Deep Effect Trajectory Prediction in Robot Manipulation

M. Yunus Seker, Ahmet E. Tekden, Emre Ugur*

Bogazici University
Computer Engineering Department

Abstract

Imagining the consequences of one's own actions, before and during their execution, allows the agents to choose actions based on their simulated performance, and to monitor the progress by comparing observed to simulated behavior. In this study, we propose a deep model that enables a robot to learn to predict the consequences of its manipulation actions from its own interaction experience on objects of various shapes. Given the top-down image of the object, the robot learns to predict the movement trajectory of the object during execution of a lever-up action performed with a screwdriver in a physics-based simulator. The prediction is realized in two stages; the system first computes a number of features from the object and then generates the complete motion trajectory of the center of mass of the object using Long Short Term Memory (LSTM) models. In the first step, we investigated use of various feature descriptors such as shape context that encodes a distributed representation of positions of the object boundary points, unsupervised fea-

*M. Yunus Seker and Ahmet E. Tekden equally contributed to this work.
Email address: emre.ugur@boun.edu.tr (M. Yunus Seker, Ahmet E. Tekden, Emre Ugur)

tures that are extracted from autoencoders, Convolutional Neural Network (CNN) based features that are conjointly trained with the LSTMs, and finally task-specific supervised features that are engineered to well-encode the underlying dynamics of the lever-up action. The models are trained in simulation with objects of varying edge numbers and tested in the simulated and the real world. Our deep and generic CNN-based LSTM model outperformed the predictors that use unsupervised representations such as shape descriptors or autoencoder features in the simulated test set. Additionally, it was shown to generalize well to novel object shapes that were not experienced during model training. Finally, our model was shown to perform well in predicting the consequences of lever-up actions generated by a screwdriver that was attached to the gripper of the real UR10 robot. We further showed that our system can predict qualitatively different trajectories of objects that roll off the table or tumble over as the result of lever-up action.
Keywords: Robot Learning, Predictive Models, Long Short Term Memory, Shape Context, Manipulation, Convolutional Neural Networks

1. Introduction

Predicting the consequences of one's own actions is an important requirement for intelligent control and decision making in both biological and artificial systems. Neurophysiological data suggests that human brain benefits from internal forward models that continuously predict the outcomes of the generated motor commands for trajectory planning and movement control [21]. For higher-level cognitive functions, behavioral data suggest that forward prediction is used to generate multi-step plans to achieve different

goals. For example, given a number of objects with different affordances such as pushability, stackability and climbability, the chimpanzees can predict the outcomes of potential chains of actions and execute a particular sequence such as stacking boxes on top of each other and climbing over the stack to acquire a banana that was initially out of the reach [22, 33]. While the underlying mechanisms for more complex reasoning in humans are unknown, several experiments showed that different parts of the brain become active in deductive and inductive reasoning [17] in predicting the consequences of different actions in different states [2, Ch. 9]. Reasoning in artificial agents also benefits from state transition and prediction models in standard search and complex planning domains [28, Ch. 3–6,10-11]. In these systems, given the current state of the environment and the parameterized actions, the next state predictions can be encoded in different levels of abstractions. For example, an upper body humanoid robot with manipulation capabilities can exploit predictions of high-level categorical effects such as lifted, grasped or rolled or low-level continuous effects such as the low-level trajectories of the manipulated objects or the haptic profiles expected to be measured during action executions.

Prediction of consequences of the robot's own actions and learning this prediction capability from the robot's own interaction experience have been vastly studied in robotics under the umbrella term of affordances [20]. Starting from the seminal studies [24, 29], affordance learning has corresponded to acquiring the capability of inferring the effects given objects and actions. Most of affordance studies have focused on learning robust object representations to predict effects on challenging objects or on learning how to parame-

terize actions to achieve desired effects [39]. In these studies, the effects were discrete and the temporal aspect of the effect representation was ignored. We require methods that predict how effects temporally evolve during action execution for better control, execution monitoring, and online planning of actions.

In our previous work, we showed that learned effect predictions in sub-symbolic [36] and symbolic [35] spaces enabled the robots to make multi-step plans to achieve given goals. However, those predictions provided only approximate estimations about the next states as the effects were represented with categorical variables. Lower-level effect prediction that takes into account real-valued state variables and temporal information, on the other hand, would provide the possibility to mentally simulate any action on any object. This can be exploited to train inverse models for specific tasks by imagining the effects of actions instead of actually executing them, or to monitor the execution performance of the executed actions.

In this paper, our aim is to enable the robot to learn predicting the motion trajectories of the objects generated by robot actions. The robot's own manipulation experience is used for training. Given an object with an arbitrary shape, we expect the robot to learn the effect of the manipulation action that is applied to an arbitrary point on the object. This paper is a part of a larger research agenda studied in the IMAGINE project[1], where imagining effects of the actions are expected to be achieved using machine learning techniques and physics simulators. The IMAGINE project is implemented

[1]https://www.imagine-h2020.eu/

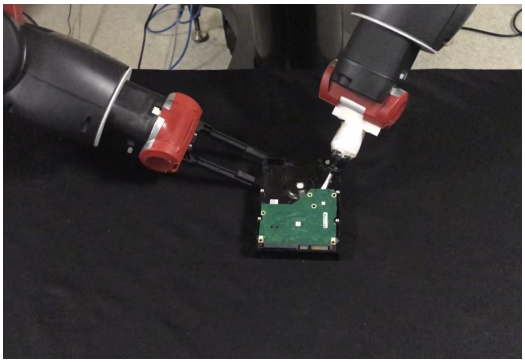


Figure 1: Baxter robot performing lever-up action in order to extract the PCB module of a hard drive.

in the context of recycling of electromechanical appliances as the current recycling practices do not automate disassembly. One of the most common actions in disassembly operations, namely lever-up action, is selected as the sample action in this paper.

The effect prediction is studied in two coupled sequential stages. In the feature extraction stage, we investigated the use of various object descriptors such as shape context features that well-represent the geometry of the objects, autoencoder features that encode the shape in a compact reduced space and task-specific features that are trained along with the subsequent stage. In the next stage, recurrent neural networks are used to predict the complete motion trajectory of the object given the object features. To the best of our knowledge, the learning of prediction of the action effects that depend on the shape of various objects in trajectory level has not been studied before. Given the top-down 2D images of objects, our system can learn to predict such low-level effects using simulated interaction experience, and can also predict the effects of real robot actions on analogous real objects.

The rest of this paper is organized as follows: The next section provides

an overview of the studies on effect prediction in robotic systems and the object descriptors that are suitable for such predictions. Section III provides the details of our model and the alternative approaches that are used for comparison. Sections IV and V give the experimental setup and the experimental results, and the final section concludes the paper.

2. Related Work

The consequences of the robotic actions can be predicted perfectly if the dynamics of the system was exactly known. As it is not realistic to model the complete dynamics of real tasks and situations, the modeling can be done either through approximating the underlying physical model through mathematical equations (e.g. simulations with physics engines such as ODE [32]) or via learning from data. In this section, we will review the relevant data-driven methods and the relevant feature representations. We will first review the object representations that encode the geometrical and physical properties of objects, and then the learning methods that have been used for effect prediction.

2.1. Representations for effect prediction

Explicit geometric descriptors:. We argue that the representations required for effect prediction should be compact enough to be processed by the learning machine and rich enough to encode the geometrical properties of the objects on which the effects of the actions depend on. Object representation is an important and well-studied topic in computer vision that has used inspiration from human visual processes. Initial studies focused on the geometrical properties of the objects and their silhouettes. From a constructive

perspective, pre-defined geometric object templates were used to segment complex objects into simple parts with generalized cones [25] and geons [6] such as ellipsoids, cylinders and cuboids. From a more bottom-up perspective, on the other hand, Hoffman [19] proposed a geometrical concept called minima rule that divides the objects into pieces from their local concave points. The similarity between objects was computed based on how well those points match across the objects. Extending this work, Singh *et al.* [31] segmented the objects by cutting their silhouette from the closest pairs of local concave points and represented the objects by the composition of the convex sub-parts. A more distributed and generic representation was provided by Belongie and Malik [5] who also exploited a similar representation to compute object similarity. In their method, a number of points were sampled from the border of the object, and the so-called shape context features on a reference point were computed based on the distribution of the relative positions of other sample points, offering a measure between different shape characteristics and geometries. The distance between shape context features was used as the similarity metric in their study successfully. Recently, Bogh and Kragic exploited the shape context descriptor in vision-based robotic object grasping problems [7]. In their method, after the global contour of an object was extracted using a stereo camera system, the robot learned to predict whether an object was graspable or not by training a non-linear classifier (Support Vector Machine) that used shape context features of the sampled (graspable or non-graspable) points. Shape context was argued to provide information about the physical characteristics of the objects, such as their center of gravities; and was shown to be more effective in predicting

graspability of novel objects compared to local appearance features such as filter banks that are composed of edge, texture and colour filters.

Deep descriptors:. With the recent developments in deep learning and the increasing amount of data, deep object representations have been heavily used as input representations to learning machines in 2D object recognition tasks. For 3D object classification, on the other hand, Yi Fang *et al.* [13] developed a deep shape descriptor that provides geometrically descriptive shape features that are robust to noise and structural variations in various 3D datasets. They used 3D object models as input to their model, making it difficult to be used in real-world circumstances due to the hardness of complete 3D model retrieval from the environment. We are aiming to use raw object images instead of using the complete 3D models. Zhu *et al.* [40] proposed a new approach for constructing 2D representations from 3D object data. By projecting 3D shapes into 2D space from a number of perspectives and using autoencoders to reduce the size of the feature representations, they show that their method gives high accuracy performances on object matching as well as leading to a faster and more efficient training procedure for feature learning. In our work, we investigated the use of several autoencoders in effect prediction rather than object comparison.

2.2. Learning methods

Probabilistic methods:. Deisenroth *et al.* [12] proposed to create a probabilistic dynamic model that predicts the next state given the current state and robot actions using Gaussian Processes. By exploiting the uncertainty represented in the learned probabilistic model, they showed that a policy

search algorithm could learn to solve a real cart-pole task efficiently and performed long-term planning effectively. Battaglia *et al.* [4] proposed to use Bayesian models, named as Intuitive Physic Engines (IPE), which predicted the motion of the stacked cuboids that fall to different places on the ground. They trained their model to predict discrete effects, i.e. whether they fall or not; and continuous effects such as the direction of fall with the features that could be extracted from vision. They used only one type of object, therefore, did not focus on the effect of the shape of the objects in interactions. Furthermore, they used manually-engineered features such as the angle of minimum critical angle across sub-towers.

Deep methods:. It is not realistic to manually engineer different feature sets for all different tasks, and deep learning methods provide means to automatically extract suitable features for the corresponding problems. Wu *et al.* [37] used deep learning to find the parameters of a simulation engine that predicted the future positions of the objects that slide on various tilted surfaces. Lerer *et al.* [23] trained deep networks to predict the stability of the block towers given their raw images obtained from a simulator. Fragkiadaki *et al.* [15] used a deep model architecture where the output of Convolutional Neural Networks was used as inputs of Long Short Term Memory cells [18] to predict the motion of balls in simulated environments. Battaglia *et al.* [3] proposed the so-called interaction networks that perform object-centric and relation-centric reasoning to predict the dynamics of objects. Similarly, Chang *et al.* [10] used neural networks to factorize object dynamics into pairwise interactions and predict the future trajectory of objects by aggregating the results of these pairwise interactions. These networks achieved high per-

formance but did not consider the shapes of the objects. Furthermore, they mostly used single object types in their experiments and have not considered setups that involve interactions with robotic end-effectors.

Learning in robotic tasks:. Agrawal *et al.* [1] trained deep forward and inverse models that used the output of CNNs in order to learn how to poke an object to move it to a given target location. Initial and final images of the scene were transformed into their latent feature representations using convolutional layers of CNNs. The forward model took parameters of the poke action and the latent representation of the initial image to predict the latent representation of the final image. The inverse model took latent representations of both final and initial images to find the parameters of the poke action. They showed that their model could generalize to novel shapes and novel objects. While they could find the required parameters for the desired goal state, the effect of the poke action was not explicitly computed.

End-to-end learning:. The action-effect prediction has also been investigated in pure end to end learning systems. Finn *et al.* [14] used convolutional LSTMs [38] to predict future image frames using only current image frames and actions of the robot. Similarly Byravan *et al.* [9] used encoder-decoders to predict SE(3) motions of rigid bodies in-depth data. Their network was similar to Finn *et al.*'s network but their model is designed to work on depth data and to predict transformation matrix. In end-to-end methods, as the next RGB(-d) frame was predicted directly, the predictions became more and more blurred in multi-step predictions. Furthermore, object mask discovery and prediction of the next positions of the pixels were the main focus of

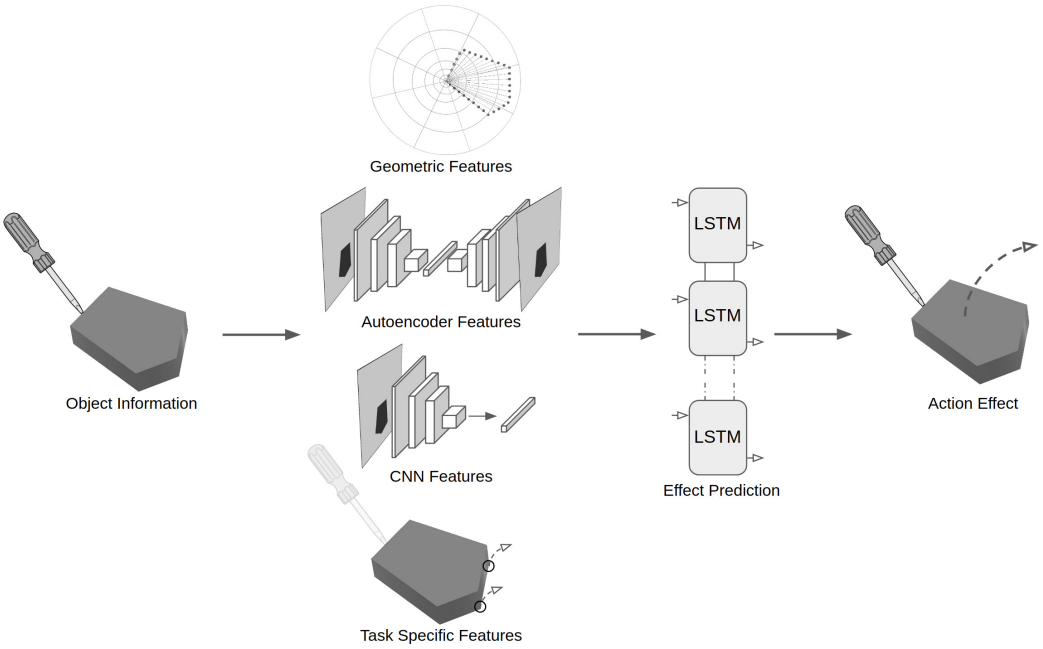


Figure 2: Our general framework. The robot executes an action on the object using a tool. The left-most figure shows the initial state of the object before action execution and the right-most figure provides the trajectory of the object observed during action execution. Given the visual perception of the object, one of the feature extraction methods is used to compute the object features. These features are fed to the LSTM effect predictor that is trained to predict the observed trajectory.

these studies, whereas the shapes of the objects and their influence in the generated effects are very central in our study.

3. Proposed Method

Our model predicts the effect of actions given the object and action related information as shown in Fig. 2. Given top-down image of an object and the contact point of the tool attached to the robot, our system learns to predict the motion trajectory of the object position expected to be observed

during action execution of the robot. This prediction is done by first computing various features from the image of the object and using these features as inputs of a recurrent neural network (RNN) that outputs the position trajectory. A number of models that differ in the utilized feature descriptors are proposed:

- Generic unsupervised features based models: A number of generic features such as autoencoder features or shape context features are used as inputs to the RNN.

- Supervised features based models: A number of features are computed conjointly with the effect predictor. Given raw images or shape context features, Convolutional Neural Networks are trained using the error signal back-propagated from the RNN.

- Task-specific features based models: A number of specific features that are considered to well-represent the particular task and action are manually designed and used as inputs to RNNs. For the lever-up action, the edge of the object that supports rotation movement of the object directly determines the trajectory. Therefore, support points are either directly provided (as the ground truth) or learned and used as inputs to the RNN.

Our model will be described in detail in the rest of this section. As the effect predictor RNN is common in all our models, the specific Recurrent Neural Network model, namely Long Short Term Memory Network, is described first, and the details of the feature representations will be presented next.

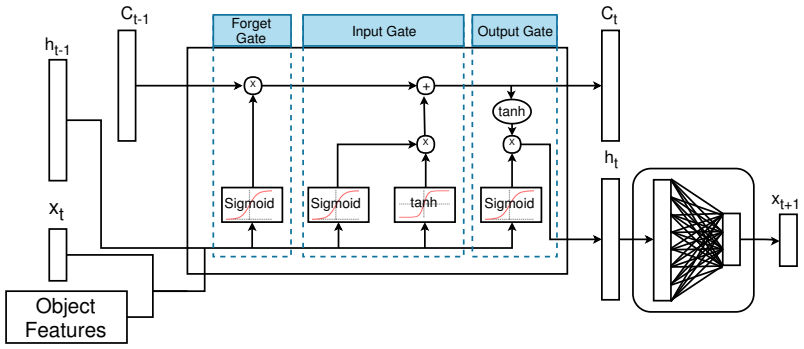


Figure 3: Trajectory Prediction Model. x_t: Current object state. h_t: Message passed from LSTM network to itself and to the output dense layer. C_t: Memory cell of LSTM network. Object features: Features passed to the LSTM network.

3.1. Long-Short Term Memory (LSTM) networks for effect prediction

Our aim is to predict not only the single snapshot of the outcome obtained at the end of the action, but also how the environment changes during the action execution of the robot. For this, our system predicts the trajectory of the successive states expected to be observed during the execution of the corresponding action. Recurrent Neural Networks (RNNs) with the capability of storing the history of their successive inputs in compact state-like representations are suitable for learning in generating such multi-step predictions.

In our model, Long-Short Term Memory [18] (LSTM) networks that are RNN networks with special LSTM units are used. LSTM units are special RNN units that can better handle long term dependencies. In addition to the so-called message (h_t), output unit that is used as the input to the next iteration, LSTM unit also uses an internal memory and specific processing units, called memory cell and gates, respectively. Figure 3 provides the structure of LSTM where the information processing is illustrated from left to right sequentially. First, the current state (x_t), the pose of the object, is

concatenated with the object features and passed to the LSTM network. It is then processed along with the previous output message (h_{t-1}) to predict the next output message(h_t). Output message (h_t) is passed to a dense layer to predict the next n states ($\mathrm{x}_{t+1:t+n}$), and to the LSTM network for calculation of the future states. The value of n can be set flexibly.

The simulated interaction scene is arranged such that the robot always contacts the object from the same position. Therefore, the point of contact is not used as an additional input to the network. This setting prevents our model from learning trajectories based on their sizes and initial poses.

3.2. Generic unsupervised features

In this study, we used two types of generic features: shape context features that encode geometric boundary of the object with distance and angle histograms, and autoencoders that represent the raw top-down image of the object in reduced dimensionality.

3.2.1. Shape context features

Shape context is a distributed representation that encodes histogram of relative positions of boundary points of an object with respect to a reference point and direction. The 2D geometry of the object is represented with a compact histogram representation. In order to exploit the advantage of action-grounded representations [26], the reference point is set as the point of contact between the robotic tool and the object, and the reference direction is set as the direction of the action. After reference point is selected, the contour of the image is extracted (Fig. 4a), a number of points are sampled uniformly from the contour, and vectors that connect the reference point to

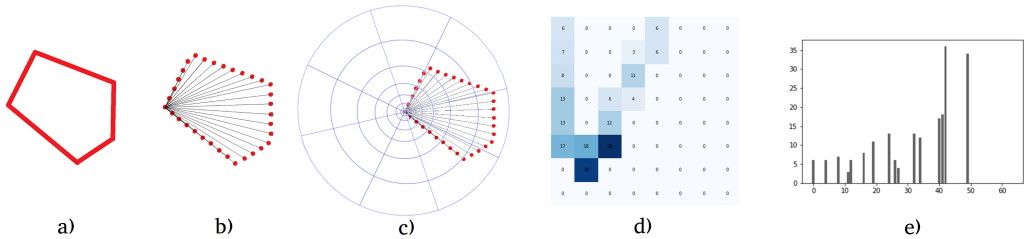


Figure 4: Steps of shape context histogram calculation. a) Finding the contours of the shape. b) Sampling contour points and constructing vectors from the reference point to all sample points. c) Placing the log-polar coordinate system onto reference point and arranging frame according to that the first wedge of the log-polar coordinate system fits the same orientation of the edge that reference point is on. d) Counting sample points according to wedge and ring indexes and constructing a 2D shape context matrix. e) Flattening the matrix to get shape context histograms

all sampled points are calculated (Fig. 4b). Each vector provides the distance between each sampled point and the reference point, and the angle between the vector and the reference direction. All distances can be scaled by the mean distance of all points in order to obtain a size-invariant representation. Finally, a log-polar system is located on the reference point and vectors are counted as which one of them is placed inside of the which wedge and which ring of the log-polar system (Fig. 4c). In this way, a feature matrix size of $R \times W$ (Fig. 4d) is extracted where R and W are ring and wedge numbers of the log-polar system respectively. The matrix can also be viewed as a flat $R \times W$ vector (Fig. 4e). Note that the log-polar coordinate system is used when measuring shape context because it provides more detailed information for the contour points that is closer to the reference point.

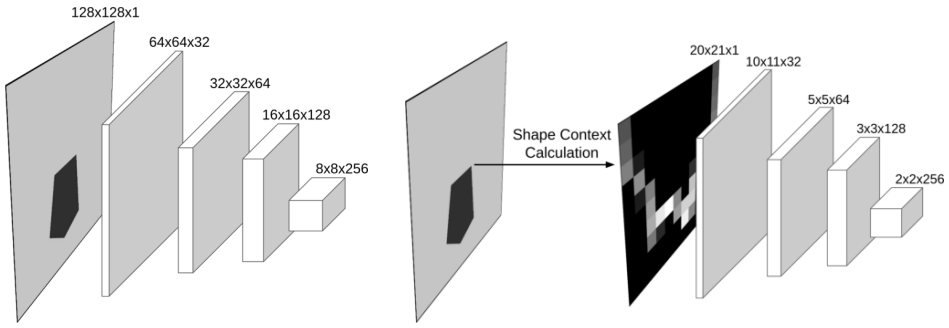
A number of different shape context histograms with different density and quality measurements can be constructed by changing R and W values. In

our work, we use 3 different shape context features called as Sc-200 (10×20), Sc-800 (20×40) and Sc-3200 (40×80).

3.2.2. Autoencoder features

Autoencoder is a dimensionality reduction technique that compresses high dimensional data into a small code such that when the code is decompressed the match between the reconstructed data points and the original ones is maximum. Bottleneck structure of the autoencoders allows the neural network to represent the original input in a lower-dimensional space while preserving the important characteristics of the data.

In our work, we used two convolutional autoencoders to compute representations of the top-down images of the objects in various reduced dimensions: Autoencoder512 and Autoencoder256. The autoencoder used in this study is structured as in [11] starting with the large number of channels and decreasing the channel numbers towards to the latent layer as the image shrinks. The Autoencoder512 network takes an (128x128x1) image as input and outputs another image with the same size. In encoder network, filter numbers of convolutional layers are chosen as 128, 128, 64 and 8 ordered from the first to the last layer. For the decoding part, all layers are the same but in backwards order. Filter sizes of all convolutional layers are (3x3), and each convolutional layer is followed by a max-pooling (2x2) layer for encoding or an upsampling (2x2) layer for decoding parts of the network. At the end we have (8x8x8) encoded features. In Autoencoder256 on the other hand, the filter number of the last encoder layer is 4 instead of 8. So the output of the Autoencoder256 is (8x8x4). After training the autoencoders, the activation in the bottleneck layer is used as the input for the effect prediction model.



(a) Image CNN (Im-CNN) (b) Shape Context CNN (Sc-CNN)

Figure 5: Convolutional Neural Networks are used as supervised feature extractors

3.3. Generic supervised features

Shape context and autoencoder features are generic features, however, they arrange the feature space independent of the learning problem while generally reducing dimension. In this section, we will provide feature extractors that are trained along with the subsequent RNN predictor. While training our model to predict the next poses of the object, a convolutional neural network (CNN) based feature extractor is also trained in the meantime. Our CNN can use two different inputs, namely, shape context matrices or raw images. In the remaining of this subsection, we will provide the details of CNNs.

Convolutional Neural Networks are variations of feed-forward neural networks that run over the 2D structure of image through convolution operation and pooling layers. CNNs can have multiple convolution layers with pooling and activation layers between them. They generally take a raw image as input and learn the network weights that are needed to predict the desired output from that image. Convolutional layers with (3x3) filters and ReLu activation functions are used in this study. Every convolutional layer is followed

by a (2x2) maxpooling. We implemented two different CNN-based feature descriptors. Im-CNN (Fig. 5a) takes the raw top-down images of the objects as inputs whereas SC-CNN (Fig. 5b) uses shape context matrices that are computed from the boundaries of the objects as inputs. In this study, we assume that the visual perception is encoded relative to the executed action. As the lever-up action follows the same trajectory, Im-CNN does not need to explicitly use any action related information. For both of the CNN models, we followed the structure of VGG-16 [30] architecture, which is a widely used standard convolution architecture, where each convolution layer is followed by a pooling layer while the channel sizes are doubling starting from 32 to 256.

3.4. Task specific features: Support points and their predictors

In order to evaluate the performance of the proposed features above, in this section, we introduce manually designed task-specific features that have a direct relation with the physics of the particular task in a compact form. The performance of the effect predictors with these features will be used as a baseline. For the particular action of lever-up, the trajectory of the object depends on the edge which the object rotates around. The support points, i.e. the object corners that define the supporting edge of the object rotation, are used as the manually designed ground-truth inputs for the RNN trajectory predictor. These features are named as support point ground truth, $\mathrm{SP}_{\mathrm{truth}}$.

The support points can be directly extracted from observing the object movement during action execution, however, our purpose is to predict the trajectory prior to executing any action. Therefore, we propose a neural network model that learns to predict support points and provide the support

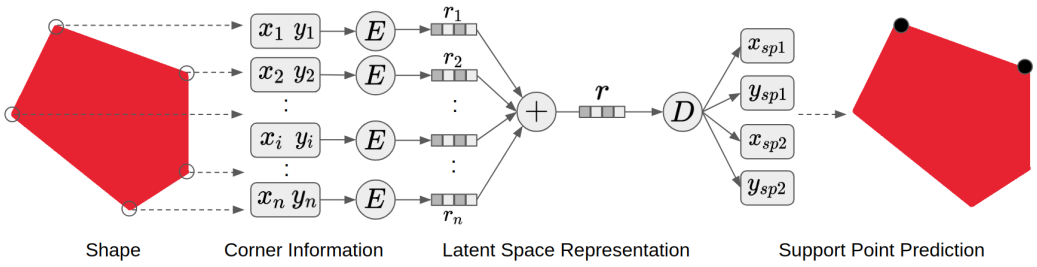
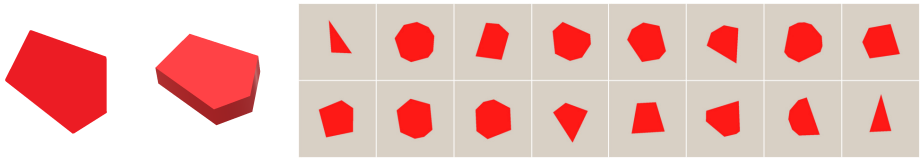


Figure 6: Support point prediction model. Corner information is separately transformed to latent space representation using a single shared encoder (E). Representations are merged into one by using a summation layer. Resulting general representation is given to a decoder structure to predict the support point of the shape.

point predictions to the effect predictor. These features are called $\mathrm{SP}_{\mathrm{pred}}$. Figure 6 shows the neural network structure that uses the corner coordinate information from the shapes to predict the two support points. The network always outputs two support points but input size is changing because edge numbers of the shapes are changing between 3 and 10. To construct a neural network that can adapt to the changing number of inputs (corners), we used a neural network structure inspired from Conditional Neural Processes [16]. A single shared encoder, which is a three-layer MLP with 128 neurons, is used to transform the corner inputs into their latent space representations. After that, all representations are merged into one using an aggregation layer which sums up the given representation vectors. Using a shared encoder followed by an aggregation layer, we ensure that our network can handle different numbers of input sizes. The resulting representation holds the general information about the shape according to its corner information. Finally, this representation is used as an input to a decoder structure, which is a two-layer MLP with 128 neurons, to predict the support points. Finally,



(a) (b)

Figure 7: (a) A sample random shape (pentagon) and the corresponding object generated in the V-REP simulator. (b) A number of randomly generated sample shapes.

the error of the network is back-propagated according to mean square error between predicted support points and $\mathrm{SP}_{\mathrm{truth}}$ information.

4. Experimental Setup

The robot is required to learn to predict the effects of its actions through self-exploration and observation. V-REP[2] physics-based simulator with Bullet engine is used for the training platform. To verify our system with a rich set of shapes, an object dataset is created in the simulator from randomly generated shapes. In order to generate an object, a circle is generated with a radius of 10 cm, and between 3 and 10 corner points are picked randomly on the circle. The center of the circle is connected with each pair of successive corner point forming a triangle, and the formed triangles are grouped into triangle meshes that can be efficiently created in and handled by the physics engine of the V-REP simulator (Figure 7b). 160 objects are generated in total for interaction.

The generated objects are imported to the V-REP simulator and placed on the table. The lever-up action is applied to the same object several times

[2] www.coppeliarobotics.com/

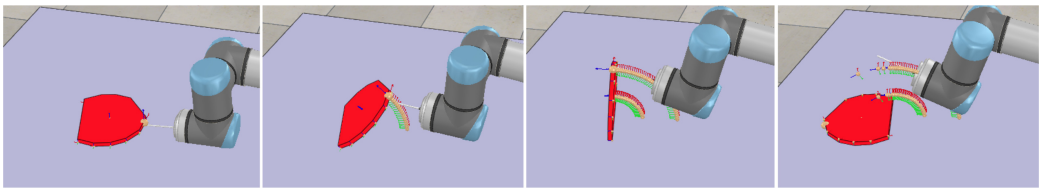


Figure 8: An example of lever-up experiment with a sample shape. Lever-up action is performed from the indicated point of action.

such that the object is contacted from two uniformly sampled points at each edge. A screwdriver that is attached to the end effector of the simulated UR10 robot arm is programmed to generate an open-loop lever-up action, always following the same trajectory as shown in Fig. 8. 1800 interactions are performed in total, storing the shape of the object, the top-down image taken by a simulated Kinect camera, and the trajectory of the center-of-mass of the object in each interaction. The points that support the rotation of the object are also stored in each interaction.

In order to evaluate our proposed model, the dataset with 1800 interactions is shuffled and then divided into training, validation and test sets. To make a better comparison between models, 10 randomization seeds are used for all models. This is to prevent the possibility of inconclusive experimental results that can possibly be obtained from a seed that randomly perform better for certain features. 1400 of the trajectories are used on training, 100 of the trajectories are used on validation, 300 of the trajectories are used for testing. Our models were trained to predict the next pose of the object given the last 15 poses obtained from the simulator. Number of hidden units of the LSTM was empirically selected as 256. To prevent over-fitting, the models were validated on the whole trajectory prediction which was done

by recurrently predicting the next n poses given the current predictions. Intermediate outputs of LSTM obtained from previous time-steps were used for prediction at training time, which boosted the overall performance of the model. Validation error, which was defined as the mean square error on the complete trajectory of the levered-up object, was used to select the best performing models at the end. Keras framework with tensorflow backend [3] is used to train and test the neural network models. ADAM optimizer with default hyper-parameters and mean squared error loss is used for this purpose where the number of epochs is limited to an empirically set number, 100.

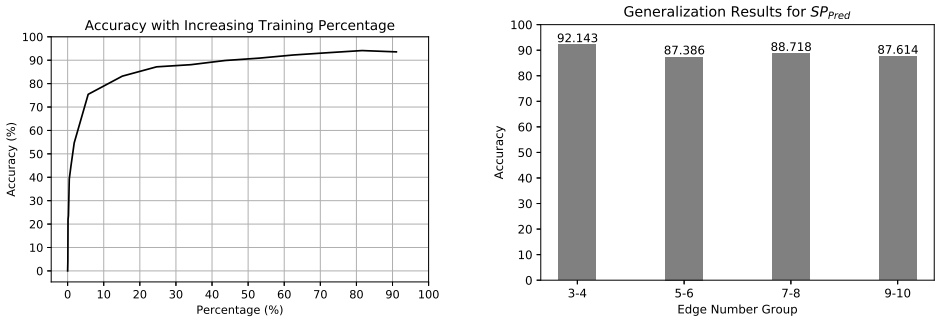
5. Results

In this section, first, we will show the results of our support point prediction ($\mathrm{SP}_{\mathrm{pred}}$) model.

5.1. Support point prediction

Support points correspond to the object corners around which object rotates during lever-up action. As a task-specific feature, we propose to predict these points using a neural network and use them as inputs in predicting object movement trajectory. Recall that the corner coordinates of the objects are used as inputs to our neural network (Figure 6). The increase in performance of the network, i.e. the increase in its accuracy in predicting support points for the test samples, is shown in Figure 9a. Results show that our model reaches up to 93% accuracy predicting the true support points. Comparing with a random guess accuracy of 23%, it can be clearly seen that

[3] `https://keras.io/`, `https://www.tensorflow.org/`

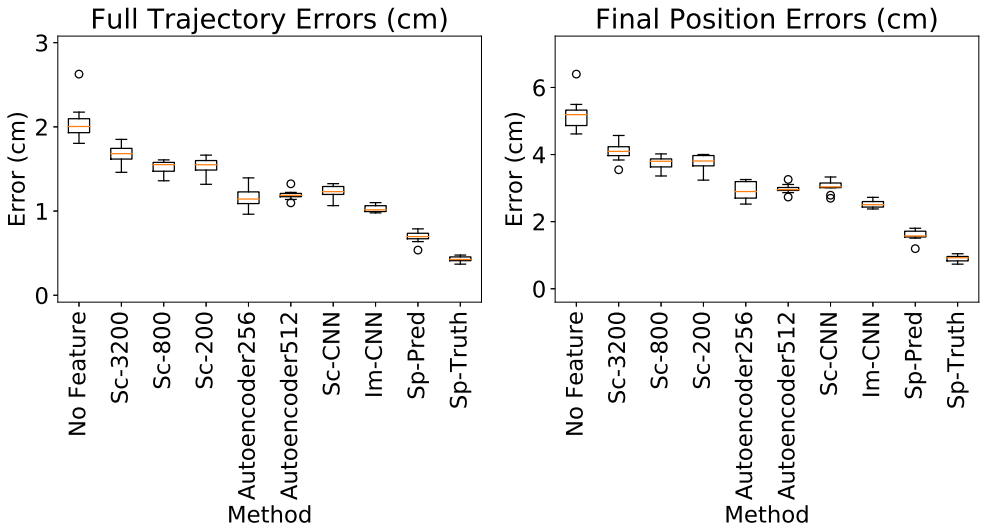


(a) (b)

Figure 9: (a) Change in support point prediction with increasing training size. (b) Generalization performance of the model. Our model is tested with the selected group of shapes while trained with the rest.

our model can learn to predict the support points successfully with a very high accuracy rate. Results also show that our model can reach around 90% accuracy using nearly 50% of the training data.

Next, we evaluated the generalization performance of our support prediction model. For this, instead of generating training and test sets from the completely shuffled data, objects with six shapes are used for training and objects with the remaining two shapes are used for the test. The objects that are used for the test was with 3&4 edges first, with 5&6 edges second, with 6&7 edges third and with 8&9 edges last. Figure 9b provides the generalization results where our models achieved around 89% prediction accuracy which is close to the full model accuracy (93%), in predicting novel shaped objects that was not provided in training phase. Results show that the parameter sharing encoder structure that we used in our support point prediction model allowed the network to generalize to the changing number



(a) Prediction error for full trajectory (b) Prediction error on final position

Figure 10: This figure shows prediction performance of the models over full trajectory error and final position error. Boxes corresponds distribution of errors for 10 different randomization seeds.

of corners successfully.

5.2. Object movement trajectory prediction

The performances of the models that predict the movement trajectory of the object are evaluated in this section. Figure 10 provides the model prediction errors where No-Feature (similar to random guess), shape context (Sc-200, Sc-800, Sc-3200), autoencoder (Autoencoder256, Autoencoder512), shape context based CNN (Sc-CNN), top-down raw image based CNN (Im-CNN), predicted support points ($\mathrm{SP}_{\mathrm{pred}}$) and real support points ($\mathrm{SP}_{\mathrm{truth}}$) are used as inputs to the LSTM trajectory predictor. For all features, one step ahead prediction models (n=1) are used. The error is defined as Euclidean distance between the real and predicted trajectory. For validation

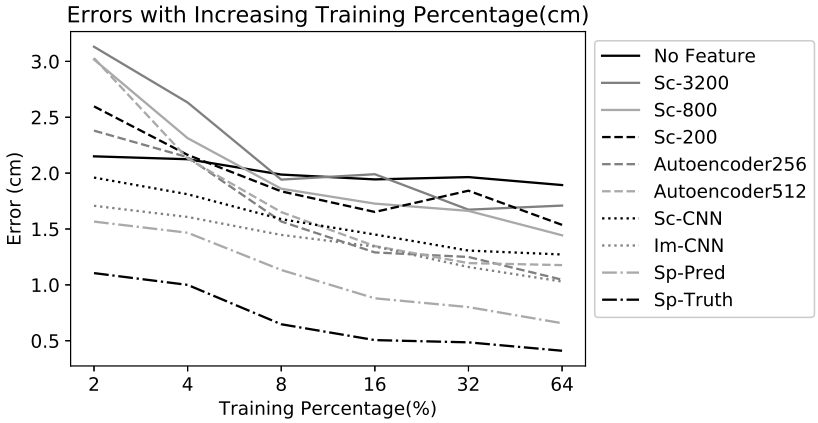


Figure 11: Effect of increasing training size on prediction performance.

of methods, Euclidean distance between the end-points of the real and the predicted trajectories are also shown. These errors are shown in Figure 10. Boxes of each method correspond to the distribution of mean error for data shuffled with 10 different seeds. In case no input features are used to predict the effect, the mean error is about 2 cm as shown with the first bar in the plot. The prediction error decrease through the use of unsupervised generic features such as shape context or autoencoders; and further decrease if supervised features such as image CNN's are used. Additionally, models using unsupervised features have higher variance on error compared to models using supervised features. Finally, the best accuracy is obtained through the use of task-specific features, real or predicted support points as shown. While the low error in models with manually designed features was expected, our general conclusion is that deep predictor that uses the raw image as input and effect as output provided better performance compared to unsupervised generic features that are considered to well-represent the geometry of the objects.

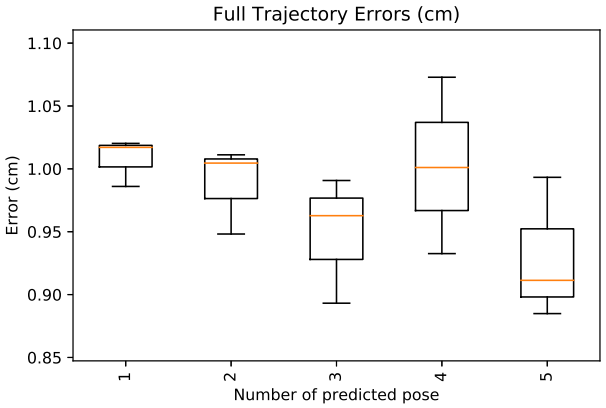


Figure 12: This figure shows full trajectory errors (cm) according to predicted step number at each iteration. It is shown that changing predicted pose numbers at each iteration results in a difference in the error less than 5 millimeters.

LSTMs enable predicting more than one step ahead at each iteration, which would decrease the run time of the model. For example if 50 time-step trajectory was predicted, performing one-step ahead prediction would require 50 iterations while two-step ahead predictions would require 25 iterations. We investigated the effect of number of future states the model predicts in one pass through training models with one to five step ahead predictions. As shown in Figure 12, no significant difference in error was found between these models.

We further analyzed how the prediction performance is affected by the training size. For this purpose, we trained our models with 2%, 4%, 8%, 16%, 32%, and 64% of the data set, and calculated the error with the remaining samples. Figure 11 shows the effect of increasing training set size on prediction error. Errors for CNN and especially autoencoder models quickly decrease as data size increases. As autoencoders see wider range of shape data by increased sample size, they can generate more general and robust

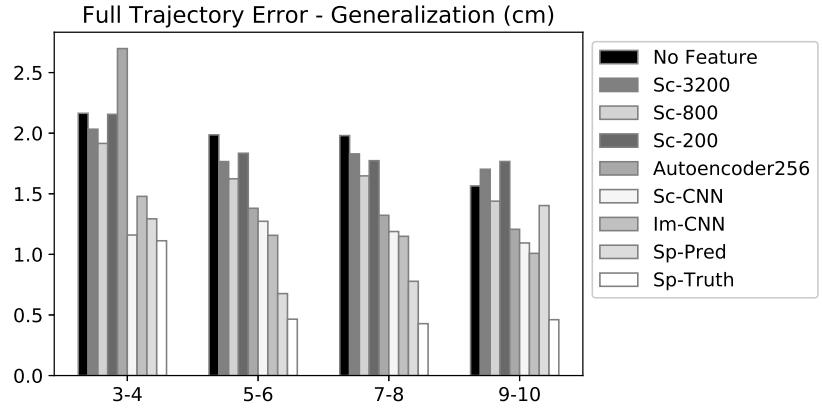


Figure 13: Generalization results. Each model is tested with a particular set of shapes and is trained with the remaining shapes. For example, the first group of bars correspond to the errors of different models that were tested with triangles and squares and trained with the remaining object types.

latent vectors, and models trained on these latent vectors provide better performance even catching the performance of Im-CNN. As expected, $\mathrm{SP}_{\mathrm{truth}}$ and $\mathrm{SP}_{\mathrm{pred}}$ models perform well even with small amount of data as they are fine-tuned to the task. Note that $\mathrm{SP}_{\mathrm{truth}}$ and $\mathrm{SP}_{\mathrm{pred}}$ are task-specific high-level features that already include knowledge about dynamics of the learning problem, therefore can bootstrap the learning whereas the supervised feature based Im-CNN performance catches up only with sufficient number of training data.

5.3. Model generalization analysis

In this section, we analyzed the generalization performance of the learned models to novel shapes. For this purpose, we trained our models faced with novel environments. In particular, each model is tested with two types of objects (e.g. triangle and rectangle) and trained with others. The prediction

performances of the models on novel shapes are provided in Fig. 13. Overall, performance degradation is graceful in all models. We observed that $\mathrm{SP}_{\mathrm{truth}}$ and $\mathrm{SP}_{\mathrm{pred}}$, and autoencoder exhibited poor generalization performance in the triangle-rectangle test set, whereas Im-CNN achieved similar generalization performance independent of the particular test set. Interestingly, the performance of the Sc-CNN is stable across different generalization test sets whereas the best performing shape context feature, Sc-800, fails in generalization tests. One can conclude that whether applied on raw image mask or on distributed shape context representation, convolution layers that are trained based on backpropagated errors enable stable generalization performance. This is probably due to the fact that spatial/topological information that has a direct influence on the effects is well-organized in 2d form, and can be extracted by convolution operation effectively. These results suggest that among the general models, Im-CNN achieves the best performance in both accuracy and generalization.

5.4. Verification in the real robot

In this section, we provide the evaluation of our model in the real world. For this, the prediction model that was trained in the simulator was transferred to the real robot and the predicted and actual manipulation trajectories are compared. A screwdriver was attached to the gripper of the real UR10 robot (Fig. 14). Two objects, a pentagon shaped object and a heptagon shaped one, were generated and printed using a 3D printer. The size of the generated objects were kept similar to their simulated counterparts. To test the model, each object was placed in different orientations so that it could be levered-up from each of its edges using the real screwdriver. The ob-

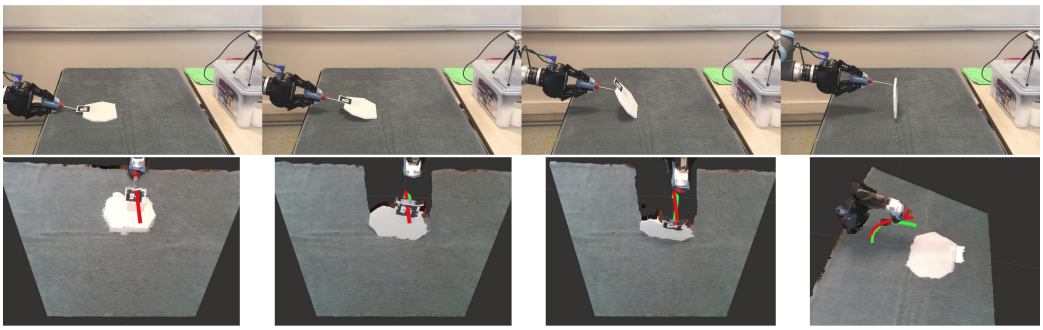


Figure 14: Real robot execution. The first row shows the environment while the execution. In the second row, first three images show the predicted trajectory (red line) and the trajectory of the object during the execution (green line) from the point of view of the camera. The last image shows the side view of the real and predicted trajectories after the execution is completed.

ject poses were tracked with an Intel real sense camera using ARtags placed on the objects. The best performing Im-CNN-based LSTM model was used to predict the trajectory from the top-down masks of the objects.

The sampling rates of the real and the simulated systems and therefore the lengths of the trajectories observed are different from each other. Therefore, Dynamic Time Warping method is applied to remove the timing discrepancy before comparing the trajectories observed in the real world and the trajectories predicted based on the simulation experience of the robot. Then, the distance between trajectories is calculated by taking the average of the point-by-point Euclidean distances of these trajectories. For comparison of 3D printed shapes, we generated the real-world setups of the by placing simulated objects to the same positions in the simulator and by measuring the simulated object movement trajectories. Tables 1 and 2 provide the distances between the real and predicted trajectories and the distances between the real and simulated trajectories. As shown, our predictions were very

	Edge 1	Edge 2	Edge 3	Edge 4	Edge 5
Real vs Simulation	0.485	0.520	0.379	0.385	0.474
Real vs Predicted	0.981	0.446	0.480	0.611	0.401

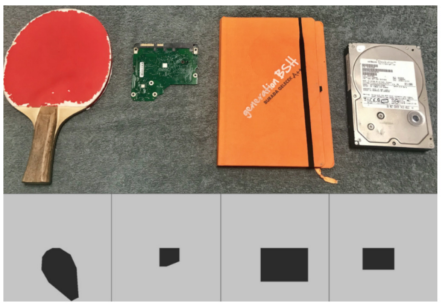
Table 1: The prediction and simulation error that was observed in pentagon shaped object experiments. Each column corresponds to a lever-up action that is applied to a particular edge of the object.

	Edge 1	Edge 2	Edge 3	Edge 4	Edge 5	Edge 6	Edge 7
Real vs Simulation	0.447	0.910	0.368	0.333	0.720	0.864	0.338
Real vs Predicted	0.601	0.633	0.460	0.316	0.531	0.499	0.471

Table 2: The prediction and simulation error that was observed in heptagon shaped object experiments. Each column corresponds to a lever-up action that is applied to a particular edge of the object

close to the ground truth: the distances between the trajectories were always below 1 cm in both pentagon and heptagon shaped objects, and lower than 0.5 cm for real life objects. In some cases, the prediction error was smaller than the simulation error, probably because the object was placed in the simulator with slight position/orientation error. These results show that our model can predict the effects of the real robot actions successfully.

To visualize our prediction model, we implemented a system that overlays the effect prediction, i.e. predicted movement trajectory of the object, on a graphical user interface (`http://wiki.ros.org/rviz`) that shows the online RGB-D output of the Kinect sensor. A number of snapshots from the real execution are provided in Fig. 14 where the match between the predicted (red colored) and the real (green colored) trajectories can be seen. We further



(a) Real Objects

HDD	PCB	Book	Racket
0.123	0.344	0.415	0.479

(b) Prediction errors for real objects

Figure 15: (a) Daily objects, namely table tennis racket, PCB, book and HDD , used in the real world experiments and their corresponding image masks. The masks are constructed using convex hulls of the object images and the configuration of the image is adjusted according to the action points. (b) Average errors (cm) between the predicted trajectory using the configured images and observed trajectory for each object.

provided a link to the robot execution video.[4]

We further provide the effect prediction results obtained from a number of flat daily objects, namely a book, a PCB, an HDD and a table tennis racket (Fig 15a). The masks of the objects were constructed by finding their corresponding convex hulls, and these masks were provided to the Im-CNN predictor as in the previous subsection. As shown in Figure 15b, the errors between the predicted and the real trajectories were around 0.5 cm for all objects, therefore indicating that our model can be successfully used to predict the effects of the actions on various daily objects.

5.5. Non-flat objects and depth-enriched image masks

In the previous sections, we focused on flat-like objects whose top-down images are sufficient to represent the physical features of the objects. Our

[4]Video: https://youtu.be/dFPOH1C3DeY

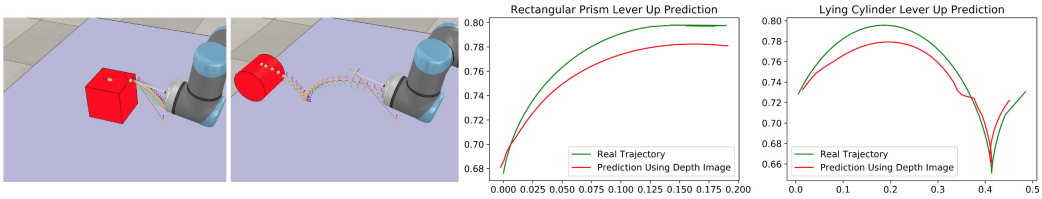


Figure 16: On the left side of the figure, two different type of objects, a cube and a lying cylinder, whose top-down image projections are not distinguishable but their resulting effects are completely different are shown. On the right side, it is shown that our model can differentiate the two different types of object effects by using depth-enriched images.

system would fail to distinguish action effects on non-flat objects that have the same top-down projections but different shapes. In this section, we enrich the top-down masks of the objects by adding depth information that is provided by a Kinect camera. We also expanded our dataset by including lying cylindrical and rectangular objects of various sizes that can only be differentiable by the depth knowledge. When original the top-down projection was used as input to Im-CNN, the average full-trajectory error in prediction was found to be 7.23 cm, whereas when depth-enriched masks were used, the average full-trajectory error dropped to 2.77 cm. Note that when lever-up action is applied on a lying cylinder object, the object starts rolling off the table, generating a completely different trajectory compared to the trajectory of a levered-up rectangular object that tumbles around its edge (Fig. 16). As shown in the figure, our system was to able predict the trajectories that correspond to qualitatively different effect categories.

6. Conclusion

In this study, we proposed and implemented a deep prediction model that enables a robot to predict the consequences of its manipulation actions

from its own interaction experience on objects of various shapes. Given the top-down image of the object, the robot learned to predict the movement trajectory of the object during execution of a lever-up action performed with a screwdriver in a physics-based simulator. We investigated the use of various feature descriptors such as shape context representation, autoencoder features, CNN features and finally manually engineered lever-up specific features. Our experimental results show that the highest prediction and generalization performance was obtained with manually engineered features which are real and predicted support points. However, as it is not feasible to manually encode different features for each different action and task, we used the performance obtained with these features as the upper bound. On the other hand, our proposed deep CNN-based LSTM model outperformed other models that used unsupervised (autoencoders) or generic (shape descriptor) features. It is shown that errors of the autoencoder models had a higher variance compared to Im-CNN model because autoencoders were unsupervised feature extractors that were trained independently of the subsequent recurrent neural network structure. Autoencoders aimed to reduce the dimensionality while preserving the important features of the data but this did not ensure that each resulting latent activation in the autoencoders is always the optimal one. In each training, the autoencoder may find a different way to reduce the dimensionality of the data and reconstruct it to the original size successfully while having different impacts with a high variance on the latter task. However, in Im-CNN model (or even in Sc-CNN model), image representations were directly connected to the recurrent neural network, allowing the model to learn always similar representations across different runs

using backpropagated error. As we used the top-down masks of the objects as inputs to CNNs, our model was easily transferred to the real-robot interaction settings where the movement of the real objects were successfully predicted.

Furthermore, we showed that our architecture is versatile and can also use depth images. When the rollable objects are considered, the trajectories of the affected objects are completely different (making spirals instead of an arc) as these objects are pushed away instead of being levered-up. Still, our model could learn to encode and predict these qualitatively different trajectories. On this basis, although our framework is verified using a lever-up action in this study, we believe that it can be used for other motion trajectory prediction tasks that require learning of the relations between geometries of objects and the corresponding non-linear motion trajectories.

Our robot learns to predict the action consequences that are determined by the related parameters of the environment, including friction coefficients. In case the robot engages with tool-object pairs with completely different friction coefficients, the predictions would fail as the robot did not learn how the friction affects the consequences of its actions. Therefore, for generalization, the robot is required to experience tool-object pairs of different friction coefficients. If the friction coefficient can be extracted from the environment, it can be used as input to the predictor; otherwise, the system needs to learn to infer this information from active exploration or use closed-loop controllers in executing its actions. While the focus of this paper is on object representations and learning methods for effect prediction, domain randomization techniques [34] that benefit from high-variability in simulations can be used

to handle different environment dynamics.

It is important to note that while our framework accepts action parameters as input, we did not add explicit action parameters in learning or prediction. In order not to be affected by kinematic constraints of the robotic platform, the lever-up action was always applied to a reference point with respect to the screwdriver. This is equivalent to encoding the perception and action relative to the tool or the end-effector. Such representations are known to be used in biological systems: target locations are encoded relative to hand in Parietal Cortex Area 5d during reaching action [8]. Previous robotics studies also exploited such representations in increasing the performance of object affordance classification [27]. In the future, we plan to study learning of consequences of actions that have complex and varying execution trajectories.

Acknowledgement(s)

This research has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 731761, IMAGINE.

References

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

[2] Al N, Nolen-Hoeksema S (2014) Atkinson and Hilgard's Introduction to

Psychology. Cengage Learning, URL https://books.google.com.tr/books?id=iEpZXwAACAAJ

[3] Battaglia P, Pascanu R, Lai M, Rezende DJ, et al. (2016) Interaction networks for learning about objects, relations and physics. In: Advances in neural information processing systems, pp 4502–4510

[4] Battaglia PW, Hamrick JB, Tenenbaum JB (2013) Simulation as an engine of physical scene understanding. Proceedings of the National Academy of Sciences 110(45):18327–18332

[5] Belongie S, Malik J, Puzicha J (2002) Shape matching and object recognition using shape contexts. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(4):509–522, DOI 10.1109/34.993558

[6] Biederman I (1987) Recognition-by-components: A theory of human image understanding. Psychological Review, 94(2), 115-147 DOI http://dx.doi.org/10.1037/0033-295X.94.2.115, URL http://www.sciencedirect.com/science/article/pii/0010027784900222

[7] Bohg J, Kragic D (2010) Learning grasping points with shape context. Robotics and Autonomous Systems 58(4):362–377

[8] Bremner LR, Andersen RA (2014) Temporal analysis of reference frames in parietal cortex area 5d during reach planning. Journal of Neuroscience 34(15):5273–5284

[9] Byravan A, Fox D (2017) Se3-nets: Learning rigid body motion using deep neural networks. In: Robotics and Automation (ICRA), 2017 IEEE International Conference on, IEEE, pp 173–180

[10] Chang MB, Ullman T, Torralba A, Tenenbaum JB (2016) A compositional object-based approach to learning physical dynamics. arXiv preprint arXiv:161200341

[11] Chollet F (2016) Building Autoencoders in Keras. `https://blog.keras.io/building-autoencoders-in-keras.html`

[12] Deisenroth M, Rasmussen CE (2011) Pilco: A model-based and data-efficient approach to policy search. In: Proceedings of the 28th International Conference on machine learning (ICML-11), pp 465–472

[13] Fang Y, Xie J, Dai G, Wang M, Zhu F, Xu T, Wong E (2015) 3d deep shape descriptor. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 2319–2328, DOI 10.1109/CVPR.2015.7298845

[14] Finn C, Goodfellow I, Levine S (2016) Unsupervised learning for physical interaction through video prediction. In: Advances in neural information processing systems, pp 64–72

[15] Fragkiadaki K, Agrawal P, Levine S, Malik J (2015) Learning visual predictive models of physics for playing billiards. arXiv preprint arXiv:151107404

[16] Garnelo M, Rosenbaum D, Maddison CJ, Ramalho T, Saxton D, Shanahan M, Teh YW, Rezende DJ, Eslami S (2018) Conditional neural processes. arXiv preprint arXiv:180701613

[17] Goel V, Gold B, Kapur S, Houle S (1998) Neuroanatomical correlates of human reasoning. Journal of cognitive neuroscience 10(3):293–302

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

[19] Hoffman D, Richards W (1984) Parts of recognition. Cognition 18(1):65 – 96, DOI https://doi.org/10.1016/0010-0277(84)90022-2, URL http://www.sciencedirect.com/science/article/pii/0010027784900222

[20] Jamone L, Ugur E, Cangelosi A, Fadiga L, Bernardino A, Piater J, Santos-Victor J (2018) Affordances in psychology, neuroscience and robotics: a survey. IEEE Transactions on Cognitive and Developmental Systems 10(1):4–25

[21] Kawato M (1999) Internal models for motor control and trajectory planning. Current opinion in neurobiology 9(6):718–727

[22] Köhler W, Winter E (1925) The Mentality of Apes. International library of psychology, philosophy, and scientific method, K. Paul, Trench, Trubner & Company, Limited, URL https://books.google.com.tr/books?id=j1IlAAAAMAAJ

[23] Lerer A, Gross S, Fergus R (2016) Learning physical intuition of block towers by example. arXiv preprint arXiv:160301312

[24] Montesano L, Lopes M, Bernardino A, Santos-Victor J (2008) Learning object affordances: from sensory–motor coordination to imitation. IEEE Transactions on Robotics 24(1):15–26

[25] Poggio T (1981) Marr's computational approach to vision. Trends in Neurosciences 4:258 – 262, DOI https://doi.org/10.1016/0166-2236(81)

90081-3, URL http://www.sciencedirect.com/science/article/pii/0166223681900813

[26] Ridge B, Ugur E, Ude A (2015) Comparison of action-grounded and non-action-grounded 3-d shape features for object affordance classification. In: Advanced Robotics (ICAR), 2015 International Conference on, IEEE, pp 635–641

[27] Ridge B, Ugur E, Ude A (2015) Comparison of action-grounded and non-action-grounded 3-d shape features for object affordance classification. In: Advanced Robotics (ICAR), 2015 International Conference on, IEEE, pp 635–641

[28] Russell SJ, Norvig P (2016) Artificial intelligence: a modern approach. Malaysia; Pearson Education Limited,

[29] Sahin E, Cakmak M, Dogar M, Ugur E, Ucoluk G (2007) To afford or not to afford: A new formalization of affordances towards affordance-based robot control. Adaptive Behavior 15(4):447–472

[30] Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:14091556

[31] Singh M, Seyranian GD, Hoffman DD (1999) Parsing silhouettes: The short-cut rule. Perception & Psychophysics 61(4):636–660

[32] Smith R, et al. (2005) Open dynamics engine

[33] Steedman M (2002) Plans, affordances, and combinatory grammar. Linguistics and Philosophy 25(5-6):723–753

[34] Tobin J, Fong R, Ray A, Schneider J, Zaremba W, Abbeel P (2017) Domain randomization for transferring deep neural networks from simulation to the real world. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, pp 23–30

[35] Ugur E, Piater J (2015) Bottom-up learning of object categories, action effects and logical rules: From continuous manipulative exploration to symbolic planning. In: Robotics and Automation (ICRA), 2015 IEEE International Conference on, IEEE, pp 2627–2633

[36] Ugur E, Oztop E, Sahin E (2011) Goal emulation and planning in perceptual space using learned affordances. Robotics and Autonomous Systems 59(7-8):580–595

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

[38] Xingjian S, Chen Z, Wang H, Yeung DY, Wong WK, Woo Wc (2015) Convolutional lstm network: A machine learning approach for precipitation nowcasting. In: Advances in neural information processing systems, pp 802–810

[39] Zech P, Haller S, Lakani SR, Ridge B, Ugur E, Piater J (2017) Computational models of affordance in robotics: A taxonomy and systematic classification. Adaptive Behavior 25(5):235–271, doi: 10.1177/1059712317726357

[40] Zhu Z, Wang X, Bai S, Yao C, Bai X (2016) Deep learning representation using autoencoder for 3d shape retrieval. Neurocomputing 204:41 – 50, DOI https://doi.org/10.1016/j.neucom.2015.08.127, URL http://www.sciencedirect.com/science/article/pii/S0925231216301047, big Learning in Social Media Analytics