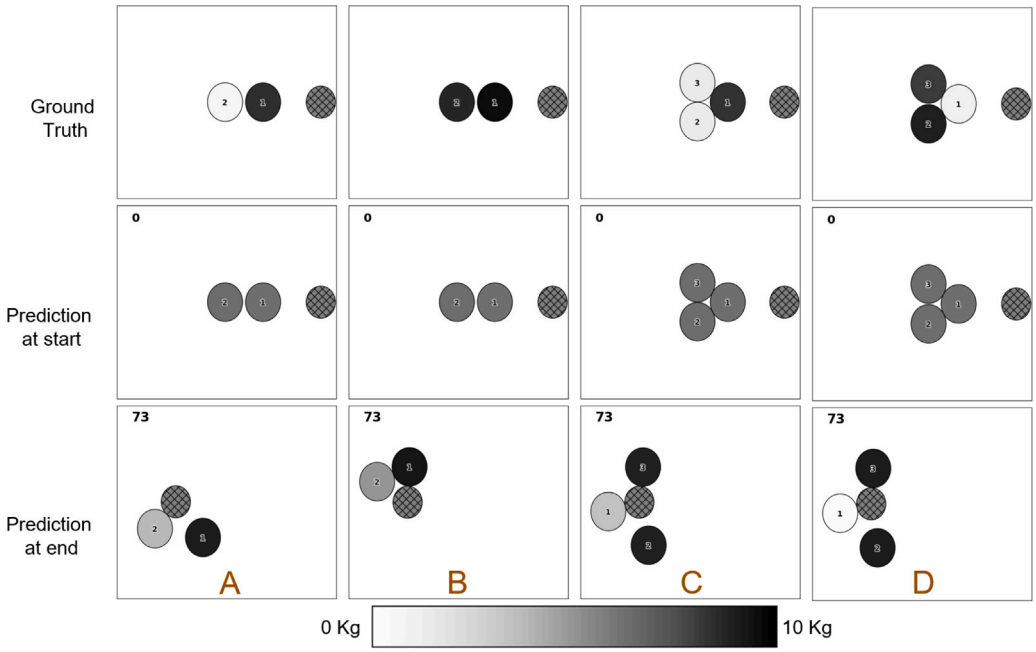


Fig. 11. Mass predictions for two very close observations. For the given two-object (A–B) and three-object (C–D) setups, the same observations are acquired for two different mass configurations, and our framework could not differentiate between the two. Our framework makes the correct prediction for scenes A and D, and incorrect prediction for scenes B and C.

shows that our framework confuses light and heavy objects in rare cases.[4] A number of representative correct and incorrect predictions are provided in Fig. 11. We investigated setups where the network made high-error mass predictions and observe that there are cases where objects of different mass configurations have similar object motions. In Fig. 11C and D, 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.

5.3. 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 modeling novel tools that are not encountered in the training distribution.

In this experiment, we use novel tool objects as shown in Fig. 12A: a stick, L-shaped tool, inv-L-shaped tool, and their various configurations. These tools are represented as multi-part objects composed of cuboids and fixed joints, and are attached to robot end-or. 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. It is important to 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. For every test scenario, the initial end-effector position is fixed, as depicted in Fig. 12B. 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 $-0.5 \leq y \leq 0.5$ (these initial points are shown in Fig. 12B). Then, on each of these initial positions, 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. In addition, 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 way, a dataset for tool and action selection that contains 166 completely diverse solvable initial and final position pairs is 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 RMSE between the final position of the manipulated object and the target position is selected. In addition, 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 test their correctness.

The results can be seen in Fig. 13. 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 in simulation. The first row shows the 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. In the last bars of each bar plot, corresponding to all tools label, results for both tool and action selection are shown. From the figure, it can be seen that our framework manages 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 it not being designed for such a task.

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

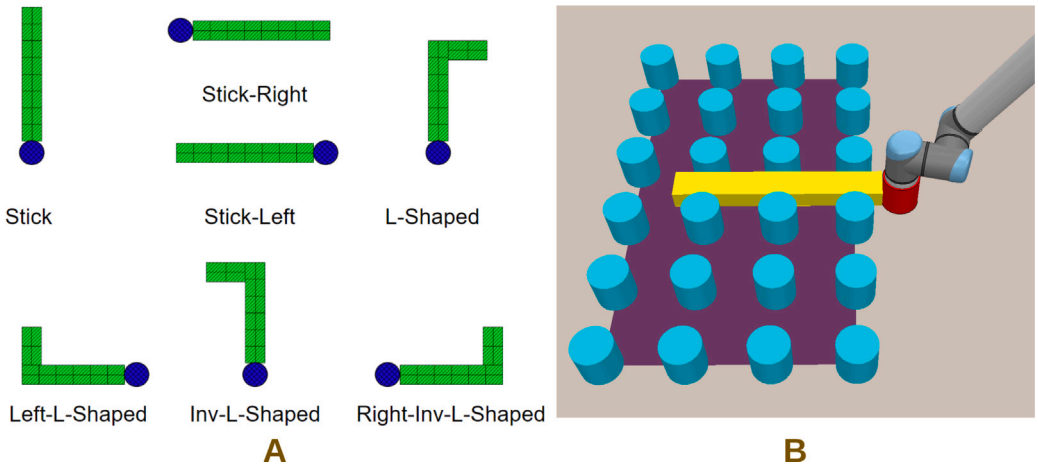


Fig. 12. (A) Tools used in tool selection and planning experiments, (B) initial end-effector position and initial positions utilized for generating feasible goal points.

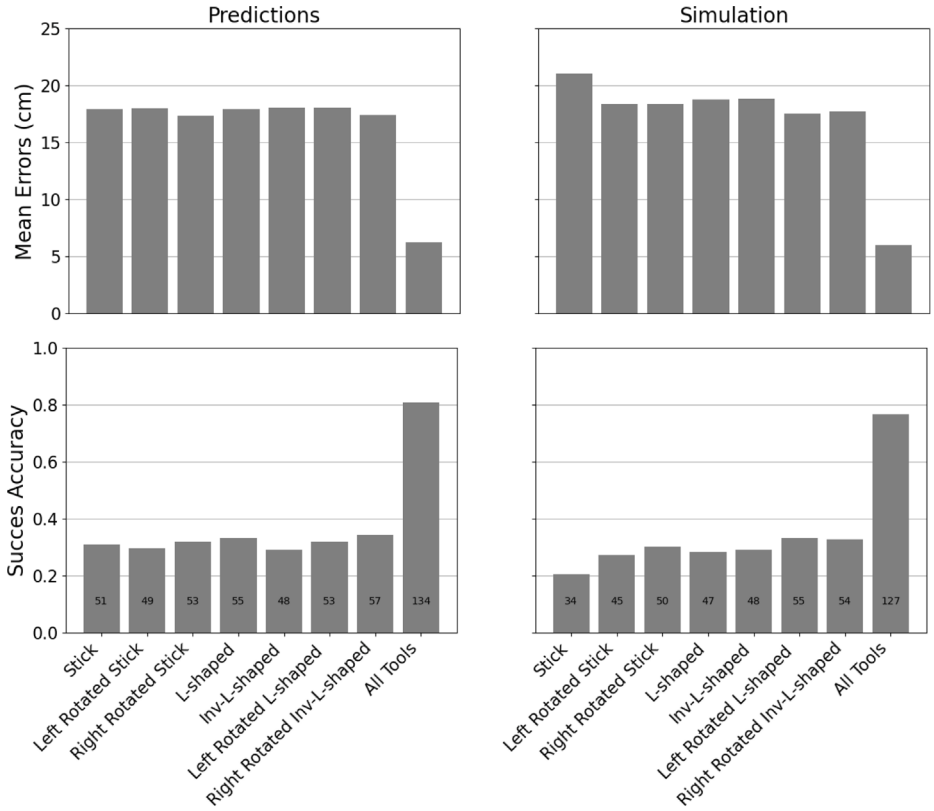


Fig. 13. Tool results. As the robot is allowed to use wider variety of tools, success rate increases and error amount decreases.

5.4. Qualitative analysis — model predictive control

In this experiment, we show that our physics prediction model can be employed in model predictive control (MPC). Given a goal position on the table, the task of the robot is to push the given cylindrical or cuboid target object to the goal position in the presence of other objects. These other objects may affect how the pushed target object moves in the scene.

For generating each test scene, four objects, which are placed closely to one another and with random sizes and joint configurations, are generated on the table. In addition, the goal location is sampled on the left side of the table. The target object is always initialized in the same location. These generation areas are shown in Fig. 14. We utilize the shooting method [78] to acquire the robot control vector u that moves the target object as close as possible to the goal position on the table. We employ our physics prediction network to perform a 20-step simulation of the scene. We then use gradient descent to update the control vector by minimizing the distance between the predicted positions of the target object and the goal location:

$$\text{argmin}_u \frac{1}{20} \sum_{t=0}^{20} L_{\text{goal}}(\phi(G_0, u_{1:t}), q_{\text{goal}}) = (\hat{q}_{\text{target},t} - q_{\text{goal}})^2 \tag{11}$$

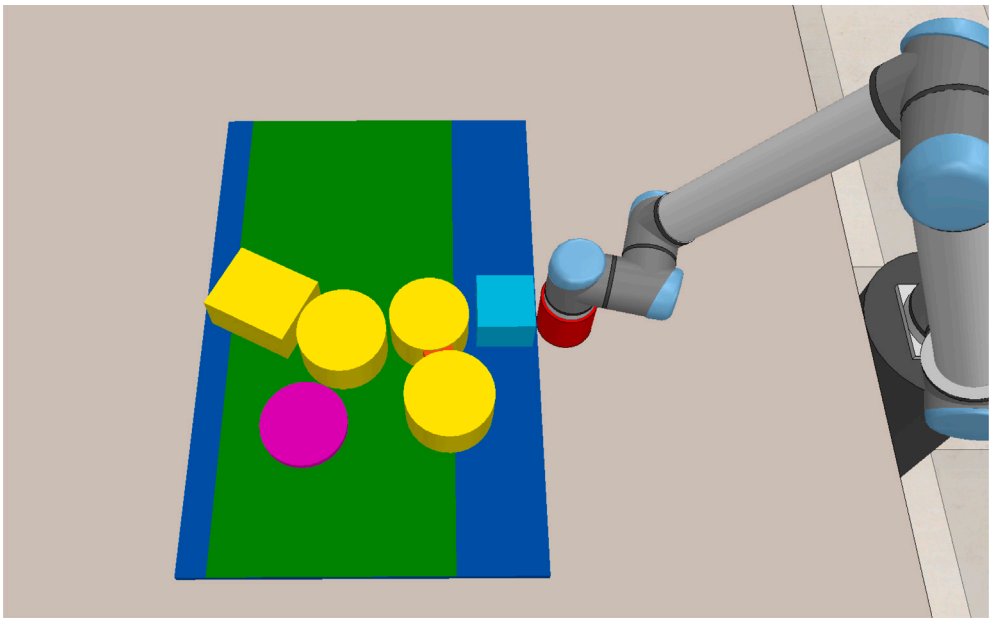


Fig. 14. Scene generation for the MPC task. The task of the robot is to push the light blue object to the center of the purple area. Dark blue and green region approximately corresponds to generation areas for the four yellow objects and the goal location, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 2
MPC results.

	Target object	Ours	Coarse physics
Position error (cm)	Cuboid	**2.18**	9.52
	Cylinder	**2.28**	7.64
Accuracy (%)	Cuboid	**95**	30
	Cylinder	**95**	40

Table 3
6D effect prediction results (cm)

	Ours	Coarse physics
Final position error	**5.04**	20.99
Trajectory error	**1.72**	11.20

where ϕ takes the current state of the scene G_0 and sequence of control vector u up to time t and predicts the pose of the target object $\hat{q}_{\text{target},t}$. In order for the gradient descent to find a plausible solution for object pushing, it requires the robot end-effector to be in contact with the target object during the motion, as the push dynamics are discontinuous. To handle this, we initialize the MPC with 16 different initial robot locations that are in contact with the target object and initial control vectors that move the robot to the goal location. We then optimize these 16 control vectors and select the one that leads to the minimum loss value. The robot performs the first 5 steps of this control vector, and then proceeds to generate a new control vector. This continues until (1) the target object is within one cm of the goal location, (2) it does not move in the last robot motion, or (3) when the robot finishes its 10th motion.

We compare our method with a coarse physics model from [79]. In this model, during contact, the pushed object moves with the same velocity as the robot end-effector. We use two metrics to evaluate our method. The first is the Euclidean distance between the final location of the target object and the goal location. The second is the accuracy of task completion: if the target object is within 5 cm of the goal location, the task is counted as a success. We perform the experiment with 20 test scenes each for cuboid and cylindrical target objects. Our results are shown in Table 2. Our method acquires a high task success rate and significantly outperforms this baseline, as the baseline does not consider the multi-object dynamics present in the scene. One important detail is that the coarse physics model, being a simpler model, is faster than our method. In contrast, our method might require a longer execution time due to the iterative nature of gradient descent steps required to estimate optimal actions. However, it is crucial to emphasize that our proposed method offers the substantial advantage in handling complex scenarios, as demonstrated by the results.

5.5. 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. A PCB is placed 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, the lengths of both sides of the 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 the HDD. In the middle of the HDD, a PCB with its side lengths between 10 to 20 cm is generated. For each generated scene, one lever-up motion with the screwdriver tool is applied from a random location on each side of the PCB. During the PCB level, our method needs to correctly determine when the PCB should start rotating around the ledge or continue to slide linearly on the HDD case in the absence of a ledge. 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 the HDD).

A sample prediction can be seen in Fig. 15. To evaluate our method, we compare it with a coarse physics model [79], in which the PCB moves with the same velocity as the screwdriver tool. Specifically, we evaluate our method on two metrics: final position error and trajectory error. The final position error quantifies the RMSE between the final

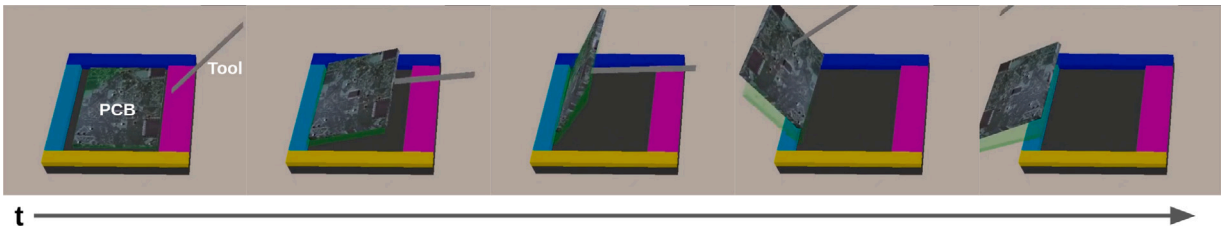


Fig. 15. Snapshots of 6D effect prediction. The ground truth pose of the object is shown with a transparent cuboid. For easier visibility, the ledges of the HDD are shown with different colors. Our method successfully models how the ledges work, and can correctly predict the trajectory of PCB. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Fig. 16. Snapshots of a robot interaction in a real-world setting. Our framework continuously updates its joint predictions as it observes the motion of objects and predicts their future positions.

timesteps of predicted and ground truth PCB trajectories. For the trajectory error, we compute the RMSE for all timesteps and calculate their average. Our results are found in Table 3. Our method significantly outperforms the baseline model, as our model successfully models the changes in the ledge sizes and their absence, and adapts to different scenes. Additional results on this setup can be found on the project page. In this setup, our network makes plausible predictions that match well with the ground truth.

5.6. 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 uses 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 Fig. 16. As the dataset does 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 joints, predictions are limited only to no-joint and fixed joint relations[5] (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 an average RMSE of 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 Figs. 17 and 18. In these analyses, for each trajectory, we use joint relations estimated using the observed timesteps for predicting the rest of the trajectory. The number of observed timesteps is given on the x axis for each bar. We then calculate the RMSE for all timesteps of the predicted trajectory and calculate their averages. In Fig. 17, similar to [30], we test 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 $t + 5$. In Fig. 18, 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.

6. 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 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, where there is a sim-to-real gap between learned physics. 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.

One important assumption of our work is that the belief regulation requires accurate object tracking for parameter estimation. Unless the object or relation parameters are known beforehand, our method has to estimate them by taking object trajectories as input. However, once these parameters are identified, our method can even aid in predicting future object poses when they are not available through tracking. To

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

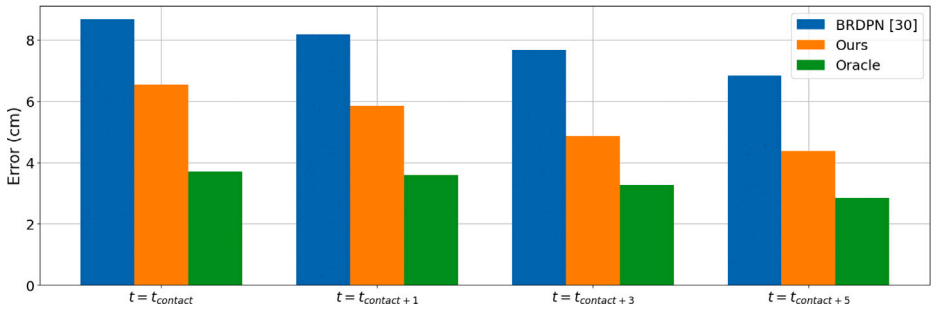


Fig. 17. Average errors (in cm) change in real world as robot makes its first contact with the objects.

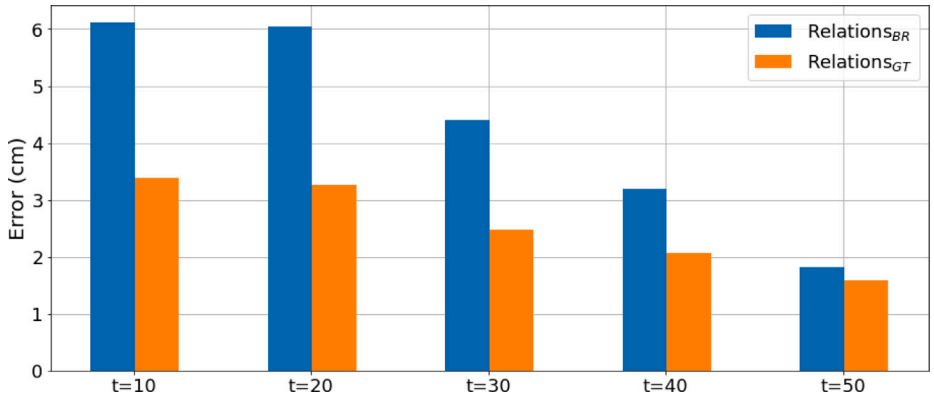


Fig. 18. Average errors (in cm) change in real world as our framework acquires more object tracking information.

address this assumption and reduce our reliance on object tracking, we plan to explore the use of signed distance function based object representations [80]. This will not only alleviate the need for precise tracking but also expand our method's applicability beyond cylindrical and cuboid objects, or their combinations, as different object shapes can be effectively represented using signed distance functions.

As our framework is very generic, we believe it can be further refined and extended. Firstly, our framework can benefit from intelligent exploration strategies that can generalize to a changing number of objects. Secondly, our physics prediction network can be finetuned in the real world to narrow the sim-to-real gap, further improving the model's real world applicability. Moreover, analytic models such as the ones used in [68–71] can be incorporated into our method. Analytic articulation models have been mainly employed in tracking, specifically on semi-dynamic objects, where only one part undergoes motion (i.e., the tracked part), while the rest remains static. In the future, we plan to investigate how the analytic articulation dynamic models could be integrated into our framework. We believe this will yield substantial benefits, making our system more data-efficient and robust in handling articulated objects. It has been shown that combining analytic and learned dynamic models have advantages for learning pushing dynamics [48], and a similar approach could be considered for articulation dynamics. Finally, it is important to note that learning of unsupervised representations for objects via interactions can be very powerful for the visual grounding of objects. In future work, we plan to extend our framework for these adaptations.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Used dataset is available as a drive link.

Acknowledgments

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).

References

[1] F. Ruggiero, V. Lippiello, B. Siciliano, Nonprehensile dynamic manipulation: A survey, IEEE Robot. Autom. Lett. 3 (3) (2018) 1711–1718.
[2] J. Stüber, C. Zito, R. Stolkin, Let's push things forward: A survey on robot pushing, Front. Robot. AI 7 (2020) 8.
[3] J. Stüber, M. Kopicki, 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, 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, S.S. Srinivasa, Pregrasp manipulation as trajectory optimization, in: Robotics: Science and Systems, Berlin, 2013.
[6] F. Paus, T. Huang, 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. 10584–10590.
[7] T. Meriçli, M. Veloso, H.L. Akın, Push-manipulation of complex passive mobile objects using experimentally acquired motion models, Auton. Robots 38 (3) (2015) 317–329.

[8] H. Van Hoof, O. Kroemer, H.B. Amor, 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, 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, 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, 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, R. Dillmann, Templates for pre-grasp sliding interactions, Robot. Auton. Syst. 60 (3) (2012) 411–423.

[13] S. Elliott, M. Valente, 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, 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, S. Levine, Unsupervised learning for physical interaction through video prediction, in: Advances in Neural Information Processing Systems, 2016, pp. 64–72.

[16] C. Finn, S. Levine, Deep visual foresight for planning robot motion, in: 2017 IEEE International Conference on Robotics and Automation, ICRA, 2017, pp. 2786–2793, http://dx.doi.org/10.1109/ICRA.2017.7989324.

[17] A. Byravan, 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, W. Burgard, Hindsight for foresight: Unsupervised structured dynamics models from physical interaction, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2020, pp. 5319–5326.

[19] E.S. Spelke, K. Breinlinger, J. Macomber, K. Jacobson, Origins of knowledge., Psychol. Rev. 99 (4) (1992) 605.

[20] J.B. Tenenbaum, C. Kemp, T.L. Griffiths, N.D. Goodman, How to grow a mind: Statistics, structure, and abstraction, Science 331 (6022) (2011) 1279–1285.

[21] D. Mrowca, C. Zhuang, E. Wang, N. Haber, L.F. Fei-Fei, J. Tenenbaum, 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, R. Pascanu, Relational inductive biases, deep learning, and graph networks, 2018, arXiv URL 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. Chang, T. Ullman, A. Torralba, J. Tenenbaum, A compositional object-based approach to learning physical dynamics, in: International Conference on Learning Representations, 2016.

[25] Y. Li, J. Wu, J.-Y. Zhu, J.B. Tenenbaum, A. Torralba, 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, 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, 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, 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, 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, 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. 10556–10562.

[31] S. Bengio, O. Vinyals, N. Jaitly, 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, H. Lu, Intuitive physics: Current research and controversies, Trends Cognit. Sci. 21 (10) (2017) 749–759.

[33] P.W. Battaglia, J.B. Hamrick, J.B. Tenenbaum, Simulation as an engine of physical scene understanding, Proc. Natl. Acad. Sci. 110 (45) (2013) 18327–18332.

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

[35] K. Smith, L. Mei, S. Yao, J. Wu, E. Spelke, J. Tenenbaum, 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, 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, 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, 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, M. Fritz, To fall or not to fall: A visual approach to physical stability prediction, 2016, arXiv preprint arXiv:1604.00066.

[40] R. Mottaghi, H. Bagherinezhad, M. Rastegari, 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, 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, J. Malik, Learning visual predictive models of physics for playing billiards, in: International Conference on Learning Representations, 2016.

[43] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.

[44] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, 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, S. Tulsiani, Compositional video prediction, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 10353–10362.

[46] F.R. Hogan, 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, D. Bagnell, A convex polynomial model for planar sliding mechanics: theory, application, and experimental validation, Int. J. Robot. Res. 37 (2–3) (2018) 249–265.

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

[49] J. King, J.A. Haustein, S.S. Srinivasa, 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, 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, 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, J.L. Wyatt, Learning modular and transferable forward models of the motions of push manipulated objects, Auton. Robots 41 (5) (2017) 1061–1082.

[53] M.Y. Seker, A.E. Tekden, E. Ugur, Deep effect trajectory prediction in robot manipulation, Robot. Auton. Syst. (ISSN: 0921-8890) 119 (2019) 173–184, http://dx.doi.org/10.1016/j.robot.2019.07.003.

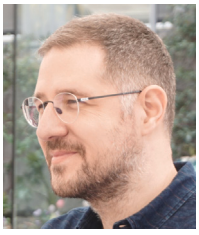
[54] P. Agrawal, A.V. Nair, P. Abbeel, J. Malik, S. Levine, Learning to poke by poking: Experiential learning of intuitive physics, in: Advances in Neural Information Processing Systems, 2016, pp. 5074–5082.

[55] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, 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] C. Nam, J. Lee, S.H. Cheong, B.Y. Cho, C. Kim, Fast and resilient manipulation planning for target retrieval in clutter, in: 2020 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2020, pp. 3777–3783.

[57] M. Janner, S. Levine, W.T. Freeman, J.B. Tenenbaum, C. Finn, J. Wu, Reasoning about physical interactions with object-centric models, in: International Conference on Learning Representations, 2019.
[58] Y. Ye, D. Gandhi, A. Gupta, S. Tulsiani, Object-centric forward modeling for model predictive control, in: Conference on Robot Learning, PMLR, 2020, pp. 100–109.
[59] H.-Y. Tung, Z. Xian, M. Prabhudesai, S. Lal, K. Fragkiadaki, 3D-OES: Viewpoint-invariant object-factorized environment simulators, in: Conference on Robot Learning, PMLR, 2021, pp. 1669–1683.
[60] X. Lin, Y. Wang, Z. Huang, D. Held, Learning visible connectivity dynamics for cloth smoothing, in: Conference on Robot Learning, 2021.
[61] A. Sanchez-Gonzalez, N. Heess, J.T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, P. Battaglia, Graph networks as learnable physics engines for inference and control, in: International Conference on Machine Learning, PMLR, 2018, pp. 4470–4479.
[62] J. Wu, I. Yildirim, J.J. Lim, B. Freeman, 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.
[63] D. Zheng, V. Luo, J. Wu, J.B. Tenenbaum, Unsupervised learning of latent physical properties using perception-prediction networks, 2018, arXiv preprint arXiv:1807.09244.
[64] J. Bohg, K. Hausman, B. Sankaran, O. Brock, D. Kragic, S. Schaal, G.S. Sukhatme, Interactive perception: Leveraging action in perception and perception in action, IEEE Trans. Robot. 33 (6) (2017) 1273–1291.
[65] J.K. Li, W.S. Lee, D. Hsu, Push-net: Deep planar pushing for objects with unknown physical properties., in: Robotics: Science and Systems, Vol. 14, Pittsburgh, Pennsylvania, 2018, http://dx.doi.org/10.15607/RSS.2018.XIV.024.
[66] Z. Xu, J. Wu, A. Zeng, J.B. Tenenbaum, S. Song, DensePhysNet: Learning dense physical object representations via multi-step dynamic interactions, in: Robotics: Science and Systems, RSS, 2019.
[67] N.K. Kannabiran, I. Essa, C.K. Liu, Estimating mass distribution of articulated objects through physical interaction, 2019, arXiv preprint arXiv:1907.03964.
[68] J. Sturm, V. Pradeep, C. Stachniss, C. Plagemann, K. Konolige, W. Burgard, Learning kinematic models for articulated objects, in: Twenty-First International Joint Conference on Artificial Intelligence, 2009.
[69] T. Schmidt, R.A. Newcombe, D. Fox, DART: Dense articulated real-time tracking, in: Robotics: Science and Systems, Vol. 2, Berkeley, CA, 2014.
[70] R. Martín-Martín, S. Höfer, O. Brock, An integrated approach to visual perception of articulated objects, in: International Conference on Robotics and Automation, IEEE, 2016, pp. 5091–5097.
[71] R. Martín-Martín, O. Brock, Coupled recursive estimation for online interactive perception of articulated objects, Int. J. Robot. Res. (2019).
[72] B. Deng, K. Genova, S. Yazdani, S. Bouaziz, G. Hinton, A. Tagliasacchi, Cvxnet: Learnable convex decomposition, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 31–44.
[73] A. Pashevich, I. Kalevatykh, I. Laptev, C. Schmid, Learning visual policies for building 3D shape categories, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2020, pp. 8073–8080.
[74] E. Rohmer, S.P. Singh, 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.
[75] S. James, M. Freese, A.J. Davison, PyRep: Bringing V-REP to deep robot learning, 2019, arXiv preprint arXiv:1906.11176.
[76] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[77] S.J. Reddi, S. Kale, S. Kumar, On the convergence of adam and beyond, in: International Conference on Learning Representations, 2018.
[78] R. Tedrake, Underactuated robotics: Learning, planning, and control for efficient and agile machines course notes for MIT 6.832, Work. Draft Ed. 3 (4) (2009) 2.
[79] W.C. Agboh, D. Ruprecht, M.R. Dogar, Combining coarse and fine physics for manipulation using parallel-in-time integration, in: The International Symposium of Robotics Research, Springer, 2019, pp. 725–740.
[80] D. Driess, J.-S. Ha, M. Toussaint, R. Tedrake, Learning models as functionals of signed-distance fields for manipulation planning, in: Conference on Robot Learning, PMLR, 2022, pp. 245–255.

Ahmet E. Tekden is a Ph.D. student in Department of Electrical Engineering, Chalmers University of Technology, Sweden. He received his B.sc. and M.sc. Degree from the same department in 2017 and 2020 from Bogazici University. He visited Nagai Group, Osaka University in 2018 for 3 months as a research intern. He worked in H2020 project IMAGINE, and is interested in robot learning and intuitive physics.



Aykut Erdem received the Ph.D. degree from Middle East Technical University in 2008. He was a Postdoctoral Researcher at Ca' Foscari University, Venice, in the EU-FP7 SIMBAD Project, from 2008 to 2010. Previously, he was with the Computer Engineering Department, Hacettepe University. Currently, he is an Associate Professor of computer science with Koç University. The broad goal of his research is to explore better ways to understand, interpret, and manipulate visual data. His current research focuses on investigating learning-based approaches to image editing, visual saliency estimation, and connecting vision and language.

Erkut Erdem received the Ph.D. degree from Middle East Technical University in 2008. After completing his Ph.D. degree, he continued his postdoctoral studies with the École Nationale Supérieure des Télécommunications (Télécom Paris) France, from 2009 to 2010. Currently, he is a Professor in the Department of Computer Engineering, Hacettepe University. His research interests include semantic image editing, visual saliency prediction, and integrated vision and language applications.

Tamim Asfour received the Diploma degree in electrical engineering in 1994 and the Ph.D. degree in computer science in 2003, both from the University of Karlsruhe, Germany. Currently, he is full Professor at the Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology (KIT) where he is head of the High Performance Humanoid Technologies Lab. His current research interest is high performance 24/7 humanoid robotics which focus on engineering humanoid robots that can act and interact in the real world, learn from human observation and experience. He is developer of the ARMAR humanoid robot family.

Emre Ugur is an Associate Professor at Department of Computer Engineering, Bogazici University, Turkey. After receiving his Ph.D. in Computer Engineering from Middle East, he worked at ATR Japan as a researcher (2009–2013), at University of Innsbruck as a senior researcher (2013–2016), and at Osaka University as specially appointed assistant professor (2015, 2016). He is interested in developmental and cognitive robotics, and intelligent and adaptive manipulation.