Learning 6-DoF Grasping and Pick-Place Using Attention Focus

Marcus Gualtieri
College of Computer and Information Science
Northeastern University, United States
mgualti@ccs.neu.edu

Robert Platt
College of Computer and Information Science
Northeastern University, United States
rplatt@ccs.neu.edu

Abstract: We address a class of manipulation problems where the robot perceives the scene with a depth sensor and can move its end effector in a space with six degrees of freedom – 3D position and orientation. Our approach is to formulate the problem as a Markov decision process (MDP) with abstract yet generally applicable state and action representations. Finding a good solution to the MDP requires adding constraints on the allowed actions. We develop a specific set of constraints called hierarchical SE(3) sampling (HSE3S) which causes the robot to learn a sequence of gazes to focus attention on the task-relevant parts of the scene. We demonstrate the effectiveness of our approach on three challenging pick-place tasks (with novel objects in clutter and nontrivial places) both in simulation and on a real robot, even though all training is done in simulation.

Keywords: Grasping, Manipulation, Reinforcement Learning

1 Introduction

Reinforcement learning (RL) promises a way to train robots in dynamic environments with little supervision – task is specified through a reward signal that could be changing or uninformative for a significant period of time. Unfortunately, classical RL does not work in realistic settings due to the high dimensional, continuous space of robot actions (position and orientation in 3D) and due to the complexity of the Markov state (all objects with their poses, velocities, shapes, masses, friction coefficients, etc.) Recent research addresses the complex state space by combining deep learning with RL and by representing state as a history of high resolution images [1]. However, the problem remains difficult due to noisy, incomplete sensor readings (partial observability), novel objects (generalization), and the high dimensional, continuous action space (curse of dimensionality). Additionally, deep RL methods tend to be data hungry which limits rapid, on-line learning.

Building on deep RL, we begin to address the issues of generalization and the complex action space by training a robot to focus on task-relevant parts of the scene. Restricting the robot's attention to task relevant areas is useful because incidental differences in scenes can be ignored and thus generalization is improved. Attention focus also narrows the space under which actions plausibly can be applied. In order to achieve this, our first step is to reformulate the problem as an MDP with abstract sense and motion actions. These abstractions induce a state space where it is at least possible for the robot to reason over only task-relevant information. Second, we develop a set of constraints on the sense actions which forces the robot to learn to focus on the task-relevant parts of the scene. This comes in the form of a hierarchical sequence of gazes that the robot must follow in order to focus attention on the task-relevant part of the action space. Third, we train in a simulated environment that can run many times faster than the real world and automatically provides information on task success. Direct transfer from simulation to the real world is possible due to the compact state representation and the way the robot learns to use it.

Specifically, contributions include (a) an abstract state-action representation to enable task-relevant focus (sense-move-effect MDP), (b) a set of constraints on the robot's action choices to enable learning of task-relevant focus (HSE3S), and (c) a single system which implements these ideas and can learn either 6-DoF grasping, 6-DoF placing, or joint 6-DoF grasping and placing by varying only the reward function. The effectiveness of the system is demonstrated both in simulation and on

2nd Conference on Robot Learning (CoRL 2018), Zürich, Switzerland.

a physical UR5 robot on three challenging pick-place benchmark tasks. Although we focus on pick-place problems (the most fundamental manipulation primitives – the robot must begin by grasping an object and end by placing it) many of the ideas are quite general, as discussed in Section 7.

2 Related Work

Traditional approaches to robotic manipulation require accurate 3D models of the objects to be manipulated [2]. Each object model has predetermined grasp and place poses associated with it, and the problem reduces to fitting the model to the sensor observation and calculating appropriate motion plans. Recent research has taken another direction – grasping and placing novel objects, i.e., objects the robot has not been explicitly trained to handle. Most of these projects focus on grasping [3, 4, 5, 6, 7], and the few that extend to pick-place (where the goal placement is non-trivial) assume either fixed grasp choices [8] or fixed place choices [9]. The present work generalizes these approaches by introducing a single system that can learn both grasping and placing.

Our approach to novel object manipulation leverages advances in deep RL. In particular, our learning agent is similar to that of deep Q-network (DQN) [1]; however, DQN has a simple, discrete list of actions, whereas our actions are continuous and high dimensional. Additionally, DQN operates on the full images of the scene, whereas our robot learns to focus on the parts of the scene that are relevant for the task. More recent approaches to RL allow continuous, high-dimensional actions, in addition to operating on the full image of the scene [10, 11, 12]. These approaches take raw sensor observations (camera images) and output low-level robot actions (motor torques, velocities, or Cartesian motions). In contrast, state and action in our approach are more abstract – state includes task-focused images and action includes target end effector pose – in order to speed up learning and introduce robustness to transfer from a simulated domain to a real robot. James, Davison, and Johns employ domain randomization to enable transfer of a policy learned in simulation to the real world [13], and Andrychowicz *et al.* address the problem of exploration by replaying experiences with different goals [14]. We view both of these improvements as complimentary: it is possible that our method could be improved by combining with these ideas.

Our strategy for sensing relies on learning a sequence of gazes in order to focus the robot's attention on the task-relevant part of the scene. This idea was inspired by work done with attention models and active sensing. Attention models are posed with the task of finding an object of interest in a 2D image [15, 16, 17]. Our problem is much more difficult in that we select 6-DoF actions to maximize a generic reward signal. Active sensing is the problem of optimizing sensor placements for the achievement of some task [18, 19]. However, we are mainly just interested in finding constraints to the sensing actions that allow efficient selection of end effector actions.

3 Problem Statement

The state of the world consists of a robotic manipulator and several objects. Rigidly attached to the robot are an end effector and a depth sensor. The robot's configuration is completely described by a list of joint angles, $q = [q_1, \ldots, q_n]$, $q_i \in \mathbb{R}$. The objects can have arbitrary poses and shapes. The robot's actions are delta motions of each joint, including the end effector joints: $\delta = [\delta_1, \ldots, \delta_n]$, $\delta_i \in \mathbb{R}$. If there is no collision, $q_{t+1} = q_t + \delta_t$. On each time step the robot observes the environment through its depth sensor as a point cloud – a list of 3D points in the reference frame of the sensor. The robot has a finite memory for storing up to the last k observations.

The task is specified by a human operator or programmer via a reward function, e.g. $+1$ when an object is grasped and placed into a desired configuration and 0 otherwise. The primary sources of uncertainty in the environment include (a) object surfaces are only partially visible in the current observation and (b) actions may not move the robot by the desired δ due to collisions. The goal of the learning algorithm is to learn a policy, i.e. a mapping from a history of observations to actions, that maximizes the expected sum of rewards per episode. The robot is not provided with a model of the environment – it must learn through trial and error. The robot is allowed to act for a maximum number of time steps per episode.

4 Approach

The first part of our approach is to reformulate the problem as an MDP with abstract states and actions (Section 4.1). This makes it possible for the robot to reduce the amount of sensor data it must process, assuming it takes the right actions. The second part of our approach is to add constraints to the allowed sense actions (Section 4.2) which forces the robot to learn to focus attention on task-relevant parts of the scene. The problem is then amenable to solution via deep RL (Sections 4.3 and 4.4).

4.1 Sense-Move-Effect MDP

An MDP consists of states, actions, a transition model, and a reward function. The reward function is task specific, and we give examples in Sections 5 and 6. A diagram of the abstract robot is shown in Figure 1 (left).

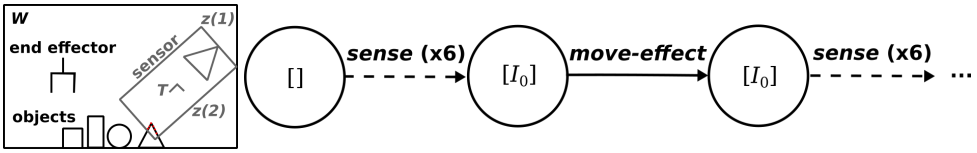

Figure 1: *Left.* The abstract robot has a mobile sensor which observes points in a rectangular volume at pose T with size z. The robot also has an end effector which can be moved to T to perform some operation o. *Right.* Sequence of actions the robot is allowed to take with state shown in circles.

Action consists of two high-level capabilities: *sense*(T, z) and *move-effect*(o). *sense*(T, z) uses the robot's sensor to acquire a point cloud C with origin at $T \in W$, where $W \subseteq \mathrm{SE}(3)$ is the reachable workspace of the robot's end effector, and where the point cloud is cropped to a rectangular volume of length, width, height $z \in \mathbb{R}^{+3}$. *move-effect*(o) moves the robot's end effector frame to T and activates controller $o \in O$, where O is a set of preprogrammed controllers actuating only end effector joints. As a running example, a binary gripper could be attached to the end effector with controllers $O = \{open, close\}$ providing capabilities for pick-place tasks.

State consists of k images, $(I_1, \ldots, I_k)$, with $k \geq 1$. Each image has three channels corresponding to three, fixed-resolution height maps of C as shown in Figure 2. Since the height maps are of fixed resolution, larger values of z will show more objects in the scene at less detail, and smaller values of z will show only an object part in more detail. The purpose of using images instead of point clouds is to make the state encoding more compact and to allow it to work with modern convolutional neural network (CNN) architectures. To further reduce the size of the state representation, we only include the most recent observation – because it represents the current location T of the abstract sensor – and only observations just before *move-effect* actions – because the environment changes only after *move-effect* actions. For many pick-place tasks $k = 2$ is sufficient because the two observations represent (a) the current target of the grasp/place and (b) the current hand contents.

Transition dynamics are initially unknown to the robot and depend on the domain, e.g. simulation or real world. We assume a static environment except during *move-effect* actions. In practice, *move-effect* actions rely on a motion planner and controller to make the continuous motion from its current pose to T. These motions may fail due to planning failure or collisions.

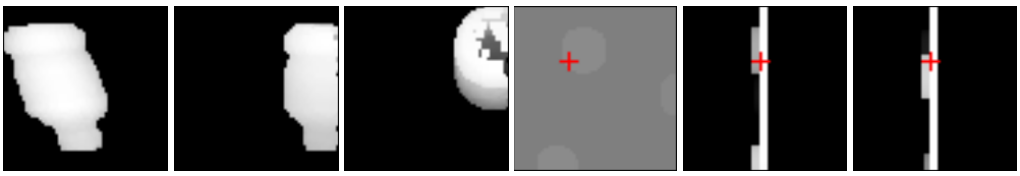

Figure 2: *Left three.* Height maps of a the sensed point cloud of a bottle along z, y, and x directions just before *move-effect(close)* was called. *Right three.* Height maps of the currently sensed point cloud along z, y, and x directions. The red cross indicates the next position the robot intends to move to in order to place the bottle on a coaster.

4.2 HSE3S

The above MDP is still very difficult to solve if any action in the combined $W \times \mathbb{R}^{+3} \times O$ space is allowed. To make learning feasible, before each *move-effect* action, the robot is constrained to follow a fixed number of *sense* actions with predetermined values for z. This process is illustrated in Figure 1 (right). We call this approach *hierarchical SE*(3) *sampling* (HSE3S). The basic idea is to select the next $sense$ position within the first, zoomed-out observation, decrease z to focus on that point of interest, select another $sense$ position within this new observation, and so on for a fixed number of time steps, at which point the focused volume is small enough that few samples are needed to precisely choose the end effector pose.

In particular, the HSE3S approach specifies a list of tuples $[(z_1, d_1) \ldots, (z_{\bar{t}}, d_{\bar{t}})]$, where z_i is a predetermined z value that the agent must use when taking *sense*(T, z) actions, and where $d_i \in \mathbb{R}^6$ is a predetermined allowed range of motion relative to the sensor's current location for choosing T. In other words, $z_i = [z(1), z(2), z(3)]$ sets the observation volume's length, width, and height for the i^{th} sense action in the sequence; and $d_i = [x, y, z, \theta, \phi, \rho]$ limits the allowed position offset to $\pm x$, $\pm y$, and $\pm z$ and rotational offset to $\pm\theta$, $\pm\phi$, and $\pm\rho$.

To see how this improves sample efficiency, imagine the problem of selecting position in a 3D volume. Let α be the largest volume allowed per sample. The number of samples n_s needed is proportional to the initial volume $v_0 = z_0(1)z_0(2)z_0(3)$, i.e. $n_s = \lceil (1/\alpha)v_0 \rceil$. But with HSE3S, we select position sequentially, by say, halving the volume size in each direction at each time step, i.e. $z_{t+1} = 0.5z_t$. In this case 8 samples are needed for each part of the divided space at each time step, so the sample complexity is $8\bar{t}$, where $\bar{t}$ is the total number of time steps, and $v_t = v_0/8^t$. To get $v_{\bar{t}} \leq \alpha$, $\bar{t} = \lceil \log_8(v_0/\alpha) \rceil$, so the sample complexity is now logarithmic in v_0, but this is at the expense of having to learn to select the correct partition in each step.

We typically use HSE3S in situations where the observation is of a fixed resolution, due to limited sensing or processing capabilities, so that each consecutive *sense* action reveals more detail of a smaller part of the scene. One potential issue with this partially observable case is that at the most zoomed-out level, objects may be blurry, and the robot may choose the wrong object without having a way to go back. Precisely, let *sense-mislead* be when the highest valued *sense* choice at time t does not lead to the highest valued possible action at time $t + i$ for some positive i. To mitigate issues with sense-misleads, we can run HSE3S in n independent trials, where the samples are different for each trial, and choose the trial that ends with the highest valued action choice. In practice, we take the 1-trial action during training when computational efficiency is more important than performance and the n-trial action during test time when performance is more important than efficiency. This is analogous to how Monte Carlo tree search is used on-line to improve policies [20].

The specific sense action sequence developed for our pick-place system is illustrated in Figures 3. The robot is initially presented with height maps of a point cloud centered in the robot's workspace and truncated at 36 cm^3. The robot then chooses a position, constrained to be on a point in this initial cloud. At the second level, the robot chooses any position in a 9 cm^3 volume, centered on the position selected in the previous level. At the third level, the robot chooses orientation about the current z-axis in $\pm\pi$ with $z = 9$ cm^3. At the fourth level, the robot chooses orientation about the current y-axis in $\pm\pi/2$ with $z = 9$ cm^3. At the fifth level, orientation about z-axis in $\pm\pi$ with $z = 9$ cm^3. Finally, at the sixth level, position is chosen with $z = 10.5$ cm^3, but this time the samples are ± 10.5 cm only on the current x, y, or z axes. The purpose of this last level is to allow for a large offset when placing, such as placing on top of another object.

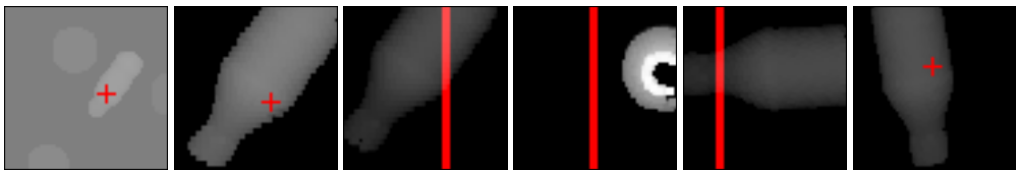

Figure 3: Example sense sequence for grasping a bottle: (a) coarse position, (b) fine position, (c) orientation about z, (d) orientation about y, (e) orientation about z, (f) final position. Crosses denote the position selected for the next action, and lines denote the orientation selected for the next action (where the center is 0 and edges $\pm\pi$). For all images besides (d) the gripper approaches the image plane with the fingers on the left and right sides.

In addition to these the HSE3S constraints, we also constrain our pick-place system to follow a *move-effect(close)-move-effect(open)* sequence, since it does not make sense to open the gripper when it is already open and vis versa. We further ensure the gripper is in its open configuration at the beginning of each episode.

4.3 Learning Model

For each time step in our MDP, the allowed actions and image zoom levels are different. Thus, we factor the Q-function into several independent CNNs, one for each time step. The architecture for each is the same and is illustrated in Figure 4 (left). Any unused input is set to zero; e.g., when selecting orientation, one element of the action vector is set to an Euler angle and the other five are zero.

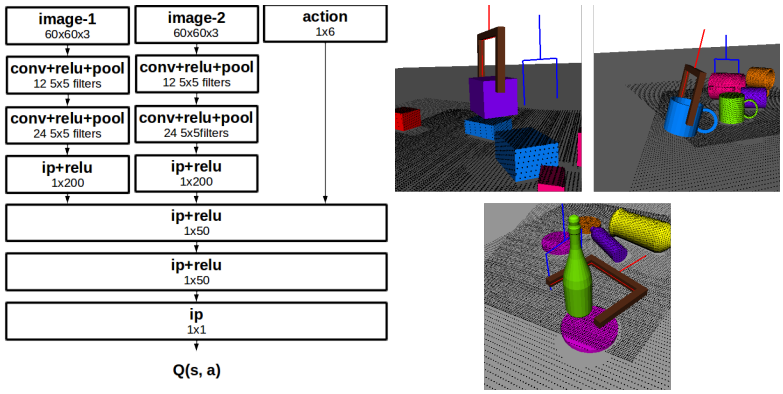

Figure 4: *Left.* CNN architecture used to approximate $Q_t(s, a)$ for each step t in a pick-place process. *Right.* Example places for the three tasks in simulation. The blue marker shows from where the object was grasped. In each case, all required and partial credit conditions were met, except with the bottle, where it was placed too high for the partial credit height condition.

4.4 Learning Algorithm

Our learning algorithm is similar to DQN [1] except instead of Q-learning we use gradient Monte Carlo [21] for the update rule. That is, for each state-action pair (s_t, a_t) in an experience replay buffer, we assign the sum of rewards experienced from that time forward $r_t + r_{t+1} + \cdots + r_{\bar{t}}$ as the training label. We empirically found Monte Carlo to perform just as well as Sarsa [22] for a grasping task (*c.f.* [23]), and it is computationally more efficient due to not having to evaluate the Q-function when generating training labels.

5 Simulation Experiments

We evaluated our approach on three pick-place tasks in simulation as depicted in Figure 4. For the first task, the robot is presented with 2 to 10 blocks cluttered together. The goal is to grasp one block and stack it on top of another block. For the second task, the robot is presented with 1 to 5 mugs in clutter. The goal is to grasp one mug and place it upright on the table. In the third task the robot is presented with 1 to 3 bottles and 3 round coasters. The goal is to grasp a bottle and place it upright on a coaster. We used OpenRAVE for the simulation environment [24] and 3DNet for non-primitive objects (i.e. bottles and mugs) [25]. The robot is given no prior information on how to grasp or place other than the constraints mentioned in Section 4 – the robot must learn these skills from trial and error.

Ideally, the reward function for each scenario should simply describe the set of goal states, e.g. +1 when the object is placed correctly and 0 otherwise. However, successful 6-DoF pick-places are rare at the beginning of learning when actions are random. To make the reward signal more informative, we assign a reward in $[0, 1]$ for a successful grasp and an additional reward in $[0, 1]$ for a successful place. For both grasping and placing there are required conditions and partial credit conditions. All

required conditions must be met for the reward to be nonzero; otherwise the reward is proportional to the number of partial credit conditions met. For example, when placing bottles, the required conditions include: (a) all required grasp conditions, (b) the bottle is within 30° of upright, (c) above a coaster, (d) no more than 4 cm above the coaster, and (c) not in collision or not in collision if moved up 2 cm. The partial credit conditions include: (a) within 15° of upright, (b) no more than 2 cm above the coaster, and (d) not in collision. Similar reward functions were developed for both grasping and placing stages for the other tasks.

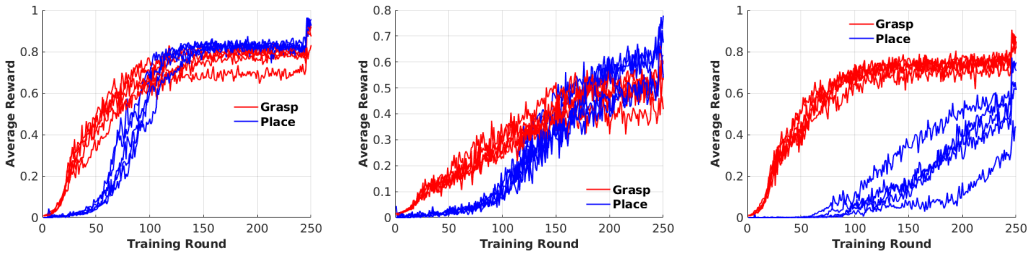

Figure 5: *Left.* Blocks. *Center.* Mugs. *Right.* Bottles. Grasp and place rewards shown in different colors – episode return is the sum of the two. Rewards are averages over all of the episodes in a training round. Each training round consisted of $2,000$ episodes followed by $3,000$ iterations of stochastic gradient descent. Maximum possible reward for a grasp/place is 1. Each plot includes learning curves for five independent runs. All episodes used 1-trial sampling (Section 4.2).

Learning curves for the three benchmark tasks are shown in Figure 5. The exploration strategy was ϵ-greedy, where $\epsilon = 1$ at the beginning (i.e. completely random actions), gradually drops down to 0.05 by the 25^{th} training round for grasping (ϵ-place drops as more place attempts are experienced due to successful grasps), and $\epsilon = 0$ for the last 5 training rounds. In terms of task difficulty, placing blocks appears easiest, followed by placing bottles on coasters, followed by placing mugs on the table. This ordering is consistent with the ambiguity of the object pose given a partial point cloud: blocks have three planes of symmetry, bottles have two orthogonal planes of symmetry, and mugs have at most one plane of symmetry (an upright plane dividing the handle). Also, there is a high variance between training realizations placing bottles on coasters, and this in turn may be due to the longer time it takes to learn the first several successful placements. We suspect this is because the space of goal placements is small compared to the blocks scenario (where the desired orientation is less specific) or the mugs scenario (where anywhere on the table is fine).

	Blocks A	Blocks A∧CF	Mugs A	Mugs A∧CF	Bottles A	Bottles A∧CF
GPD	0.95	0.51	0.77	0.52	**0.98**	0.71
1-trial	0.86	0.82	0.65	0.58	0.83	0.81
10-trial	**0.99**	**0.98**	**0.85**	**0.80**	0.94	**0.93**

Table 1: Grasp success rates averaged over $1,000$ independent episodes. Performance is shown with the antipodal condition [6] only (A) and with both antipodal and collision-free conditions (A∧CF). The comparison is with GPD [6], 1-trial HSE3S, and 10-trial HSE3S. GPD used 500 grasp samples and the highest-scoring grasp for evaluation. Both GPD and HSE3S were trained with 3DNet objects in the named category.

For the three pick-place tasks, grasping and placing are learned jointly. Unlike related work where mechanically stable grasping is an assumed capability [8, 9], a 6-DoF grasp pose is selected by the robot to maximize the expected reward for the full pick-place task. However, our same system can be easily modified to learn mechanically stable grasping independently of placing. We compared the grasping performance of our system to an off-the-shelf grasping software, grasp pose detection (GPD) [6]. The result is summarized in Table 1. For blocks, the HSE3S approach gets perfect grasps about 98% of the time whereas GPD gets perfect grasps only 51% of the time. Both systems were trained on the same set of blocks; however, the evaluation was done in the same environment (e.g. same simulated depth sensor) that the HSE3S system was trained in, so it is not surprising HSE3S does better. However, the evaluation for stable grasps is nearly the same for both methods (antipodal, force closure grasps [6]), and GPD uses a much larger representation for the state images (includes surface normals). The collision criteria in simulation may be overly conservative, so we also show success rates with the antipodal condition alone.

In the above analysis with grasping, we used the n-trial sampling strategy to minimize sense-mislead errors as described in Section 4.2. The HSE3S sequence was run 10 independent times with random sampling, and the highest valued state-action at the end was taken for *move-effect(close)*. Without n-trial sampling, HSE3S gets perfect grasps only 82% of the time for blocks. We would expect similar performance gains at the end of the learning curves in Figure 5 by n-trial sampling.

6 Robot Experiments

We tested our approach on a UR5 robot equipped with a Robotiq 85 parallel-jaw gripper and a wrist-mounted Structure depth sensor. The policies learned in simulation were directly transfered to the physical system without additional training. Objects were selected to be diverse, able to fit in the robot's gripper, and visible to the sensor. The complete test sets are shown in Figure 6. Prior to each episode, the robot acquired two point clouds approximately $45°$ above the table on opposite sides. This point cloud was transformed, cropped, and projected into images as the robot followed the HSE3S sequence. That is, in our experiments the *sense* actions operate virtually on a point cloud acquired from fixed sensor positions; however, in principle, for each *sense* action, the robot could physically move the sensor to acquire a point cloud centered on the target *sense* volume.

Figure 6: *Left.* Blocks test set. 10/15 blocks were randomly selected for each round. *Center.* Mugs test set. 5/6 mugs were randomly selected for each round. *Right.* Bottles test set. 3/7 bottles were randomly selected for each round. All 3 coasters were present in each round.

The tasks were much the same as in simulation. For the first task, 10 blocks were randomly selected from the test set and dumped onto the table. The robot's goal was to place one block on top of another. After each place or each failure, the block interacted with by the robot was removed by a human. This continued until only 1 block remained. For the second task, 5 mugs were selected from the test set and dumped onto the table, and the robot was to place them upright on the table. After each place or failure, the mug interacted with was removed. This continued until no mugs remained. For the third task, the 3 bottles and 3 coasters were placed arbitrarily except no bottle was allowed to touch any coaster. Again, after each place or failure, a bottle was removed, and this continued until no bottles remained. Examples of these tasks being attempted by the robot are shown in Figure 7.

In the real robot experiments, n-trial sampling, described in Section 4.2, played an additional role besides mitigating issues with sense-misleads. Several (about 100) samples were evaluated for grasps and places, rejected at each level if values were below a threshold, and sorted by the last value in the sense sequence. Then IK and motion planning was attempted for each grasp/place pose, starting with the highest valued pose, and continuing down the list until a reachable pose was found. For the mug and bottle tasks, additional place poses were added at the original place position and rotated about the gravity axis. This degree of freedom in the object reference frame should not affect place success, and this helped with finding reachable, high valued place poses.

Table 2 shows the grasp, place, and task success rates for each task. Results are shown with and without detection failures. A *detection failure* occurred when all grasp or place values were below a threshold. In these cases, the robot believes there is no good option available in the scene and, in a deployed scenario, would ask for human assistance. The major failure for block placing was misalignments with the bottom block. We believe this is primarily due to incorrect point cloud registration with the robot's base frame. Precise point cloud alignment was important because many of the place surfaces were small – the smallest block size was 2 cm^3. For mugs, grasp failures were the main problem. This corresponds to simulation results where mug grasp performance was lower than in the other two scenarios (Figure 5). For bottles, there were a combination of problems, mainly placing upside-down, diagonally, or too close to the edge of the coaster. The poorer place performance for bottles also corresponds with simulation results.

Figure 7: *Left.* Blocks task: one block must be placed on another. *Center.* Mugs task: one mug must be placed upright on the table. Notice the grasp is diagonal to the mug axis, and the robot compensates for this by placing diagonally with respect to the table surface. *Right.* Bottles task: place a bottle on a coaster.

	Blocks (w/ d.f.)	Blocks	Mugs (w/ d.f.)	Mugs	Bottles (w/ d.f.)	Bottles
Grasp	0.96	0.96	0.86	0.86	0.89	0.90
Place	0.67	0.67	0.89	0.91	0.64	0.67
Task	0.64	0.64	0.76	0.78	0.57	0.60
n Grasps	50	50	51	50	53	50
n Places	48	48	44	43	47	45
DF	0	N/A	1	N/A	3	N/A
GF	2	2	7	7	5	5
FOS	13	13	N/A	N/A	6	6
PUD	N/A	N/A	4	4	7	7
PIS	3	3	0	0	2	2

Table 2: *Top.* Success rates for blocks, mugs, and bottles tasks, separately including detection failures (w/ d.f.) and not counting attempts where there was a detection failure. *Middle.* Number of grasp and place attempts. A place was attempted only after the grasp was successful. *Bottom.* Failure counts broken down by type: detection failure (DF), grasp failure (GF), fell off support surface (FOS), placed upside-down (PUD), and placed into support surface (PIS).

7 Conclusion

Although we focus on pick-place tasks with simple grippers, the ideas developed in this paper could be extended to other tasks. HSE3S could be used to improve sample efficiency of any method that samples poses in $\mathrm{SE}(3)$, at the cost of having to learn where to gaze. Also, the deictic [26] state representation has been used in this and previous work [9] with much empirical success at transferring a policy learned in simulation to the real world. Also, one may imagine additional tasks that could be performed with the present system if more primitive actions were available, such as opening a drawer may require a *move-in-line*(v) action that moves from the current pose along vector v.

For future work, we plan to compare HSE3S to actor critic methods like DDPG [10] where action is output from the function approximator. Outputting the action directly avoids the need to sample actions; however, sampling actions has the benefit of inducing a nondeterministic policy useful in situations with reachability constraints. We also plan to examine how this approach scales to broader object categories, e.g. kitchen items. Finally, we would would like to demonstrate the approach on problems with longer time horizons. For instance, the three benchmark tasks could become block stacking, placing all mugs upright, and placing all bottles on coasters. Such problems may require an even more compact state representation – since state would consist of a longer history – and further speedups to learning – since informative rewards would be delayed.

Acknowledgments

We thank Andreas ten Pas for many helpful discussions while this project was underway and the reviewers for their valuable feedback. This work has been supported in part by the NSF through IIS-1427081, IIS-1724191, and IIS-1724257; NASA through NNX16AC48A and NNX13AQ85G; ONR through N000141410047; Amazon through an ARA to Platt; and Google through an FRA to Platt.

References

[1] 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*, 518(7540):529–533, 2015.

[2] T. Lozano-Pérez. Motion planning and the design of orienting devices for vibratory part feeders. *IEEE Journal Of Robotics And Automation*, 1986.

[3] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *The Int'l Journal of Robotics Research*, 34(4-5):705–724, 2015.

[4] 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*, pages 3406–3413, 2016.

[5] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. 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*, 13, 2017.

[6] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt. Grasp pose detection in point clouds. *The Int'l Journal of Robotics Research*, 36(13-14):1455–1473, 2017.

[7] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo, et al. 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.

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

[9] 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.

[10] 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.

[11] 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*, pages 173–184. Springer, 2016.

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

[13] 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*, volume 78, pages 334–343. Proceedings of Machine Learning Research, 2017.

[14] 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*, pages 5048–5058, 2017.

[15] N. Sprague and D. Ballard. Eye movements for reward maximization. In *Advances in neural information processing systems*, pages 1467–1474, 2004.

[16] H. Larochelle and G. Hinton. Learning to combine foveal glimpses with a third-order Boltzmann machine. In *Advances in neural information processing systems*, pages 1243–1251, 2010.

[17] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu. Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212, 2014.

[18] S. Chen, Y. Li, and N. M. Kwok. Active vision in robotic systems: A survey of recent developments. *The Int'l Journal of Robotics Research*, 30(11):1343–1377, 2011.

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

[20] G. Tesauro and G. Galperin. On-line policy improvement using Monte-Carlo search. In *Advances in Neural Information Processing Systems*, pages 1068–1074, 1997.

[21] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press Cambridge, second edition, 2018.

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

[23] 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.

[24] R. Diankov. *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Robotics Institute, Carnegie Mellon University, 2010.

[25] 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*, pages 5384–5391, 2012.

[26] S. Whitehead and D. Ballard. Learning to perceive and act by trial and error. *Machine Learning*, 7(1):45–83, 1991.