Category Level Pick and Place Using Deep Reinforcement Learning

Marcus Gualtieri, Andreas ten Pas, and Robert Platt
College of Computer and Information Science, Northeastern University

***Abstract*— General purpose robots need the ability to perceive and manipulate novel objects. In this work, we focus on robotic pick and place where the object category (e.g. bottles) is known but a model of the exact object instance is unavailable. We formulate the problem as a Markov Decision Process (MDP) and solve it with reinforcement learning (RL). The action-value function (Q-function) is approximated with a deep convolutional neural network (CNN). We train the robot in simulation, and we show the robot can successfully pick and place novel objects in both simulation and on a UR5.**

I. Introduction

We imagine that, in the future, general-purpose robots will provide us assistance in cluttered, human environments. A key capability that such robots would need is the ability to perceive and manipulate novel objects – novel in the sense that the robot has seen the same type before but not the exact instance. Previously we have worked towards this goal by training a robot to grasp novel objects in clutter [4]. The next step is to do something useful with the grasped object, such as place it down in a gentle and organized fashion. Typically robots perform this pick and place task by first estimating the object's pose and placing it in a known, target pose [5], [14], [21]. Object pose estimation in clutter for object category is a difficult, ongoing challenge for researchers [23].

We approach the problem not by explicitly finding object pose, but by allowing the robot learn correct grasp and place decisions through trial-and-error. This is an RL problem: the robot must learn a policy for grasp and place decisions through a reward signal, where the robot receives positive reward for a correct placement.

Success for our approach involves three key points. The first is we can make use of recent advances in grasp detection and motion planning. Assuming these capabilities simplifies the learning problem. The second is how sensor data is represented to the robot. We follow the recent trend in using deep learning for function approximation in RL [18]. Thus, we need to represent the sensor information to the neural network in some fashion. Mnih *et al.* showed that, in cluttered scenarios, it may be helpful to restrict the perceptions input into the network, i.e. what is called attention focus in humans [17]. We follow a similar idea and represent a grasp as local geometric information. The object surface geometry local to a grasp essentially encodes the object pose in the hand of the robot. Finally, we would like to train the robot in simulation so as to save time and wear on the robot hardware. This introduces a difficult problem of transferring knowledge obtained in simulation to a very different, physical environment, but the cost benefits of training in simulation are too substantial to ignore.

Fig. 1. Illustration of pick and place scenario that requires multiple picks and places. (Left) Object is grasped in initial configuration. (Center) First place is to make the mug upright so it can be grasped from the top. (Right) Places mug deep in box – the gripper must be vertical to box.

We demonstrate the utility of our approach in both simulation and on a real robot. In simulation the robot can correctly place bottles in clutter 87% of the time. On the actual robot this drops to about 73% (97% grasp success rate and 76% place success rate). Although this is still below human performance, we expect this result to scale up with simulation fidelity and continued learning in the physical environment. A video of our approach running on the UR5 robot can be found here: http://www.ccs.neu.edu/home/atp/bottles_clutter.mp4.

II. Problem Definition

Given a sensor perception of a scene, the robot's task is to grasp an object and place it gently in a target configuration. The object category is known, but the robot may have never seen the specific object instance before. The sensor perception we consider here is a colorless point cloud obtained from a depth sensor. There may be constraints on the robot arm configuration during placement. For example, it may be the gripper during the place needs to be vertical to a box so the object and gripper can fit into the box (see Figure 1). What counts as a "gentle" and "target" placement depends on context and will be made precise in Sections IV and V.

In order to simplify this problem we make use of recent advances in robotic grasp perception. Specifically, we assume that there is a black-box which converts the point cloud data into a sampling of grasps which are likely to be geometrically stable grasps. A grasp detection black-box which can predict stable grasps on a wide range of objects is not an unreasonable assumption as several have become available [4], [12], [24].

We also assume motion planning and control of the robot can be handled by a motion planning black-box. For example, we use Trajopt and OpenRAVE [2], [25]. Thus, the robot only needs to know the target grasp or the target

placement, and a motion plan can be generated to move the arm there. The problem then simplifies to that of selecting a grasp from the set provided by the grasp detector and selecting a placement.

We formulate the problem as an MDP. An MDP is a tuple (S, A, R, T, γ) of states S, actions A, rewards R, a transition function T, and a discount factor γ [27]. For the pick and place problem each of these components is described as follows:

- $s \in S$: (G, P, g, p) where G is a sampling of grasps obtained from a grasp detection sensor, P is a fixed set of allowed placements (hand configurations), g is a grasp that was selected in a previous time step, if any, and p is a placement where the robot believes the object has been placed in a previous time step, if any.
- $a \in A$: (g, p) where $g \in G$ is the grasp the robot picks and $p \in P$ is the hand configuration the robot uses to place the object. We alternate between *Pick* and *Place* actions: at the first time step pick actions are available, at the second time step place actions are available, and so on.
- $r \in R$: 1 if the object has been gently placed in a target configuration (while meeting any placement constraints) and 0 otherwise.
- T: An initially unknown, stochastic function of the robot's actions and the environment. Thus the problem is *model-free* – the robot must learn the transition dynamics from trial-and-error [27]. Note that T will be different for simulation versus for a physical robot.
- γ: It may be useful to discount rewards distant in the future, so the robot will learn to place the object quickly, all other things being equal.

We distinguish two types of placements: *temporary* and *final*. Final placements are placements that meet all the constraints required by the problem (see Figure 1). Temporary placements may or may not meet the constraints (e.g. Figure 1, center). If a temporary place is selected and executed successfully, the robot will need to re-grasp and re-place the object until it achieves a final place. The purpose of the temporary placement is to reorient the object to detect better grasps or to improve the robot's confidence about the object's pose.

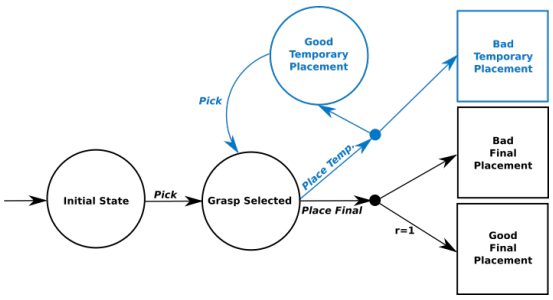

Fig. 2. Summary diagram for our MDP. Terminal states are square. The reward on each arrow is 0 unless indicated otherwise. The blue section is only needed if the problem requires re-grasping and re-placing.

A summary diagram of our MDP is shown in Figure 2. It is a summary diagram because the states shown in the diagram are actually large collections of states. For example, the "Grasp Selected" state covers every possible grasp that could be selected. The part of the diagram shown in blue is only required if temporary placements are available.

Given the MDP formulation, the goal of the pick and place problem is to find a policy that maximizes the long-term, discounted reward (i.e. maximize Equation 1). This is known as an RL problem [27]. The task is *episodic* because each trial always ends in a finite number of time steps [27].

$$G_t \equiv r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{T-1} r_T \qquad (1)$$

III. Approach

A. Training Algorithm

Our training algorithm for pick and place is shown in Algorithm 1. It is an instance of Sarsa, which is a method for estimating action values (Q-values) and for taking actions that maximize G_t (Equation 1) [26]. The outer loop of the training algorithm is one training round, i.e. one round of adding experiences to the database. After obtaining new experiences from a training round, we prune experiences if the database is too large, starting with the oldest experiences. We then label the data using the last network weights. We then update the weights using a neural network optimization framework, Caffe [7].

for $i \leftarrow 1 : nTrainingRounds$ **do**
  for $j \leftarrow 1 : nEpisodes$ **do**
    Choose random object from training set
    Place object in a random configuration
    Sense point cloud C and detect grasps G
    $s \leftarrow$ initial state
    $a \leftarrow$ `Pick(.)` (ϵ-greedy)
    for $t \leftarrow 1 : maxTime$ **do**
      $(r', s') \leftarrow T(s, a)$
      if $a =$ `Pick(.)` **then**
        $a' \leftarrow$ `Place(.)` (ϵ-greedy)
      else if $a =$ `Place`$(p) | p \in P_{temp}$ **then**
        Sense point cloud C and detect grasps G
        $a' \leftarrow$ `Pick(.)` (ϵ-greedy)
      else if $a =$ `Place`$(p) | p \in P_{final}$ **then**
        $a' \leftarrow null$
    Add (s, a, r', s', a') to database
    if s' *is terminal* **then** break
  Prune database if it is larger than *maxExperiences*
  Label database using $Q(s, a) \leftarrow r' + Q(s', a')$
  Run Caffe for *nIterations* on database

Algorithm 1: Our Sarsa implementation for pick and place

For each training round, we run some number of episodes. The algorithm enforces the transitions shown in Figure 2. As shown in the figure, actions alternate between *Pick* and *Place*, and the episode ends if the place was a final placement

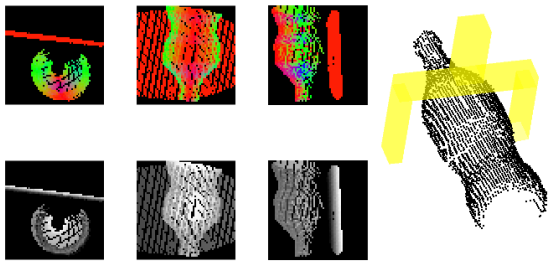

Fig. 3. Grasp images representing point cloud local to gripper. Top: surface normals (r,g,b) = (x,y,z). Bottom: height maps of the points. This grasp is nearly centered on the bottle and upside-down.

or a harsh temporary placement. The transition function T propagates the simulator forward one step and evaluates the reward. In this work we always train in simulation, but it would be straightforward to implement this algorithm on the robot, where the transition calls the motion planner and robot control functions and a human annotator decides if the place was gentle enough to merit a reward.

B. Neural Network for Q-Function Approximation

We use a deep CNN to approximate the Q-function. 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. Let's first describe how the action is encoded. The actions at any time step can either be *Pick* or *Place*. For the *Pick* action, a grasp is input into the network and the place vector is all zeros. For the *Place* action, a place is selected and the input grasp image is all zeros. The grasp is encoded to the network as a 12-channel image, similar to the one used in our prior work on grasp detection [4]. An example grasp image is shown in Figure 3. The place vector is a 1-hot binary vector indicating which place was selected from a fixed, discrete set $P = [P_{temp}, P_{final}]$.

Each of the 12 channels in the grasp image is a parallel projection of points in the point cloud from a standard basis axis in the reference frame of the grasp. This way, the grasp image encodes information about the orientation of the grasp relative to the object. The points used in the projection do not include the entire point cloud. They only include points within a small rectangular volume about the grasp. This is important in cluttered scenes where other objects will be in the full point cloud, making it necessary for the network to learn what parts of the image are relevant. Mnih *et al.* empirically showed that a small image focused on the object of interest in a larger image can improve neural network accuracy [17]. They refer to this approach as an *attention model* and the small window within the larger image as a *glimpse*. We hypothesize here that a large grasp image will have trouble ignoring clutter and a small window should be sufficient to infer the essential information for pick and place, i.e. the pose of the object in the hand.

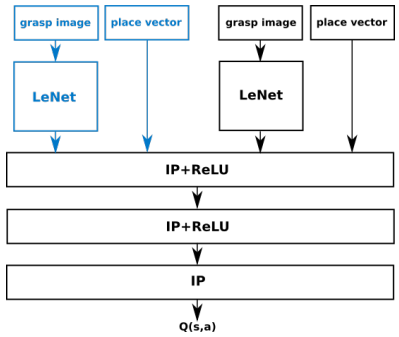

Fig. 4. CNN architecture used for the pick and place task. The blue portion is only necessary if temporary placements are allowed.

The encoding of the state to the network is very similar to the encoding of the action: a grasp image and a 1-hot place vector. At the first time step these are both zeros. Once a *Pick* action has been performed, the image becomes the encoding of the selected grasp. Once a *Place* action has been performed, the place vector gets populated with the believed, current object placement. These two pieces of information together should inform the robot of the pose of the object, assuming the place action was successful. Note that we never estimate the object pose and provide this as input to the Q-function. The only inputs are grasp images and placements.

A diagram of the CNN architecture is shown in Figure 4. The inputs are shown at the top and the output, the Q-value, at the bottom. Each image goes into a CNN component exactly the same as LeNet [11] except the output has 100 digits 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 output. If the pick and place problem does not include temporary places (i.e. there is no re-grasping), then only one image and one place vector needs to be input to the network. This is because not enough time steps are reached to observe and make use of the current object pose (see the blue region in Figure 4).

IV. Simulation Experiments

We trained and evaluated the performance of an RL robot in simulation. We first evaluated the algorithms for a scenario where all placements were final placements, i.e. $P = P_{final}$. In this case every episode has exactly two time-steps – a *Pick* time step and a *Place* time step. Let's call this the *two-step* scenario. Next we evaluated a scenario where temporary placements were allowed. Let's call this the *re-grasp* scenario because the robot may grasp and place the object multiple times. In the re-grasp scenario we designated all of the side placements as temporary and all of the top placements as final. In both scenarios the robot receives a reward of 1 at the end of an episode for a correct, final placement. Otherwise, the robot receives 0 reward. For a final placement to be considered correct, the object up-axis must be nearly aligned with the world up-axis, the object cannot be too high from the table, and the object cannot be in collision with anything, including the robot's fingers. The requirements are the same for a temporary placement except the object can have any orientation.

The simulation environment was OpenRAVE [2]. The robot model was just a floating gripper, which emphasizes our assumption from Section II that a motion planning black-box can be used to move the gripper to the desired pose. We also assumed grasps will succeed unless the fingers penetrate the object. Two point clouds were sensed in all scenarios, taken from views above the object and 90° apart. The table plane was removed from the point cloud for all of these simulations because otherwise, the cylinders baseline (described below) does not work.

We separately evaluated the robot with two different object categories: mugs and bottles from 3DNet [30]. 25% of the objects from each category were randomly selected for the test set. The object scale was sampled uniformly from a fixed range. Single objects were placed in the center of the workspace in a semi-random (collision-free) orientation. In clutter, there were 7 objects of the same class with a the orientation sampled uniformly at random and the position sampled from a Gaussian.

Training for the two-step scenario proceeded according to Algorithm 1. Results for the two-step scenario are summarized at the tops of Tables I and II. Clutter is more challenging than single objects and mugs are more challenging than bottles. We attribute the difference in mugs to: 1) if no points in the point cloud cover the bottom of the mug, it may be impossible to distinguish which side is upright, and 2) if no points cover the handle, the robot may choose a grasp where the fingers penetrate the handle.

We compared our approach to a similar approach with a larger grasp volume (LV). This is where the volume generating the grasp images is doubled in each dimension. While the larger volume will contain more information about the object, we expect the LV approach to perform worse in clutter due to the attention focus results of Mnih *et al.* [17]. We do in fact observe this performance drop for both objects in clutter, irregardless of if the network was trained in clutter.

Next we compared to a cylinder-fitting approach. Since bottles are nearly cylindrical one may ask if this problem can be solved using a more traditional model-fitting technique. In this experiment we first segment the scene into k clusters, using k-means [15]. $k = 1$ for single objects (i.e. no clustering) and $k = 7$ for clutter. Then a cylinder is fit to the object using Matlab's cylinder fitting algorithm (MLESAC) [28]. A grasp aligned with and near to the center of the cylinder is selected, and the placement is decided by the cylinder's length. The grasp up direction is chosen to be aligned with the cylinder half which contains fewer points. This is because objects are rarely top-heavy, so the bottom should be larger and have more points. Note this cylinder baseline makes use of domain knowledge that is learned automatically by the RL agent. This includes the fact that bottles are typically larger on the bottom, the approximate shape of the object (i.e. cylindrical with the least-curvature axis needing to be placed vertically), and the number of objects for k-means. Also, cylinder fitting does not work at all unless the table plane is removed (it gets the cylinder length wrong), whereas the RL agent is not crippled if the table is included. On the other hand, the cylinders method does not require several hours of training.

The cylinders method places objects correctly less than half the time. The "up heuristic" does not work well because, in many views, more of the top is visible than the bottom. Also, cylinder fitting does not get the length of bottles correctly if the neck is much thinner than the bottom. With oddly shaped bottles (e.g. mouthwash and bourbon), the cylinder radius is overestimated. Cylinders fit much more poorly in clutter because of occlusions and errors in clustering. Unsurprisingly, mugs do worse than bottles because handles make the object less cylindrical.

The last baseline is where the robot chooses grasps and places uniformly at random. Of course this performs poorly, but it is an interesting result because it gauges the problem difficulty. Because the problem is model-free, the robot must learn the values of grasps and places from trial-and-error, starting with random actions that only succeed 2% of the time.

Training for the re-grasping scenario proceeds in the same way as the two-step scenario except about double the number of training rounds is required. We only used mugs for this analysis because our grasp detection sensor did not find many grasps on the tops of bottles. Results for the re-grasping scenario are shown in Table III.

TABLE I

AVERAGE CORRECT PLACEMENTS OVER 300 EPISODES FOR BOTTLES.

Method / # Bottles	1 (Train)	1 (Test)	7 (Train)	7 (Test)
1 Bottle	0.94	**1.00**	0.54	0.67
7 Bottles	0.74	0.78	**0.78**	**0.87**
1 Bottle LV	**0.95**	0.99	0.32	0.47
7 Bottles LV	0.88	0.96	0.66	0.80
Cylinders	0.41	0.43	0.23	0.24
Random	0.02	0.02	0.03	0.02

TABLE II

AVERAGE CORRECT PLACEMENTS OVER 300 EPISODES FOR MUGS.

Method / # Mugs	1 (Train)	1 (Test)	7 (Train)	7 (Test)
1 Mug	**0.92**	0.84	0.62	0.60
7 Mugs	0.74	0.74	**0.71**	**0.75**
1 Mug LV	0.90	**0.91**	0.43	0.40
7 Mugs LV	0.67	0.67	0.70	0.70
Cylinders	0.14	0.08	0.09	0.12
Random	0.01	0.02	0.02	0.02

TABLE III

RE-GRASPING RESULTS FOR SINGLE MUGS OVER 300 EPISODES.

Metric / Tested With	1 Mug (Train)	1 Mug (Test)
Correct Placements	0.85	0.84
Number of Placements	1.85	1.85

V. ROBOT EXPERIMENTS

We ran a series of experiments to evaluate our method on a UR5 robot. The weights learned in simulation were kept fixed. As with the tests in simulation, we separately evaluated the system with single objects and with clutter for

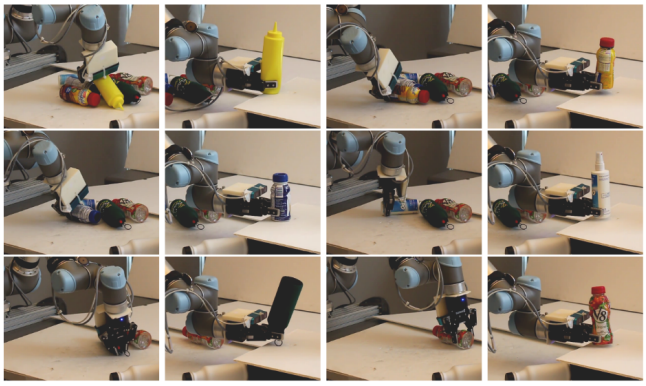

Fig. 5. Sequence of picks and places for bottles in clutter. The bottle with the green cover was placed upside-down, but the rest (not counting the one that rolled out) were successful.

both bottles and mugs. Unlike in simulation, grasps were only considered successful if the object was transported to the placement area. Placements were considered successful if, after the fingers opened, the object was placed upright on the target placement area.

The experiments were performed by a UR5 robot, equipped with a Robotiq parallel-jaw gripper and a wrist-mounted Structure depth sensor. Two sensor views were always taken from fixed poses, 90° apart. The object set included 7 bottles and 6 mugs. The objects were required to fit into the gripper and be mostly visible to the sensor. Some of the lighter bottles were partially filled so small disturbances (e.g. sticking to fingers) would not cause a failure. An example of a two-step clutter experiment with bottles is shown in Figure 5.

The success rates and failure counts for robot experiments are shown in Table IV. As we saw in simulation, mugs were more challenging to place than bottles, partly for the same reasons, and also because our grasp detector finds fewer grasps on mugs. Additionally, the point cloud in reality appears to have more noise and less visibility than the simulated sensor at high incidence angles.

TABLE IV

(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).

	1 Bottle	7 Bottles	1 Mug	6 Mugs	Re-grasp
Grasp	0.96	0.97	0.88	0.80	0.97
Final Place	0.89	0.76	0.80	0.61	0.76
Temp. Place	-	-	-	-	0.77
Entire Task	0.84	0.73	0.70	0.48	0.55
# of Trials	112	97	96	91	72
Upside-down	5	19	5	4	0
Sideways	0	0	7	18	9
Fell Over	2	3	1	2	0

VI. RELATED WORK

The traditional approach to robotic pick and place is to first estimate the object's pose and then to plan motions that re-orient the object to a desired, target pose. For example, Nguyen *et al.* estimate the object's pose, grasp a part of the object that known to be graspable, and then reorient and place the object [21]. This procedure can repeat multiple times until the object reaches the target pose. Lin *et al.* focus on the perception part of the pick and place problem where objects are presented in light clutter [14]. They also view determining the object's pose as the necessary first step. Harada *et al.* use a cylinder fitting method to estimate the object's pose to pick from a dense pile of bananas [5]. Their cylinder fitting mechanism is admittedly more sophisticated than our cylinders baseline as they use a probabilistic model to adjust the cylinder size and to rank the grasps. However, each of these examples either only works on the exact object used in training or generalize to objects very similar in appearance. Pose estimation for novel objects in a given category is an active area of research [6], [22], [29].

On the other hand, Jiang *et al.* show how to place new objects in new place areas without explicitly estimating the object's pose [8]. 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. Also their learning method relies on segmenting the object, segmenting the place area, hand-picked features, and human annotation for place appropriateness.

The idea of assigning task semantics to grasps is related to the our pick and place problem. Dang and Allen introduced the term "semantic grasp" and showed how to learn semantic grasps given object category [1]. Myers *et al.* and Nguyen *et al.* more recently proposed methods for detecting object affordances [19], [20]. With these methods the robot learns semantic grasps in a supervised fashion, whereas with our approach the robot learns from trial-and-error and a reward signal. Also, these papers are focused on detecting the task-relevant object parts and/or grasps whereas ours is focused on performing the task itself.

RL has long been studied for use in robot control. Kober *et al.* give a comprehensive (but now somewhat dated) survey [10]. Most of these operate on the torque or acceleration level of the robot controller whereas ours operates at a higher level (because we assume grasp detection and motion planning black-boxes). The RL alternative to a Q-learning is to optimize the policy directly with respect to the reward. This is called *policy search*, of which there are impressive examples [3], [13]. Policy search seems to work well in cases where human demonstrations are available [10]. It is also possible to use *actor-critic* methods, a combination of policy search with value approximation [10], [27].

VII. CONCLUSION

In this paper we have formulated the robotic pick and place problem as an MDP. We showed a practical way for how it can be solved in simulation without human demonstrations. Key to our approach was the representation of the point cloud sensor data to the CNN. The primary input to the CNN was a multi-channel image oriented in the reference frame of a grasp. This image contains information about the future pose of the object in the robot's hand. Further, we showed it is

important the image cover a relatively small volume if clutter is present. This is because less clutter is visible in the image, thus simplifying the learning problem. Also key to making the problem tractable was to assume a grasp-detection sensor and a motion planning black box. Apparently this knowledge can be transferred to a physical robot in spite of these assumptions as we showed with experiments on a UR5.

Formulating the problem as an MDP is attractive not only because the literature is full of RL solutions but also because the framework is highly extensible. For example, with a small change to the transition function we could easily relax the simulation assumption that grasps always succeed. The only difficulty is deciding how to simulate the grasps because there are so many options [4], [9], [16]. Another simple change would be to add sensor views as actions. This way, the robot should learn to take as few views as possible while getting enough information to perform the task. Another straightforward extension would be to incorporate automatic error-recovery: the robot would keep placing the object until the robot was satisfied with the result. Although these extensions are straightforward, the caveat is that it is not clear how much longer training will take or how much larger a neural network and/or state representation is needed. We leave this, with additional domain transfer improvements, to future work.

Known limitations include the time required for training (about 12 hours for each object class), inability to recognize correct objects when the clutter contains objects from multiple classes, and fixed place options.

REFERENCES

[1] Hao Dang and Peter Allen. Semantic grasping: Planning robotic grasps functionally suitable for an object manipulation task. In *IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pages 1311–1317, 2012.

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

[3] Justin Fu, Sergey Levine, and Pieter Abbeel. One-shot learning of manipulation skills with online dynamics adaptation and neural network priors. In *IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pages 4019–4026, 2016.

[4] Marcus Gualtieri, Andreas ten Pas, Kate Saenko, and Robert Platt. High precision grasp pose detection in dense clutter. In *IEEE Int'l Conf. on Intelligent Robots and Systems*, 2016.

[5] Kensuke Harada, Kazuyuki Nagata, Tokuo Tsuji, Natsuki Yamanobe, Akira Nakamura, and Yoshihiro Kawai. Probabilistic approach for object bin picking approximated by cylinders. In *IEEE Int'l Conf. on Robotics and Automation*, pages 3742–3747, 2013.

[6] Stefan Hinterstoisser, Cedric Cagniart, Slobodan Ilic, Peter Sturm, Nassir Navab, Pascal Fua, and Vincent Lepetit. Gradient response maps for real-time detection of textureless objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):876–888, 2012.

[7] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Ssergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM Int'l Conf. on Multimedia*, pages 675–678, 2014.

[8] Yun Jiang, Changxi Zheng, Marcus Lim, and Ashutosh Saxena. Learning to place new objects. In *Int'l Conf. on Robotics and Automation*, pages 3088–3095, 2012.

[9] Edward Johns, Stefan Leutenegger, and Andrew Davison. Deep learning a grasp function for grasping under gripper pose uncertainty. In *IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pages 4461–4468, 2016.

[10] Jens Kober, Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The Int'l Journal of Robotics Research*, 32(11):1238–1274, 2013.

[11] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

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

[13] Sergey Levine, Nolan Wagener, and Pieter Abbeel. Learning contact-rich manipulation skills with guided policy search. In *IEEE Int'l Conf. on Robotics and Automation*, pages 156–163, 2015.

[14] Hsien-I Lin, Yi-Yu Chen, and Yung-Yao Chen. Robot vision to recognize both object and rotation for robot pick-and-place operation. In *Int'l Conf. on Advanced Robotics and Intelligent Systems*, pages 1–6. IEEE, 2015.

[15] James MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symp. on mathematical statistics and probability*, pages 281–297, 1967.

[16] Andrew Miller and Peter Allen. Graspit! a versatile simulator for robotic grasping. *IEEE Robotics & Automation Magazine*, 11(4):110–122, 2004.

[17] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212, 2014.

[18] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei Rusu, Joel Veness, Marc Bellemare, Alex Graves, Martin Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[19] Austin Myers, Ching L Teo, Cornelia Fermüller, and Yiannis Aloimonos. Affordance detection of tool parts from geometric features. In *IEEE Int'l Conf. on Robotics and Automation*, pages 1374–1381, 2015.

[20] Anh Nguyen, Dimitrios Kanoulas, Darwin Caldwell, and Nikos Tsagarakis. Detecting object affordances with convolutional neural networks. In *IEEE/RSJ Int'l Conf.on Intelligent Robots and Systems*, pages 2765–2770, 2016.

[21] Anh Nguyen, Dimitrios Kanoulas, Darwin Caldwell, and Nikos Tsagarakis. Preparatory object reorientation for task-oriented grasping. In *IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pages 893–899, 2016.

[22] Karl Pauwels and Danica 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*, pages 1300–1307, 2015.

[23] Colin Rennie, Rahul Shome, Kostas Bekris, and Alberto De Souza. A dataset for improved rgbd-based object detection and pose estimation for warehouse pick-and-place. *IEEE Robotics and Automation Letters*, 1(2):1179–1185, 2016.

[24] Ashutosh Saxena, Justin Driemeyer, and Andrew Ng. Robotic grasping of novel objects using vision. *The Int'l Journal of Robotics Research*, 27(2):157–173, 2008.

[25] John Schulman, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, and Pieter Abbeel. Finding locally optimal, collision-free trajectories with sequential convex optimization. In *Robotics: Science and Systems IX*, 2013.

[26] Richard Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in neural information processing systems*, pages 1038–1044, 1996.

[27] Richard Sutton and Andrew Barto. *Introduction to reinforcement learning*, volume 135. MIT Press Cambridge, 1998.

[28] Philip Torr and Andrew Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1):138–156, 2000.

[29] Paul Wohlhart and Vincent Lepetit. Learning descriptors for object recognition and 3d pose estimation. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition*, pages 3109–3118, 2015.

[30] Walter Wohlkinger, Aitor Aldoma, Radu Rusu, and Markus Vincze. 3dnet: Large-scale object class recognition from cad models. In *IEEE Int'l Conf. on Robotics and Automation*, pages 5384–5391, 2012.