Learning Manipulation Skills Via Hierarchical Spatial Attention

Marcus Gualtieri and Robert Platt

***Abstract*—Learning generalizable skills in robotic manipulation has long been challenging due to real-world sized observation and action spaces. One method for addressing this problem is attention focus – the robot learns where to attend its sensors and irrelevant details are ignored. However, these methods have largely not caught on due to the difficulty of learning a good attention policy and the added partial observability induced by a narrowed window of focus. This article addresses the first issue by constraining gazes to a spatial hierarchy. For the second issue, we identify a case where the partial observability induced by attention does not prevent Q-learning from finding an optimal policy. We conclude with real-robot experiments on challenging pick-place tasks demonstrating the applicability of the approach.**

I. Introduction

Learning robotic manipulation has remained an active and challenging research area. This is because the real-world environments in which robots exist are large, dynamic, and complex. Partial observability – where the robot does not at once perceive the entire environment – is common and requires reasoning over past perceptions. Additionally, the ability to generalize to new situations is critical because, in the real world, new objects can appear in different places unexpectedly.

The particular problem addressed in this paper is the large space of possible robot observations and actions – how the robot processes its past and current perceptions to make high-dimensional decisions. Visual attention has long been suggested as a solution to this problem [1]. Focused perceptions can ignore irrelevant details, and generalization is improved by the elimination of the many irrelevant combinations of object arrangements [1]. Additionally, as we later show, attention can result in a substantial reduction to the number of actions that need considered. Indeed, when selecting position, the number of action choices can become logarithmic rather than linear in the volume of the robot's workspace. In spite of these benefits, visual attention has largely not caught on due to (a) the additional burden of learning where to attend and (b) additional partial observability caused by the narrowed focus.

We address the first challenge – efficiently learning where to attend – by constraining the system to a spatial hierarchy of attention. On a high level this means the robot must first see a large part of the scene in low detail, select a position within that observation, and see the next observation in more detail at the position previously selected, and so on for a fixed number of gazes. We address the second challenge – partial observability induced by the narrowed focus – by identifying attention with a type of state-abstraction which preserves the ability to learn optimal policies with efficient reinforcement learning (RL) algorithms.

This article extends our prior work [2], wherein we introduced the hierarchical spatial attention (HSA) approach and demonstrated it on 3 challenging, 6-DoF, pick-place tasks. New additions include (a) faster training and inference times, (b) more ablation studies and comparisons to related work, (c) better understanding of when an optimal policy can be learned when using this approach, (d) longer time horizons, and (e) improved real-robot experimental results.

The rest of the paper is organized as follows. First is related work (Section II). Next, the general manipulation problem is described and the visual

Khoury College of Computer Sciences, Northeastern University, Boston, MA, 02115 USA e-mail: mgualti@ccs.neu.edu.

attention aspect is added (Sections III and IV-A). After that, the HSA constraints are added, and this approach is viewed as a generalization of earlier approaches (Section IV-B to IV-E). The bulk of the paper includes analysis and comparisons in 4 domains of increasing complexity (Section V). Real robot experiments are described close to the end (Sections V-C and V-D). Finally, we conclude with what we learned and future directions (Section VI).

II. Related Work

This work is most related to robotic manipulation, reinforcement learning, and attention models. It is extends our prior research on 6-DoF pick-place [2] and primarily builds on DQN [3] and Deictic Image Mapping [4].

A. Learning Robotic Manipulation

Traditional approaches to robotic manipulation consider known objects – a model of every object to be manipulated is provided in advance [5], [6], [7]. While these systems can be quite robust in controlled environments, they encounter failures when the shapes of the objects differ from expected. Recent work has demonstrated grasping of novel objects by employing techniques intended to address the problem of generalization in machine learning [8], [9], [10], [11], [12], [13], [14], [15], [16].

There have been attempts to extend novel object grasping to more complex tasks such as pick-place. However, these have assumed either fixed grasp choices [17] or fixed place choices [18]. The objective of the present work is to generalize these attempts – a single system that can find 6-DoF grasp and place poses.

Other research considers grasping and pushing novel objects to a target location [19]. Their approach is quite different: a predictive model of the environment is learned and used for planning, whereas we aim to learn a policy directly. Other work has considered the problem of domain transfer [20] and sparse rewards in RL [21]. We view these as complimentary ideas that could be combined with our approach for an improvement.

B. Reinforcement Learning

Like several others, we apply RL techniques to the problem of robotic manipulation (see above-mentioned [10], [13], [15], [18], [21] and survey [22]). RL is appealing for robotic control for several reasons. First, several algorithms (e.g., [23], [24]) do not require a complete model of the environment. This is of particular relevance to robotics, where the environment is dynamic and difficult to describe exactly. Additionally, observations are often encoded as camera or depth sensor images. Deep Q-Networks (DQN) demonstrated an agent learning difficult tasks (Atari games) where observations were image sequences and actions were discrete [3]. An alternative to DQN that can handle continuous action spaces are actor-critic methods like DDPG [25]. Finally, RL – which has its roots in optimal control – provides tools for the analysis of learning optimal behavior (e.g. [26], [27], [28]), which we refer to in Section V-A.

C. Attention Models

Our approach is inspired by models of visual attention. Following the early work of Whitehead and Ballard [1], we distinguish overt actions (which directly affect change to the environment) from perceptual actions (which retrieve information). Similar to their agent model, our abstract robot has a virtual sensor which can be used to focus attention on task-relevant parts of the scene. The present work updates their methodology to address more realistic problems, and we extend their analysis by describing a situation where an optimal policy can be learned even in the presence of "perceptual aliasing" (i.e. partial observability).

Attention mechanisms have also been used with artificial neural networks to identify an object of interest in a 2D image [29], [30], [31], [32]. Our situation is more complex in that we identify 6-DoF poses of the robot's hand. Improved grasp performance has been observed by active control of the robot's sensor [33], [34]. These methods attempt to identify the best sensor placement for grasp success. In contrast, our robot learns to control a virtual sensor for the purpose of reducing the complexity of action selection and learning.

Work contemporary with ours also considered attention for controlling high-dimensional manipulators [35]. Important differences from our approach include the use of policy gradient instead of value-based methods, sensing 2D depth instead of 3D point clouds, and learned instead of fixed attention parameters.

III. Problem Statement

The problem considered herein is learning to control a move-effect system (Fig. 1, cf. [4]):

Definition 1 (Move-Effect System)**.** *A* move-effect system *is a discrete time system consisting of a robot, equipped with a depth sensor and end effector (e.e.), and rigid objects of various shapes and configurations. The robot perceives a history of point clouds $C_1, \dots, C_k$, where $C_i \in \mathbb{R}^{n_c \times 3}$ is acquired by the depth sensor; an e.e. status, $h \in \{1, \dots, n_h\}$; and a reward $r \in \mathbb{R}$. The robot's action is $move\text{-}effect(T_{ee}, o)$, where $T_{ee} \in W$ is the pose of the e.e., $W \subseteq SE(3)$ is the robot's workspace, and $o \in \{1, \dots, n_o\}$ is a preprogrammed controller for the e.e. For each stage $t = 1, \dots, t_{max}$, the robot receives a new perception and takes an action.*

The reward is usually instrumented by the system engineer to indicate progress toward completion of some desired task. The robot initially has no knowledge of the system's state transition dynamics. The objective is, by experiencing a sequence of episodes, for the robot to learn a policy – a mapping from observations to actions – which maximizes the expected sum of per-episode rewards.

For example, suppose the e.e. is a 2-fingered gripper, $o \in \{open, close\}$, $h \in \{empty, holding\}$, the objects are bottles and coasters, and the task is to place all the bottles on the coasters. The reward could be 1 for placing a bottle on a coaster, -1 for removing a placed bottle, and 0 otherwise.

IV. Approach

Our approach has two parts. The first part is to reformulate the problem as a Markov decision process (MDP) with abstract states and actions (Section IV-A). With this reformulation, the resulting state representation is substantially reduced, and it becomes possible for the robot to learn to restrict attention to task-relevant parts of the scene. The second part is to add constraints to the actions so that e.e. pose is decided sequentially (Section IV-B). After these improvements, the problem is then amenable to solution via standard RL algorithms like DQN.

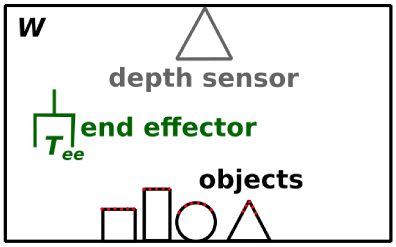

Fig. 1: The move-effect system. The robot has an e.e. which can be moved to pose T_{ee} to perform operation o.

A. Sense-Move-Effect MDP

The sense-move-effect system adds a controllable, virtual sensor which perceives a portion of the point cloud from a parameterizable perspective (Fig. 2).

Definition 2 (Sense-Move-Effect System)**.** *A* sense-move-effect system *is a move-effect system where the robot's actions are augmented with $sense(T_s, z)$ (where $T_s \in W$ and $z \in \mathbb{R}^3_{>0}$) and the point cloud observations $C_1, \dots, C_k$ are replaced with a history of k images, $I_1, \dots, I_k$ (where $I \in \mathbb{R}^{n_{ch} \times n_x \times n_y}$). The sense action has the effect of adding $I = Proj(Crop(Trans(T_s, C_k), z))$ to the history.*[1]

The *sense* action makes it possible for the robot to get either a compact overview of the scene or to

[1] $Proj : \mathbb{R}^{n_c \times 3} \rightarrow \mathbb{R}^{n_{ch} \times n_x \times n_y}$ is n_{ch} orthographic projections of points onto n_{ch}, $n_x \times n_y$ images. Each image plane is positioned at the origin with a different orientation. Image values are the point to plane distance, ambiguities resolved with the nearest distance. $Crop : \mathbb{R}^{n_c \times 3} \rightarrow \mathbb{R}^{n_{c'} \times 3}$ returns the $n_{c'} \leq n_c$ points of C which lie inside a rectangular volume situated at the origin with length, width, height z. $Trans(T_s, C)$ expresses C (initially expressed w.rt. the world frame) w.r.t. T_s.

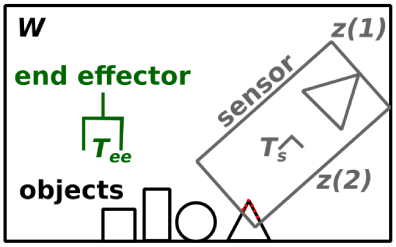

Fig. 2: The sense-move-effect system adds a virtual, mobile sensor which observes points in a rectangular volume at pose T_s with size z.

attend to a small part of the scene in detail. Since the resolution of the images is fixed, large values of z correspond to seeing more objects in less detail, and small values of z correspond to seeing less objects in more detail.

The robot's memory need not include the last k images – it can include any previous k images selected according to a predetermined strategy. Because the environment only changes after *move-effect* actions, we keep the latest image, I_k, and the last $k-1$ images that appeared just before *move-effect* actions. Fig. 3 shows an example in the bottles on coasters domain.

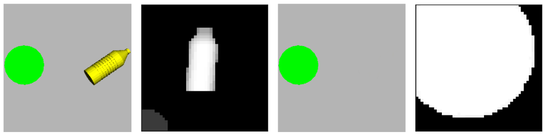

Fig. 3: Scene and observed images for $k = 2$ and $n_{ch} = 1$. **Left**. Scene's initial appearance. **Left center**. *sense* image (large z) just before $move\text{-}effect(T_{ee}, close)$. **Right center**. Scene's current appearance. **Right**. Current *sense* image, focused on the coaster (small z).

In order to apply standard RL algorithms to the problem of learning to control a sense-move-effect system, we define the sense-move-effect MDP.

Definition 3 (Sense-Move-Effect MDP)**.** *Given a sense-move-effect system, a reward function, and transition dynamics, a* sense-move-effect MDP *is a finite horizon MDP where states are sense-move-effect system observations and actions are sense-move-effect system actions.*

The reward function and transition details are task and domain specific, respectively, examples of which are given in Section V.

B. Hierarchical Spatial Attention

The observation is now similar to that of DQN – a short history of images plus the e.e. status – and can be used by a Q-network to approximate Q-values. However, the action space remains large due to the 6-DoF choice for T_s or T_{ee} and the 3-DoF choice for z. Additionally, it may take a long time for the robot to learn which *sense* actions result in useful observations. To remedy both issues, we design constraints to the sense-move-effect actions.

Definition 4 (Hierarchical Spatial Attention)**.** *Given a sense-move-effect system, $L \in \mathbb{N}_{>0}$, $T_s^1 \in W$, and the list of pairs $[(z_1, d_1), \ldots, (z_L, d_L)]$, (where $z_i \in \mathbb{R}^3_{>0}$ and $d_i \in \mathbb{R}^6$),* hierarchical spatial attention (HSA) *constrains the robot to take L $sense(T_s, z)$ actions, with $z = z_i$ for $i = 1, \ldots, L$, prior to each $move\text{-}effect$ action. Furthermore, the first sensor pose in this sequence must be T_s^1; the sensor poses T_s^{i+1}, for $i = 1, \ldots, L-1$, must be offset no more than d_i from T_s^i; and e.e. pose T_{ee} must be offset no more than d_L of T_s^L.*[2]

The process is thus divided into t_{max} *overt stages*, where, for each stage, L *sense* actions are followed by 1 *move-effect* action (Fig. 4). The constraints should be set such that the observation size z_i and offset d_i decrease as i increases, so the point cloud under observation decreases in size, and the volume within which the e.e. pose can be selected is also decreasing. These constraints are called hierarchical spatial attention because the robot is forced to learn to attend to a small part of the scene (e.g., Fig. 5).

To see how HSA can improve action sample efficiency, consider the problem of selecting position in a 3D volume. Let α be the largest volume allowed per sample. With naive sampling, the

[2] Concretely, $d_i = [x, y, z, \theta, \phi, \rho]$ indicates a position offset of $\pm x/2$, $\pm y/2$, and $\pm z/2$ and a rotation offset of $\pm\theta/2$, $\pm\phi/2$, and $\pm\rho/2$.

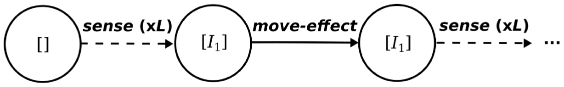

Fig. 4: Initially, the state is empty. Then, L sense actions are taken, at each point the latest image is state. After this, the robot takes 1 *move-effect* action. Then, the process repeats, but with the last image before *move-effect* saved to memory.

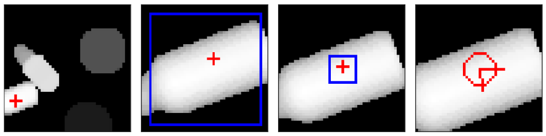

Fig. 5: HSA applied to grasping in the bottles on coasters domain (Section V-C). There are 4 levels (i.e. $L = 4$). The sensor's volume size z is $36 \times 36 \times 23.75$ cm for level 1, $10.5 \times 10.5 \times 47.5$ cm for levels 2 and 3, and $9 \times 9 \times 47.5$ cm for level 4. As indicated by blue squares, d constrains position in the range of $\pm 18 \times 18 \times 6.875$ cm for level 1, ± 4.5 cm^3 for level 2, and ± 1.125 cm^3 for level 3. Orientation is selected for level 4 in the range of $\pm 90°$ about the hand approach axis. Red crosses indicate the x, y position selected by the robot, and the red circle indicates the angle selected by the robot. Positions are sampled uniformly on a $6 \times 6 \times 6$ grid and 60 orientations are uniformly sampled. Pixel values normalized and height selection not shown for improved visualization.

required number of samples n_s is proportional to the workspace volume $v_0 = d_1(1)d_1(2)d_1(3)$, i.e., $n_s = \lceil v_0/\alpha \rceil$. But with HSA, we select position sequentially, by say, halving the volume size in each direction at each step, i.e., $d_{i+1} = 0.5 d_i$. In this case $8L$ samples are needed, i.e., a sample for each octant at each step. The volume represented by each sample at step i, for $i = 1, \dots, L$, is $v_i = v_0/8^i$. To get $v_L \leq \alpha$, i.e., to get the volume represented by samples used for selecting e.e. position to be no more than α, $L = \lceil \log_8(v_0/\alpha) \rceil$. Thus, with HSA, the sample complexity becomes logarithmic, rather than linear, in v_0.

C. Lookahead Sense-Move-Effect

So far we have not specified how action parameters T_s, T_{ee}, and z are encoded. For *standard sense-move-effect*, these are typically encoded as 6 floating point numbers representing the pose T and 3 floating point numbers representing the volume size z. Alternatively, the pair (T, z) could be encoded as the *sense* image that *would* be seen if the sensor were to move to pose T with zoom z. This is as if the action was "looking ahead" at the pose the sensor or e.e. would move to if this action were selected.

In particular, the *lookahead sense-move-effect* MDP has actions $\mathit{sense}(T_s, z_s)$ and $\mathit{move\text{-}effect}(T_{ee}, z_{ee}, o)$, the difference being the additional parameter $z_{ee} \in \mathbb{R}^3_{>0}$ for $\mathit{move\text{-}effect}$. The action samples are encoded as the height map that would be generated by $\mathit{sense}(T, z)$. Because action has this rich encoding, state is just the e.e. status and a history of k actions.

The HSA constraints for the lookahead variant have the same parameterization – an initial pose T_s^1 and a list of pairs $[(z_1, d_1), \dots, (z_L, d_L)]$. The semantics are slightly different. z_i for $i = 1, \dots, L-1$ is the z_s parameter for the ith sense, and z_L is the z_{ee} parameter. The d_i for $i = 1, \dots, L-1$ specify the offset of the sense action samples relative to the last pose decided, T_s^i. d_L specifies the offset of T_{ee} relative to T_s^L.

D. Relation to Other Approaches in the Literature

1) DQN: Consider a sense-move-effect MDP with HSA constraints $L = 1$, T_s^1 centered in the robot's workspace, and z_1 and d_1 large enough to capture the entire workspace. The only free action parameters for this system are the e.e. pose, which is sampled uniformly and spaced appropriately for the task, and the e.e. operation. In this case, the observations and actions are similar to that of DQN [3], and the DQN algorithm can be applied to the resulting MDP.

However, this approach is problematic in robotics because the required number of action samples is large, and the image resolution would need to be high in order to capture the required details of the scene. For example, a pick-place task where e.e. poses are in $SE(3)$, the robot workspace is 1 m^3, the required position precision is 1 mm, and the required orientation resolution is 1° per Euler angle requires on the order of 10^{17} actions. Adding more levels (i.e. $L > 1$) alleviates this problem.

2) Deictic Image Mapping: With $L = 1$, T_s^1 centered in the robot's workspace, z_1 the deictic marker size (e.g., the size of the largest object to be manipulated), and d_1 large enough to capture the entire workspace, HSA applied to the lookahead sense-move-effect MDP is the Deictic Image Mapping representation [4]. Similar to the case with DQN, if the space of e.e. poses is large, and precise positioning is needed, many actions need to be sampled. In fact, the computational burden with the Deictic Image Mapping representation is even larger than that of DQN due to the need to create images for each action. Yet, the deictic representation has significant advantages over DQN in terms of efficient learning due to its small observations [4].

HSA generalizes and improves upon both DQN and Deictic Image Mapping by overcoming the burden for the agent to select from many actions in a single time step. Instead, the agent sequentially refines its choice of e.e. pose over a sequence of L decisions. We provide comparisons between these approaches in Section V.

E. Implementation Methods

To implement HSA for a sense-move-effect MDP, it is necessary to select values for HSA parameters and a training algorithm. Here we provide rough guidelines for making both choices for standard HSA.

1) HSA Parameter Values: Ideal values for T_s^1, L, and $[(z_1, d_1), \dots (z_L, d_L)]$ depend on the position and size of the robot's workspace, the desired e.e. precision, and available computing resources. In our implementations, we have separate levels for selecting position and orientation, with position selecting levels occurring first. The procedure for deciding position selecting levels is as follows. First, the position component of the initial sensor pose T_s^1 is set to the center of the robot's workspace. Second, the number of action samples n_s depends on computing resources, e.g., the number of Q-values that can be evaluated in parallel. If $n_s = n^3$, where n is the number of position samples spaced evenly along an axis, then n is set to the largest integer such that n_s samples can be evaluated efficiently. Third, the number of levels L is the minimum number of times the workspace needs divided to achieve the desired e.e. precision. If $p \in \mathbb{R}^3_{>0}$ is the desired e.e. precision and $w \in \mathbb{R}^3_{>0}$ is the size of the workspace, $L = \max_{i=1,\dots,3} \lceil \log_n(w(i)/p(i)) \rceil$. Fourth, sampling regions d_i for $i = 1, \dots, L$ should be large enough so that, if patches size d_i are centered on samples in level $i-1$, the entire region is covered: $d_i = w/n^{i-1}$. Lastly, observation sizes z_i for $i = 1, \dots, L$ should be equal to d_i or the size of the largest object to be manipulated, whichever is largest. The latter condition is necessary if the entire object must be visible to determine the appropriate action. For example, when grasping bottles to be placed upright, either the top or bottom of the bottle must be visible to determine bottle orientation in the hand. Deciding orientation selecting levels is simpler: add 1 level per Euler angle, each with the desired angular e.e. precision.

2) Training Algorithm: Algorithm 1 is a variant of DQN [3] that follows the HSA constraints. For concreteness, this implementation stores experiences for Q-learning; modification for other temporal difference (TD) update rules, such as Sarsa [24] or Monte Carlo (MC) [36], is straight-forward. For simplicity of exposition, we also restrict to the case where image history consists of the current image I and the image I_h before the last grasp, e.e. status is binary *empty* or *holding*, and the e.e. operation is binary *open* or *close*.

Initially, the Q-function gets random weights, the experience replay database is empty, and the probability of taking random actions $\epsilon = 1$ (line 1). The environment is initialized to a scene unique to each episode (line 2). For each time step, the e.e. status is observed (line 5), and I_h is the previously observed image if the e.e. is holding something (lines 6-8). Then, for each HSA level, a *sense* action is taken (line 10), the pose of the next *sense* action is determined either randomly or according to Q (line 12), and the experience is saved (line 15).

```
Input: $nEpisodes$, $t_{max}$, $T_1$, $n_s$, $L$,
       $[(z_1, d_1), \ldots, (z_L, d_L)]$,
       $maxExperiences$, $trainEvery$
1  Initialize $Q$, $D$, $\epsilon$
2  for $i \leftarrow 1, ..., nEpisodes$ do
3      $env \leftarrow initialize\text{-}environment(i)$
4      for $t \leftarrow 1, \ldots, t_{max}$ do
5          $h = get\text{-}ee\text{-}status(env)$
6          $I_h = \text{NULL}$
7          if $t > 1 \wedge h = holding$ then
8              $I_h \leftarrow I$
9          for $l \leftarrow 1, \ldots, L$ do
10             $I \leftarrow sense(T_l, z_l)$
11             $o' \leftarrow (h, I_h, I)$
12             $T_{l+1} \leftarrow$
                   $get\text{-}pose(Q, o', T_l, d_l, n_s, \epsilon)$
13             $a' \leftarrow T_l^{-1} T_{l+1}$
14             if $t > 1 \vee l > 1$ then
15                 $D \leftarrow D \cup \{(o, a, o', r)\}$
16             $o \leftarrow o'$; $a \leftarrow a'$; $r \leftarrow 0$
17         $op \leftarrow get\text{-}ee\text{-}op(h)$
18         $overtAct \leftarrow move\text{-}effect(T_{L+1}, op)$
19         $r \leftarrow transition(env, overtAct)$
20     $D \leftarrow D \cup \{(o, a, \text{NULL}, r)\}$
21     if $modulo(i, trainEvery) = 0$ then
22         $D \leftarrow prune\text{-}exp(D, maxExperiences)$
23         $Q \leftarrow update\text{-}q\text{-}function(D, Q)$
24     $\epsilon \leftarrow decrease\text{-}epsilon(|D|)$
```

Algorithm 1: Train standard HSA.

Actions are encoded relative to the previous sense pose (line 13). Next, the robot moves the e.e. to T_{L+1} and performs an operation op, after which a reward is observed (lines 17 - 19). Finally, after $trainEvery$ episodes, the Q function is updated with the current experiences (lines 21 - 23), and ϵ is set inversely proportional to the number of experiences (line 24).

V. Application Domains

In this section we compare the HSA approach in 4 application domains of increasing complexity. The complexity increases in terms of the size of the action space and in terms of the diversity of object poses and geometries. We analyze simpler domains because the results are more interpretable and learning is faster (Table I). More complex domains are included to demonstrate the practicality of the approach. All training is in simulation, and Sections V-C and V-D include test results for a psychical robotic system. Source code for reproducing the simulated experiments is available at [37].

A. Tabular Pegs on Disks

Here we analyze the HSA approach applied to a simple, tabular domain, where the number of states and actions is finite. The domain consists of 2 types of objects – pegs and disks – which are situated on a 3D grid (Fig. 6). The robot can move its e.e. to a location on the grid and open/close its gripper. The goal is for the robot to place all the pegs onto disks.

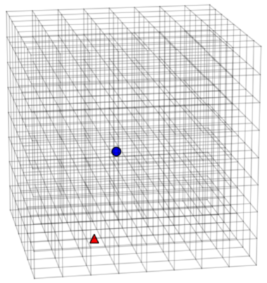

Fig. 6: Tabular pegs on disks with an $8 \times 8 \times 8$ grid, 1 peg (red triangle), and 1 disk (blue circle).

If this problem is described as a finite MDP, eventual convergence to the optimal policy is guaranteed for standard RL algorithms [26], [27]. However, the number of state-action pairs is too large for practical implementation unless some abstraction is applied. The main question addressed here is if convergence guarantees are maintained with the HSA abstraction.

1) Ground MDP: Tabular pegs on disks is first described without the sense-move-effect abstraction.

- **State.** A set of pegs $P = \{p_1, \ldots, p_n\}$, a set of disks $D = \{d_1, \ldots, d_n\}$, and the current time $t \in \{1, \ldots, t_{max}\}$. A peg (resp. disk) is a location

	Tabular Pegs on Disks	Upright Pegs on Disks	Bottles on Coasters	6-Dof Pick-Place
Time (hours)	0.23	1.29	8.12	96.54

TABLE I: Average simulation time for the 4 test domains. Times are averaged over 10 or more simulations on 4 different workstations, each equipped with an Intel Core i7 processor and an NVIDIA GTX 1080 graphics card.

$(x, y, z) \in \{1, \dots, m\}^3$ except peg locations are augmented with a special in-hand location h. Pegs (resp. disks) cannot occupy the same location at the same time, but 1 peg and 1 disk can occupy the same location at the same time.

- **Action.** $\textit{move-effect}(x, y, z)$, which moves the e.e. to $(x, y, z) \in \{1, \dots, m\}^3$ and opens/closes. It opens if a peg is located at h and closes otherwise.
- **Transition.** t increments by 1. If no peg is at h and a peg p is at the action location, then the peg is grasped ($p = h$). If a peg is located at h and the action location a does not contain a peg, the peg is placed ($p = a$). Otherwise, the state remains unchanged.
- **Reward.** 1 if a peg is placed on an unoccupied disk, -1 if a placed peg is removed, and 0 otherwise.

Initially, pegs and disks are at distinct locations, and no peg is in the e.e. The time horizon is $t_{\text{max}} = 2n$, where there is enough time to grasp and place each peg. This MDP satisfies the Markov property because the next state is completely determined from the current state and action. The number of possible states is shown in Eq. 1, and the number of actions is $|A| = m^3$. It is not practical to learn the optimal policy by enumerating all state-action pairs for this MDP: for example, if $m = 16$ and $n = 3$, the state-action value lookup table size is on the order of 10^{24}.

$$|S| = \binom{m^3 + 1}{n}\binom{m^3}{n} t_{max} \tag{1}$$

2) Sense-Move-Effect MDP: We apply the sense-move-effect abstraction of Section IV-A and HSA constraints of Section IV-B to the tabular pegs on disks problem. The process is illustrated in Fig. 7. At level 1, the sensor perceives the entire m^3 grid as 8 cells, each cell summarizing the contents of an octant of space in the underlying grid. The robot then selects one of these cells to attend to. At levels $2, \dots, L - 1$, the sensor perceives 8 cells revealing more detail of the octant selected in the previous level. At level L, the sensor perceives 8 cells in the underlying grid, and the location of the underlying action is determined by the cell selected here. Without loss of generality, assume the grid size m of the ground MDP is a power of 2 and the number of levels L is $\log_2(m)$.

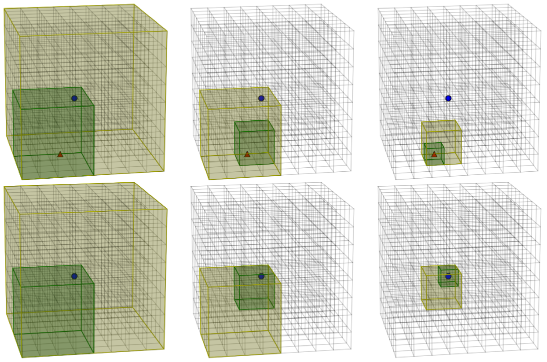

Fig. 7: HSA applied to the grid in Fig. 6. **Columns** correspond to levels 1, 2, and 3. The observed volume appears yellow, and the octant selected by the robot appears green. **Top**. Robot selects the peg and is holding it afterward. **Bottom**. Robot selects the disk.

- **State.** The current level $l \in \{1, \dots, L\}$, the overt time step $t \in \{1, \dots, t_{max}\}$, a bit $h \in \{0, 1\}$ indicating if a peg is held, and the tuple $G = (G_p, G_d, G_{pd}, G_e)$ where each $G_i \in \{0, 1\}^8$. G_p indicates the presence of unplaced pegs, G_d unoccupied disks, G_{pd} placed pegs, and G_e empty space.
- **Action.** The action is $a \in \{1, \dots, 8\}$, a location in the observed grids.

- **Transition.** For levels $l = 1, \ldots, L-1$, the robot selects a cell in G which corresponds to some partition of space in the underlying grid. The sensor perceives this part of the underlying grid and generates the observation at level $l+1$. For level L, the L selections determine the location of the underlying $move\text{-}effect$ action, l is reset to 1, and otherwise the transition is the same as in the ground MDP.
- **Reward.** The reward is 0 for levels $1, \ldots, L-1$. Otherwise, the reward is the same as for the ground MDP.

The above process is no longer Markov because a history of states and actions could be used to better predict the next state. For instance, for a sufficiently long random walk, the exact location of all pegs and disks could be determined from the history of observations, and the underlying grid could be reconstructed.

On the other hand, this abstraction results in substantial savings in terms of the number of states (Eq. 2) and actions ($|A| = 8$). The only nonconstant term (besides t_{max}) is logarithmic in m. Referring to the earlier example with $m = 16$ and $n = 3$, the state-action lookup table size is the order of 10^{11}.

$$|S| \leq 2^{33} \log_2(m) t_{max} \tag{2}$$

3) Theoretical Results: The sense-move-effect MDP with HSA constraints can be classified according to the state abstraction ordering defined in Li et al. [28]. In particular, we show Q^*-irrelevance, which is sufficient for the convergence of a number of RL algorithms, including Q-learning, to a policy optimal in the ground MDP.

Definition 5 (Q^*-irrelevance Abstraction [28])**.** *Given an MDP $M = \langle S, A, P, R, \gamma \rangle$, any states $s_1, s_2 \in S$, and an arbitrary but fixed weighting function $w(s)$, a Q^**-irrelevance abstraction *ϕ_{Q^*} is such that for any action a, $\phi_{Q^*}(s_1) = \phi_{Q^*}(s_2)$ implies $Q^*(s_1, a) = Q^*(s_2, a)$.*

ϕ_{Q^*} is a mapping from ground states to abstract states and defines the abstract MDP.[3]

Theorem 1 (Convergence of Q-learning under Q^*-irrelevance [28])**.** *Assume that each state-action pair is visited infinitely often and the step-size parameters decay appropriately. Q-learning with abstraction ϕ_{Q^*} converges to the optimal state-action value function in the ground MDP. Therefore, the resulting optimal abstract policy is also optimal in the ground MDP.*

Because Li et al. do not consider action abstractions, we redefine the ground MDP to have the same actions as sense-move-effect MDP. Additionally, to keep the ground MDP Markov, we add the current level l, and the current point of focus $v \in \{1, \ldots, m\}^3$, to the state. This does not essentially change the tabular pegs on disks domain but merely allows us to rigorously make the following connection.

Let states and actions of the ground MDP be denoted by s and a respectively. Similarly, let states and actions of the sense-move-effect MDP be denoted by $\bar{s}$ and $\bar{a}$ respectively. Let $\phi_{SME} : S \to \bar{S}$ be the observation function.

Theorem 2 (ϕ_{SME} is Q^*-irrelevant)**.** *The sense-move-effect abstraction, ϕ_{SME}, is a Q^*-irrelevance abstraction.*

Proof. $Q^*(s, a)$ can be computed from $\bar{s}$ and $\bar{a}$. The reward after the current overt stage t depends on h, whether or not it is possible to select a peg/disk, and whether or not it is possible to avoid selecting a placed peg. These are known from $\bar{s}$ and $\bar{a}$. Furthermore, whether or not a peg will be held after the current stage can be determined from $\bar{s}$ and $\bar{a}$. Finally, due to $t_{max} = 2n$ and the fact that all pegs are initially unplaced, the sum of future rewards following an optimal policy from the current stage depends only on (a) whether or not a peg will be

[3] Although the definition is for infinite-horizon problems (due to γ), our finite-horizon problem readily converts to an infinite-horizon problem by adding an absorbing state that is reached after t_{max} overt stages. The weight $w(s)$ is the probability the underlying state is s given its abstract state $\phi(s)$ is observed. Any fixed policy, e.g. ϵ-greedy with fixed ϵ, induces a valid $w(s)$ and satisfies the definition.

held after the current stage and (b) the amount of time left, $t-1$. □

4) Simulation Results: In these experiments, there were $n = 3$ objects, and the grid size was $m = 16$. Besides Deictic Image Mapping (where $L = 1$), the number of levels was $L = 4$. A comparison with no abstraction or HSA with $L = 1$ was not possible because the system quickly ran out of memory (Eq. 1). The learning algorithm was Sarsa [24], and actions were taken greedily w.r.t. the current Q-estimate. An optimistic initialization of action-values and random tie-breaking were relied on for exploration.

The proof to Theorem 2 suggests the observability of pegs, disks, placed pegs, and empty space are all important for learning the optimal policy. On the other hand, we empirically found no disadvantage to removing the G_{pd} (placed pegs) and G_e (empty space) grids. However, it is important to distinguish unplaced pegs and placed pegs. Fig. 8 shows learning curves for an HSA agent with G_p and G_d grids versus an HSA agent with the same grids but showing pegs/disks irregardless of whether or not they are placed/occupied.

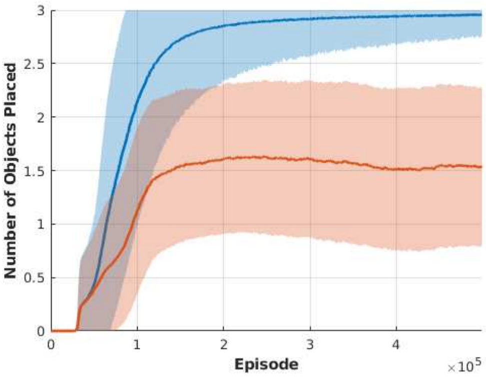

Fig. 8: Number of objects placed for the standard HSA agent (blue) and a standard HSA agent with a faulty sensor (red). Curves are first mean and $\pm\sigma$ over each episode in 30 realizations, then averaged over $1,000$-epsisode segments for visualization.

Lookahead HSA and Deictic Image Mapping variants (Section IV-C and IV-D) result in an even smaller state-action space than standard HSA. In the tabular domain, this means faster convergence (Fig. 9). Although the deictic representation seems superior in these results, it has a serious drawback. The action-selection time scales linearly with m^3 because there is one action for each cell in the underlying grid. The lookahead variant captures the best of both worlds – small representation and fast execution. Thus, in the tabular domain, lookahead appears to be the satisfactory middle ground between the two approaches. However, for more complex domains, where Q-function approximation is required, the constant time needed to generate the action images becomes more significant, and the advantage of lookahead in terms of episodes to train diminishes (Section V-B).

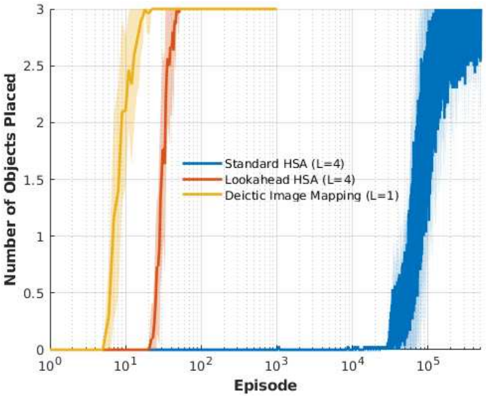

Fig. 9: Number of objects placed for standard HSA (blue), lookahead HSA (red), and Deictic Image Mapping (yellow) agents. Curves are mean (solid) and $\pm\sigma$ (shaded) over 30 realizations. Plot in log scale for lookahead and deictic results to be visible.

B. Upright Pegs on Disks

In this domain, pegs and disks are modeled as tall and flat cylinders, respectively, where the cylinder axis is always vertical (Fig. 10, left). Unlike the tabular domain, object size and position are sampled from a continuous space. Grasp and place success are checked with a set of simple conditions appropriate for upright cylinders.[4] The reward is 1 for grasping an unplaced peg, -1 for grasping a placed peg, 1 for placing a peg on an unoccupied disk, and 0 otherwise.

Observations consist of 1 or 2 images ($k = 2$, $n_{ch} = 1$, $n_x = n_y = 64$); the current HSA level, $l \in \{1, 2, 3\}$; and the e.e. status, $h \in \{empty, holding\}$. Each HSA level selects (x, y, z) position (Fig. 10, right). Gripper orientation is not critical for this problem.

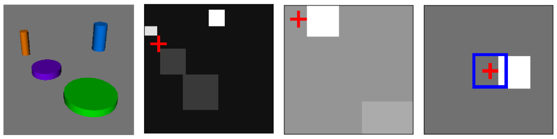

Fig. 10: **Left**. Example upright pegs on disks scene. **Right**. Level 1, 2, and 3 images for grasping the orange peg. Red cross denotes the (x, y) position selected by the robot and the blue rectangle denotes the allowed (x, y) offset. $z_x = z_y = 36$ cm^2 for level 1 and 9 cm^2 for levels 2 and 3. $d_x = d_y = 36$ cm^2 for level 1, 9 cm^2 for level 2, and 2.25 cm^2 for level 3. Pixel values normalized and height selection not shown for improved visualization.

1) Network Architecture and Algorithm: The Q-function consists of 6 convolutional neural networks (CNNs), 1 for each level and e.e. status, with identical architecture (Table II). This architecture results in faster execution time compared with our previous version [2]. The loss is the squared difference between predicted and actual action-value target, averaged over a mini-batch. The action-value target is the reward received at the end of the current overt stage.[5] For CNN optimization, Adam [38] is used with a base learning rate of 0.0001, weight decay of 0.0001, and mini-batch size of 64.

layer	kernel size	stride	output size
conv-1	7×7	2	$32 \times 32 \times 32$
conv-2	7×7	2	$16 \times 16 \times 64$
conv-3	7×7	2	$8 \times 8 \times 32$
conv-4	7×7	2	$4 \times 4 \times 32$
conv-5	7×7	1	$4 \times 4 \times 4$

TABLE II: CNN architecture for the upright pegs on disks domain. Each layer besides conv-4 and conv-5 has a rectified linear unit (ReLU) as the activation.

2) Simulation Results: We tested standard HSA with 1, 2, and 3 levels. The number of actions (CNN outputs) per level was adjusted so that each case had the same 5.625 mm precision in positioning of the e.e.: 1 level had 4^9 outputs, 2 levels 4^3 outputs and 4^6 outputs, and 3 levels each had 4^3 outputs. Note that with 1 level this is the DQN (i.e. no-hierarchy) approach (Section IV-D). Exploration was ϵ-greedy with $\epsilon = 0.04$.

Results are shown in Fig. 11. The 1 level case is faster in terms of episodes because learning is over fewer time steps. The 2 levels case initially learns faster for the same reason. The 1 and 2 level cases converge to higher values because, with 3 levels, there is a higher chance of taking a random action during an overt stage. This is because more levels imply more time steps over which a random action could be selected w.p. ϵ. What is important is that, in the last $5,000$ episodes when $\epsilon = 0$, all scenarios have similar performance. However, HSA trains faster than DQN in terms of wall clock time (1.29 versus 2.55 hours) because fewer actions need evaluated (192 versus $262,144$). This advantage becomes more staggering as dimensionality of the action space increases, as in following sections.

In another experiment we tested the sensitivity of standard HSA to the choice of z and d parameters. As explained in Section IV-E1, these parameters are selected based on task geometry. If z (resp. d) is too small, parts of the workspace will not be perceivable (resp. reachable). On the other hand, if z is too large, the scene will not be visible in detail (because the perceived images are of fixed resolution), and if d is too large, the samples at the last level will not be dense, resulting in low e.e. precision. Results for different values of z

[4] Grasp conditions: gripper is collision-free and the top-center of exactly 1 cylinder is in the gripper's closing region. Place conditions: entire cylinder is above an unoccupied disk and the cylinder bottom is at most 1 cm below or 2 cm above the disk surface.

[5] With standard MC and $\gamma = 1$, the action-value target would be the sum of rewards received after the current time step [36]. Since, for this problem, no positively rewarding grasp precludes a positively rewarding place, ignoring rewards after the current overt stage is acceptable.

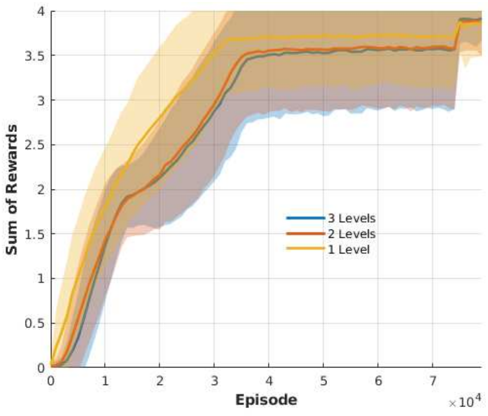

Fig. 11: Standard HSA with varying number of levels. (Blue) $L = 3$, (red) $L = 2$, and (yellow) $L = 1$. Curves are mean $\pm\sigma$ over 10 realizations then averaged over $1,000$ episode segments.

and d are shown in Table III. The “ideal” values are those selected according to the principles in Section IV-E1 and correspond to the 3-levels case in Fig. 11. As expected, performance is much worse when selecting z and d without consideration to task geometry.

	Small	**Ideal**	**Large**
level-1, $z_{xy} =$	36.0	36.0	36.0
level-1, $d_{xy} =$	36.0	36.0	36.0
level-2, $z_{xy} =$	6.00	9.00	12.00
level-2, $d_{xy} =$	6.00	9.00	12.00
levle-3, $z_{xy} =$	6.00	9.00	12.00
level-3, $d_{xy} =$	1.50	2.25	3.00
μ Return	2.69	3.91	2.83
σ Return	1.32	0.01	1.75

TABLE III: Varying standard HSA parameters z_{xy} and d_{xy} (in cm). “Ideal” values were selected according to Section IV-E1. “Small” (resp. “Large”) values are smaller (resp. larger) than ideal. Last 2 rows are average and standard deviation over sum of rewards per episode, after 10 different training sessions and $1,000$ episodes per session.

We also compared standard HSA to lookahead HSA, both with 3 levels. We did not compare to the Deictic Image Mapping approach (Lookahead HSA with 1 level) because computation of all 4^9 images was prohibitively expensive. Results are shown in Fig. 12. In contrast to the tabular results, both scenarios perform similarly. We hypothesize that the advantage of lookahead HSA is lost due to the equivariance property of CNNs. Since execution time for standard HSA is less than half that of lookahead (1.29 versus 3.67 hours), from now on we only consider standard HSA.

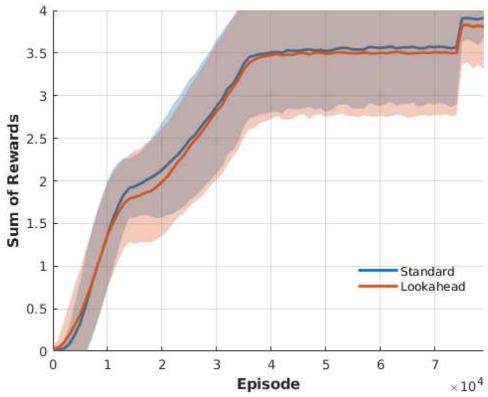

Fig. 12: Standard HSA (blue) versus lookahead HSA (red).

C. Bottles on Coasters

The main question addressed here is if HSA can be applied to a practical problem and implemented on a physical robotic system. The bottles on coasters domain is similar to the pegs on disks domain, but now objects have complex shapes and are required to be placed upright.[6] The reward is 1 for grasping an unplaced object more than 4 cm from the bottom (placing with bottom grasps is kinematically infeasible in the physical system), -1 for grasping a placed object, 1 for placing a bottle, and 0 otherwise.

Observations are similar to before except now the image resolution is lower ($n_x = n_y = 48$), and the overt time step is always input to grasp networks (and never input to place networks). HSA

[6] Grasp conditions: gripper closing region intersects exactly 1 object and the antipodal condition from [12] with $15°$ friction cone. Place conditions: bottle is upright, center of mass (CoM) (x, y) position at least 2 cm inside an unoccupied coaster, and bottom within ± 2 cm of coaster surface.

has 3 levels selecting (x, y, z) position and 1 level selecting orientation about the gripper approach axis (Fig. 5).

To achieve the target precision in e.e. pose (3.75 mm position and 6° orientation for grasping), DQN (or 1-level HSA) would need to evaluate over 53 million actions. Evaluation was prohibitively expensive with our computing hardware. HSA only needs 404 actions (although we use 708 to achieve redundancy, with little loss in computation time as the evaluation is done in parallel).

1) Network Architecture and Algorithm: The network architecture is shown in Table IV. There is 1 network for each HSA level and e.e. status. Weight decay is 0. Q-network targets are the reward after the current overt stage.

layer	kernel size	stride	output size
conv-1	8×8	2	$24 \times 24 \times 64$
conv-2	4×4	2	$12 \times 12 \times 64$
conv-3	3×3	2	$6 \times 6 \times 64$
conv-4 / ip-1	2×2 / -	1 / -	$6^3/n_{orient}$

TABLE IV: CNN architecture for the bottles on coasters domain. Each layer besides the last has a ReLU activation. The last layer is a convolution layer for levels 1-3 (selecting position) and an inner product (IP) layer for level 4 (selecting orientation). $n_{orient} = 60$ for grasp networks and $n_{orient} = 3$ for place networks.

2) Simulation Results: 70 bottles from 3DNet [39] were randomly scaled to height 10-20 cm. Bottles were placed upright with probability $1/3$ and on their sides with probability $2/3$. Learning curves for 2 bottles and 2 coasters are shown in Fig. 13. Performance is lower than that of the upright pegs on disks domain, reflective of the additional problem complexity.

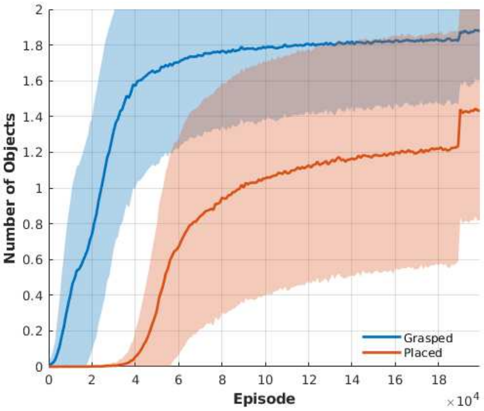

Fig. 13: Number of bottles grasped (blue) and placed (red). Curves are mean $\pm\sigma$ over 10 realizations then averaged over $1,000$ episode segments. Standard HSA with $L = 4$.

To test robustness of the system to background noise, we ran the same experiment with the addition of distractor objects. These distractors are 3 rectangular blocks, with side lengths 1 to 4 cm, scattered randomly in the scene (e.g., Fig. 14, left). Learning performance is only slightly lower (Fig. 14, right). However, if clutter is present at test time, it is important to train the system with clutter. The robot trained without clutter places an average of 1.24 bottles in the cluttered environment (versus 1.55 if trained with clutter). The distractors are visible at some levels (e.g., level 1), so the robot does need to learn to ignore (and avoid collisions with) them.

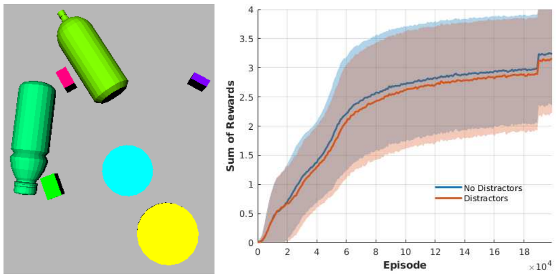

Fig. 14: **Left.** Scene with clutter. **Right.** Learning curves comparing average sum of rewards when distractors are not present (blue) and present (red).

3) Top-n Sampling: Before considering experiments on a physical robotic system, we address an important assumption of the move-effect system of Section III. The assumption is the e.e. can move to any pose, T_{ee}, in the robot's workspace. Recent advances in motion planning algorithms make this a reasonable assumption for the most part; nonetheless, a pose can still sometimes be unreachable due to obstacles, motion planning failure, or IK failure.

To address this issue, multiple, high-valued actions are sampled from the policy learned in simulation. In particular, for each level l of an overt stage, we take the top-n samples according to Eq. 3, where Q_l is the action-value estimate at level l, Q_{max} is the maximum possible action-value, Q_{min} is the minimum possible action-value, and $p_0 = 1$.

$$p_l = p_{l-1} \frac{Q_l - Q_{min}}{Q_{max} - Q_{min}}, \quad l = 1, \ldots, L \tag{3}$$

Preliminary tests in simulation showed sampling top-n p_l values performs better than sampling top-n Q_L values, as was done previously [2]. Sampling top-n p_l values may be viewed as an ensemble method where each level votes on the final overt action (cf. [40]).

During test time, the resulting n, T_{ee} samples are checked for IK and motion plan solution in descending order of p_L value. As n increases, the probability of failing to find a reachable e.e. pose decreases; however, the more poses that are unreachable, the lower the p_L value. Thus, when designing an HSA system, it is important to not over constrain the space of actions.

4) Robot Experiments: We tested the bottles on coasters task with the physical system depicted in Fig. 15. The system consists of a Universal Robots 5 (UR5) arm, a Robotiq 85 parallel-jaw gripper, and a Structure depth sensor. The test objects (Fig.16) were not observed during training. The CNN weight files had about average performance out of the 10 realizations (Fig. 13).

Initially, 2 coasters were randomly selected and placed in arbitrary positions in the back half of the robot's workspace (too close resulted in unreachable places). Then, 2 bottles were randomly selected and placed upright with probability $1/3$ and on the side with probability $2/3$. The bottles were not allowed to be placed over a coaster.[7] Top-n sampling with $n = 200$ was used. A threshold was set for the final grasp/place approach, whereby, if the magnitude of the force on the arm exceeded this threshold, the motion canceled and the open/close action was immediately performed.

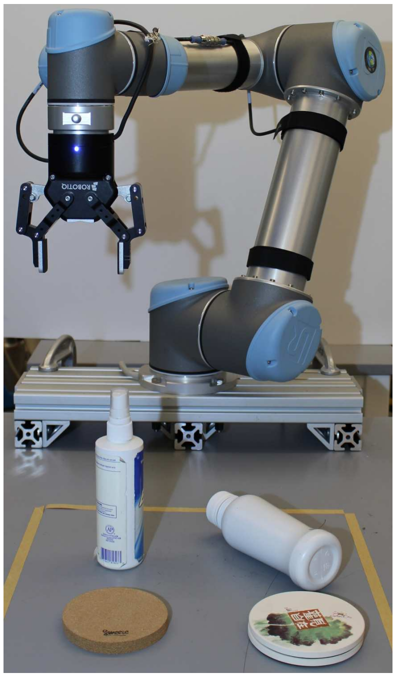

Fig. 15: Test setup for bottles on coasters task: a UR5 arm, Robotiq 85 gripper, Structure depth sensor (mounted out of view above the table and looking down), 2 bottles, and 2 coasters.

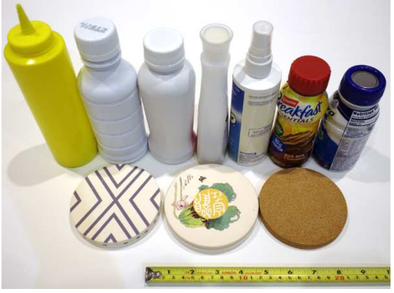

Fig. 16: Test objects used in UR5 experiments.

Results are summarized in Table V, and a successful sequence is depicted in Fig. 17. A grasp was considered successful if a bottle was lifted to the "home" configuration; a place was considered

[7] Python's pseudorandom number generator was used to decide the objects used and upright/side placement. Object position was decided by a human instructed to make the scenes diverse.

successful if a bottle was placed upright on an unoccupied coaster and remained there after the gripper withdrew. Failures were: grasped a placed object (×3), placed too close to the edge of a coaster and fell over (×3), placed upside-down (×2), object slip in hand after grasp caused a place failure (×1), and object fell out of hand after grasp (×1).

	Grasp	Place
Attempts	60	59
Success Rate	0.98	0.90
Number of Objects	1.97 ± 0.18	1.67 ± 0.48

TABLE V: Performance for UR5 experiments placing 2 bottles on 2 coasters averaged over 30 episodes with $\pm\sigma$. Task success rate with $t_{max} = 4$ was 0.67.

D. 6-DoF Pick-Place

The HSA method was also implemented for 6-DoF manipulation, and the same system was tested on 3 different pick-place tasks [2].[8] The tasks included stacking a block on top of another, placing a mug upright on the table, and (similar to Section V-C) placing a bottle on a coaster. All tasks included novel objects in light to moderate clutter (Fig. 18). To handle perceptual ambiguities in mugs, the observations were 3-channel images ($k = 2$, $n_{ch} = 3$, $n_x = n_y = 60$) projected from a point cloud obtained from 2 camera poses. HSA included 6 levels ($L = 6$) – 3 for (x, y, z) position and 1 for each Euler angle. Results from UR5 experiments are shown in Table VI.

	Blocks	Mugs	Bottles
Grasp	0.96	0.86	0.89
Place	0.67	0.89	0.64
Task	0.64	0.76	0.57
n Grasps	50	51	53
n Places	48	44	47

TABLE VI: **Top.** Grasp, place, and task success rates for the 3 tasks with $t_{max} = 2$ (i.e., 1 pick 1 place). **Bottom.** Number of grasp and place attempts.

VI. Conclusion

The primary conclusion is that the sense-move-effect abstraction, when coupled with hierarchical spatial attention, is an effective way of simultaneously handling (a) high-resolution 3D observations and (b) high-dimensional, continuous action spaces. These two issues are intrinsic to realistic problems of robot learning. We provide several other considerations relevant to systems employing spatial attention:

A. Secondary Conclusions

- Compared to a flat representation, HSA can result in an exponential reduction in the number of actions that need to be sampled (Section IV-B).
- HSA generalizes DQN, and lookahead HSA generalizes Deictic Image Mapping (Section IV-D).
- The partial observability induced by an HSA observation does not preclude learning an optimal policy (Section V-A).
- HSA may take longer to learn than DQN in terms of the number of episodes to convergence, but HSA executes faster when the number of actions is large (Section V-B).
- Lookahead HSA is preferred to standard HSA in terms of the number of the episodes to train, but execution time is longer by a constant and the learning benefit diminishes when coupled with function approximation (Sections V-A and V-B).
- HSA can be applied to realistic problems on a physical robotic system (Sections V-C and V-D).

B. Limitations and Future Work

A concern with all deep RL methods is that modeling and optimization errors induced by the use of function approximation prevent the robot from learning an optimal policy. This is true for even simple problems, such as the upright pegs on disks problem of Section V-B. Also, how manipulation skills can be automatically and efficiently transferred to different but related tasks remains an open question. Even small changes to the task, such as the inclusion of distractor objects, requires complete retraining of the system for maximum performance. Finally, HSA is a competing approach to

[8] This section refers to an earlier version of our system, so the simulations took longer and the success rates for bottles are lower. The setup was similar to that in Fig. 15 except the sensor was mounted to the wrist. See [2] for more details.

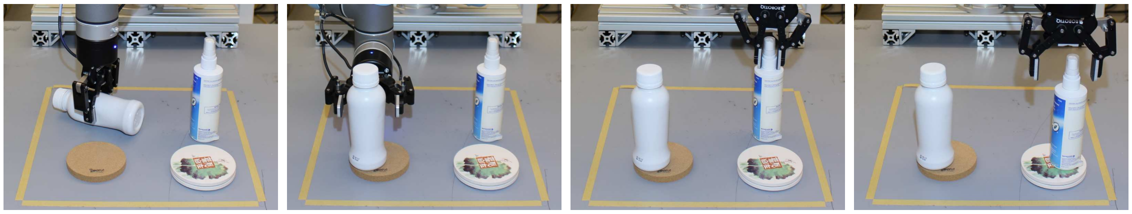

Fig. 17: Successful trial – all bottles placed in 4 overt stages. Image taken immediately after open/close.

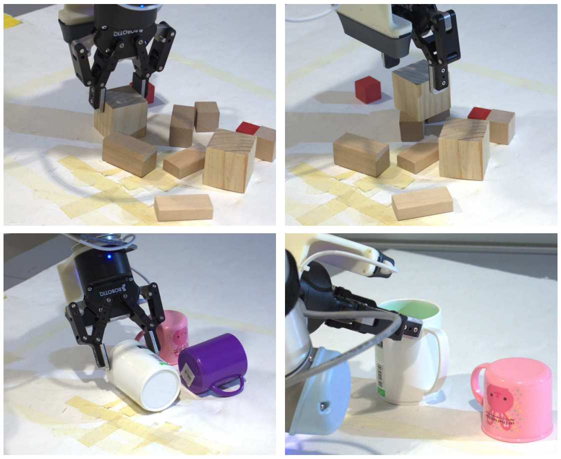

Fig. 18: 6-DoF pick place on the UR5 system. **Top**. Blocks task. **Bottom**. Mugs task. Notice the grasp is diagonal to the mug axis, and the robot compensates for this by placing diagonally with respect to the table surface.

policy search methods in that both can handle high-dimensional, continuous action spaces. It would be interesting to see a comparison between these approaches. Previous value-based approaches like DQN cannot handle the high-dimensional action spaces prevalent in robotics; thus, HSA enables a comparison between value and policy search methods for these domains.

Acknowledgment

We thank Andreas ten Pas for reviewing early drafts of this paper and the anonymous reviewers for their insightful comments. Funding was provided by NSF (IIS-1724257, IIS-1724191, IIS-1750649, IIS-1830425, IIS-1763878), ONR (N00014-19-1-2131), and NASA (80NSSC19K1474).

References

[1] S. Whitehead and D. Ballard, "Learning to perceive and act by trial and error," *Machine Learning*, vol. 7, no. 1, pp. 45–83, 1991.

[2] M. Gualtieri and R. Platt, "Learning 6-DoF grasping and pick-place using attention focus," in *Proceedings of The 2nd Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 87, Oct 2018, pp. 477–486.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[4] R. Platt, C. Kohler, and M. Gualtieri, "Deictic image maps: An abstraction for learning pose invariant manipulation policies," in *AAAI Conf. on Artificial Intelligence*, 2019.

[5] T. Lozano-Pérez, "Motion planning and the design of orienting devices for vibratory part feeders," *MIT Artificial Intelligence Laboratory Technical Report*, 1986.

[6] P. Tournassoud, T. Lozano-Pérez, and E. Mazer, "Regrasping," in *IEEE Int'l Conf. on Robotics and Automation*, vol. 4, 1987, pp. 1924–1928.

[7] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," in *Proceedings of The 2nd Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 87. PMLR, Oct 2018, pp. 306–316.

[8] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The Int'l Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.

[9] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *IEEE Int'l Conf. on Robotics and Automation*, 2016, pp. 3406–3413.

[10] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with large-scale data collection," in *Int'l Symp. on Experimental Robotics*. Springer, 2016, pp. 173–184.

[11] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," *Robotics: Science and Systems*, vol. 13, 2017.

[12] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt, "Grasp pose detection in point clouds," *The Int'l Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1455–1473, 2017.

[13] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Proceedings of The 2nd Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 87. PMLR, 29–31 Oct 2018, pp. 651–673.

[14] D. Morrison, P. Corke, and J. Leitner, "Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach," in *Robotics: Science and Systems*, 2018.

[15] D. Quillen, E. Jang, O. Nachum, C. Finn, J. Ibarz, and S. Levine, "Deep reinforcement learning for vision-based robotic grasping: A simulated comparative evaluation of off-policy methods," in *IEEE Int'l Conf. on Robotics and Automation*, 2018.

[16] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo, N. Fazeli, F. Alet, N. Chavan-Dafle, R. Holladay, I. Morona, Q.-N. Prem, D. Green, I. Taylor, W. Liu, T. Funkhouser, and A. Rodriguez, "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching," in *IEEE Int'l Conf. on Robotics and Automation*, 2018.

[17] Y. Jiang, C. Zheng, M. Lim, and A. Saxena, "Learning to place new objects," in *Int'l Conf. on Robotics and Automation*, 2012, pp. 3088–3095.

[18] M. Gualtieri, A. ten Pas, and R. Platt, "Pick and place without geometric object models," in *IEEE Int'l Conf. on Robotics and Automation*, 2018.

[19] A. Xie, A. Singh, S. Levine, and C. Finn, "Few-shot goal inference for visuomotor learning and planning," in *Proceedings of The 2nd Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 87. PMLR, 29–31 Oct 2018, pp. 40–52.

[20] S. James, A. Davison, and E. Johns, "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task," in *Conf. on Robot Learning*, vol. 78. Proceedings of Machine Learning Research, 2017, pp. 334–343.

[21] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *Advances in Neural Information Processing Systems*, 2017, pp. 5048–5058.

[22] J. Kober, A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The Int'l Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[23] C. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Cambridge, 1989.

[24] G. Rummery and M. Niranjan, "On-line Q-learning using connectionist systems," Cambridge University Engineering Department, CUED/F-INFENG/TR 166, September 1994.

[25] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[26] C. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[27] T. Jaakkola, M. Jordan, and S. Singh, "Convergence of stochastic iterative dynamic programming algorithms," in *Advances in Neural Information Processing Systems*, 1994, pp. 703–710.

[28] L. Li, T. Walsh, and M. Littman, "Towards a unified theory of state abstraction for MDPs," in *Int'l Symp. on Artificial Intelligence and Mathematics*, 2006.

[29] N. Sprague and D. Ballard, "Eye movements for reward maximization," in *Advances in Neural Information Processing Systems*, 2004, pp. 1467–1474.

[30] H. Larochelle and G. Hinton, "Learning to combine foveal glimpses with a third-order Boltzmann machine," in *Advances in Neural Information Processing Systems*, 2010, pp. 1243–1251.

[31] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu, "Recurrent models of visual attention," in *Advances in Neural Information Processing Systems*, 2014, pp. 2204–2212.

[32] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2017–2025.

[33] M. Gualtieri and R. Platt, "Viewpoint selection for grasp detection," in *IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, 2017, pp. 258–264.

[34] D. Morrison, P. Corke, and J. Leitner, "Multi-view picking: Next-best-view reaching for improved grasping in clutter," in *IEEE Int'l Conf. on Robotics and Automation*, 2019.

[35] B. Wu, I. Akinola, and P. Allen, "Pixel-attentive policy gradient for multi-fingered grasping in cluttered scenes," in *IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, 2019.

[36] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press Cambridge, 2018.

[37] "Source code for: Learning manipulation skills via HSA," https://github.com/mgualti/Seq6DofManip, accessed: 2019-11-13.

[38] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Int'l Conf. on Learning Representations*, 2015.

[39] W. Wohlkinger, A. Aldoma, R. Rusu, and M. Vincze, "3DNet: Large-scale object class recognition from CAD models," in *IEEE Int'l Conf. on Robotics and Automation*, 2012, pp. 5384–5391.

[40] O. Anschel, N. Baram, and N. Shimkin, "Averaged-DQN: Variance reduction and stabilization for deep reinforcement learning," in *Int'l Conf. on Machine Learning*, 2017, pp. 176–185.

Marcus Gualtieri is a PhD student at Northeastern University in Boston, Massachusetts. In 2017 he received the MS degree in computer science from Northeastern, and 2008 he received the BS degree in software engineering from Florida Institute of Technology. His research interests include robot learning and planning in unstructured environments.

Robert Platt is an associate professor at Northeastern. Prior to that, he was a research scientist at MIT and a robotics engineer at NASA. He earned his PhD in Computer Science in 2006 from the University of Massachusetts, Amherst. His research interests primarily include perception, planning, and control for robotic manipulation.