

From explanation to synthesis: Compositional program induction for learning from demonstration

Michael Burke*, Svetlin Penkov†, Subramanian Ramamoorthy*†

*Institute for Perception action and Behaviour
School of Informatics, University of Edinburgh

†FiveAI

Email: {mburke33, sv.penkov, s.ramamoorthy}ed.ac.uk

Abstract—Hybrid systems are a compact and natural mechanism with which to address problems in robotics. This work introduces an approach to learning hybrid systems from demonstrations, with an emphasis on extracting models that are explicitly verifiable and easily interpreted by robot operators. We fit a sequence of controllers using sequential importance sampling under a generative switching proportional controller task model. Here, we parameterise controllers using a proportional gain and a visually verifiable joint angle goal. Inference under this model is challenging, but we address this by introducing an attribution prior extracted from a neural end-to-end visuomotor control model. Given the sequence of controllers comprising a task, we simplify the trace using grammar parsing strategies, taking advantage of the sequence compositionality, before grounding the controllers by training perception networks to predict goals given images. Using this approach, we are successfully able to induce a program for a visuomotor reaching task involving loops and conditionals from a single demonstration and a neural end-to-end model. In addition, we are able to discover the program used for a tower building task. We argue that computer program-like control systems are more interpretable than alternative end-to-end learning approaches, and that hybrid systems inherently allow for better generalisation across task configurations.

I. INTRODUCTION

Recent work in end-to-end learning has resulted in significant advances in the synthesis of visuomotor robot controllers. However, many of these approaches require extensive amounts of data for training, and there are concerns regarding the reliability and verifiability of controllers obtained using these approaches. This is particularly true in industrial settings, where robot operators require interpretable systems with easily verifiable behaviours, and which can be reconfigured with relative ease when minor task changes occur.

As robotics moves beyond task-level learning and starts to address more challenging scenarios, particularly those requiring memory and conditional perception-action loops, end-to-end models are required to become more and more complex, and as a result are less interpretable and more difficult to alter for different scenarios. Addressing these challenges often requires capturing additional data and model retraining, a particularly time consuming exercise.

Hybrid systems [12] are a natural remedy to these challenges, and have a long history of application in robotics. This paper seeks to learn hybrid systems from one-shot demonstrations, inferring high-level switching logic that can be easily adjusted by a robot operator, while ensuring that

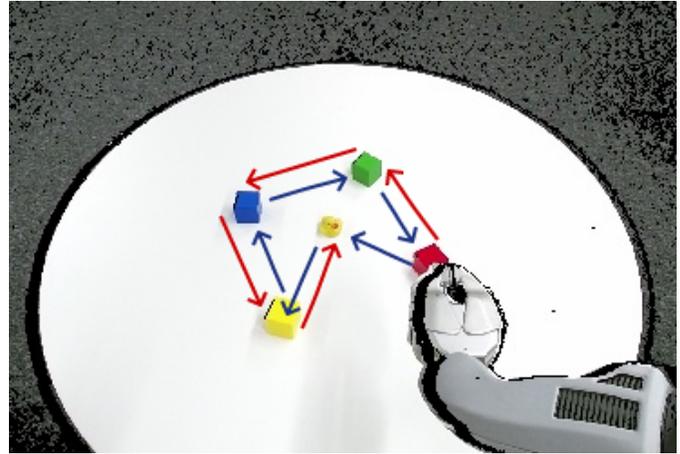


Fig. 1. We consider a continual inspection task where a PR2 robot is required to inspect four coloured blocks in a particular order, as indicated by the red arrows above. Upon reaching the rubber duck, the robot is required to reverse its inspection direction, as indicated by the blue arrows.

lower-level motion primitives correspond to controllers that are easily verified and inspected, but retaining the benefits of neural learning through perceptual grounding networks.

In order to achieve this, we propose building a hybrid system in a bottom-up fashion, using a generative switching proportional controller model to identify sub-controllers that constitute a demonstration, and inferring a high-level program that executes these controllers in an appropriate manner to reproduce the observed behaviour. Our motivation for this is to align with sequential composition theories in robotics [5], where tasks are solved by moving between sub-controllers lying within the domains of one another. Under our inference framework, each sub-controller is represented by a goal (a joint angle configuration) and an appropriate proportional gain. These sub-controllers can be grounded using a perception network, which predicts goals for a given scene and controller. Importantly, this framework allows for goals to be visually verified by a robot operator, and follows an established perception-action control loop. We use proportional controllers not only for the level of interpretability they provide, but because they allow very rich behaviours to be constructed when chained. Pose-goal-based task representations naturally

allow for the use of planning and collision avoidance schemes for safe execution in manipulation tasks. Moreover, grounded proportional control systems can be considered to be visual-servoing instances of the funnelling compositions of Burrige et al. [5], and are thus a natural mechanism with which to investigate visuomotor control.

Our primary contribution is a method for extracting a programmatic hybrid dynamical system from models trained using end-to-end learning, allowing for the flexibility and interpretability of hybrid control to be used in conjunction with end-to-end learning, and facilitating generalisation across task configurations. In line with this:

- We introduce a probabilistic generative task model that infers proportional controller motion primitives from demonstrations and by performing sensitivity analysis on end-to-end models.
- We bridge statistical learning and symbolic reasoning in order to move beyond low-level task learning, and allow for programmatic reasoning that facilitates generalisation across task configurations.
- We incorporate kinematic structure into the inference process to ensure that inferred symbols map to motion primitives and concepts that are easily verified by robot operators.

We demonstrate this method on a task (Figure 1) requiring the identification of loops, conditionals and low-level motion primitives.

II. RELATED WORK

Learning from demonstration (LfD) [1] is widely acknowledged as a particularly useful paradigm for robot programming. Significant progress has been made in LfD, moving beyond the direct replication of motions to produce more robust approaches [2] through the introduction of more general schemes for modelling motion like dynamic motion primitives [32], linear dynamical attractor systems [10], sparse online Gaussian processes [6, 13] or conditionally linear Gaussian models [7, 23] that can be used for trajectory optimisation.

More recently, trajectory optimisation approaches have been extended to incorporate end-to-end learning, demonstrating robust task level visuomotor control [24] through guided policy search. End-to-end learning has allowed for the use of domain transfer to facilitate one-shot learning [43] from human video demonstrations, and for the use of reinforcement learning to learn optimised control policies [37, 45].

Unfortunately, end-to-end learning approaches typically lack interpretability, are difficult to verify without policy distillation [3] and require significant amounts of training data [36]. In addition, many of these approaches have been criticised as only seeking to imitate and reproduce motions, with little thought to higher level conceptual learning and reasoning [19, 22]. Work on options learning [40, 18] and skill identification [28, 38] has paid greater attention to this aspect, but has arguably slowed with a recent emphasis on end-to-end learning.

End-to-end learning systems often lack flexibility, for example if a task goal is modified [20] or when systematic compositional differences between test and training data are present [26]. Hybrid systems combining the high level symbolic reasoning (long upheld as an essential ingredient of cognition) with sub-symbolic machine learning systems are a natural means of addressing these challenges [21]. For robotics, a key consideration in leveraging symbolic reasoning is the symbol grounding problem [15], which seeks to relate high level conceptual reasoning to physical phenomena or behaviours. A number of symbol grounding approaches have been proposed. For example, Dantam et al. [8] tokenise human demonstrations using a grammar based on detected object connections, and manually link these to robot controllers for an assembly task. Penkov et al. [35] use eye-tracking to identify symbol locations and facilitate instance learning and symbol grounding. Lázaro-Gredilla et al. [22] learn high level concepts (programs) by inducing linear sequences of instructions using example programs, given a known set of atomic instructions for perception and control.

Unfortunately, much work on concept learning has tended to assume that low level motion primitives are known or specified in advance [33, 22] and neglected aspects related to how best to combine a set of low-level primitives to solve a high-level task. Compositional options learning strategies like skill chaining [18] or LQR-trees [41] have sought to address this. This work moves towards addressing the disconnect between options learning, end-to-end learning and symbolic reasoning by extracting compositional low level motion primitives from an end-to-end model and synthesising a program using these.

Our work is similar to Niekum et al. [29], who apply concept learning using a Beta-Process Autoregressive Hidden Markov Model (BP-AR-HMM) to extract high-level skills from demonstrations and then use dynamic motion primitives to model the low-level motions associated with these skills. They extend this work by learning a finite state automaton for skill sequencing, grounding skills with external visual observations for greater flexibility [30, 31]. However, our work differs by extracting an explicit program from demonstrations and in that it pays specific attention to the recovery of both compositional and interpretable motion primitives from neural visuomotor controllers, allowing for its potential application as an explanatory mechanism for models trained using end-to-end learning.

Programmatic structure is a particularly useful means of capturing human-like concepts [19] that provides strong inductive bias for learning symbolic representations [35, 34, 11]. Discovering programmatic structure from observations is a challenging problem that has been studied in the domains of grammar inference [9], program synthesis [14], programming by demonstration [4] and end-user programming [25]. Learning and working with programs not only enables model checking and interpretation of black box policies [42, 33], but also results in powerful abstractions allowing knowledge transfer to novel environments.

III. SYNTHESISING INTERPRETABLE CONTROLLERS

As discussed previously, our goal is to synthesise a program comprising low-level motion primitives that are easily interpretable by a robot operator. We accomplish this using a bottom-up approach, by first inferring low-level controllers for a given demonstration using a probabilistic generative task model. These low level controllers are then grounded using controller specific perception networks, so as to facilitate generalisation across task configurations. Finally, trace level program induction is used to extract a program from the sequence of controller primitives inferred using the generative task model. Each of these components is discussed below.

We test this approach using a reaching task in the spirit of Lázaro-Gredilla et al. [22]. Here, we require our robot to continually inspect a set of 4 coloured blocks and a toy duck in a known order. However, upon reaching the duck, the robot is expected to reverse its inspection direction, as is illustrated in Figure 1. Although seemingly simple, this task comprises both loops and conditionals, in addition to low level control primitives, and so is a challenge for many existing approaches to learning from demonstration.

A. Generative task models

Taking inspiration from the work of Burrige et al. [5] on compositional funnels, we model high-level robot tasks using a switching task model comprising sub-tasks described by a set of proportional controllers acting on joint angles θ . Here, we assume that any robot task can be completed using a sequence ($j = 1 \dots J$) of proportional controllers determining robot joint velocities,

$$\mathbf{u}_\theta = K_p^j(\theta - \theta_d^j), \quad (1)$$

parameterised by joint angle goals θ_d^j and controller gains K_p^j . This approach contrasts with those used in [24] (a conditioned linear Gaussian transition model) or the Beta process autoregressive hidden Markov model [31] (a compact vector autoregressive process), in that these low level motion primitives are inherently interpretable, that is, the j^{th} sub-controller can be directly inspected by a robot operator, with goals verified for feasibility.

At any stage in a robot task demonstration, a robot may switch to a new controller, $j + 1$, or continue towards a goal defined by the current controller, j . We model this switching behaviour using a Bernoulli trial, sampling goals and gains from a prior distribution $\Phi(\theta)$ if a switch has occurred, or from Gaussian jitter if no switch has occurred. This produces the generative task model,

$$k \sim \text{Bernoulli}(p) \quad (2)$$

$$\theta_d^j(t) \sim \begin{cases} \mathcal{N}(\theta_d^j(t-1), \mathbf{Q}_\theta) & \text{if } k = 0 \\ \Phi(\theta) & \text{if } k = 1 \end{cases} \quad (3)$$

$$K_p^j(t) \sim \mathcal{N}(K_p^j(t-1), \mathbf{Q}_{kp}) \quad (4)$$

$$\mathbf{u}_\theta \sim \mathcal{N}(K_p^j(\theta - \theta_d^j), \mathbf{R}). \quad (5)$$

Here, \mathbf{Q}_θ and \mathbf{Q}_{kp} are transition uncertainty terms, while \mathbf{R} represents the control measurement noise. p denotes the

probability of switching to a new controller. Given this model and a task demonstration (a sequence of T joint angles and velocities) we infer controller parameters using sequential importance sampling, as described in Algorithm 1.

Algorithm 1 Inferring controllers using Sequential importance sampling re-sampling

```

Initialise  $N$  particles
for  $t = 0$  to  $T$  do
  for  $k = 0$  to  $pN$  do
    Sample  $\theta_d^k(t) \sim \mathcal{N}(\theta_d^k(t-1), \mathbf{Q}_\theta)$ 
    Sample  $K_p^k(t) \sim \mathcal{N}(K_p^k(t-1), \mathbf{Q}_{kp})$ 
    Evaluate  $L^k = \mathcal{N}(K_p^j(\theta - \theta_d^j), \mathbf{R})$ 
  end for
  for  $k = pN$  to  $N$  do
    Sample  $\theta_d^k(t) \sim \Phi(\theta)$ 
    Sample  $K_p^k(t) \sim \mathcal{N}(K_p^k(t-1), \mathbf{Q}_{kp})$ 
    Evaluate  $L^k = \mathcal{N}(K_p^j(\theta - \theta_d^j), \mathbf{R})$ 
  end for
  Draw  $N$  samples:  $\theta_d^k, K_p^k \sim L^k$ 
end for

```

The generative task model described above places no limits on the number of controllers required by a task, and is thus flexible enough to extend to almost any feasible manipulation task, through the sequential composition of controllers.

B. Attribution priors

Inference under the model above can be challenging, as there are numerous joint angle goal possibilities to consider, particularly for robots with large work-spaces. As a result, special attention needs to be paid to the switching prior, $\Phi(\theta)$.

Here, we propose a switching prior based on sensitivity information extracted from an end-to-end model trained to predict robot controls given robot states and images. As mentioned previously, end-to-end learning has been shown to be effective for many robot control tasks, but is often undesirable for robot operators and in industrial applications, given the lack of interpretability of these systems. This lack of interpretability extends across much of modern machine learning, and has led to the development of sensitivity analysis techniques that seek to identify which inputs contribute the most to a model's output [39, 44]. Gradient-based sensitivity maps are a particularly powerful means of extracting the image content that is salient to a deep learning model, and reliably highlight image content of interest for well trained models, even in cluttered scenes.

We make use of these techniques to construct a prior over controller goals. First, we train an end-to-end prediction model $g(\mathbf{I}, \theta)$ using 1700 velocity-image pairs, following the architecture in Figure 2. This model is unable to solve our conditional sequential reaching task as it lacks memory, but is still able to identify image regions of potential importance to a prediction, as shown in the saliency map of Figure 3. Our hypothesis is that many of these salient regions correspond to

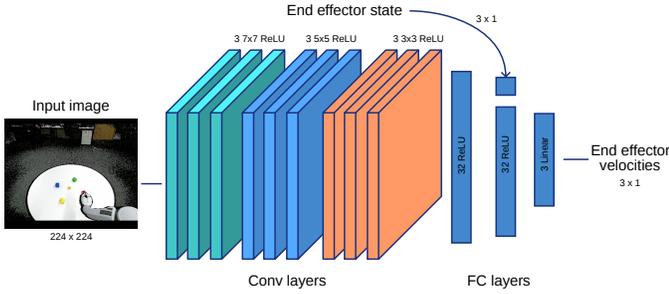


Fig. 2. A simple end-to-end model trained (Adam optimiser, learning rate of $1e-2$) to predict end effector given images and joint states is used to provide a saliency prior of positions of interest in a scene. This prior is then used to sample potential controller goals.

joint goals of importance and that these saliency maps provide a useful prior over regions of interest in a scene.

With this in mind, we sample joint angles using a saliency map and the rejection sampling described in Algorithm 2. Here, goal samples are initially drawn from the set of M joint angles in the demonstration sequence, starting from current joint state $\theta(t)$. This restricts joint angle goals to future joint angles in the demonstration. These samples are then projected into the image plane using robot forward kinematics and a camera model, and an attribution likelihood

$$L_a^k = \frac{\partial g(\mathbf{I}, \theta(t))}{\partial \mathbf{I}(\mathbf{x}_e^k)} \quad (6)$$

is evaluated at this image coordinate, \mathbf{x}_e^k . Finally, we re-sample and jitter the sampled angles given this likelihood to obtain joint angle goal samples.

Algorithm 2 Sampling from attribution priors

- Draw $k = 1 \dots N_p$ samples: $\theta_d^k \sim [\theta(t) \dots \theta(t + M)]$
 - Project samples into image plane: $\mathbf{x}_e^k = \mathbf{K} \text{FK}(\theta_d^k)$
 - Evaluate attribution likelihood: $L_a^k = \frac{\partial g(\mathbf{I}, \theta(t))}{\partial \mathbf{I}(\mathbf{x}_e^k)}$
 - Draw $k = 1 \dots N_p$ samples: $\theta_d^k \sim L_a^k$
-

Figure 3 shows projected joint angle goal samples drawn using the sampling process above, along with controller roll-outs towards these. It is clear that the proposed sampling process is able draw upon the visual information captured by the end-to-end prediction model to provide good priors over joint angle goals and potential destinations. It should be noted that the saliency prior and proportional goal configuration inductively bias the inference towards goals that correspond with specific objects present in the scene. As a result, saliency-based sampling is unlikely to be useful in situations where image content causes repulsion, for example in a collision avoidance setting, and may result in multiple unused particles here.

C. Symbol extraction

Inference using sequential importance sampling and the attribution prior produces a distribution over possible controllers

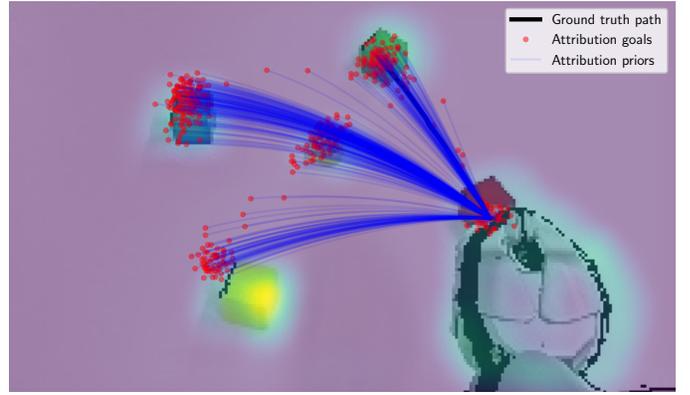


Fig. 3. Controller roll-outs to joint states drawn from attribution priors reduce the space of possible joint angle goals. Goal projections are marked in red, with roll-outs in blue. The saliency map used for sampling is extracted by performing sensitivity analysis on the end-to-end prediction model depicted in Figure 2.

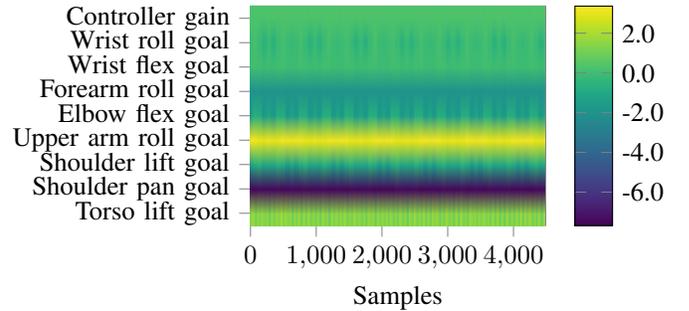


Fig. 4. A distribution over joint angle goals and controller gains is obtained at each time step in the sequence. The figure shows the maximum-a-posteriori estimates for each controller parameter (y-axis) at each time step. The colour bar indicates the inferred parameter values. (Figure best viewed electronically).

for each time step in a demonstration sequence (Figure 4). Importantly, the periodicity in inferred controller parameters is clearly visible. However, our goal is to build a high level sequence of controllers that describe the demonstrated task at a symbolic level. We isolate this sequence of controllers constituting a demonstration by making use of the effective particle size [17],

$$N_{eff} = \frac{1}{\sum_{k=0}^N (L^k)^2}, \quad (7)$$

at each time step of the inference process. The intuition behind this is as follows: When a demonstration requires a controller switch towards a new goal, a significant portion of the particles will have low probability mass and therefore a low effective particle size. In contrast, the effective particle size will be greatest when only a single controller is required and a clear joint goal and gain has been established.

This is illustrated for our running example in Figure 5, which shows the effective particle size for each sample in

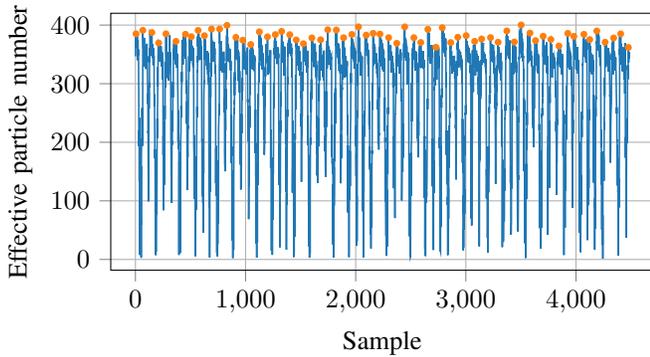


Fig. 5. The effective particle size is used to identify the chain of proportional controllers used in a demonstration sequence.

the demonstration sequence. A simple peak detector is able to identify points at which a clear joint goal and gain are present, which typically occurs mid-way through a motion. We construct a sequence of controllers by selecting the maximum-a-posteriori controller at each of these points in the demonstration.

This sequence of controller primitives is then clustered (K -means clustering using an elbow criterion to select the number of clusters) in order to produce a symbolic behaviour trace of the demonstration, as shown at the top of Figure 6. Here, each controller is represented by a distinct colour and the sequence of goals constituting a task is clearly visible.

Visual inspection of the image projection of the joint angle goals associated with the identified controller primitives clearly shows that we have successfully extracted the key primitives making up our demonstration sequence, along with the order in which these were visited. Importantly, the task model structure used for inference means that these goals are verifiable and easily interpretable by a robot programmer.

1) *Comparison with linear Gaussian systems:* We contrast this with the linear motion primitives learned using the conditional Gaussian mixture model approach [24], which lack a clear interpretation. These take the form

$$\theta(t) = \mathbf{A}^j \theta(t-1) + \mathbf{B}^j \mathbf{u}_\theta, \quad (8)$$

and require that a linear quadratic regulator be used to find controls given the current state.

This formulation is not particularly compact, as it requires multiple models to describe the motion towards a particular goal. For example, when this approach was used to model motions in our running example, **15** models were required to describe trajectories, using the Bayesian information criterion for model selection. In contrast, the proposed task model produces **5** clearly interpretable models. Fewer models are required for the latter because the proportional controller formulation is invariant to the direction from which a goal is approached.

2) *Evaluating the effects of attribution:* We compared inference using visual attribution priors to a particle filter under

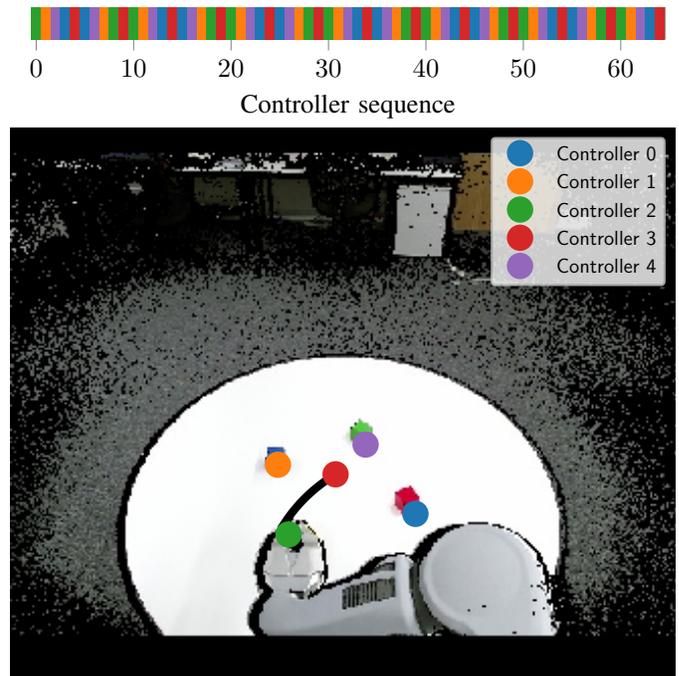


Fig. 6. Maximum-a-posteriori controllers are clustered to produce a symbolic trace of controller primitives comprising a demonstration. The black line in the figure illustrates a roll out from the current robot state using Controller 3.

the proposed generative model, but with the sensitivity-based rejection sampling step replaced by a direct draw from the M future demonstration states directly. As shown in Figure 7, inference using this approach failed to correctly identify suitable proportional controller goals. This can be attributed to a poor effective sample size (see top of Figure 7 and Table I), which makes it challenging to identify controller switching and infer controllers.

TABLE I
INFERENCE QUALITY

Statistics: N_{eff}	Mean	Max	Min	IQR
Attribution ($N_p = 50$)	32.12	48.73	3.80	12.45
Baseline ($N_p = 50$)	28.12	42.96	1.00	7.60

Importantly, the improved inference quality introduced by the attribution prior results in a greater interquartile range in effective particle number, which allows for the improved detection of controller switches. These experiments clearly show that the proposed rejection sampling approach extracts visual information from the end-to-end model to inform the task identification process.

D. Program induction

The symbolic trace identified above can be simplified into a programmatic representation that can easily be inspected and modified by a robot operator, thereby providing greater levels of flexibility. As discussed earlier, a number of program induction techniques have been developed previously, to deal

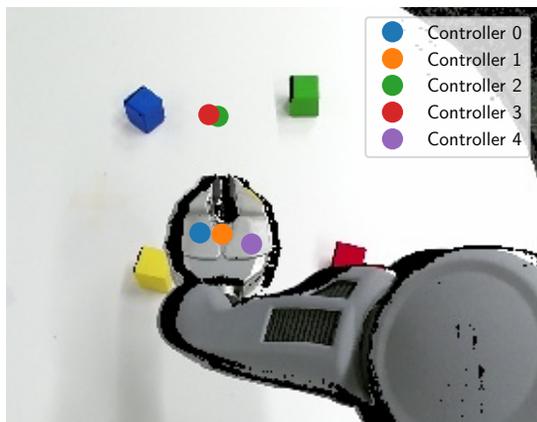
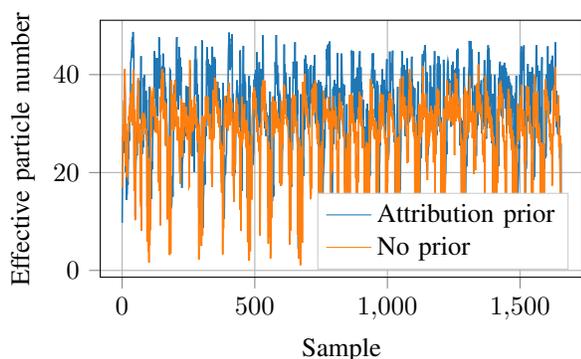


Fig. 7. The effective particle size is significantly improved by the inclusion of attribution information, indicating improved inference. Inference without this information fails to extract sensible controller goals.

with a variety of grammars or program primitives. These approaches can be expensive and often introduce errors if they fail to reproduce an instruction trace perfectly. In addition, program induction is complicated by the fact that any number of programs could produce a given instruction trace, and the choice of whether one program is more interpretable or elegant than another is highly subjective. The latter is a particular challenge for program induction, as the solution space for programs is extremely large.

In light of this, we propose a string parsing approach that deals with three key aspects (Figure 8) that we consider likely to arise from the compositional structure of the trace inferred using the proposed generative task model. Importantly, these correspond to commonly observed mission requirements in robotics [27], specifically those required for patrolling and sequential visits. This trace is formed of a sequence of proportional controllers, driving a robot through a series of goal states to solve some task. These motion primitives mean that we are likely to observe the following aspects in general tasks.

The first is a looping structure, where a cycle of goal states is visited a number of times. This structure can be identified by finding repeated consecutive sub-sequences in a trace. Since our goal is to produce shorter program structures, we condense traces by replacing repeated sub-sequences in a trace with loops, using an exhaustive search to identify repeated sub-

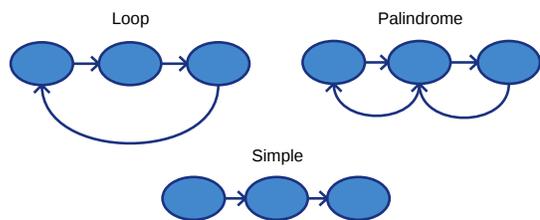


Fig. 8. Three key sub-structures are identified to extract a program from symbolic traces. Loops are a sequence of controllers executed cyclically, palindromes are a sequence of controllers executed in a path reversal process, and simple controller sequences have a clear start and end goal.

sequences, and replacing those sub-sequences that repeat the most with looped instructions. We repeat this process for code segments both outside and inside the replaced loops, until no repeated sub-sequences remain.

The second structure we consider is a palindromic or retreating sequence. Here, a task may require a robot to pass through a set of goal states, and then return by reversing through the same set of goal states. We identify this structure by exhaustively searching through code segments after loop detection, and finding all palindromes in the instruction trace of a code segment. We replace palindromic sub-sequences (only those longer than 6 instructions to avoid unnecessary complication) in code by a controller list and a counter that increments until the palindrome centre, before decrementing until the palindrome ends. The palindrome centre state can be considered a goal condition that needs to be met before a trace continues.

Listing 1. Simplified program extracted from trace.

```
# Looping sequence
c_list = [2,1,4,0,3]
for j in range(6):

    # Palindromic sequence
    count = 0
    for k in range(len(c_list)*2-1):
        execute(c_list[count])
        if (k >= len(c_list)-1):
            count = count-1
        else:
            count = count+1

    # Simple sequence
    execute(3)

# Simple sequence
execute(2)
execute(1)
execute(4)
execute(0)
execute(3)
```

Once these structures have been removed, the only remaining code blocks in the program will contain simple sequences with clear starting and terminating goal states. Program listing 1 illustrates this for our running example. It is clear that the synthesised program captures the demonstrated behaviour compactly, and could easily be modified by a robot operator, say to extend the number of loop repetitions or to insert a new

goal if required.

This program is both conceptually similar and of equivalent complexity to the hand-coded example originally used to generate the demonstration (see Program listing 2), which looped indefinitely, executing a simple transition model on each iteration. Differences arise from the finite demonstration length used to infer symbolic traces and the constraints of the proposed triple structure program grammar, which is restricted to loops, palindromes and simple sequences in order to ensure generalisability across tasks.

Listing 2. Original program used to generate demonstration.

```
state_explanation = [3,2,1,4,0,3]

class sm():
    def __init__(self):
        self.step = 1
        self.state = 0

    def transition(self, state):
        self.state = self.state + self.step
        if (self.state == 5):
            self.step = -1
        if (self.state == 0):
            self.step = 1
        return self.state

seq = sm()
state = 0
while (1):
    state = seq.transition(state)
    mc.move(state_explanation[state])
```

E. Symbol grounding

The previous section showed how we can infer low-level controller primitives and construct a corresponding program describing a robot demonstration using these. However, symbol grounding is required if the inferred symbols are to generalise to different configurations of the demonstrated task.

For the purpose of illustration, we ground our symbols by means of symbol-conditioned perception networks (4 convolutional and two dense fully connected layers), $g_c(\mathbf{I}, \phi)$, trained to predict controller goal locations for a given scene, using a mean square error loss,

$$J = \mathbb{E} [\|g_c(\mathbf{I}, \phi) - \mathbf{y}_{\theta}(c)\|]. \quad (9)$$

Here $\mathbf{y}_{\theta}(c)$ denotes the goal angles extracted from the demonstration sequence for controller c , and ϕ represents network parameters. We leverage the fact that controller goals correspond to visual information for data augmentation, cropping and re-positioning image patches around joint angle goals at projected image locations of joint angles sampled from the robot workspace (Figure 9). This augmentation scheme is inductively biased towards object-based goal prediction, implicitly making the assumption that goals are associated with content in image patches around these. Alternative grounding networks may be required for other situations and for better generalisation to scene variation or multi-camera settings, but this is beyond the scope of this work. Goal prediction via more advanced deep neural networks trained using supervised

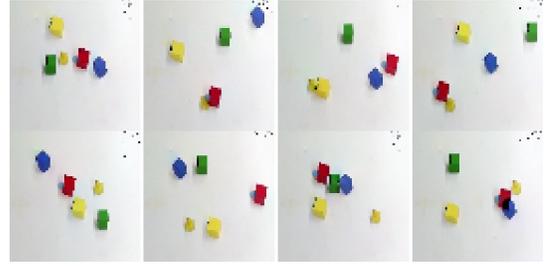


Fig. 9. Grounding networks are trained to predict controller goals (joint angle states) using 3200 synthetic images generated by cropping patches around identified controller goal states and pasting these in varying configurations corresponding to projected joint angle states.

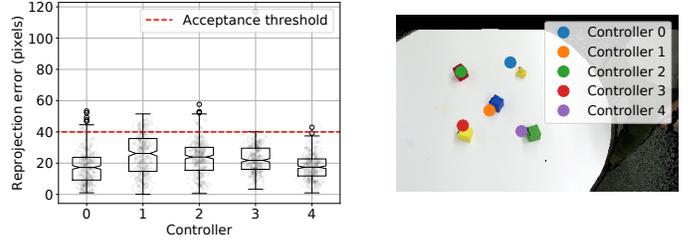


Fig. 10. Symbol grounding allows the synthesis of the high level program in different contexts, thereby exhibiting high levels of generalisation. Boxplots of the error between projected joint angle goals and manually labelled positions are shown above for 400 arbitrarily arranged test images. This allows generalisation to previously unseen object configurations.

learning has recently shown excellent performance in real-time drone racing [16], and this performance is likely to extend to more complex tasks.

F. Program synthesis

Given the inferred program and controller primitives, program synthesis becomes trivial. Here, we simply follow the high level program, and execute the required controllers as requested. Controller execution is a simple matter of selecting a joint angle goal using the symbol grounding network, and using a proportional control law with the inferred gains.

This approach allows for generalisation to different task configurations, for example the case in Figure 10, where we have re-ordered the blocks to be visited and the robot is still able to execute the underlying program in the original order. Modifying the program, say to visit blocks in a different order, is easily accomplished using the hybrid system representation. This level of generalisation cannot be easily obtained with end-to-end models like those used for guided policy search [24], and the original model (Figure 2) fails in this configuration. Although approaches like this are often able to generalise well to variations within a task, the abstraction provided by learning a grounded program inherently introduces flexibility with regard to alternative task configurations.

It should be noted that the guided policy search model is unable to replicate the task in our running example as it lacks a recurrent memory component, and is thus unable to model repeated palindromic motions, which require motion conditioned on a longer history of states.

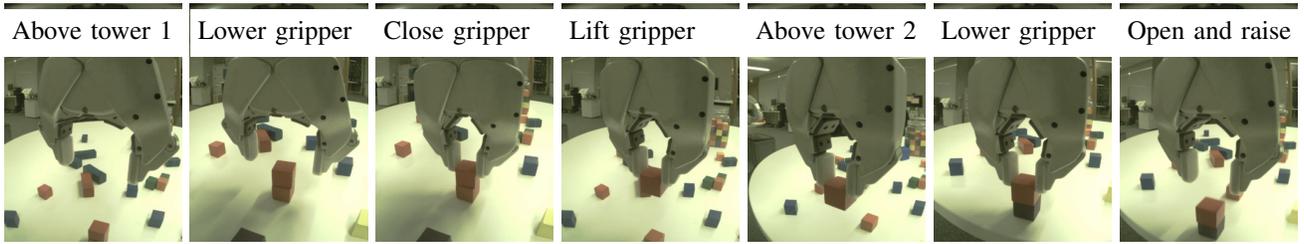


Fig. 11. A tower assembly task using the motion primitives pictured in the figure is used to investigate the proposed task decomposition framework.

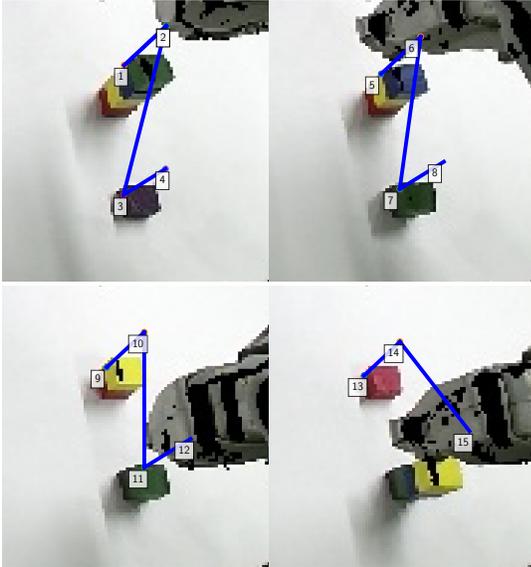


Fig. 12. The tower assembly task is decomposed into 4 key controllers (numbered in the figure) using the generative task model. Interestingly, this decomposition is simpler than our original hand-coded rules.

IV. TOWER BUILDING EXAMPLE

Thus far, the proposed framework has been demonstrated on a relatively simple reaching task. This section illustrates our approach in a more complex setting, where we inferred a program and controller primitives to explain a tower disassembly and reassembly task. Here, we programmed a PR2 robot to move blocks from one tower to another, relying on hard-coded positions and offsets, following the general rubric indicated in Figure 11: move above block, lower gripper, close gripper, lift gripper, move above goal, lower gripper, open and raise gripper.

We recorded a single demonstration sequence, and used the generative task model to identify sub-controllers with an end-to-end model trained using supervised learning. Figure 12 shows the segmented controller goal projections for each sub-task identified. Interestingly, the inferred controllers combine aspects that we had decoupled in code, decomposing each stage of the process above into four components: position gripper, close and raise gripper, position block, open and raise gripper.

Clustering the identified controllers results in 15 unique controllers, connected in a simple sequence. A key observation

is that the proposed programmatic abstraction means that similar concepts, for example placing a block on a tower, are considered different tasks, as they occur in different positions. This highlights the importance of choosing motion primitives that map to likely concepts in a given scenario. Compositional goal-like motion primitives are inductively biased towards the identification of concepts related to inspection or the manipulation of objects, but program abstraction techniques [14] could be applied to identify higher order concepts if necessary.

V. CONCLUSION

This paper has introduced a method for extracting a program from demonstrations of robot manipulation tasks. We proposed a probabilistic generative model for learning from one-shot demonstrations that decomposes tasks into a sequence of proportional controllers in joint angle space, with corresponding joint angle goals. Inference under this model draws on prior information extracted from an end-to-end visuomotor controller, leveraging sensitivity analysis for goal identification. Program induction was used to reduce the inferred trace into a form that could be easily modified by a robot operator, and perceptual symbol grounding networks trained to allow task generalisation. We argued that this programmatic compositional formulation is inherently interpretable, as joint goals could be verified by inspection, and that the incorporation of structure (kinematic models, knowledge of controller primitives) into the inference process both simplifies and ensures interpretability of learned behaviours.

Importantly, the proposed approach can be used as an explanation mechanism for reinforcement learning agents or existing end-to-end systems. Synthesising a program by observing an agent can provide valuable insights into its learned behaviours. This also allows for the transfer of learned behaviours when tasks or goals are modified, thereby introducing greater flexibility and improving generalisation across task configurations. Finally, the proposed approach eases system verification, which currently poses a significant barrier to the deployment of end-to-end learning systems.

ACKNOWLEDGEMENTS

This work was supported by funding from the ORCA Hub EPSRC (EP/R026173/1, 2017-2021) and an Alan Turing Institute funded project, Safe AI for Surgical Assistance.

REFERENCES

- [1] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469 – 483, 2009. ISSN 0921-8890.
- [2] Christopher G Atkeson and Stefan Schaal. Robot learning from demonstration. In *ICML*, volume 97, pages 12–20. Citeseer, 1997.
- [3] Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. Verifiable Reinforcement Learning via Policy Extraction. *arXiv preprint arXiv:1805.08328*, 2018.
- [4] Aude Billard, Sylvain Calinon, Ruediger Dillmann, and Stefan Schaal. Robot programming by demonstration. In *Springer handbook of robotics*, pages 1371–1394. Springer, 2008.
- [5] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek. Sequential Composition of Dynamically Dexterous Robot Behaviors. *The International Journal of Robotics Research*, 18(6):534–555, 1999.
- [6] J. Butterfield, S. Osentoski, G. Jay, and O. C. Jenkins. Learning from demonstration using a multi-valued function regressor for time-series data. In *2010 10th IEEE-RAS International Conference on Humanoid Robots*, pages 328–333, Dec 2010.
- [7] Silvia Chiappa and Jan R. Peters. Movement extraction by detecting dynamics switches and repetitions. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 388–396. Curran Associates, Inc., 2010.
- [8] N. Dantam, I. Essa, and M. Stilman. Linguistic transfer of human assembly tasks to robots. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 237–242, Oct 2012.
- [9] Colin De la Higuera. *Grammatical inference: learning automata and grammars*. Cambridge University Press, 2010.
- [10] K. R. Dixon and P. K. Khosla. Trajectory representation using sequenced linear dynamical systems. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 4, pages 3925–3930 Vol.4, April 2004.
- [11] Alexander L. Gaunt, Marc Brockschmidt, Nate Kushman, and Daniel Tarlow. Differentiable Programs with Neural Libraries. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1213–1222, International Convention Centre, Sydney, Australia, August 2017. PMLR.
- [12] Rafal Goebel, Ricardo G Sanfelice, and Andrew R Teel. Hybrid dynamical systems. *IEEE Control Systems*, 29(2):28–93, 2009.
- [13] Daniel H Grollman and Odest Chadwicke Jenkins. Sparse incremental learning for interactive robot control policy estimation. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 3315–3320. IEEE, 2008.
- [14] Sumit Gulwani, Oleksandr Polozov, Rishabh Singh, et al. Program synthesis. *Foundations and Trends® in Programming Languages*, 4(1-2):1–119, 2017.
- [15] Stevan Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1):335 – 346, 1990.
- [16] Elia Kaufmann, Antonio Loquercio, Rene Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. Deep Drone Racing: Learning Agile Flight in Dynamic Environments. In *Proceedings of the 2nd Annual Conference on Robot Learning*, 13–15 Nov 2018.
- [17] Augustine Kong, Jun S Liu, and Wing Hung Wong. Sequential imputations and Bayesian missing data problems. *Journal of the American statistical association*, 89(425):278–288, 1994.
- [18] George Konidaris and Andrew G Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. In *Advances in neural information processing systems*, pages 1015–1023, 2009.
- [19] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.
- [20] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40, 2017.
- [21] Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Ingredients of intelligence: From classic debates to an engineering roadmap. *Behavioral and Brain Sciences*, 40:e281, 2017.
- [22] Miguel Lázaro-Gredilla, Dianhuan Lin, J. Swaroop Guntupalli, and Dileep George. Beyond imitation: Zero-shot task transfer on robots by learning concepts as cognitive programs. *Science Robotics*, 4(26), 2019.
- [23] Sergey Levine and Pieter Abbeel. Learning neural network policies with guided policy search under unknown dynamics. In *Advances in Neural Information Processing Systems*, pages 1071–1079, 2014.
- [24] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.
- [25] Henry Lieberman, Fabio Paternò, Markus Klann, and Volker Wulf. *End-User Development: An Emerging Paradigm*. Springer Netherlands, 2006.
- [26] Gary Marcus. *The algebraic mind: Integrating connectionism and cognitive science*. MIT Press, 2003.
- [27] Claudio Menghi, Christos Tsigkanos, Patrizio Pelliccione, Carlo Ghezzi, and Thorsten Berger. Specification Patterns for Robotic Missions. *arXiv preprint arXiv:1901.02077*, 2019.
- [28] Scott Niekum and Andrew G. Barto. Clustering via Dirichlet Process Mixture Models for Portable Skill Discovery. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett,

- F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1818–1826. Curran Associates, Inc., 2011.
- [29] Scott Niekum, S. Osentoski, George Konidaris, and Andrew G. Barto. Learning and generalization of complex tasks from unstructured demonstrations. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5239–5246, Oct 2012.
- [30] Scott Niekum, Sachin Chitta, Andrew Barto, Bhaskara Marthi, and Sarah Osentoski. Incremental Semantically Grounded Learning from Demonstration. In *Proceedings of Robotics: Science and Systems*, Berlin, Germany, June 2013.
- [31] Scott Niekum, Sarah Osentoski, George Konidaris, Sachin Chitta, Bhaskara Marthi, and Andrew G Barto. Learning grounded finite-state representations from unstructured demonstrations. *The International Journal of Robotics Research*, 34(2):131–157, 2015.
- [32] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 763–768. IEEE, 2009.
- [33] Svetlin Penkov and Subramanian Ramamoorthy. Explaining Transition Systems through Program Induction. *arXiv preprint arXiv:1705.08320*, 2017.
- [34] Svetlin Penkov and Subramanian Ramamoorthy. Learning Programmatically Structured Representations with Perceptor Gradients. In *International Conference on Learning Representations*, 2019.
- [35] Svetlin Penkov, Alejandro Bordallo, and Subramanian Ramamoorthy. Physical symbol grounding and instance learning through demonstration and eye tracking. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5921–5928, May 2017.
- [36] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 3406–3413. IEEE, 2016.
- [37] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018.
- [38] Pravesh Ranchod, Benjamin Rosman, and George Konidaris. Nonparametric bayesian reward segmentation for skill discovery using inverse reinforcement learning. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 471–477. IEEE, 2015.
- [39] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626. IEEE, 2017.
- [40] Richard S Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.
- [41] R. Tedrake. LQR-trees: Feedback motion planning on sparse randomized trees. In *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009.
- [42] Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. Programmatically Interpretable Reinforcement Learning. *arXiv preprint arXiv:1804.02477*, 2018.
- [43] Tianhe Yu, Chelsea Finn, Sudeep Dasari, Annie Xie, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-Shot Imitation from Observing Humans via Domain-Adaptive Meta-Learning. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018.
- [44] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.
- [45] Yuke Zhu, Ziyu Wang, Josh Merel, Andrei Rusu, Tom Erez, Serkan Cabi, Saran Tunyasuvunakool, János Kramár, Raia Hadsell, Nando de Freitas, and Nicolas Heess. Reinforcement and Imitation Learning for Diverse Visuomotor Skills. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018.