A Visually-Grounded Library of Behaviors for Learning to Manipulate Diverse Objects across Diverse Configurations and Views

Anonymous Author(s)
Affiliation
Address
email

Abstract: We propose a visually-grounded library of behaviors approach for learning to manipulate diverse objects across varying initial and goal configurations and camera placements. Our key innovation is to disentangle the standard image-to-action mapping into two separate modules that use different types of perceptual input: (1) a behavior selector which conditions on *intrinsic* and *semantically-rich* object appearance features to select the behaviors that can successfully perform the desired tasks on the object in hand, and (2) a library of behaviors each of which conditions on *extrinsic* and *abstract* object properties, such as object location and pose, to predict actions to execute over time. The selector uses a semantically-rich 3D object feature representation extracted from images in a differential end-to-end manner. This representation is trained to be view-invariant and affordance-aware using self-supervision, by predicting varying views and successful object manipulations. We test our framework on pushing and grasping diverse objects in simulation as well as transporting rigid, granular, and liquid food ingredients in a real robot setup. Our model outperforms image-to-action mappings that do not factorize static and dynamic object properties. We further ablate the contribution of the selector's input and show the benefits of the proposed view-predictive, affordance-aware 3D visual object representations.

Keywords: Robot Learning, Visual Representation, Interactive Perception

1 Introduction

Object manipulation in unstructured environments is challenging since methods to manipulate objects largely depend on the object's visual appearance. One approach to capture the dependence between actions and visual features is to learn a direct mapping from image to actions with deep neural networks [1, 2, 3]. Despite their flexibility, such end-to-end image-to-action mappings have been shown to be very data aggressive [4], and cannot easily generalize across objects, camera viewpoints, or scene configurations [5].

Approaches that abstract away object details and encode only a subset of their properties, e.g., their 3D locations and velocities [6] or 3D keypoints [7, 8] make the state-to-action mapping easier to learn with less data. However, this abstraction may substantially limit the range of objects that a policy can handle, since useful information (object shapes, softness, weight, and material, for example) for the downstream task may be ignored. The challenging question is: how can we design a framework for object manipulation that uses **abstract** representations for sample efficient behavior learning, but at the same time is capable of utilizing **semantically-rich** representations for handling diverse objects and views.

We propose Visually-grounded library of BEhaviors (V-BEs), a hierarchical framework for vision-based object manipulation. Our main contribution is that the two levels of our policy hierarchy use different visual representations. At the lower level of the hierarchy, a behavior library contains a set of distinct behaviors each of which operates on an **abstract** object state representation that captures dynamic properties of objects in the environment, such as object and gripper positions over time.

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.

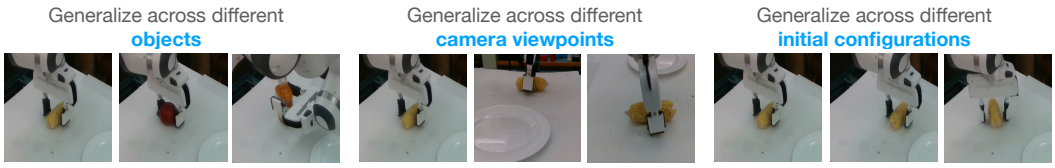


Figure 1: We propose a novel policy representation that generalizes to unseen objects, object placements and camera views.

Behaviors can be redundant, use different state abstractions, be closed loop or open loop, and be learned or manually engineered with different algorithms [9, 10, 11]. At the higher level of the hierarchy, a selector is trained using environment interactions to associate objects to behaviors that can successfully manipulate them. This selector takes as input RGB-D images of the scene, maps them to **semantically-rich** and view-invariant 3D object feature representations [12], and outputs the behavior that best suits the presented object and task. The decomposition of visual information into a perceptually rich selector network and behaviors with state abstraction makes the behaviours easier to learn with reinforcement learning or be designed by an engineer, and the selector easier to learn by trial-and-error learning as it does not bear the burden of producing low-level actions. At test time, we use 3D object detectors to localize the object, and present its location and size to the low-level controllers when necessary.

Our second contribution is proposing a visual representation for our behaviour selector that is robust to changing camera viewpoint (with respect to a non-changing robot torso), as well as object placements. We use the differentiable 2D-to-3D neural networks of Tung et al. [12] to map an input RGB-D image alongside camera intrinsics and extrinsics to transform 3D visual scene feature maps from the current camera viewpoint to the world frame, ensuring that visual features are bound to a static world frame—that coincides with the frame of the agent's body—and do not change dramatically while the camera moves. The 3D visual feature representations learn based on environment interactions to embed objects close to the behaviours that can manipulate them. They are further trained from the auxiliary task of view prediction using a multi-view camera setup, which encourages the model to complete missing information from the current view so the 3D visual feature maps are consistent across views and through occlusions.

We test the proposed model in two manipulation tasks: pushing and grasping. We show that our proposed framework can successfully manipulate diverse objects and generalize the learned skills to unseen objects with varying object starting positions, initial poses, goal locations, and camera viewpoints in simulation. We empirically show the proposed model outperforms existing end-to-end image-to-action mappings [4] and state abstracted object-to-action mappings that use only 3D object locations [6] in simulation. We ablate the contribution of our higher-level 3D feature representation used in our selector and then compare against 2D pretrained baselines. Lastly, we test our model in a real robot setup where the robot transports rigid, granular, and liquid objects onto a plate.

2 Related Work

Libraries of Behaviors: Libraries of behaviors have been considered in many previous works, and library elements have been called action, motion, behavioral, or motor primitives, parameterized policies, options, chunks, macros, subroutines, behavioral units, and skills, as well as many other terms. However, most previous work either assumes that the environment state is known and do not use perception, or has a different approach to perception than ours. Work of Neumann et al. on modular policies [13] defines a motor primitive as a mapping from a robot's joint angles and velocities and not the locations of objects. Rather, object locations are used as context for selecting a motor primitive. This means i) motor primitives are "blind" to the object 3D location throughout the episode, and ii) object appearance is discarded which means the library of primitives cannot handle object variability. Work of de Silva [14] on parameterized policies similarly considers as the contextual vector the desired goal location of a dart, and learns a mapping from this location to policy parameters, where each policy operates only over the robots' joints and is represented as a Dynamic Motion Primitive (DMP) [15, 16, 17]. To extend DMPs to tasks involving physical interactions, Kroemer et al. [18] parameterize DMPs based on the distance of an object part from the gripper, but only consider known object 3D shapes. Strudel et al. [19] use a depth frame as

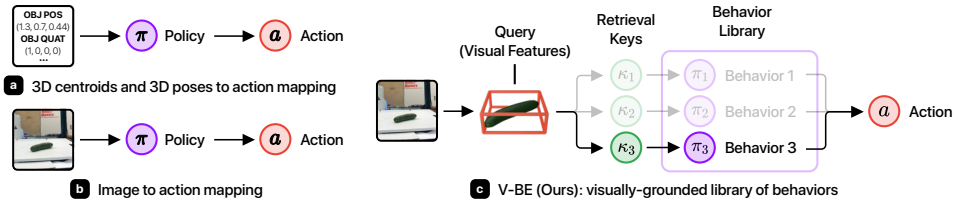


Figure 2: Comparison of our model with previous approaches. In contrast to state-to-action or image-to-action mapping, the proposed framework decomposes a policy into a behavior selection module and a library of behaviors to select from. This decomposition enables the selection module to work on a different state representation from each behavior in the library of behaviors: the selection module operates in a semantically-rich visual feature space, while the behaviors operates in an abstract object state space that facilitate efficient policy learning.

input to a meta-policy that selects skills to execute. However, the use of modularity over skills is for temporal sequencing: there is no mixture of behaviors per manipulation task, rather an image to action mapping is learnt with behavior cloning from demonstrations.

Deep Reinforcement and Imitation Learning for Object Manipulation: Current work in (deep) reinforcement learning typically considers only one monolithic policy is learned via reinforcement learning [9, 20, 21], imitation learning [22, 23, 1], or a combination of the two [24, 25]. Most approaches take as input 3D object and gripper centroids, orientations, and velocities, and ignore contextual visual information [26]. Existing methods are commonly trained in a fixed environment and evaluated in terms of their ability to discover a policy rather than their ability to generalize it to previously unseen circumstances, e.g., novel objects, which is the goal of this work.

Methods that do aim to generalize across objects learn a mapping from images-to-actions [4], depth-to-actions [27] or pointcloud-to-actions [28, 29]. They have been successful in object grasping [27, 30] and object pushing from a fixed camera view [31, 4]. Seminal works of [4, 32] train an image-to-action deep neural network by imitating trajectories obtained from planners that operate in a low-dimensional state space. Nevertheless, such monlithic image-to-action deep mappings don't generalize well across different camera viewpoints [5, 33, 34] and objects. Works that attempt to transfer visuomotor policies learned in simulation to the real-world often require identical placement of the camera in the real world [35, 6]. Florence et al. [36] train manipulation policies parameterized by the 2D or 3D locations of a designated set of visual descriptors obtained from RGBD images, and show their policies generalize across objects of the same object category, e.g., shoes of different shapes. The proposed work further allows generalization across object categories, and no explicit descriptor selection or optimization is necessary. Furthermore, our framework permits each behaviour to use a different state representation.

Hierarchical Reinforcement Learning (HRL): Most existing HRL methods [37, 38] focus on discovering behaviors and temporally sequencing them for long-horizon tasks. This paper instead uses a hierarchical vision-based policy architecture to improve model generalization across objects, configurations, and viewpoints.

3 Methods

Problem Setup We consider the problem of manipulating (e.g. grasping, pushing, etc.) diverse sets of objects from various initial configurations and camera viewpoints. At training time, the agent has access to training objects $\mathcal{O}_{\text{train}}$, RGB-D images from J viewpoints $v_1, \ldots, v_J \in \mathcal{V}$, and it can interact with the training object from randomly selected viewpoints. At test time, the agent has to perform the same manipulation task as at training time on test objects $\mathcal{O}_{\text{test}}$ from selected viewpoints. Performance is measured by the success rate of manipulating all test objects with random initial configurations and camera viewpoints.

Our framework is comprised of two major components — (1) a library of behaviors $\Pi = \{\pi_i \mid i = 1, 2, \ldots, K\}$ and (2) a selector function G. Given an RGBD image I of the scene captured from camera view v, denote by $I_v = \{I, v\}$, and an object 3D bounding box o, the selector G obtains the probability of successfully manipulating the object when applying behavior π_i on the object by computing a query object feature representation $f(I_v, o)$ and compares it with learned key embeddings

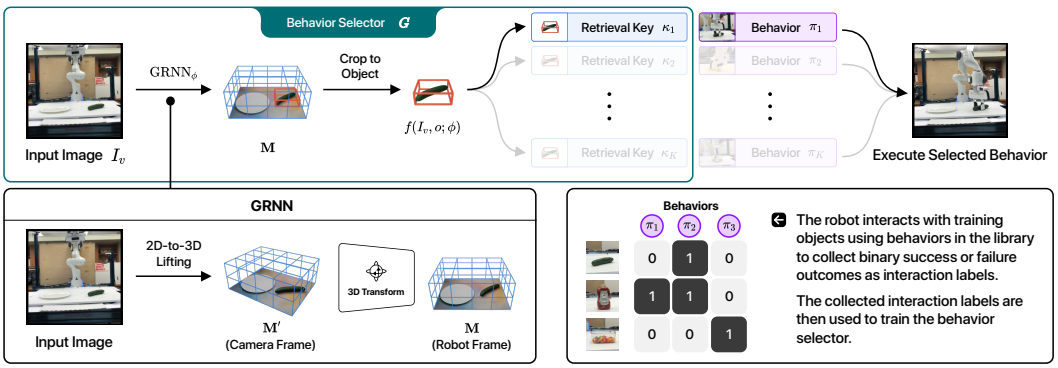


Figure 3: Overview of the proposed framework.

$\kappa_i = \kappa(\pi_i)$, $i = 1, ..., K$ associated with each behavior π_i,

$$\mathrm{G}(I_v, o, \pi_i; \phi, \kappa) = \sigma(\langle f(I_v, \mathrm{o}; \phi), \kappa_i \rangle) \in [0, 1], \tag{1}$$

where ϕ is the learnable neural network parameters for function f, $\langle \cdot, \cdot \rangle$ is the inner product operation and σ is the sigmoid function. We will detail the feature dimension of $f(I_v, o; \phi)$ and κ in the next section. At test time, we select the behavior π_i that has the highest probability of leading to successful manipulation.

Each behavior π_i uses abstract state representations for the objects to manipulate, for example, 3D object locations and 3D poses, and does not take into account object shape or appearance. As a result, each behaviour can handle only a subset of object shapes and orientations. To handle a diverse set of objects and orientations, our selector learns to pick different behaviors for different objects or orientations. One advantage of the modularity of our policy architecture is that each behavior can use its own state abstraction, such as object 6D-poses, 3D bounding boxes, part-based 3D boxes, or 2D or 3D object keypoints locations [7]. Our framework supports integrating a wide range of existing models, representations, and controllers as selectable behaviors.

In the rest of this section, we detail the architecture and training details of the behavior selector in Section 3.1, and describe how we acquire a library of behaviors for grasping and pushing—the two robot manipulation tasks we evaluate our framework on—in Section 3.2.

3.1 Visually-Grounded Behavior Selector (G)

The behavior selector G is a classifier that learns object-centric visual feature representation $f(I_v, \mathrm{o}; \phi)$ for the object box o in image view I_v, and behavioural key embeddings $\kappa^i, i = 1..K$ for the behaviours in the library to retrieve behaviours compatible with a particular object.

Training the selector with interaction labels. We learn the object feature representation and behavioural keys through trial-and-error. In each trial, our agent applies a randomly sampled behaviour π_i on an object o in the workspace which results in binary success or failure outcome $\ell \in \{0, 1\}$, which we call *interaction labels*. Agent interactions are organized as tuples of the form $(I_v, \mathrm{o}, \pi_i, \ell)$. With the interaction experience as training data $\mathcal{T}$, we train the feature encoder f and behavioural keys $\kappa_i = \kappa(\pi_i), i = 1...K$ with the loss:

$$\mathcal{L}_{\text{afford}}(\kappa, \phi) = \sum_{(I_v, \mathrm{o}, \pi_i, \ell) \in \mathcal{T}} \mathrm{BCE}(\ell, \mathrm{G}(I_v, \mathrm{o}, \pi_i; \phi, \kappa)) = \sum_{(I_v, \mathrm{o}, \pi_i, \ell) \in \mathcal{T}} \mathrm{BCE}(\ell, \sigma(\langle f(I_v, \mathrm{o}; \phi), \kappa_i \rangle)) \tag{2}$$

where $\mathrm{BCE}(y, p) = -y \cdot \log(p) - (1 - y) \log(1 - p)$ is the binary cross entropy loss.

3D object feature representation. Our object feature representation $f(I_v, \mathrm{o}; \phi)$ is computed using Geometry-aware Recurrent Neural Networks (GRNNs) of Tung et al. [12], which are end-to-end differentiable architectures that map a single RGB-D image or a set of multi-view images to 3D feature representation of the scene the image(s) depict. Given a posed image I_v, GRNNs obtain scene-centric 3D feature representation $\mathbf{M} = \mathrm{GRNN}_\phi(I_v) \in \mathbb{R}^{H \times W \times D \times C}$ through differentiable 2D-to-3D operations, 3D convolution-based refinement, and 3D rotation operations that align the

3D feature representations with the robot's coordinate frame (as opposed to camera frame), with W, H, D denoting the width, height and depth of the scene map and C denotes the feature dimension of the 3D scene feature map.

From the scene map $\mathbf{M}$, we obtain object-centric feature representation $f(I_v, \mathrm{o}; \phi) = \mathrm{crop}(\mathbf{M}, o) \in \mathbb{R}^{64\times64\times64\times32}$ by cropping the scene map using a fixed-size axis-aligned box o, centered around the object we wish to manipulate. The feature cropping operation is similar to the one used in Mask-RCNN [39]. The retrieval policies keys are also learned in the same representation space. Thus, both $f(I_v, \mathrm{o}; \phi)$ and $\kappa_i, i = 1..K$ have length $64 \times 64 \times 64 \times 32$ in the experiments.

3D Object detector We learn a detector with 3D Mask-RCNN built on top of the GRNNs feature encoder [12]. We use groundtruth 3D object boxes at training time, and predicted 3D object boxes at test time, where we train our representation to detect objects in 3D.

View prediction and occupancy prediction as an auxiliary task. We use view prediction and occupancy prediction as an auxiliary task to help our image encoder generalize better in its ability to select behaviours. These two self-supervised prediction tasks has been shown to provide a useful pretraining or co-training objective for 3D object detection in [40]. Given an input posed image I_v^n and a query view q^n, the overall self-supervised prediction loss reads:

$$\mathcal{L}_{\text{self-pred}}(\phi, \theta, \eta) = \sum_{n=1}^{N} \underbrace{\|P_\theta(\text{GRNN}_\phi(I_v^n), q^n) - \bar{I}_q^n\|_2^2}_{\text{view prediction loss}} + \underbrace{\|\text{Occ}_\eta(\text{GRNN}_\phi(I_v^n)) - \text{occ}^n\|_1}_{\text{occupancy prediction loss}}, \quad (3)$$

where $P_\theta(\mathbf{M}, q)$ is a projection function that projects a 3D feature map $\mathbf{M}$ from the query viewpoint q to a 2D feature map and decodes it to a target image $\bar{I}_q^n$ using image decoder with neural network weights θ, $\text{Occ}_\eta(\mathbf{M}) \in \mathbb{R}^{64\times64\times64}$ is a voxel occupancy prediction function that predicts 3D occupancy map from an input 3D feature map $\mathbf{M}$ using a single 3D convolution layer with weights η, and occ^n is the estimated occupancy map computed from all available input views in the nth data point by voxelizing the unprojected point clouds from all available depth images. We train the model from unlabelled multi-view images captured around the table by simply moving the cameras, capturing the images, and recording the corresponding camera locations.

The final objective for training the affordance-based visual features is

$$\underset{\kappa,\phi,\theta,\eta}{\text{minimize}}\, \mathcal{L}(\kappa, \phi, \theta, \eta) = \mathcal{L}_{\text{self-pred}}(\phi, \theta, \eta) + \lambda_{\text{a}} \cdot \mathcal{L}_{\text{afford}}(\kappa, \phi), \quad (4)$$

where λ_{a} is a hyperparameter for balancing the two losses.

3.2 Building a Library of State Abstracted Behaviors

Any existing behaviors, whether engineered or learned using reinforcement or imitation learning, can be included in our library. This flexibility is a contribution of our modular architecture. In this paper, we consider two common manipulation tasks: pushing and grasping, and we build appropriate behavioral libraries for each.

For pushing, the behaviors are deterministic goal conditioned policies $a_t = \pi(s_t, g)$ that map a state of the environment and the robot $s_t = [s_t^e, s_t^r]$ and a goal state g to an action a_t at time step t. The environment state s_t^e is the 3D object centroid and the robot state s_t^r is the gripper 3D location, pose, and whether it is opened or closed. Actions include 3D translation, opening (position control), and closing (force control) of the gripper. A goal state g is a target centroid location for the object. We use a total of 25 goal conditioned policies – one is trained from the whole set of objects, while the others are trained on disjoint subsets of object configurations organized based on object category and initial poses. We train all policies using deterministic policy gradients (DDPG) [25] with goal relabelling (HER) [9] while randomizing initial and goal object 3D locations.

For grasping, we design controllers $\pi(a_t|g; p^{grasp}, q^{grasp})$ which given a 3D grasping point $p^{grasp} \in \mathbb{R}^3$ relative to the center of the object and a grasping 3D angle $q^{grasp} \in \mathbb{R}^2$, move the gripper (open loop) to the grasping 3D point location, close it, and move it to the desired goal location. The grasping angle q^{grasp} consists of two numbers describing the yaw of the gripper and the elevation angles between the gripper and the table surface. When the elevation angle is smaller than 90 degrees (not top-down grasps), we constraint the gripper to point toward the center of the object on the x-y plane. We manually select 30 different controllers including top-down grasps with

different yaw orientations (top-grasps) and grasps from the side with different elevation angles of the gripper (side-grasps). We empirically found that these parameterized controllers are quite stable and can be shared across multiple objects.

4 Experiments

Our experiments aim to answer the following questions: (1) Does the proposed library-based approach outperform existing methods that use a single combined perception and policy module, either using 2D images, 3D object locations, or 3D scene feature maps as input? (2) Is the proposed view-invariant and affordance-aware 3D feature representation a necessary choice for the selector? (3) Does the method work on a real robot? We test our model on grasping and pushing a wide variety of objects in the MuJoCo simulator [41] and further test a transporting task on a real-world Franka Panda robot arm.

4.1 Simulation Experiments

Our simulated environment consists of a Fetch Robot equipped with a parallel-jaw gripper. The robot is positioned in front of a table of height $0.4m$. To obtain the visual observations, on each episode we choose 3 random cameras from cameras placed at 30 nominal different views including 10 different azimuths ranging from 0° to 360° combined with 3 different elevation angles from 20°, 40°, 60°. All cameras are looking at the center of the table top, and are 0.5 meter away from that point. All images have size 128×128.

Task Descriptions: In the grasping task, the agent has to grasp an object and move it to a specified target location above the table. We use 274 distinct object meshes from 6 categories in ShapeNet [42] including toy buses, toy cars, cans, bowls, plates, and bottles. We randomly split the dataset into 207 training objects, and 67 testing objects. After augmenting the meshes with random scaling from 0.8 to 1.5 and random rotations around the vertical z-axis, we get a total of 800 distinct object configurations (object instance and pose), 600 for training and 200 for testing. At the start of each episode, an object is placed in an area of $30cm \times 16cm$ around the center of the table, and a goal is sampled uniformly $10 \sim 30cm$ away from the gripper's initial position. An episode is successful if the object centroid is within $5cm$ of the target at the final timestep.

In the pushing task, the agent has to push an object placed on the table to a specified target location. We use 100 objects from 12 categories in ShapeNet [42]: baskets, bowls, bottles, toy buses, cameras, cans, caps, toy cars, earphones, keyboards, knives, and mugs. After augmentation and splitting to train and test sets, we obtain 615 training object configurations and 200 for testing. The initial and the goal position of the object are both uniformly sampled to be within $15cm$ of the center of the table along both x-axis and y-axis, although we resample if that location is already in the goal area. An episode is successful if the object centroid is within $5cm$ of the goal within 50 timesteps.

Baselines: We compare our method with various learning and non-learning based methods for object manipulation:

(a) *Single Behavior w/ Abstract 3D State (Abstract 3D) [9, 25]:* a policy takes as input ground truth 3D bounding box of the object and gripper and outputs actions.

(b) *Single Behavior w/ Abstract 3D State and 2D Images (Abstract 3D + Image):* a policy takes both RGB-D images and the ground truth 3D bounding box as inputs and outputs actions. Our architecture resembles that of [43], but we further include ground truth object position as extra inputs to the model. For fair comparisons to other methods, the model only takes as input the current state as opposed to the states in 5 past steps, as in [43].

(c) *Single Behavior w/ 3D Feature Tensor (Contextual 3D):* a policy that takes as an input RGB-D images and the ground truth 3D bounding box and outputs actions. Different from (b), the model first transforms the image into a view-invariant 3D feature tensor using GRNNs [12], then converts the 3D feature tensor into a feature vector though three 3D-convolutional layers and a fully connected layer, and concatenates it with the rest of the inputs to predict actions.

(d) *Ours, Library of Behaviors w/ Visual Selector (V-BEs):* Our model takes the same input as (b) and (c). The 3D bounding boxes are used as input to all the behaviors. The RGB-D images are transformed into 3D affordance-aware visual features and treated as input to the selector.

	Single Behavior			Library of Behaviors	Ablation study on the selector's visual feature representation	
	Abstract 3D [9, 25]	Abstract 3D + Image	Contextual 3D	V-BEs (Ours)	V-BEs w/ 2D features	V-BEs w/o Fine-tuning on Interaction Labels [45, 12]
grasping	0.30	0.35	0.20	**0.78**	0.46	0.31
pushing	0.83	0.70	0.10	**0.88**	0.81	0.46

Table 1: Success rates on grasping and pushing unseen objects. We also ablate the proposed method with selectors operating on varying representations.

We train the baselines with different learning methods including behavior cloning [4], DDPG-HER [11, 9] and DAGGER [44]. We report the best performance we got by training the model with these different methods. We also attempt to make all the models have similar number of parameters so the comparison is fair. However, larger networks are empirically harder to train and do not converge well, so we instead increase the number of parameters in smaller networks until their performance saturates. For pushing, we found that using DDPG-HER is enough to lean a good *Abstract 3D* policy from scratch. For *abstract 3D + Image*, we found it is critical to use behavior cloning from expert demonstrations to obtain good policies. The expert demonstrations are obtained from trained expert policies on single objects. For *Contextual 3D*, we include DAGGER to enforce behavior cloning during execution. To train the grasping policies, we further include human demonstrations in the replay buffer when training it with DDPG-HER. Both *abstract 3D + Image* and *Contextual 3D* are trained with DAGGER since offline behavior cloning is insufficient.

4.2 Single Behavior versus a Library of Behaviors

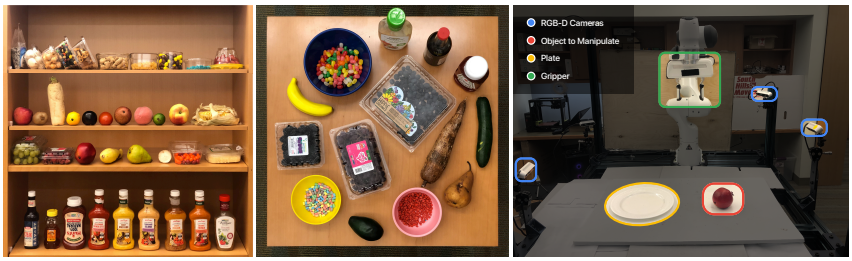
We compare the proposed model with models that do not use a library-based approach, i.e., single behavior approaches. As shown in Table 1, our method outperforms all the single behavior baselines. *Abstract 3D* performs well, but since it does not use any visual information, its performance saturates at around 0.8 for pushing and 0.3 for grasping. *Abstract 3D* performs poorly for grasping. The learned behaviors do not transfer well to new objects. Adding a 2D image helps, but not dramatically (see *Abstract 3D + Image* in Table 1). Although 3D feature maps obtained from GRNNs are semantically rich and can handle varying viewpoints, the mapping to actions is harder to learn due to the higher dimensionality of the 3D scene map, resulting in under-fitting models. Our model takes advantage of both abstract and semantically rich representation and thus can handle better object variability and transferability. The combinatorial nature of the proposed method allows the model to capture the multi-modality in trajectory generation.

4.3 The Necessity of Building the Selector with the Proposed 3D Representations

Next, we show the importance of using view-invariant 3D visual feature representations and fine-tuning the selector with interaction labels. We compare our method with two baselines: (a) a model with a selector that learns the visual affordance features using 2D visual features extracted from 2D CNNs, and (a) a model with a selector that operates over 3D visual feature representation learned only with the view and occupancy prediction loss, as suggested in [45, 12], without fine tuning with interaction labels. See Table 1 for the results. Our method significantly outperforms these two baselines, which shows the importance of both proposed components. To fully test the power of existing 2D CNNs, we also tested 2D feature selector with existing VGG network [46] pretrained on ImageNet and fine-tuned on our interaction labels. However, the performance does not differ too much with shallower 2D CNNs training from scratch (We obtain a success rate of 0.78 on pushing with the pretrained VGG network).

4.4 Transporting Task on a Real Robot

We test our model on a 7-DOF Franka robot arm equipped with a parallel-jaw gripper for a transporting task, where the robot needs to transport various rigid, granular, or liquid food ingredients from random initial positions and poses onto a plate (see Figure 4). We set up 4 Intel RealSense RGB-D cameras that have full view of the workspace around the the center of the table. In each trial, an object is placed in a 50cm $\times$ 30cm region on the table, and the goal is transport all objects to a plate 25cm to the left of the starting region. Granular objects and liquids are placed inside containers in the beginning of the trial. An episode is considered successful if the object is successfully



Training Objects Testing Objects Workspace Setup

Figure 4: Real robot experimental setup. We set 4 cameras around the table that cover different viewpoints of the workspace. 3 out of 4 cameras are shown in the right most image. An extra topdown camera is used, but it is clipped from the image.

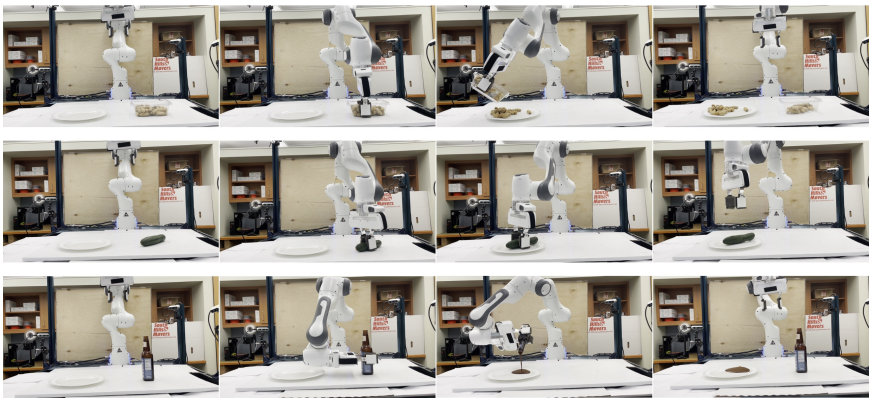


Figure 5: Sample real robot executions. Each row shows an execution trial during test time. Our robot can successfully transport rigid, granular, and liquid food ingredients to a target plate.

transported into the plate. For granular objects and liquid, an episode is considered successful if at least half of the total quantity ends up in the plate.

We construct our library of behaviors with 26 controllers, in which 13 of them are pick-and-place controllers from various grasping angles and the other 13 are pick-and-pour controllers using the same grasping angles as the pick-and-place ones. At evaluation, we randomly select 3 views from 4 possible camera views to obtain RGB-D images as inputs to the learned selector. We use 20 rigid objects, 20 granular objects, and 12 bottles of liquids for our experiment, and we split them into 38 training object and 14 testing objects. To train our model, we collect a total of 3510 interaction labels by running the 26 behaviors on the training objects with random initial poses and using the labels to finetune the visual selector with the objective specified in Equation (4). During evaluation, our model can achieve a success rate of **88.6%** on the test set. Sample executions and transporting objects in clutter are visualized in Figure 5 and included in our supplementary video.

Limitations and Future Work Our current framework only supports manipulating single objects. For behaviours that involve multiple objects, we would need a scene graph representation that considers the spatial interactions across different objects. Another important avenue is to scale up the library of behaviours to solve diverse tasks. Recent advances in large-scale imitation learning from human demonstration [47] will be a promising direction to investigate.

5 Conclusion

We have presented V-BEs, a hierarchical policy architecture where 3D object visual feature representations are used to select from a library of behaviours. The proposed modular architecture supports both low-level behaviours and the selector to be learnt in a data efficient manner. We have shown results on pushing and grasping diverse objects in simulation and the real world, across diverse viewpoints. Our method outperforms image-to-action monolithic policies of previous works, as well as policies that operate on 3D locations and velocities alone. Since our framework is sample-efficient and simple to run, we can easily deploy complex transporting skills on a real robot arm.

References

[1] T. Zhang, Z. McCarthy, O. Jow, D. Lee, K. Goldberg, and P. Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. *ICRA*, pages 1–8, 2018.

[2] D. Gandhi, L. Pinto, and A. Gupta. Learning to fly by crashing. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3948–3955. IEEE, 2017.

[3] D. Pathak, P. Mahmoudieh, G. Luo, P. Agrawal, D. Chen, Y. Shentu, E. Shelhamer, J. Malik, A. A. Efros, and T. Darrell. Zero-shot visual imitation. In *ICLR*, 2018.

[4] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[5] S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn. Robonet: Large-scale multi-robot learning, 2019.

[6] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[7] Z. Qin, K. Fang, Y. Zhu, L. Fei-Fei, and S. Savarese. Keto: Learning keypoint representations for tool manipulation. *arXiv preprint arXiv:1910.11977*, 2019.

[8] L. Manuelli, Y. Li, P. Florence, and R. Tedrake. Keypoints into the future: Self-supervised correspondence in model-based reinforcement learning, 2020.

[9] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba. Hindsight experience replay. *arXiv preprint arXiv:1707.01495*, 2017.

[10] S. Feng, X. Xinjilefu, C. G. Atkeson, and J. Kim. Optimization based controller design and implementation for the atlas robot in the darpa robotics challenge finals. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 1028–1035, 2015.

[11] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. M. O. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2016.

[12] H.-Y. F. Tung, R. Cheng, and K. Fragkiadaki. Learning spatial common sense with geometry-aware recurrent networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[13] G. Neumann, C. Daniel, A. Paraschos, A. Kupcsik, and J. Peters. Learning modular policies for robotics. *Frontiers in computational neuroscience*, 8:62, 2014.

[14] B. D. Silva, G. Konidaris, and A. Barto. Learning parameterized skills, 2012.

[15] E. Rückert and A. d'Avella. Learned parametrized dynamic movement primitives with shared synergies for controlling robotic and musculoskeletal systems. *Frontiers in computational neuroscience*, 2013.

[16] S. Schaal. Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In *Adaptive motion of animals and machines*, pages 261–280. Springer, 2006.

[17] F. Stulp, E. Theodorou, and S. Schaal. Reinforcement learning with sequences of motion primitives for robust manipulation. *Robotics, IEEE Transactions on*, 28:1360–1370, 12 2012. doi:10.1109/TRO.2012.2210294.

[18] O. Kroemer and G. S. Sukhatme. Learning relevant features for manipulation skills using meta-level priors. *arXiv preprint arXiv:1605.04439*, 2016.

[19] R. Strudel, A. Pashevich, I. Kalevatykh, I. Laptev, J. Sivic, and C. Schmid. Learning to combine primitive skills: A step towards versatile robotic manipulation, 2019.

[20] O. Kilinc, Y. Hu, and G. Montana. Reinforcement learning for robotic manipulation using simulated locomotion demonstrations, 2019.

[21] N. Wahlström, T. B. Schön, and M. P. Deisenroth. From pixels to torques: Policy learning with deep dynamical models. *arXiv preprint arXiv:1502.02251*, 2015.

[22] N. Ratliff, J. A. Bagnell, and S. S. Srinivasa. Imitation learning for locomotion and manipulation. In *2007 7th IEEE-RAS International Conference on Humanoid Robots*, pages 392–397, 2007.

[23] M. Sieb, X. Zhou, A. Huang, O. Kroemer, and K. Fragkiadaki. Graph-structured visual imitation. In *Conference on Robot Learning (CoRL)*, 2019.

[24] L. Fan, Y. Zhu, J. Zhu, Z. Liu, O. Zeng, A. Gupta, J. Creus-Costa, S. Savarese, and L. Fei-Fei. Surreal: Open-source reinforcement learning framework and robot manipulation benchmark. In *Conference on Robot Learning*, pages 767–782. PMLR, 2018.

[25] A. Rajeswaran, V. Kumar, A. Gupta, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *CoRR*, abs/1709.10087, 2017. URL http://arxiv.org/abs/1709.10087.

[26] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang. Solving rubik's cube with a robot hand, 2019.

[27] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. Aparicio, and K. Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. 07 2017. doi:10.15607/RSS.2017.XIII.058.

[28] A. Mousavian, C. Eppner, and D. Fox. 6-dof graspnet: Variational grasp generation for object manipulation. *CoRR*, abs/1905.10520, 2019. URL http://arxiv.org/abs/1905.10520.

[29] Y. Qin, R. Chen, H. Zhu, M. Song, J. Xu, and H. Su. S4g: Amodal single-view single-shot se(3) grasp detection in cluttered scenes, 2019.

[30] L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. *CoRR*, abs/1509.06825, 2015. URL http://arxiv.org/abs/1509.06825.

[31] P. Agrawal, A. Nair, P. Abbeel, J. Malik, and S. Levine. Learning to poke by poking: Experiential learning of intuitive physics. *arXiv preprint arXiv:1606.07419*, 2016.

[32] T. Tosun, E. Mitchell, B. Eisner, J. Huh, B. Lee, D. Lee, V. Isler, H. S. Seung, and D. Lee. Pixels to plans: Learning non-prehensile manipulation by imitating a planner. *arXiv preprint arXiv:1904.03260*, 2019.

[33] M. Riedmiller, R. Hafner, T. Lampe, M. Neunert, J. Degrave, T. Wiele, V. Mnih, N. Heess, and J. T. Springenberg. Learning by playing solving sparse reward tasks from scratch. In *International Conference on Machine Learning*, pages 4344–4353. PMLR, 2018.

[34] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel. Asymmetric actor critic for image-based robot learning. *arXiv preprint arXiv:1710.06542*, 2017.

[35] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *CVPR*, pages 12627–12637, 2019.

[36] P. Florence, L. Manuelli, and R. Tedrake. Self-supervised correspondence in visuomotor policy learning, 2019.

[37] T. D. Kulkarni, K. R. Narasimhan, A. Saeedi, and J. B. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *arXiv preprint arXiv:1604.06057*, 2016.

[38] J. T. Springenberg, K. Hausman, M. Riedmiller, N. Heess, and Z. Wang. Learning an embedding space for transferable robot skills. 2018.

[39] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017. URL http://arxiv.org/abs/1703.06870.

[40] A. W. Harley, S. K. Lakshmikanth, F. Li, X. Zhou, H.-Y. F. Tung, and K. Fragkiadaki. Learning from unlabelled videos using contrastive predictive neural 3D mapping. *ICLR*, 2020.

[41] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[42] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[43] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *ICRA*, pages 1–8. IEEE, 2018.

[44] S. Ross, G. J. Gordon, and J. A. Bagnell. No-regret reductions for imitation learning and structured prediction. In *In AISTATS*. Citeseer, 2011.

[45] A. W. Harley, F. Li, S. K. Lakshmikanth, X. Zhou, H.-Y. F. Tung, and K. Fragkiadaki. Embodied view-contrastive 3d feature learning. *arXiv preprint arXiv:1906.03764*, 2019.

[46] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[47] S. Young, D. Gandhi, S. Tulsiani, A. Gupta, P. Abbeel, and L. Pinto. Visual imitation made easy, 2020.