2018 IEEE International Conference on Robotics and Automation (ICRA)
May 21-25, 2018, Brisbane, Australia

Pick and Place Without Geometric Object Models

Marcus Gualtieri, Andreas ten Pas, and Robert Platt

Abstract— **We propose a novel formulation of robotic pick and place as a deep reinforcement learning (RL) problem. Whereas most deep RL approaches to robotic manipulation frame the problem in terms of low level states and actions, we propose a more abstract formulation. In this formulation, actions are target reach poses for the hand and states are a history of such reaches. We show this approach can solve a challenging class of pick-place and regrasping problems where the exact geometry of the objects to be handled is unknown. The only information our method requires is: 1) the sensor perception available to the robot at test time; 2) prior knowledge of the general class of objects for which the system was trained. We evaluate our method using objects belonging to two different categories, mugs and bottles, both in simulation and on real hardware. Results show a major improvement relative to a shape primitives baseline.**

I. Introduction

Traditional approaches to pick-place and regrasping require precise estimates of the shape and pose of all relevant objects [1], [2]. For example, consider the task of placing a mug on a saucer. To solve this problem using traditional techniques, it is necessary to plan a path in the combined space of the mug pose, the saucer pose, and the manipulator configuration. This requires the pose and shape of the mug to be fully observable. Unfortunately, even when the exact shape of the mug is known in advance, it can be hard to estimate the mug's pose precisely and track it during task execution. The problem is more difficult in the more realistic scenario where the exact shape of the mug is unknown.

Approaches based on deep RL are an alternative to the model based approach described above [3]. Recent work has shown that deep RL has the potential to alleviate some of the perceptual challenges in manipulation. For example, Levine *et al.* showed deep learning in conjunction with policy gradient RL can learn a control policy expressed directly in terms of sensed RGB images [4]. Not only does this eliminate the need to develop a separate perceptual process for estimating state, but it also simplifies the perceptual problem by enabling the system to focus on only the perceptual information relevant to the specific manipulation task to be solved. This, along with encoding actions using low level robot commands (such as motor torque or Cartesian motion commands [5], [4]), means the approach is quite flexible: a variety of different manipulation behaviors can be learned by varying only the reward function.

Unfortunately, deep RL approaches to robotics have an important weakness. While the convolutional layers of a deep network facilitate generalization over horizontal and vertical position in an image, they do not facilitate generalization over depth or in/out of plane orientation, i.e., the full 6-DOF pose space in which robots operate. This is a significant problem for robotics because deep RL methods must learn policies for many different relative poses between the robot and the objects in the world. Not only is this inefficient, but it detracts from the ability of the deep network to learn other things like policies that generalize well to novel object geometries.

Fig. 1. Pick-place problem considered in this paper. The robot must grasp and place an object in a desired pose without prior knowledge of its shape.

We propose a new method of structuring robotic pick-place and regrasping tasks as a deep RL problem, i.e., as a Markov decision process (MDP). Our key idea is to formulate the problem using reach actions where the set of target poses that can be reached using these actions is sampled on each time step. Each reach action is represented by a descriptor that encodes the volumetric appearance of the scene in the local vicinity of the sampled reach target. In order to formulate the MDP, we note our problem is actually a partially observable MDP (POMDP) where object shape and pose are hidden state and the images or point clouds produced by the sensors are the observations. In order to solve this problem as an MDP, we encode belief state as a short history of recently taken reach actions expressed using the volumetric descriptors used to encode the reach action.

As a result of these innovations, our method is able to learn policies that work for novel objects. For example, we show that our system can learn to grasp novel mugs (for which prior geometric models are not available) from a pile of clutter and place them upright on a shelf as in Figure 1. The same system can be trained to perform a similar task for other classes of objects, such as bottles, simply by retraining. Our system can also learn policies for performing complex regrasping operations in order to achieve a desired object pose. As far as we know, this is the first system described in the literature that has been demonstrated to accomplish the above without constructing or matching against geometric models of the specific objects involved.

College of Computer and Information Science, Northeastern University, Boston, Massachusetts, USA. This work has been supported in part by NSF through IIS-1427081 and ONR through N000141410047.

978-1-5386-3081-5/18/$31.00 ©2018 IEEE

II. Related Work

One early approach to manipulation of unknown objects is based on shape primitives. Miller *et al.* explored this in the context of grasp synthesis [6]. Others have extended these ideas to segmentation and manipulation problems [7], [8], [9]. These methods have difficulty when the objects are not approximately cylindrical or cuboid and when the objects cannot be easily segmented. Our method performs much better than a cylinder-based shape primitives method, even when the objects involved (bottles and mugs) are nearly cylindrical.

Another approach to manipulating unknown objects is to estimate the object shape from a recent history of sensor feedback. For example, Dragiev and Toussaint explore an approach that models the implicit shape of an object as a Gaussian process [10]. Mahler *et al.* do something similar for the purposes of grasping while incorporating tactile feedback [11]. These methods can run into trouble when there is not enough data to fit the implicit shape with high confidence. Both of the above approaches can be viewed as ways of estimating object shape and pose in order to facilitate traditional configuration space planning. The problem of object pose and shape estimation given various amounts of prior data remains an active area of research [12], [13], [14].

Recently, there has been much advancement in grasping novel objects. Bohg *et al.* provide a survey [15]. Most of these methods are trained in a supervised fashion to predict whether a grasp is stable or not. The present paper can be viewed as extending our prior work in grasp detection [16], [17] to pick-and-place and regrasping.

The approach nearest to ours is by Jiang *et al.* who propose a method for placing new objects in new place areas without explicitly estimating the object's pose [18]. Their placements are sampled instead of, as in our case, fixed. However, they do not jointly reason about the grasp and the place – the grasp is predefined. This is an important drawback because the type of placement that is desired often has implications on how the grasp should be performed. Their learning method also relies on segmenting the object, segmenting the place area, hand-picked features, and human annotation for place appropriateness.

RL has long been studied for use in robot control. Kober *et al.* survey robotics applications that use RL [19]. Since this survey, deep RL has become prominent in robotic manipulation [4], [5], [20]. These methods operate on the motor torque or Cartesian motion command level of the robot controller whereas ours operates at a higher level.

III. Problem Statement

We consider the problem of grasping, regrasping, and placing a novel object in a desired pose using a robotic arm equipped with a simple gripper. We assume this is a first-order kinematic system such that state is fully described by the geometry of the environment. Also, we assume the agent can act only by executing parameterized reach actions.

The problem can be expressed as an MDP as follows. Let $\Gamma \subseteq \mathbb{R}^3$ denote the portion of work space that is free of obstacles. For simplicity of exposition, suppose that it is known that the free space contains a finite set of N rigid body objects, O. Let Λ denote a parameter space that describes the space of all possible object shapes. Let $\xi(o) \in \Lambda \times SE(3)$ denote the shape and pose of object $o \in O$. Let $H \in SE(3)$ denote the current pose of the robot hand. The state of the system is fully described by the pose of the hand and the shape and pose of all N objects: $s = (H, \xi(o^1), \ldots, \xi(o^N)) \in S = SE(3) \times \{\Lambda \times SE(3)\}^N$.

We will assume the robot can act only by executing the following parameterized, pre-programmed actions: REACH-GRASP(T) where $T \in SE(3)$ and REACH-PLACE(t) where $t \in$ PLACE $\subset \mathbb{N}$ belongs to a discrete set of pre-programmed reach poses expressed relative to the robot base frame. The total set of available actions is then $A = SE(3) \cup$ PLACE.

Given a goal set $G \subset S$, we define a reward function to be 1 when a goal state is reached and 0 otherwise. The episode terminates either when a goal state is reached or after a maximum number of actions. Finally, we assume access to a simulator that models the effects of an action $a \in A$ taken from state $s \in S$. For stochastic systems, we assume the simulator draws a sample from the distribution of possible outcomes of an action. Given this formalization of the manipulation problem, we might express it as an MDP $\mathcal{M} = (S, A, \mathcal{T}, r)$ with state-action space $S \times A$, unknown but stationary transition dynamics $\mathcal{T}$, and reward function $r(s, a) = 1$ if $s \in G$ and $r(s, a) = 0$ otherwise.

A key assumption in this paper is object shape and pose are not observed directly and therefore the MDP defined above is not fully observed. Instead, it is only possible to observe a volumetric occupancy grid, $C(x) \in \{\text{OCCUPIED}, \text{FREE}, \text{UNKNOWN}\}$ where $x \in \bar{\Gamma} \subset \Gamma$ is a volumetric grid over Γ. We assume that C is populated based on depth sensor data obtained during a single time step. (As such, there may be a large number of UNKNOWN cells.) Given the above assumptions, the manipulation problem can be expressed as a POMDP $\mathcal{P} = (S, A, \mathcal{T}, r, C, \mathcal{O})$, where $\mathcal{O} : S \times A \times C \to \mathbb{R}$ is the observation probabilities, assumed to be unknown. The goal of this paper is to find policies that maximize the expected sum of discounted rewards over the episode, i.e., reach the goal state in a minimum number of actions.

IV. Approach

Solving the POMDP $\mathcal{P}$ using general purpose belief space solvers (e.g. [21]) is infeasible because the underlying MDP $\mathcal{M}$ is far too large to solve even if it were fully observed. Instead we propose what we call the *descriptor-based MDP* that encodes REACH-GRASP actions using a special type of descriptor and encodes belief state implicitly as a short history of states and actions.

A. The REACH-GRASP Descriptor

The REACH-GRASP descriptor is a key element of our state and action representation, based on the grasp descriptor developed in our prior work [16], [17]. It encodes the

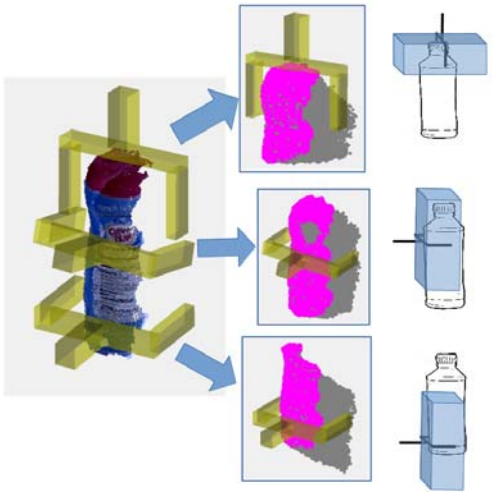

Fig. 2. Examples of the grasp descriptor for the three grasps shown on the left. The right column shows the cuboid associated with each grasp. The middle column shows the descriptor – the visible and occluded points contained within the cuboid.

relative pose between a robot hand and an object in terms of the portion of the volumetric occupancy grid in the vicinity of a prospective grasp. Let $\mathcal{C} = \{x \in \bar{\Gamma} | C(x) = \text{OCCUPIED}\}$ denote the voxelized point cloud corresponding to the occupancy grid C. Then, the REACH-GRASP descriptor at pose $T \in SE(3)$ is $D(\mathcal{C}, T) = trunc_\gamma(T\mathcal{C})$, where $T\mathcal{C}$ is the point cloud in the grasp reference frame, and where $trunc_\gamma(X)$ denotes the elements of X that lie within a cuboid centered at the origin with dimensions $\gamma = (\gamma_x, \gamma_y, \gamma_z)$. This is illustrated in Figure 2. The middle column shows the REACH-GRASP descriptors corresponding to the three grasps of the object shown on the left. A REACH-GRASP descriptor is encoded to the deep network as an image where the points are projected onto planes orthogonal to three different viewing directions and compiled into a single stacked image, $I(D)$, as described in [16], [17].

B. The Descriptor-Based MDP

Our key idea is to find goal-reaching solutions to the POMDP $\mathcal{P}$ by reformulating it as an MDP with descriptor-based states and actions. Specifically, we: 1) reparameterize the REACH-GRASP action using REACH-GRASP descriptors rather than 6-DOF poses; 2) redefine state as a history of the last two actions visited.

Action representation: Recall that the underlying MDP defines two types of actions: REACH-GRASP(T) where T denotes the pose of the grasp and REACH-PLACE(t) where $t \in$ PLACE and PLACE denotes a finite set of place poses. Since RL in a continuous action space can be challenging, we approximate the parameter space of REACH-GRASP by sampling. That is, we sample a set of m candidate poses for REACH-GRASP: $T_1, \ldots, T_m \in SE(3)$. In principle, we can use any sampling method. However, since REACH-GRASP is intended to reach toward *grasp* configurations, we use grasp pose detection (GPD) [16], [17] to generate the samples. Each of the candidate poses generated by GPD is predicted to be a pose from which closing the robot hand is expected to result in a grasp (although the grasp could be of any object).

Since we are sampling candidate parameters for REACH-GRASP, we need a way to encode these choices to the action-value function. Normally, in RL, the agent has access to a fixed set of action choices where each choice always results in the same distribution of outcomes. However, since we are now sampling actions, this is no longer the case, and we need to encode actions to the action-value function differently. In this paper, we encode each target pose candidate for REACH-GRASP by the corresponding REACH-GRASP descriptor, $D_i = D(\mathcal{C}, T_i), i \in [1, m]$. Essentially, the descriptor encodes each target pose candidate by the image describing what the point cloud nearby the target pose looks like. The total action set consists of the set of descriptors corresponding to sampled reach-grasps, REACH-GRASP$(D(\mathcal{C}, T_i)), i \in [1, m]$, and the discrete set of reach-places adopted from the underlying POMDP, REACH-PLACE$(i), i \in$ PLACE: $A = [1, m] \cup$ PLACE.

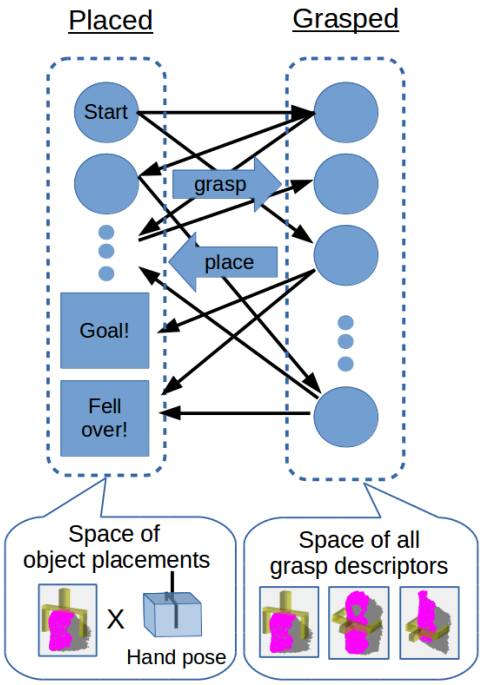

Fig. 3. The descriptor-based MDP. States on the right are those where an object is grasped. All other states are on the left.

State representation: We encode state as the history of the M most recent reach actions where REACH-GRASP actions are represented using the corresponding descriptors. In all of our experiments, $M \leq 2$. Figure 3 illustrates the resulting state-action space. The set of blue circles on the right labeled "Space of all grasp descriptors" denotes the set of states where an object has been grasped. This is a continuous space of states equal to the set of REACH-GRASP descriptors resulting from the most recent REACH-GRASP action, $\{trunc_\gamma(\mathcal{C}) | \mathcal{C} \subset \bar{\Gamma}\}$. The set of blue circles on the left labeled "Space of object placements" represents the set of states where an object has been placed somewhere in

the environment. These states are encoded as the history of the two most recent reach actions: the REACH-PLACE action taken on the last time step and the descriptor that encodes the REACH-GRASP action taken two time steps ago. All together, a state in this new MDP is a point in $S = \{trunc_\gamma(\mathcal{C}) | \mathcal{C} \subset \bar{\Gamma}\} \times$ PLACE. The state labeled "Goal!" in Figure 3 denotes an absorbing state where the object has been placed correctly, and the state labeled "Fell over!" denotes an absorbing state where the object has been placed incorrectly. When the agent reaches either of these states, it obtains a final reward and the episode ends.

Reward: Our agent obtains a reward of $+1$ when it reaches a placement state that satisfies the desired problem constraints, and otherwise, it obtains zero reward.

C. The Simulator

Deep RL requires such an enormous amount of experience that it is difficult to learn control policies on real robotic hardware without spending months or years in training [5], [4]. As a result, learning in simulation is basically a requirement. Fortunately, our formulation of the manipulation problem in terms of pre-programmed, parameterized actions simplifies the simulations. Instead of needing to simulate *arbitrary* contact interactions, we only need a mechanism for simulating the grasp that results from executing REACH-GRASP(T) and the object placement that results from executing REACH-PLACE(t). The former can be simulated by evaluating standard grasp quality metrics [22]. The later can be simulated by evaluating sufficient conditions to determine whether an object will fall over given the executed placement. Both are easy to evaluate in OpenRAVE [23], the simulator used in this work.

D. The Action-Value Function

We approximate the action-value function using the convolutional neural network (CNN) shown in Figure 4. The input is an encoding of the state and the action, and the output is a scalar, real value representing the value of that state-action pair. This structure is slightly different than that used in DQN [3] where the network has a number of outputs equal to the number of actions. Here, the fact that our MDP uses sampled reach actions means that we must take action as an input to the network. The action component of the input is comprised of the REACH-GRASP descriptor (encoded as a $60 \times 60 \times 12$ stacked image as described in Section IV-A) denoting the REACH-GRASP parameter and a one-hot vector denoting the REACH-PLACE parameter. When the agent selects REACH-GRASP, the grasp descriptor is populated and the place vector is set to zero. When a REACH-PLACE is selected, the grasp descriptor is set to zero and the place vector is populated.

The state component of the input is also comprised of a REACH-GRASP descriptor and a place vector. However, here these two parameters encode the recent history of actions taken (Section IV-B). After executing a grasp action, the grasp descriptor component of state is set to a stored version of the descriptor of the selected grasp and the place vector is set to zero. After executing a place action, the grasp descriptor retains the selected grasp and the place component is set to the just-executed place command, thereby implicitly encoding the resulting pose of the object following the place action. Each grasp image (both in the action input and the state input) is processed by a CNN similar to LeNet [24], except the output has 100 hidden nodes instead of 10. These outputs, together with the place information, are then concatenated and passed into two 60-unit fully connected, inner product (IP) layers, each followed by rectifier linear units (ReLU). After this there is one more inner product to produce the scalar output.

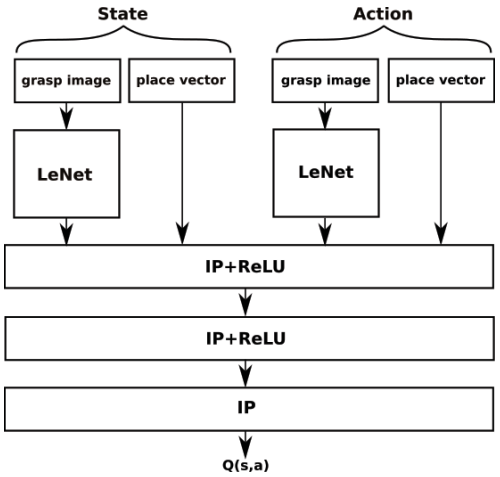

Fig. 4. Convolutional neural network architecture used to encode the action-value function, i.e., the Q-function.

E. Learning Algorithm

Our learning algorithm is shown in Algorithm 1. This is similar to standard DQN [3] with a couple of differences. First, we use a variant of Sarsa [25] rather than Q-learning because the large action branching factor makes the $\max_{a \in A} Q(s, a)$ in Q-learning expensive to evaluate and because Sarsa is known to perform slightly better on non-Markov problems. Second, we do not run a single stochastic gradient descent (SGD) step after each experience. Instead, we collect *nEpisodes* of experience before labeling the experience replay database using the most recent neural network weights. Every *nEpisodes* additional experiences, we run *nIterations* of SGD using Caffe [26]. For the experiments in this paper, the learning algorithm is run only in simulation; although it could be used to fine-tune the network weights on the actual hardware.

V. Experiments in Simulation

We performed experiments in simulation to evaluate how well our approach performs on pick-place and regrasping problems with novel objects. To do so, we obtained objects belonging to two different categories for experimentation: a set of 73 bottles and a set of 75 mugs – both in the form of mesh models from 3DNet [27]. Both object sets were partitioned into a 75%/25% train/test split.

Algorithm 1: Sarsa implementation for pick and place

```
for i ← 1 : nTrainingRounds do
    for j ← 1 : nEpisodes do
        Choose random object(s) from training set
        Place object(s) in a random configuration
        Sense point cloud C and detect grasps G
        s ← initial state
        a ← Pick(.) (ε-greedy)
        for t ← 1 : maxTime do
            (r, s') ← T(s, a)
            if a = Pick(.) then
                a' ← Place(.) (ε-greedy)
            else if a = Place(p)|p ∈ P_temp then
                Sense point cloud C and detect grasps G
                a' ← Pick(.) (ε-greedy)
            else if a = Place(p)|p ∈ P_final then
                a' ← null
            Add (s, a, r, s', a') to database
            if s' is terminal then break
            a ← a'; s ← s'
    Prune database if it is larger than maxExperiences
    Label each database entry (s, a) with r + γQ(s', a')
    Run Caffe for nIterations on database
```

A. Experimental Scenarios

There were three different experimental scenarios, *two-step-isolation*, *two-step-clutter*, and *multi-step-isolation*. In *two-step-isolation*, an object was selected at random from the training set and placed in a random pose in isolation on a tabletop. The goal condition was a right-side-up placement in a particular position on a table. In this scenario, the agent was only allowed to execute one grasp action followed by one place action (hence the "two-step" label). *Two-step-clutter* was the same as *two-step-isolation* except a set of seven objects was selected at random from the same object category and placed in random poses on a tabletop as if they had been physically dumped onto the table.

The *multi-step-isolation* scenario was like *two-step-isolation* except multiple picks/places were allowed for up to 10 actions (i.e., *maxTime*=10). Also, the goal condition was more restricted: the object needed to be placed upright, inside of a box rather than on a tabletop. Because the target pose was in a box, it became impossible to successfully reach it without grasping the object from the top before performing the final place (see Figure 7, bottom). Because the object could not always be grasped in the desired way initially, this additional constraint on the goal state sometimes forced the system to perform a regrasp in order to achieve the desired pose.

In all scenarios, point clouds were registered composites of two clouds taken from views above the object and 90° apart: a single point cloud performs worse, presumably because features relevant for determining object pose are unobserved. In simulation, we assumed picks always succeed, because the grasp detector was already trained to recognize stable grasps with high probability [16], [17] [1]. A place was considered successful only if the object was placed within 3 cm of the table and 20 degrees of the vertical in the desired pose.

B. Algorithm Variations

The algorithm was parameterized as follows. We used 70 training rounds (*nTrainingRounds* $= 70$ in Algorithm 1) for the two-step scenarios and 150 for the multi-step scenario. We used $1,000$ episodes per training round (*nEpisodes* $= 1,000$). For each training round we updated the CNN with $5,000$ iterations of SGD with a batch size of 32. *maxExperiences* was $25,000$ for the two-step scenarios and $50,000$ for the multi-step scenario. For each episode, bottles were randomly scaled in height between 10 and 20 cm. Mugs were randomly scaled in height between 6 and 12 cm. We linearly decreased the exploration factor ϵ from 100% down to 10% over the first 18 training rounds.

We compared the performance of Algorithm 1 on two different types of REACH-GRASP descriptors. In the *standard* variation, we used descriptors of the standard size ($10 \times 10 \times 20$ cm). In the *large-volume* (LV) variation, we used descriptors evaluated over a larger volume ($20 \times 20 \times 40$ cm) but with the same image resolution.

We also compared with two baselines. The first was the *random baseline*, where grasp and place actions were chosen uniformly at random. The second was the *shape primitives baseline*, where object pose was approximated by segmenting the point cloud and fitting a cylinder. Although it is generally challenging to fit a shape when the precise geometry of the object to be grasped is unknown, we hypothesized that it could be possible to obtain good pick-place success rates by fitting a cylinder and using simple heuristics to decide which end should be up. We implemented this as follows. First, we segment the scene into k clusters, using k-means ($k = 1$ for isolation and $k = 7$ for clutter). Then we fit a cylinder to the most isolated cluster using MLESAC [28]. We select the grasp most closely aligned with and nearest to the center of the fitted cylinder. The height of the placement action is determined based on the length of the fitted cylinder. The grasp up direction is chosen to be aligned with the cylinder half which contains fewer points. In order to get the shape primitive baseline to work, we had to remove points on the table plane from the point cloud. Although our learning methods do not require this and work nearly as well either way, we removed the table plane in all simulation experiments for consistency.

C. Results for the Two-Step Scenarios

Figure 5 shows learning curves for the two-step-isolation and two-step-clutter contingencies for bottles (left) and mugs (center) averaged over 10 runs. Table I shows place success rates when the test objects were used.

[1] It is possible to train grasping from the same reward signal, but this would require longer simulations. Empirically, this assumption did not lead to many grasp failures on the real robot (see Section VI).

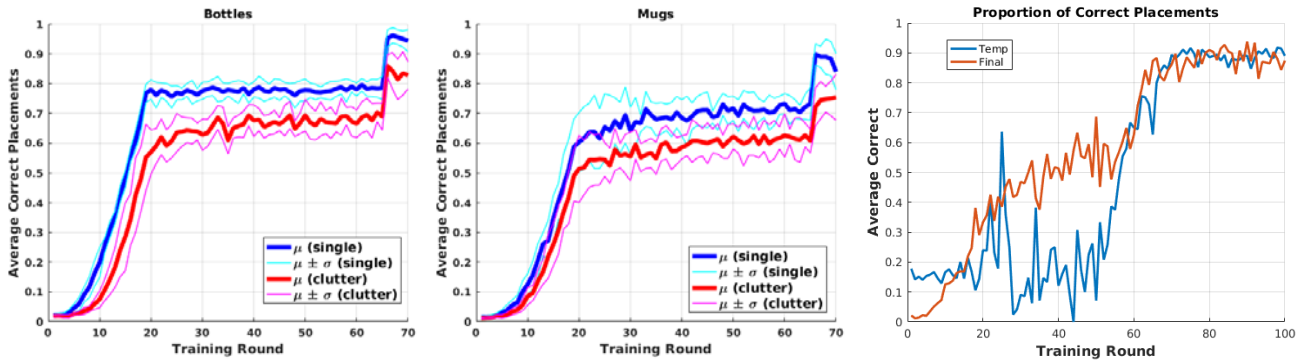

Fig. 5. **Left** and **center**. Average of 10 learning curves for the two-step scenario. The "training round" on the horizontal axis denotes the number of times Caffe had been called for a round of $5,000$ SGD iterations. The left plot is for bottles and the center for mugs. Blue denotes single objects and red denotes clutter. Curves for mean plus and minus standard deviation are shown in lighter colors. The sharp increase in performance during the last five rounds in each graph is caused by dropping the exploration factor (ϵ) from 10% to 0% during these rounds. **Right.** One multi-step realization with mugs in isolation. Red line: number of successful pick-place trials as a function of training round. Blue line: number of successful non-goal placements executed.

Trained With / Tested With	Bottle in Iso.	Bottles in Clut.
Isolation	1.00	0.67
Clutter	0.78	0.87
Isolation LV	0.99	0.47
Clutter LV	0.96	0.80
Shape Primitives Baseline	0.43	0.24
Random Baseline	0.02	0.02
Trained With / Tested With	Mug in Iso.	Mugs in Clut.
Isolation	0.84	0.60
Clutter	0.74	0.75
Isolation LV	0.91	0.40
Clutter LV	0.67	0.70
Shape Primitives Baseline	0.08	0.12
Random Baseline	0.02	0.02

TABLE I

AVERAGE CORRECT PLACEMENTS OVER 300 EPISODES FOR BOTTLES (TOP) AND MUGS (BOTTOM) USING TEST SET, AFTER TRAINING.

Several results are worth highlighting. First, our algorithm does very well with respect to the baselines. The random baseline (last row in Table I) succeeds only 2% of the time – suggesting that the problem is indeed challenging. The shape primitives baseline (where we localize objects by fitting cylinders) also does relatively poorly: it succeeds at most only 43% of the time for bottles and only 12% of the time for mugs. Second, place success rates are lower when objects are presented in clutter compared to isolation: 100% success versus 87% success rates for bottles; 84% versus 75% success for mugs. Also, if evaluation is to be in clutter (resp. isolation), then it helps to train in clutter (resp. isolation) as well: if trained only in isolation, then clutter success rates for bottles drops from 87% to 67%; clutter success rates for mugs drops from 75% to 60%. Also, using the LV descriptor can improve success rates in isolation (an increase of 84% to 91% for mugs), but hurts when evaluated in clutter: a decrease from 87% to 80% for bottles; a decrease from 75% to 70% for mugs. We suspect that this drop in performance reflects the fact that in clutter, the large receptive field of the LV descriptor encompasses "distracting" information created by other objects nearby the target object (remember we do not use segmentation) [29].

D. Results for the Multi-Step Scenario

Training for the multi-step-isolation scenario is the same as it was in the two-step scenario except we increased the number of training rounds because the longer policies took longer to learn. We only performed this experiment using mugs (not bottles) because it was difficult for our system to grasp many of the bottles in our dataset from the top. Figure 5 shows the number of successful non-goal and goal placements as a function of training round [2]. Initially, the system does not make much use of its ability to perform intermediate placements in order to achieve the desired goal placement, i.e., to pick up the mug, put it down, and then pick it up a second time in a different way. This is evidenced by the low values for non-goal placements (the blue line) prior to round 60. However, after round 60, the system learns the value of the non-goal placement, thereby enabling it to increase its final placement success rate to is maximum value (around 90%). Essentially, the agent learns to perform a non-goal placement when the mug cannot immediately be grasped from the top or if the orientation of the mug cannot be determined from the sensor perception. After learning is complete, we obtain an 84% pick and place success rate averaged over 300 test set trials.

VI. EXPERIMENTS ON A REAL ROBOT

We evaluated the same three scenarios on a real robot: *two-step-isolation*, *two-step-clutter*, and *multi-step-isolation*. As before, the two step scenarios were evaluated for both bottles and mugs, and the multi-step scenario was evaluated for only mugs. All training was done in simulation, and fixed CNN weights were used on the real robot.

The experiments were performed by a UR5 robot with 6 DOFs, equipped with a Robotiq parallel-jaw gripper and a wrist-mounted Structure depth sensor (Figure 7). Two sensor views were always taken from fixed poses, 90° apart. The object set included 7 bottles and 6 mugs, as shown in

[2]Non-goal placements were considered successful if the object was 3 cm or less above the table. Any orientation was allowed. Unsuccessful non-goal placements terminate the episode.

Fig. 6. The seven novel bottles and six novel mugs used to evaluate our approach in the robot experiments.

Figure 6. We used only objects that fit into the gripper, would not shatter when dropped, and had a non-reflective surface visible to our depth sensor. Some of the lighter bottles were partially filled so small disturbances (e.g., sticking to fingers) would not cause a failure. Figure 7 shows several examples of our two-step scenario for bottles presented in clutter.

Unlike in simulation, the UR5 requires an IK solution and motion plan for any grasp or place pose it plans to reach to. For grasps, GPD returns many grasp choices. We sort these by their pick-place Q-values in descending order and select the first reachable grasp. For places, the horizontal position on the shelf and orientation about the vertical (gravity) axis do not affect object uprightness or the height of the object. Thus, these variables were chosen to suit reachability.

After testing some trials on the UR5, we found we needed to adjust a couple of training/simulation parameters. First, we changed the conditions for a successful place in simulation because, during our initial experiments, we found the policy sometimes selected placements that caused the objects to fall over. As a result, we adjusted the maximum place height in simulation from 3 cm to 2 cm and changed the reward function to fall off exponentially from $+1$ for altitudes higher than 2 cm. Second, we raised the acceptance threshold [3] used by our grasp detector, GPD [16], [17].

	1 Bottle	7 Bottles	1 Mug	6 Mugs	Regrasp
Grasp	0.99	0.97	0.96	0.93	0.94
FinalPlace	0.98	0.94	0.93	0.87	1.00
TempPlace	-	-	-	-	1.00
EntireTask	0.97	0.92	0.90	0.80	0.68
n Trials	112	107	96	96	72
UpsideDown	0	4	5	10	0
Sideways	0	0	0	2	0
FellOver	2	2	1	0	0
$t > 10$	-	-	-	-	12

TABLE II

(TOP) SUCCESS RATES FOR GRASP, TEMPORARY PLACE, FINAL PLACE, AND ENTIRE TASK. (BOTTOM) PLACEMENT ERROR COUNTS BY TYPE. RESULTS ARE AVERAGED OVER THE NUMBER OF TRIALS (MIDDLE).

Table II summarizes the results from our robot experiments. We performed 483 pick and place trials over five different scenarios. Column one of Table II shows results for pick and place for a single bottle presented in isolation averaged over all bottles in the seven-bottle set. Out of 112 trials, 99% of the grasps were successful and 98% of the placements were successful, resulting in a complete task pick/place success rate of 97%. Column two shows similar results for the bottles-in-clutter scenario, and columns three and four include results for the same experiments with mugs. Finally, column five summarizes results from the multi-step-isolation scenario for mugs: overall, our method succeeded in placing the mug upright into the box 68% of the time. The temporary place success is perfect because a temporary placement only fails if the mug is so high it rolls away after dropped or too low it is pushed into the table, neither of which ever happened after 72 trials. The final placement is perfect because it always did get the orientation right (for all 72 trials that got far enough to reach the final placement), and it is hard for the mug to fall over in the box. The multi-step scenario has low task success rate because 12 trials failed to perform the final place after 10 time steps. Perhaps this is due to lower Q-function values on the real system (due to domain transfer issues), causing the robot to never become confident enough with its given state information to perform the final place.

Our experimental results are interesting for several reasons beyond demonstrating that the method can work. First, we noticed consistently lower place performance for the mug category relative to the bottle category. The reason for this is there is more perceptual ambiguity involved in determining the orientation of a mug compared to that of a bottle. In order to decide which end of a mug is "up", it is necessary for the sensor to view into at least one end of the mug. Second, the robot had trouble completing the multi-step task in a reasonable number of steps with the real hardware compared with simulation. This may be because fewer grasps are available on the real robot versus the simulated robot due to collision modelling. Another unexpected result was our learned policies typically prefer particular types of grasps, e.g., to grasp bottles near the bottom (see Figure 7). We suspect this is a result of the link between the location of a selected grasp and the grasp descriptor used to represent state. In order to increase the likelihood that the agent will make high-reward decisions in the future, it selects a grasp descriptor that enables it to easily determine the pose of the object. In the case of bottles, descriptors near the base of the bottle best enable it to determine which end is "up".

VII. CONCLUSION AND FUTURE WORK

This paper proposes a new way of structuring robotic pick-place and regrasping tasks as a deep RL problem. Importantly, our learned policies can place objects very accurately without using shape primitives or attempting to model object geometry in any way. Our key insight is to encode a sampled set of end-to-end reaching actions using descriptors that encode the geometry of the reach target pose. We encode state as a history of recent actions and observations. The resulting policies, which are learned in simulation, simultaneously perceive relevant features in the environment and plan the appropriate grasp and place actions in order to achieve task goals. Our experiments show that the method consistently outperforms a baseline method based on shape primitives.

[3]GPD outputs a machine-learned probability of a stable (i.e., force closure) grasp. The *threshold* is the grasp stability probability above which grasps are accepted.

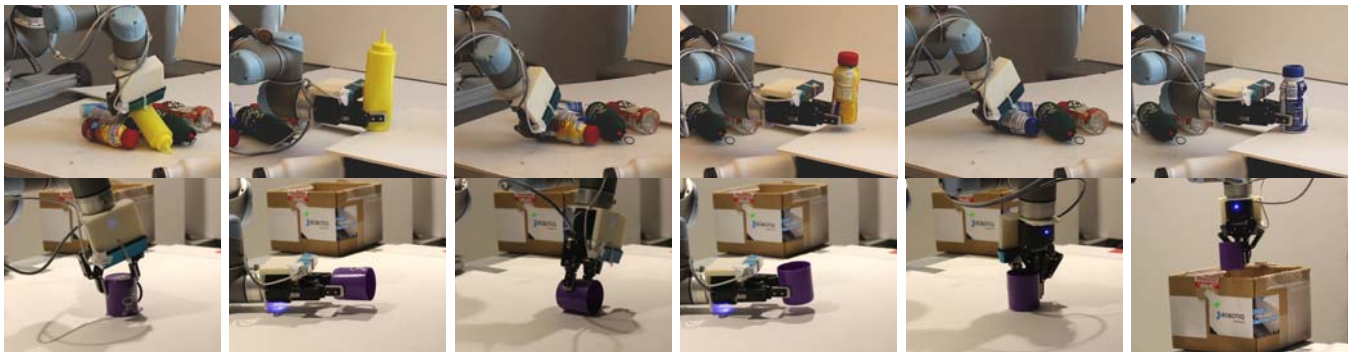

Fig. 7. **Top**. Two-step-clutter scenario for bottles. First three objects are placed right-side-up and without falling over. **Bottom**. Multi-step-isolation scenario for a mug. The mug is initially upside-down, so must be flipped around before it can be put upright into the box.

For future work, we plan to generalize the descriptor-based MDP in two ways. First, place poses could be sampled from a continuous, 6-DOF space, as grasps are. To do this we would develop a special purpose place detector in the same way GPD is a grasp detector. Second, the system should be able to work with a more diverse set of objects, e.g., kitchen items. This may require a CNN with more capacity and longer training time, motivating innovations to speed up the learning in simulation.

REFERENCES

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

[2] M. Mason, "The mechanics of manipulation," in *IEEE Int'l Conf. on Robotics and Automation*, vol. 2, 1985, pp. 544–548.

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

[5] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[6] A. Miller, S. Knoop, H. Christensen, and P. Allen, "Automatic grasp planning using shape primitives," in *IEEE Int'l Conf. on Robotics and Automation*, vol. 2, 2003, pp. 1824–1829.

[7] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Close-range scene segmentation and reconstruction of 3d point cloud maps for mobile manipulation in domestic environments," in *IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, 2009, pp. 1–6.

[8] T. Mörwald, J. Prankl, A. Richtsfeld, M. Zillich, and M. Vincze, "Blort-the blocks world robotic vision toolbox," in *ICRA Workshop: Best Practice in 3D Perception and Modeling for Mobile Manipulation*, 2010.

[9] K. Harada, K. Nagata, T. Tsuji, N. Yamanobe, A. Nakamura, and Y. Kawai, "Probabilistic approach for object bin picking approximated by cylinders," in *IEEE Int'l Conf. on Robotics and Automation*, 2013, pp. 3742–3747.

[10] S. Dragiev, M. Toussaint, and M. Gienger, "Gaussian process implicit surfaces for shape estimation and grasping," in *IEEE Int'l Conf. on Robotics and Automation*, 2011, pp. 2845–2850.

[11] J. Mahler, S. Patil, B. Kehoe, J. Van Den Berg, M. Ciocarlie, P. Abbeel, and K. Goldberg, "Gp-gpis-opt: Grasp planning with shape uncertainty using gaussian process implicit surfaces and sequential convex programming," in *IEEE Int'l Conf. on Robotics and Automation*, 2015, pp. 4919–4926.

[12] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit, "Gradient response maps for real-time detection of textureless objects," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 34, no. 5, pp. 876–888, 2012.

[13] K. Pauwels and D. Kragic, "Simtrack: A simulation-based framework for scalable real-time object pose detection and tracking," in *EEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, 2015, pp. 1300–1307.

[14] P. Wohlhart and V. Lepetit, "Learning descriptors for object recognition and 3d pose estimation," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2015, pp. 3109–3118.

[15] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesisa survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2014.

[16] M. Gualtieri, A. ten Pas, K. Saenko, and R. Platt, "High precision grasp pose detection in dense clutter," in *IEEE Int'l Conf. on Intelligent Robots and Systems*, 2016.

[17] 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.

[18] 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.

[19] 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.

[20] U. Viereck, A. ten Pas, K. Saenko, and R. Platt, "Learning a visuomotor controller for real world robotic grasping using easily simulated depth images," in *Proceedings of the Conf. on Robot Learning*, 2017.

[21] H. Kurniawati, D. Hsu, and W. S. Lee, "SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces," in *Robotics: Science and systems*, vol. 4, 2008.

[22] R. Murray, Z. Li, and S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[23] R. Diankov, "Automated construction of robotic manipulation programs," Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, 2010.

[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

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

[26] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *ACM Int'l Conf. on Multimedia*, 2014, pp. 675–678.

[27] 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.

[28] P. Torr and A. Zisserman, "Mlesac: A new robust estimator with application to estimating image geometry," *Computer Vision and Image Understanding*, vol. 78, no. 1, pp. 138–156, 2000.

[29] 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.