

Learning to Place Objects onto Flat Surfaces in Upright Orientations

Rhys Newbury^{1*}, Kerry He^{1*}, Akansel Cosgun¹ and Tom Drummond¹

Abstract—We study the problem of placing a grasped object on an empty flat surface in an upright orientation, such as placing a cup on its bottom rather than on its side. We aim to find the required object rotation such that when the gripper is opened after the object makes contact with the surface, the object would be stably placed in the upright orientation. We iteratively use two neural networks. At every iteration, we use a convolutional neural network to estimate the required object rotation, which is executed by the robot, and then a separate convolutional neural network to estimate the quality of a placement in its current orientation. Our approach places previously unseen objects in upright orientations with a success rate of 98.1% in free space and 90.3% with a simulated robotic arm, using a dataset of 50 everyday objects in simulation experiments. Real-world experiments were performed, which achieved an 88.0% success rate, which serves as a proof-of-concept for direct sim-to-real transfer.

Index Terms—Deep Learning in Grasping and Manipulation, Perception for Grasping and Manipulation

I. INTRODUCTION

Everyday objects are usually placed in specific orientations that are convenient to humans. For example, a cup is designed to be placed on its bottom rather than on its side. Placing objects down correctly in semantically preferable orientations is a fundamental skill for service robots. For example, a robot unpacking the dishwasher should place plates, glasses, and bowls on shelves in certain orientations. Research in robotic manipulation over the past decades has mostly focused on how to pick up objects [1], with recent works utilizing the advances in deep learning [2]–[5]. However, after it has been grasped what to do with the object has largely been overlooked in the field. A common practice in pick-and-place robotic manipulation scenarios is to drop the object at a height without any consideration to its resulting pose [3], [5], [6]. Only a handful of researchers have studied how to place the grasped object down [7]–[10]. Furthermore, no work to our knowledge has leveraged deep learning for the object placement problem.

Humans usually associate an upright orientation with objects, placing them in a way that they are most commonly seen in our surroundings [10]. In this paper, we study the

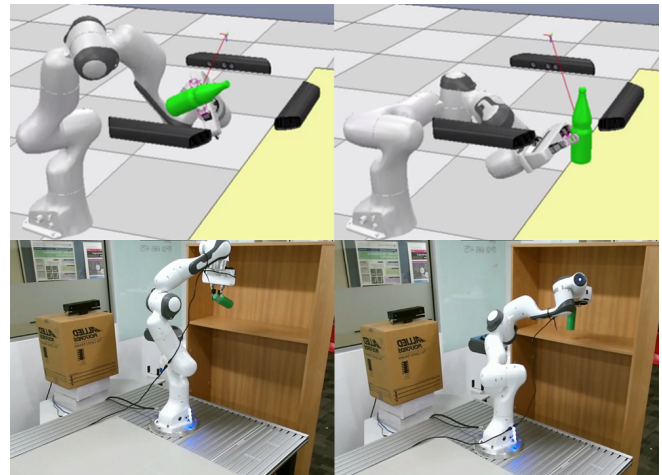


Fig. 1: Object placement of a coke bottle upright on a flat surface in simulation (top) and of a 3D-printed model using a real robot (bottom). The robot starts with a random grasp pose (left). Our approach computes a rotation required to bring the object to an upright orientation (right).

problem of placing a grasped object down on an empty flat surface in the upright orientation. We consider the solution to the problem as finding the required object rotation such that when the gripper is opened, the object comes to rest in the upright orientation under the influence of gravitational forces. We propose two convolutional neural networks, Placement Rotation Convolutional Neural Network (PR-CNN) and Placement Quality Convolutional Neural Network (PQ-CNN), used in an iterative algorithm until a satisfactory solution is found. We train both networks in simulation using 45 randomly selected object models of typical household objects. We report the experimental results on the remaining five previously unseen objects starting from random object orientations in the robotic gripper.

The contributions of this paper is twofold:

- An iterative, learning-based approach to placing previously unseen objects from input depth images, without access to object class or models.
- A proof-of-concept implementation on a robotic system, demonstrating the feasibility of direct sim-to-real transfer.

The organization of the paper is as follows. We review the relevant literature in Section II and describe the problem in Section III. We present our iterative placement method in Section IV. We investigate placement in free space in Section V, with a simulated robot in Section VI and a real robot in Section VII, before concluding in Section VIII.

*Equal contribution

¹Monash University, Australia Email: rhys.newbury@monash.edu
Digital Object Identifier (DOI): 10.1109/LRA.2021.3068122

©2021 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

II. RELATED WORK

In one of the earliest implementations of robotic object placement, Edsinger [11] use a compliant robotic arm to place an object onto a shelf by moving the arm to a fixed configuration and then lowering the end-effector using force control, hence utilizing contact with the environment. However, this approach assumes that the pose of the object in the gripper is known, which is unavailable for unknown objects. Since then, many researchers have approached the placement problem analytically, attempting to find flat features on the object and the surface on which the object can be placed [8], [9], [12]–[14]. Baumgart [12], [13] find stable poses of the object analytically by finding a point of first object contact, followed by rotating the object such that additional contact points are found. Their approach runs in real time, but does not take into account object semantics for determining placement orientations. Harada [8], [14] matches planar surface patches on the object with planar surface patches in the environment, which allows finding placements on large, flat surfaces, but also less obvious placements such as a mug hanging on a flat bar. Their approach, however, requires the 3D model of the object and its pose. Haustein [9] presents a similar approach and uses Monte Carlo Tree Search to optimize motion planning to reach the stable pose.

The majority of recent approaches in robotic manipulation are driven by machine learning approaches, and object placement is no exception. Fu [10] use hand-chosen features to find the upright orientation of man-made objects. The approach presented by Jiang [7], [15] uses learning on hand-designed features and successfully places known objects in stably 98% of the time and new objects 82% of the time. Paolini [16] estimates the probability of a successful placement, then attempts to solve for the most likely placement location, given a grasped object. Manuelli [17] develops a placement algorithm based on keypoint estimators, which applies geometric constraints on the keypoints to achieve category-level placement. This work is extended to use shape completion based on dense point clouds [18]. Mitash [19] demonstrates successful placement of previously unknown objects into constrained spaces by using a possibilistic representation to model the unobserved parts of the object. They also demonstrate a transfer of the object between two robot manipulators, from a vacuum-based to a parallel jaw gripper.

While the majority of the surveyed papers make use of 3D object models, similar to [7], [15], [19], we assume that we do not have access to the object model or class during runtime and must solve the placement problem utilizing only sensor readings from RGBD cameras. Our work is different from existing literature in two aspects: 1) The concept of upright orientations, which embodies human preferences, has been first explored by Fu [10] at a theoretical level given object models. Our work is novel in combining this theory with the robotic placement problem. 2) We propose a novel hardware-in-the-loop iterative approach, where the robot applies rotations to the object, and the new observations from the sensors are passed to the network until convergence to the upright object orientation is achieved.

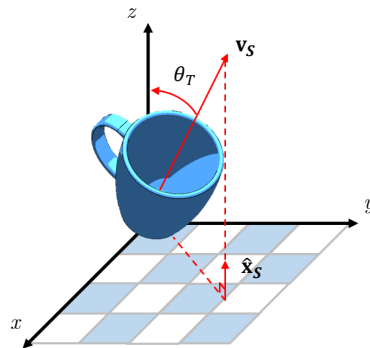


Fig. 2: We define the placement surface as the infinite xy -plane, stable axis to the placement plane ($\hat{\mathbf{x}}_S$) and upright vector for each object ($\hat{\mathbf{v}}_S$) which encodes possible upright orientations. θ_T is the shortest angle to rotate the object to an upright orientation.

III. PROBLEM DESCRIPTION

Fig. 2 illustrates the axes and angles we define for an object. The coordinate frame defined by the axes (x, y, z) represents an arbitrary and fixed world frame, in which gravity is acting in the $-z$ direction. We define the stable axis $\hat{\mathbf{x}}_S$ as the orthogonal unit vector to the surface on which objects are placed. The placement surface is set as the infinite and uncluttered xy -plane; hence we set $\hat{\mathbf{x}}_S$ to be in the direction of the z -axis. We assign an upright vector $\hat{\mathbf{v}}_S$ attached to an object, such that $\hat{\mathbf{v}}_S = \hat{\mathbf{x}}_S$ when the object is in its upright orientation. An important property of object placement on an uncluttered infinite plane is that the placement is independent of any rotation about the stable axis $\hat{\mathbf{x}}_S$. To illustrate this, any rotation of the mug shown in Fig. 2 about $\hat{\mathbf{x}}_S$ can be reframed as a rotation of the global reference frame about the same axis. Because of this, there exists a continuous set of rotations $\mathbf{R} \in SO(3)$ that can orient an object to an upright orientation. We uniquely define the ground truth rotation R_T as the shortest possible rotation required to move the object from its current orientation to an upright orientation. The ground truth rotation R_T can therefore be found as the transformation required to rotate the unit vector $\hat{\mathbf{v}}_S$ to $\hat{\mathbf{x}}_S$. This rotation is calculated most easily using axis-angle representation, which is defined by an angle θ rotated about an axis \mathbf{n} . The ground truth rotation angle $\theta_T \in [-\pi, \pi]$ is the angle between unit vectors $\hat{\mathbf{v}}_S$ and $\hat{\mathbf{x}}_S$ and can be found by:

$$\theta_T = \arccos(\hat{\mathbf{x}}_S \cdot \hat{\mathbf{v}}_S) \quad (1)$$

The ground truth axis \mathbf{n}_T is defined as the axis perpendicular to both $\hat{\mathbf{v}}_S$ and $\hat{\mathbf{x}}_S$ and can be found by:

$$\mathbf{n}_T = \hat{\mathbf{x}}_S \times \hat{\mathbf{v}}_S \quad (2)$$

We consider the robotic placement problem as follows. The robot starts with an object already in hand, and the task is to place the object down successfully. A solution to the problem is a proposed object rotation that would result in a particular object orientation (the upright orientation in this case). We assume that the robot has no a priori knowledge about the object class, 3D model, or the upright orientation, however, it has access to depth cameras and force sensing. Successful placement onto the tabletop requires that the object is stable and in the upright orientation under gravitational and contact

forces after release.

IV. ITERATIVE PLACEMENT WITH QUALITY

Our approach draws inspiration from the Iterative Error Feedback idea proposed by Carreira [20], which progressively changes an initial solution by feeding back error predictions. We also draw inspiration from actor-critic networks, where one network outputs an action for the system to execute, and the other network determines the quality of the action [21]. We propose an iterative, learning-based approach to robotic object placement. We propose two neural networks which both take as input depth images from multiple viewpoints:

- Placement Rotation Convolutional Neural Network (PR-CNN): Outputs the rotation that transforms the object to the upright orientation.
- Placement Quality Convolutional Neural Network (PQ-CNN): Estimates the confidence level that the object would be come to rest in its upright orientation if it is placed in its current orientation.

Algorithm 1 Proposed placement algorithm

```

1: function ROTATEOBJECT
2:   for  $i = 1 \rightarrow max\_restart$  do
3:     for  $i = 1 \rightarrow max\_iter$  do
4:        $img \leftarrow take\_depth\_image()$ 
5:        $rotation[i] \leftarrow PR-CNN(img)$ 
6:        $rotate\_ee\_by(rotation[i])$ 
7:        $img \leftarrow take\_depth\_image()$ 
8:        $orientation[i] \leftarrow get\_ee\_rotation()$ 
9:        $quality[i] \leftarrow PQ-CNN(img)$ 
10:      if  $1 - quality[i] < \epsilon_1$  then
11:        if  $|rotation[i]| < \epsilon_2$  then
12:          goto 18
13:        end if
14:      end if
15:    end for
16:     $rotate\_ee\_by(random\_rotation())$ 
17:  end for
18:   $max\_index \leftarrow arg\_max(quality)$ 
19:   $rotate\_ee\_to(orientation[max\_index])$ 
20: end function

```

The pseudocode can be seen in Algorithm 1. At every iteration, the PR-CNN proposes an object rotation to achieve the upright orientation, which the robot executes. Then PQ-CNN estimates the quality of the proposed object orientation. The algorithm stops if PQ-CNN predicts that the orientation would be upright if placed in the current orientation, and PR-CNN’s output is the rotational identity which suggests that the object orientation has converged. If this does not occur within a number of iterations max_iter , the algorithm restarts from a new random object orientation. If the maximum number of restarts $max_restart$ is reached without a satisfactory solution, the rotation that yields the maximum predicted quality is chosen among all iterations. The iterative approach helps in getting more observations from the object and correcting the errors of PR-CNN. Quality

estimation provided by PQ-CNN is useful to identify when PR-CNN converges to an upside-down orientation, which is a common failure mode when objects lack noticeable features distinguishing the upright and upside-down orientations.

A. Placement Rotation CNN (PR-CNN)

We aim to learn the required rotation applied to the object that would result in an upright orientation. The ground truth rotation R_T is obtained analytically using the methodology outlined in Section III.

PR-CNN has an architecture based on ResNet-50 [22] and is pre-trained on ImageNet [23]. PR-CNN takes as input n 64×64 depth images of the object, which has depth values normalized between $[-0.5, 0.5]$. The network is not specific to the number or pose of cameras and may vary between implementations discussed in this paper. We train a single CNN with shared weights, such that each depth image is passed independently to the network. The n outputs of size 1024 are then concatenated into a single output. This output is then used in two linear layers reducing the output size to 1024 and 6, respectively. This network structure (shown in Fig.3) draws inspiration from multi-viewpoint pose estimation [24]. The six-dimensional output corresponds to the six-dimensional representation of $SO(3)$ rotations as proposed by Zhou [25]. This represents the first two columns of a rotation matrix, which is then converted to a full rotation matrix using the Gram-Schmidt-like process described by Zhou [25]. This representation of rotations is continuous, which can allow for smaller approximation errors [26].

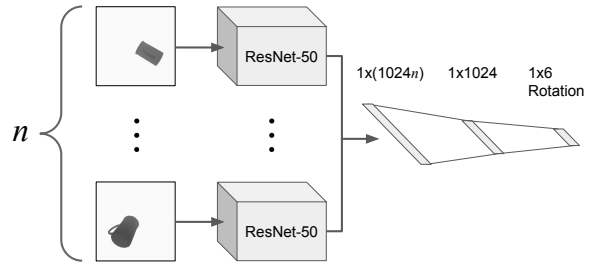


Fig. 3: We train a convolutional neural network, with shared weights between each of the n depth images. These are then passed through linear layers to get to the desired number of outputs

Furthermore, we use a similar loss function to Zhou [25], which is the geodesic distance between the output (R_s) and the ground truth (R_T):

$$\mathcal{L}_{geodesic} = \arccos \left(\frac{\text{tr}(R_s R_T^{-1}) - 1}{2} \right) \quad (3)$$

where $\text{tr}(R)$ is the matrix trace operator. For a rotation matrix, this is defined as:

$$\text{tr}(R) = R_{00} + R_{11} + R_{22} \quad (4)$$

B. Placement Quality CNN (PQ-CNN)

We learn the estimated quality of an object being placed in its current orientation, which is defined as the confidence that the object will come to rest in the upright orientation if released in the current orientation. The object quality that we are estimating is based on the assumption of placing on a flat surface. The ground truth binary label is obtained by Bullet Physics Simulation [27]. The procedure to collect data

is described in Section V-B. By learning the estimated quality of an object in its current pose, we can avoid cases where PR-CNN proposes incorrect rotations. However, this network can be used independently of PR-CNN to estimate the quality of placements using a traditional approach to find the upright pose.

Human preferred upright orientations of objects are often defined by the personal design and preference of humans, meaning an analytical approach may be difficult to generalize between different classes of objects. Hence, this motivates the use of a CNN, which can be trained using user-defined upright orientations.

PQ-CNN uses the same network structure as PR-CNN (Section IV-A). The weights are not shared between the ResNet-50 architectures as the networks are learning two different criteria, and we did not want to bias the networks to learn the same patterns. Modifications were made to the final layer to output a single number with sigmoid activation. We use a binary cross-entropy loss function for PQ-CNN.

V. EXPERIMENTS WITHOUT A ROBOT

A. Object Models

We use a dataset of 50 everyday objects, each with 3D meshes of each object. These object models are only needed for data collection and are not used during the execution of our approach. The simulated renderings of all 50 objects can be seen in Fig 4. We picked the objects such that each object had a well-defined, single upright vector. We avoided symmetrical objects such as cans and boxes due to the existence of multiple stable orientations, which are impossible for a depth camera to differentiate between, leading to an ambiguous problem. We also avoided objects without a “natural base” such as spherical objects, spoons, or toothbrushes, as these objects were not designed to stand in an upright orientation [10]. We manually label each object with an upright orientation. We use a scene where three depth cameras were positioned orthogonally from each other, to the front, left, and right of the gripper, at a 25cm radius around the point p_c at which objects and the gripper interact (as seen in Fig. 1).



Fig. 4: The 50 object models that were chosen to train and evaluate our models. Object models are depicted in their upright orientations which are annotated manually. Test sets were created by randomly selecting 5 objects as test objects to perform evaluation on, while the remaining 45 objects are used to train the networks.

B. Data Collection

We used the PyRep toolkit [28] for the simulation environment. PR-CNN and PQ-CNN need separate but overlapping

datasets. We collected two datasets, both with 100,000 data points. To collect a data point, for PR-CNN, we randomly pick an object from the 50 objects and randomly sample an orientation with a slight positional variation around p_c . We collect the three depth camera images and the ground truth rotation that would rotate the object to the upright orientation. We save the binary label for PQ-CNN, indicating whether the object would be in an upright placement or not when the object is dropped at the lowest possible height from the placement surface in the sampled orientation (obtained by physics simulation). As PQ-CNN aims to solve a binary classification task, we need a balanced dataset for both classes. As random orientations are more likely to unsuccessfully come to rest in the upright orientation, we sample additional data points with slight variations to the upright orientation. Furthermore, we use an equal amount of successful and unsuccessful data points for training PQ-CNN.

C. Methods

- **Baseline:** We combine point clouds from 6 orthogonal rotations (analogous to 6 sides of a cube) and estimate each point’s normal vector. The object is then placed such that the average normal vector of the largest flat plane is perpendicular to the table surface. Open3D [29] is used for point cloud operations.
- **Single pass (SP):** A single pass of PR-CNN is used to determine the object rotation.
- **Iterative (ITR):** PR-CNN is run iteratively until the identity rotation is achieved or a maximum number iterations (15 in this case) has been reached.
- **Iterative with Quality (ITR-Q):** Our full approach combining PR-CNN and PQ-CNN, as detailed in Section IV.

D. Metrics

- **Success Rate:** The percentage of placements where the steady-state object orientation is within $\pm 15^\circ$ of the desired ground truth orientation.
- **Stability Rate:** The percentage of placements where the object stays stationary for a minute and the final object orientation is within $\pm 15^\circ$ of the initial placement orientation. This metric was adopted from [15].
- **Angular Error:** The average angle difference between the upright vector \hat{v}'_S and the stable axis \hat{x}_S .

E. Experimental Procedure

PR-CNN was trained for 150 epochs, and the epoch with the highest ITR success rate was selected to present results. For PR-CNN, we did not use a validation set, however, we validated the model on the test objects by using physics to simulate 250 placements from random initial orientations of randomly selected objects every 25 epochs. PQ-CNN was trained for 15 epochs, and the highest validation accuracy was used during the evaluation of ITR-Q.

F. Design Parameters

To select the design parameters used to evaluate the full ITR-Q approach, we performed additional studies on PR-CNN to investigate the effect of pre-training, shared weights, output angle representation, and number of cameras. Unless

ITR (Success Rate %)	
ResNet-50 SW	54.8
ResNet-50 PT	89.6
ResNet-50 PT SW	98.4

TABLE I: Comparison of network architectures using pre-training (PT) and shared weights (SW). Pre-training yields an improvement in success rate. The architecture using three single-channel input CNN’s in parallel with shared weights is shown to outperform the alternative architecture using a single three-channel input CNN.

otherwise specified, we use pre-training, shared weights, 6D angular representation, and 3 orthogonal cameras for training the PR-CNN. To attribute any performance differences to PR-CNN, success rates are obtained using only **ITR** approaches. For these experiments, we trained and evaluated the networks on all 50 object models. The experiments were completed using a NVIDIA GeForce GTX 1080. The average network inference time was 11.89ms for PR-CNN (three depth image inputs).

Network Architecture: We investigate the effect of pre-training and shared weights on the performance of the network. Each network uses ResNet-50 [22] as the backbone. We consider two alternatives, firstly, training a singular network with three input channels, where each channel represents a depth image from a different viewpoint. Secondly, an architecture using shared weights as described in Section IV-A. The results are shown in Table I. The effect of pre-training is particularly noticeable from the results, which show an increase in success rate from 54.8% to 98.4% for **ITR**. Despite being pre-trained on an unrelated dataset (ImageNet [23]) consisting of 3 channel RGB images, the network trains faster and converges to a lower loss. We also observe a performance increase of 8.8 percentage points in using the architecture with shared weights using **ITR**. For the rest of this paper, we use pre-training along with the architecture with shared weights.

Number of Cameras: We analyze the effect of using different quantities of cameras on the placement performance. All cameras are orthogonal to each other as described in Section V-A, with additional cameras being added to the left, front, right, and back of the object in the order of increasing cameras. The results are shown in Table II. There is a noticeable performance increase when going from one to two cameras. However, the benefit of additional cameras plateaus at three cameras. Although three cameras were used in this paper, these results suggest that our approach can be flexibly transitioned to use only one or two cameras without a major drop in performance. This allows our approach to be more applicable to the real world, where access to three cameras may not be possible or practical. Our real-world robotic experiments are conducted with a single camera.

Angular Representation: We compare the effect of using different angular representations as the output of PR-CNN. For all angular representations, we use the same geodesic loss function shown in Eq. 3. The results are shown in Table III. The Euler angle representation performed worst among all representations, likely due to discontinuity issues. The 6D angular representation [25] slightly outperformed the

ITR (Success rate %)	
1 Camera	85.2
2 Cameras	96.8
3 Cameras	98.4
4 Cameras	96.4

TABLE II: Comparison of different quantities of cameras used as inputs into PR-CNN. The setup using three cameras performed the best out of all **ITR** experiments.

quaternion representation by 2.0 percentage points. While both the quaternion and 6D representations are viable options for the network, we opt to use the 6D representations for the rest of this paper.

ITR (Success rate %)	
Euler	32.4
Quaternion	96.4
6D [25]	98.4

TABLE III: Comparison of different angular representations on network performance. The Euler representation performed the worst, while the 6D representation yielded the highest **ITR** success rate.

G. Performance on Unseen Objects

To evaluate the generalizability of the networks, five test sets were created by randomly selecting 5 non-overlapping objects out of the 50. We trained PQ-CNN and PR-CNN on the remaining 45 objects. The aggregate results on all the test sets is shown in Table IV.

	Success Rate (%)	Stability Rate (%)	Angular Error (°)
Baseline	54.0 ± 10.0	96.7 ± 4.2	47.7 ± 11.5
SP	84.4 ± 8.9	89.9 ± 5.8	22.8 ± 13.1
ITR	96.1 ± 4.5	98.3 ± 1.6	8.0 ± 7.1
ITR-Q	98.1 ± 1.9	99.3 ± 1.1	5.2 ± 2.5

TABLE IV: Performance of different placement methods averaged over five different sets of 5 previously unseen objects randomly chosen from the set of 50 objects. **SP**, **ITR** and **ITR-Q** were trained on the remaining 45 known objects. Our full approach **ITR-Q** outperforms all other methods in all metrics. Standard deviations are between the metrics obtained from the five test sets.

SP, **ITR** and **ITR-Q** all outperformed the baseline. Although the baseline had a high stability rate of 96.7%, it only places objects in upright orientations only 54.0% of the time. This suggests that the largest flat surface of an object often corresponds to a stable pose but does not always correspond to a semantically preferred orientation.

The proposed **ITR-Q** algorithm performed the best in all metrics, achieving a success rate of 98.1%, a stability rate of 99.3% and an average angular error of 5.2° on previously unseen objects. Compared to the 98.4% success rate of **ITR** on seen objects (see Table I), these results suggest that the full **ITR-Q** algorithm can successfully generalize to unseen objects with only a minimal drop in performance.

The benefit of an iterative approach to using PR-CNN is validated by the 11.7 percentage point improvement in success rate between **SP** and **ITR** methods. Moreover, the consistent outperformance of **ITR** compared to **SP** was observed throughout training, as seen in Figure 5. This behavior suggests that the iterative approach can converge to a solution even if the underlying network has not been fully trained.

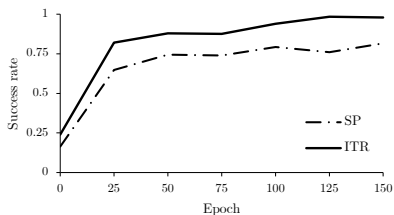


Fig. 5: **ITR** consistently outperformed **SP** during the training of PR-CNN.

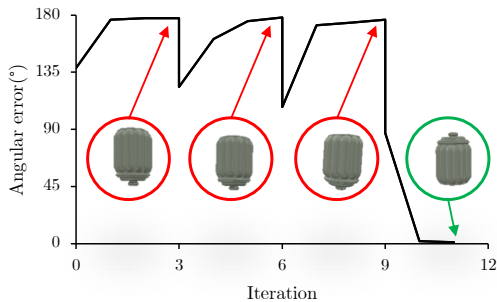


Fig. 6: Angular error of the cookie jar object is plotted at each iteration for **ITR-Q**. Red arrows point to iterations corresponding to when **ITR-Q** has identified low-quality convergence, and samples a new random orientation to seek a different final orientation. The final iteration highlighted in green is identified to be of a high quality by PQ-CNN, thus concluding the iterations.

H. Common Failure Modes

The main failure mode was placing objects upside-down, particularly objects with relatively symmetrical features at either orientation, such as the cookie jar object (as seen in Figure 6). This was particularly an issue for **SP** and **ITR** approaches if PR-CNN converged to this incorrect orientation. **ITR-Q** resolves this issue by identifying when a problematic convergence has occurred by setting a new randomized initial orientation, analogous to common optimization techniques used to escape from local minima. An example of this is illustrated in Figure 6. For the cookie jar object, **ITR** had a success rate of 56.0%, while **ITR-Q** improved the success rate of the cookie jar object to 96.1%. This highlights the effectiveness of our full approach. The performance improvement on the cookie jar and similarly ambiguous objects make up most of the 2.0 percentage point improvement in overall success rate between **ITR** and **ITR-Q**.

I. Generalization between object categories

We observed that PR-CNN generalized well to objects of the same semantic categories present in the training set. Furthermore, during preliminary testing, when PR-CNN was trained on only bottles, the network would generalize well on other bottle like objects. However, PR-CNN trained on the single semantic category, bottles, did not generalize well to completely new object categories such as plates.

To mitigate this limitation, we trained our network on a wide variety of objects from different semantic categories, which allowed PR-CNN to generalize a lot better to new objects. Within the 50 objects we have selected, we observed no performance degradation increasing the number of object categories, apart from requiring more training data and more training time. However, further exploration of a more diverse



Fig. 7: Left: RGB component of the RGB-D input to the network. Middle: ground truth depth image. Right: Output of Pix2Pix which segmented out the gripper and hallucinated the gaps behind the fingers.

object category pool and generalizing between them is an interesting direction for future work.

VI. PLACEMENT WITH A SIMULATED ROBOT

We performed additional experiments with our trained networks by introducing a robotic arm. Rather than retraining the networks, we use Pix2Pix [30] to segment the object in the image, removing the gripper. We then use **ITR-Q** on the segmented image. Assuming the gripper and grasped object act as a single rigid body (i.e. no slipping), we can execute object rotations by applying the same rotation to the robotic gripper. Placement is achieved by lowering the end effector until contact is detected via force sensing of the robot joints. The gripper fingers are then opened, and the end-effector is retracted along the reverse direction of the end-effector orientation. To increase the chances of solving the inverse kinematics, objects are placed on a surface elevated from the ground.

A. Object Segmentation

Learning-based approaches (such as [31]) have previously been proposed to segment out grippers from real scenes. However, these approaches do not reconstruct occluded pixels behind the gripper. We use Pix2Pix [30] to segment out the object and hallucinate pixels occluded by robotic fingers, similar to [32]. The input to Pix2Pix is an RGB-D image with the gripper grasping the object. The ground truth is a depth image with only the object (see Fig. 7). We collected a dataset of 1,500 images, 30 for each object. We train a single network to remove the gripper from all viewpoints. We train Pix2Pix for 40 epochs on the training set objects. We use the default parameters and loss function of the original model [30]. The average network inference time was 21.09ms for Pix2Pix (one RGB-D input).

B. End Effector Constraints

As we take an iterative approach, we cannot present the object to the cameras at a fixed gripper angle. Executing the rotations directly from PR-CNN would likely cause heavy occlusions in the images due to the gripper. See Fig. 8 (Top) for an example where the gripper almost entirely blocks one of the cameras. Therefore, we take a rotation of the gripper around the stable axis (defined as the z-axis in our problem) such that we minimize occlusions due to the gripper while still allowing for enough degrees of freedom to rotate the upright vector of the object to any orientation. After taking this action, the images become less occluded, as shown in Fig. 8 (Bottom).

C. Results

The results are evaluated on the same five test sets as Section V-G and are shown in Table V. Compared to the

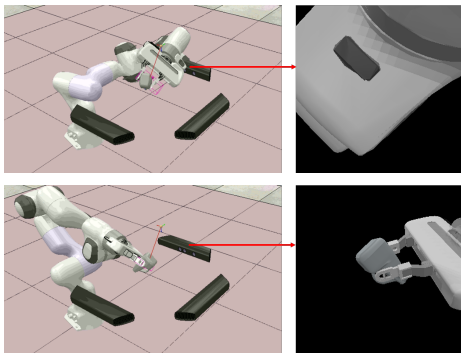


Fig. 8: Directly executing the rotations would cause the end-effector to occlude the cameras (Top), therefore, we apply a transformation to the output rotation around the stable axis to minimize occlusions (Bottom)

previous experiment without the robot arm, the performance was lower as expected. The success rate decreased by 7.8 percentage points and the stability rate decreased by 4.0 percentage points. This is due to the inherent difficulty of the scenario, which includes the robot hand being in the images and the contact physics between the end-effector and the object. However, our approach can place previously unseen objects in stable orientations 95.3% of the time and a success rate over 90%.

	Success Rate (%)	Stability Rate (%)	Angular Error (°)
ITR-Q w/ Gripper	90.3 ± 2.6	95.3 ± 1.9	16.7 ± 7.2

TABLE V: Performance of **ITR-Q** in the presence of a simulated gripper. The same training and testing objects from Section V-G were used. Despite the gripper introducing complexities into the input image, the algorithm is still able to achieve relatively high success rates. Standard Deviations are between the metrics obtained from the five test sets.

The main failure mode of this situation is Pix2Pix [30] being unable to handle severe occlusions for small objects. In these cases, less useful information is provided to the networks, which cannot produce useful outputs. This problem could be alleviated by re-grasping the object.

VII. PLACEMENT WITH A REAL ROBOT

To demonstrate the feasibility of our proposed approach and the potential for sim-to-real transfer, we implemented our object placement approach on a Franka Emika Panda robotic arm using a single Kinect V2 depth camera. To allow for performance comparisons with the previous experiment involving a single camera, the **ITR** approach was used for the real-world experiments. We evaluated our approach in a pick and place scenario, where an object starts on a table and has to be placed onto a shelf. The system consists of two computers connected via TCP/IP. Multiple computers were used due to the high USB bandwidth required for running multiple RGB-D cameras. Each computer uses the Melodic version of the Robot Operating System running on Ubuntu 18.04 LTS operating system.

To generate realistic object grasps for the algorithm to consider placement, we use GGCNN [5] to generate viable grasp points from a given depth image generated by a wrist-mounted RealSense D435 RGB-D camera. Once the object is grasped, the arm moves to a predefined location in front of the Kinect V2.



Fig. 9: The 5 objects used for real robot experiments. Left to right: Bowl, Sunscreen, 3D printed Spray Bottle, 3D printed Green Pitcher and 3D printed Green Bottle. Only the Green Bottle was in the original 50 training objects.

Some modifications were made to the original approach to increase the likelihood of success in real-world experiments. (1) The Kinect V2 images were pre-processed before being used in the **ITR** algorithm, which involved removing the background of the depth and color images based on a minimum and maximum depth threshold. (2) We consider the output of PR-CNN for a moving window of depth images (window size = 5). If the output of all 5 frames is within 10° , the average rotation across the window is executed. This helps remove possible outliers from the output. If this does not occur within 100 frames, the window with the lowest variance is used. (3) We controlled the arm with the Manipulability Motion Controller (MMC) [33], a controller designed to maximize the manipulability of the arm. This was chosen to avoid joint limits when executing complex rotations with the arm. (4) When placing the object at the predetermined position on the shelf, the target end-effector pose would often correspond to kinematically infeasible joint configurations. Hence, we randomly sampled positions about the shelf and angles around the stable axis to find a feasible placement pose. This maximized the possibility of successfully finding placement positions within the joint limits of the robotic arm.

We ran ten trials of five different objects (shown in Figure 9). One object was 3D printed from the data set. Four objects were not featured in the training set of the networks. Objects were chosen which could be segmented using Pix2Pix, as we are focusing on testing the approach to placing objects.

A. Results

Overall, we achieved an 88.0% success rate in real-world experiments. Despite the irregularities introduced in the real world, the results show that we can achieve results comparable to those we achieved in simulation. The network performed very well with the Bowl, however, this success rate is skewed, as the bowl was likely to fall into the correct orientation once placed.

B. Failure Modes

The most common failure was due to kinematic constraints, where the robotic arm could not successfully execute the orientation requested by the algorithm. This was partially mitigated by the introduction of MMC [33]. This could be improved further by considering motion planning for this application, such as the work done by [34].

Moreover, the robot would sometimes occlude the object

	Success Rate (%)	Avg. Num. Iterations
Bowl	100.0	1.8
Sunscreen	90.0	1.8
Spray Bottle	80.0	1.6
Green Pitcher	80.0	1.7
Green Bottle	90.0	1.3

TABLE VI: Real-world performance of **I**TR using a single depth camera, grasping objects from the table and placing the object on to a shelf. The results are comparable to those achieved in the simulated experiment (85.2%).

from the singular camera. However, this is a known issue, and as discussed in Section V-F, we expect the use of additional cameras to improve the performance of PR-CNN.

Limitations due to direct sim-to-real transfer also introduced errors, which is an open problem in robot learning [35]. Inaccurate segmentation of the object from the gripper using the Pix2Pix network and noise from the images captured by the Kinect V2 were observed during the experiments. Future work to investigate and mitigate the effects of these artifacts is expected to improve the performance of the algorithms further.

VIII. CONCLUSION AND FUTURE WORK

In this work, we propose an approach to rotate grasped objects into orientations to be placed in stable, upright orientations. We show the feasibility of learning to place objects from depth images without object classification or explicit pose estimation. Our simulation experiments suggest that our iterative approach **I**TR-Q performs better than placing objects on their largest supporting planes. The proposed approach is general enough so that our two proposed neural networks can be replaced by alternative approaches, however, we show that learning-based approaches provide the opportunity to learn human-annotated upright orientations, giving an edge over analytical methods. Our work also shows potential for sim-to-real transfer learning and justifies the need for more research to generalize the approach to different object classes and shapes.

Our current iterative approach only re-evaluates the object's orientation after it has completed the rotation, making it slow to react to disturbances. We reserve a closed-loop reactive approach as future work, analogous to grasping in [5], that can re-evaluate the rotation at every time step, and hence will be more effective in dealing with object slip and disturbances. A limitation of our approach is that it is designed for objects with a single defined preferred orientation, however, some objects such as cans and boxes have multiple preferred orientations. It is possible to extend our representation to include such objects by considering multiple ground truth rotations. Another interesting idea for future work is further exploring the ability to estimate the quality of placement. This would allow exploring more creative placement methods such as controlled dropping of an object [34] in a feasible configuration of the arm.

REFERENCES

- [1] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis—a survey," *IEEE Trans. Robot.*, 2014.
- [2] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *IJRR*, 2015.
- [3] 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," in *RSS*, 2017.
- [4] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *IEEE ICRA*, 2016.
- [5] D. Morrison, P. Corke, and J. Leitner, "Closing the Loop for Robotic Grasping: A Real-time, Generative Grasp Synthesis Approach," in *Robotics: Science and Systems (RSS)*, 2018.
- [6] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Ffrob: Leveraging symbolic planning for efficient task and motion planning," *IJRR*, 2018.
- [7] Y. Jiang, C. Zheng, M. Lim, and A. Saxena, "Learning to place new objects," *IEEE ICRA*, 2011.
- [8] K. Harada, T. Tsuji, K. Nagata, N. Yamanobe, H. Onda, T. Yoshimi, and Y. Kawai, "Object placement planner for robotic pick and place tasks," in *IEEE/RSJ IROS*, 2012.
- [9] J. A. Haustein, K. Hang, J. Stork, and D. Kragic, "Object placement planning and optimization for robot manipulators," in *IROS*, 2019.
- [10] H. Fu, D. Cohen-Or, G. Dror, and A. Sheffer, "Upright orientation of man-made objects," in *ACM SIGGRAPH*, 2008.
- [11] A. Eadsinger and C. C. Kemp, "Manipulation in human environments," in *IEEE-RAS International Conference on Humanoid Robots*, 2006.
- [12] J. Baumgartl, P. Kaminsky, and D. Henrich, "A geometrical placement planner for unknown sensor-modelled objects and placement areas," in *IEEE ROBIO*, 2013.
- [13] J. Baumgartl, T. Werner, P. Kaminsky, and D. Henrich, "A fast, gpu-based geometrical placement planner for unknown sensor-modelled objects and placement areas," in *IEEE ICRA*, 2014.
- [14] K. Harada, T. Tsuji, K. Nagata, N. Yamanobe, and H. Onda, "Validating an object placement planner for robotic pick-and-place tasks," *Rob Auton Syst*, 2014.
- [15] Y. Jiang, M. Lim, C. Zheng, and A. Saxena, "Learning to place new objects in a scene," *IJRR*, 2012.
- [16] R. Paolini, A. Rodriguez, S. S. Srinivasa, and M. T. Mason, "A data-driven statistical framework for post-grasp manipulation," *IJRR*, 2014.
- [17] L. Manuelli, W. Gao, P. Florence, and R. Tedrake, "kpam: Keypoint affordances for category-level robotic manipulation," in *ISRR*, 2019.
- [18] W. Gao and R. Tedrake, "kpam-sc: Generalizable manipulation planning using keypoint affordance and shape completion," 2019.
- [19] C. Mitash, R. Shome, B. Wen, A. Boularias, and K. Bekris, "Task-driven perception and manipulation for constrained placement of unknown objects," *IEEE RA-L*, vol. 5, no. 4, pp. 5605–5612, 2020.
- [20] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik, "Human pose estimation with iterative error feedback," in *IEEE CVPR*, 2016.
- [21] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft actor-critic algorithms and applications," 2019.
- [22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep learning for image recognition," in *IEEE CVPR*, 2016.
- [23] J. Deng, W. Dong, R. Socher, L. jia Li, K. Li, and L. Fei-fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.
- [24] OpenAI, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.
- [25] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," in *CVPR*, 2019.
- [26] Z. Chen and F. Cao, "The construction and approximation of neural networks operators with gaussian activation function," in *Math. Comm.*, 2013.
- [27] E. Coumans, "Bullet physics engine," <http://bulletphysics.org>, 2010.
- [28] S. James, M. Freese, and A. J. Davison, "Pyrep: Bringing v-rep to deep robot learning," *arXiv preprint arXiv:1906.11176*, 2019.
- [29] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.
- [30] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *IEEE CVPR*, 2017.
- [31] V. Florence, J. J. Corso, and B. Griffin, "Robot-supervised learning for object segmentation," in *2020 ICRA*, 2020, pp. 1343–1349.
- [32] Z. Chen, D. Ting, R. Newbury, and C. Chen, "Semantic segmentation for partially occluded apple trees based on deep learning," *Comput. Electron. Agric.*, 2021.
- [33] J. Haviland and P. Corke, "A purely-reactive manipulability-maximising motion controller," *arXiv:2002.11901*, 2020.
- [34] A. Holladay, J. Barry, L. P. Kaelbling, and T. Lozano-Pérez, "Object placement as inverse motion planning," in *IEEE ICRA*, 2013.
- [35] F. Sadeghi, A. Toshev, E. Jang, and S. Levine, "Sim2real viewpoint invariant visual servoing by recurrent control," in *CVPR*, 2018.