Abstract

This paper addresses the problem of mobile robot manipulation of novel objects via detection. Our approach uses vision and control as complementary functions that learn from real-world tasks. We develop a manipulation method based solely on detection then introduce task-focused few-shot object detection to learn new objects and settings. The current paradigm for few-shot object detection uses existing annotated examples. In contrast, we extend this paradigm by using active data collection and annotation selection that improves performance for specific downstream tasks (e.g., depth estimation and grasping). In experiments for our interactive approach to few-shot learning, we train a robot to manipulate objects directly from detection (ClickBot). ClickBot learns visual servo control from a single click of annotation, grasps novel objects in clutter and other settings, and achieves state-of-the-art results on an existing visual servo control and depth estimation benchmark. Finally, we establish a task-focused few-shot object detection benchmark to support future research: https://github.com/griffbr/TFOD.

1. Introduction

With the progression of high-quality datasets [11, 12, 19, 39], our community has seen remarkable advances in object detection [7, 52, 53]. Recently, few-shot object detection has emerged as a critical innovation that enables detection of new objects from only a few annotated examples [9, 61, 66]. Current few-shot detection benchmarks use limited [30] or randomized [59] few-shot examples from previous datasets.

Detection supports many downstream tasks [47, 64, 71]. When considering detection for evolving tasks, however, passive learning is not enough. This is especially true for robots, where visual experience is dynamic and interactive [3]. Furthermore, robots that only use passive data are wasting a critical asset—the ability to interact with the world and learn from those interactions. To improve performance on evolving tasks, we introduce a task-focused approach to select few-shot learning examples from robot-collected data.

We introduce a task-focused approach to learning de-
tection that we call task-focused few-shot object detection. Whereas previous few-shot detection methods improve detection given a static set of annotated examples, our goal is to discover the few-shot examples that improve the performance of downstream tasks. As our robot advances to more difficult tasks and environments, its detection model fails. Because this is precisely the learning opportunity that we want, we enable our robot to request annotation on data collected during that task. Thus, few-shot annotation is focused on tasks that require improved detection. In short, rather than trying to predict the best set of annotated examples a priori, we let the robot, world, and difficulty of the task become our oracle for learning (Fig. 2).

The first contribution of our paper is developing a complete set of detection-based tasks that enable mobile robot manipulation. To the best of our knowledge, this is the first work to develop end-to-end visual object manipulation entirely from detection. Using detection, we complete manipulation tasks with a single RGB camera without any 3D modeling requirements. Our detection-based tasks are also modular, which facilitates integration with other methods.

A second contribution of our paper is introducing task-focused few-shot object detection. This enables our robot to learn detection-based manipulation tasks for new objects and settings using only a few annotated examples (see result in Fig. 1). Furthermore, this approach addresses the problem of how to select annotated examples when applying few-shot detection to downstream tasks in the wild. We also establish a corresponding Task-Focused Few-Shot Object Detection (TFOD) benchmark, which is configurable for various few-shot settings and will support future detection research in this task-focused setting for manipulation.

Finally, we validate our combined approach across many robot experiments. Our robot learns detection-based visual servo control in 13.3 s using a single annotated example, achieves state-of-the-art results on an existing visual servo control and depth estimation benchmark, learns to grasp objects in clutter with as few as four annotated examples, and performs mobile pick-and-place at dynamic locations.

2. Related Work

Object Detection predicts a set of bounding boxes and category labels for objects in an RGB image. Many detectors use regression and classification over a set of region proposals [5, 53], anchors [38], or window centers [57]. Other detectors treat detection as a single regression problem [52] or use a transformer architecture [58] to predict all detections in parallel [7]. Detection also supports many downstream vision tasks such as segmentation [22], 3D shape prediction [14], depth [17] and pose estimation [47, 64], and single-view metrology [71], to name but a few. In this work, we continue this progress and introduce a novel approach to object manipulation that operates directly from detection.

Learning object detection typically requires a large number of bounding box annotations from a labeled dataset for training and evaluation [11, 12, 39], with some datasets additionally focusing on continuous recognition [40] or multi-view indoor environments [54]. However, the paradigm of learning from a dataset then detecting objects in the wild does not account for objects absent during initial training.

Few-Shot Object Detection (FSOD) expands on conventional detection by learning to detect novel objects from only a few annotated examples (few-shot objects). Among FSOD approaches, initial finetuning methods treat FSOD as a transfer learning problem from a large source domain to few-shot objects [9, 59]. Other methods use meta-learning algorithms to learn from existing detectors and quickly adapt to few-shot objects, either by using feature reweighting schemes [30, 66] or by using model parameter generation from base classes to efficiently learn few-shot objects [61]. Other FSOD approaches include using a distance metric learning-based classifier [31], incremental few-shot learning to reduce training requirements [45, 49], and one-shot detection by matching and aligning target-image-features with query-image-features [46]. Notably, FSOD is becoming a hotly studied area of object detection with increasingly rampant advances, even within just the past year [13, 20, 25, 34–36, 51, 56, 62, 68–70].

To benchmark FSOD methods, previous work [30] establishes set splits of \( k = 1, 2, 3, 5, 10 \) annotated bounding boxes for 5 few-shot objects on the PASCAL VOC dataset [11, 12] and \( k = 10, 30 \) for 20 few-shot objects on the MS-COCO dataset [39]. Subsequent work [59] revises this protocol by randomly selecting few-shot objects and \( k \) training examples for mean results over 40 repeated runs with additional results on the LVIS dataset [19].

However, the problem of creating new FSOD training examples or considering tasks downstream of detection has drawn scant attention. To collect custom detection training examples for indoor robots, Alabachi et al. [2] teleoperate an unmanned aerial vehicle to fly around an object while streaming images. This work sets a precedent for robot data...
collection for detection, but uses more training examples than typical FSOD methods ($k \gg 30$) and does not consider tasks downstream of detection. In consideration of downstream tasks, Xiao and Marlet [64] set a precedent by developing a unified framework of FSOD and viewpoint estimation using arbitrary 3D models of few-shot objects.

Inspired by these recent developments, in this work, we introduce task-focused FSOD data collection (see Fig. 2), i.e., collecting custom FSOD training examples for specific downstream tasks. Furthermore, rather than using a predetermined number of few-shot training examples, we let the difficulty of each task decide, thereby limiting annotation to only the necessary examples. Finally, we establish a new Task-Focused Few-Shot Object Detection (TFOD) benchmark, which will help guide future FSOD innovation.

Visual Servo Control (VS) uses visual data in the servo loop to control a robot. From a control perspective, VS has been understood for some time [8, 26]. Classic VS can position UAVs [18, 43] or wheeled robots [41, 42] and manipulate objects [27, 32, 60]. Although these works established the utility of VS, these early methods rely on structured visual features (e.g., fiducial markers or LED panels).

Subsequent VS methods manipulate non-structured objects using deep learning. Learning VS manipulation end-to-end can occur entirely on a robot [1, 33, 50] or in simulation with innovative sim-to-real transfer techniques [29, 48, 72]. Notably, these end-to-end methods are only demonstrated in fixed workspaces and do not address the challenges of mobile manipulation (e.g., a moving camera or dynamic positioning). A recent VS method addresses mobile manipulation of non-structured objects by combining classic VS with learned visual features [15], but this VS method pretrains vision and does not learn from tasks.

In this work, we build off of these developments to learn mobile manipulation from robot-collected data and few-shot object detection (see Table 1). Using our detection-based visual features, our robot learns state-of-the-art mobile VS and subsequent tasks like grasping. Furthermore, our task-focused few-shot learning approach lets our robot quickly adapt to new objects, tasks, and environments.

3. ClickBot: Learning Robot Manipulation via Task-Focused Few-Shot Object Detection

We introduce a method of robot manipulation using task-focused few-shot object detection (ClickBot). ClickBot operates directly from detection, learning visual control and manipulating new objects from a few clicks of annotation.

In Section 3.1, we detail how ClickBot generates training data for few-shot detection while completing tasks. In Section 3.2, we derive a visual servo controller that learns to use detection in real time. Finally, in Section 3.3, we introduce three more detection-based tasks to complete manipulation.

3.1. Task-Focused Few-Shot Object Detection

ClickBot learns to complete tasks using few-shot object detection (FSOD). FSOD typically uses an existing set of annotated examples for detection. In contrast, ClickBot collects data while performing detection-based tasks and only requests annotation if it is needed to complete a task. We call this approach task-focused few-shot object detection.

We demonstrate our approach using mobile manipulation tasks. Tasks include finding an object, moving to an object, estimating object depth, grasping an object, and placing a grasped object at a goal location. All tasks use a detection model based on robot-collected, few-shot examples.

Task-Focused Data Collection. We detail the general task-focused few-shot learning process using the object finding task (Find). First, we choose a set of object classes $O$ with corresponding few-shot detection model $D$. To find objects, ClickBot moves a camera through a series of search poses, runs $D$, and saves each pose image $I$ during the task. If $D$ detects an object, the Find task is considered a success and ClickBot can continue to another task. However, if $D$ does not detect any objects, there could be false negatives in the saved images that require an update (e.g., for new objects). To avoid future errors, ClickBot asks for guidance.

Few-Shot Annotation. We provide guidance using a custom graphical user interface for few-shot annotation. First, ClickBot shows a user each task image $I$ and asks if an object from $O$ is present. Next, if an object is present, the user draws a bounding box around each object from $O$, adding to a set of task-focused few-shot learning examples $E$ (see examples in Fig. 3). Images without objects can optionally be added to $E$ as true negatives. After user guidance, ClickBot uses $E$ to update $D$ then restarts the task. Notably, $O$, $D$, and $E$ can be task-specific or shared across multiple tasks.

As we will show in Section 4, task-focused FSOD enables ClickBot to gather data, learn new objects, and improve detection for a variety tasks with minimal annotation.
3.2. Learning Detection-Based Visual Servo Control

To move ClickBot to objects (the Move task), we develop a detection-based approach to visual servo control (VS). Using a camera and its kinematic position in the robot frame, ClickBot learns to adjust the camera to a desired pose relative to detected objects, thereby enabling ClickBot to position itself for other downstream manipulation tasks.

Detection-Based Image Features. Using detection model \( D \), input image \( I \), and a target object class label \( l \in O \), we define detection-based image features \( s \in \mathbb{R}^2 \) as

\[
s(D(I), l, s_{t-1}) := [s_x, s_y]^T, \tag{1}
\]

where \( D(I) \) outputs a set of bounding boxes with class labels, labels other than \( l \) are ignored, \( s_{t-1} \) represents \( s \) from the previous time step, and \( s_x, s_y \) denote the two image coordinates of the target object’s bounding box center.

We track \( s_{t-1} \) in (1) for two reasons. First, if there are multiple boxes with label \( l \), we select the closest match to \( s_{t-1} \) for stability. Second, we use \( \|s - s_{t-1}\|_L^1 \) to check if \( s \) indicates a physically improbable discontinuity in object position. If there is a discontinuity or detections start missing, ClickBot requests guidance on image \( I \) using the few-shot annotation process in Section 3.1.

Visual Servo Feedback Control. We use detection-based features \( s \) (1) to find our VS image feature error \( e \) as

\[
e = s - s^* = [s_x - s_x^*, s_y - s_y^*]^T, \tag{2}
\]

where \( s^* \in \mathbb{R}^2 \) is the vector of desired feature values. We also use \( s^* \) to initiate \( s \) at \( t = 0 \) as \( s(D(I), l, s^*) \), which starts VS on the target object closest to the desired position.

Standard VS [8] relates image features \( s \) to six-degrees-of-freedom (6DOF) camera velocity \( v \) using \( s = L_a v \), where \( L_a \in \mathbb{R}^{2 \times 6} \) is called the feature Jacobian. In this work, we use a constant \( s^* \) (i.e., \( s^* = 0 \)), which implies that \( e \) (2) also relates to \( v \) as \( e = s = L_a v \). Using this relationship, we control \( v \) to minimize \( e \) with

\[
v = -\lambda \hat{L}_a^+ e, \tag{3}
\]

where \( \hat{L}_a^+ \in \mathbb{R}^{6 \times 2} \) is the estimated pseudoinverse of \( L_a \) and \( \lambda \) ensures an exponential decoupled decrease of \( e \). If ClickBot decreases \( e \) below a threshold to accurately position itself relative to an object, the Move task is a success.

Learning Visual Control. In real VS experiments, it is impossible to know the exact feature Jacobian \( L_a \) [8]. Instead, some VS work [24, 27] estimates the feature Jacobian directly from observations using a Broyden update [4]. Recent work [15] modifies a Broyden update to estimate the pseudoinverse feature Jacobian by including a logical matrix that specifies which features and actuators are related.

In this work, we modify a simpler method from Broyden’s original paper [4, (4.12)] to directly estimate \( \hat{L}_a^+ \) (3). Like recent work [15], we use a logical matrix, however, our matrix \( H \) alternatively specifies which features and 6DOF camera velocities are related. We define our update as

\[
\hat{L}_a^+_{t+1} := \hat{L}_a^+_{t} + \alpha \frac{(\Delta x - \hat{L}_a^+_{t} \Delta e e^T)}{\Delta e e^T} \circ H, \tag{4}
\]

where \( \alpha \in \mathbb{R} \) determines the update speed, \( \Delta x = x_t - x_{t-1} \) is the change in 6DOF camera position since the last update, \( \Delta e = e_t - e_{t-1} \) is the change in error, and the element-wise product with logical matrix \( H \in \mathbb{R}^{6 \times 2} \) determines which \( \hat{L}_a^+ \) elements can update. Notably, we use \( H \) to prevent association between unrelated elements in \( v \) and \( e \) (3).

ClickBot learns to relate camera motion to detection-based features using (4). In plain words, ClickBot moves a camera (\( \Delta x \)), observes corresponding changes in detection-based error (\( \Delta e \)), then updates its learned motion-detection model (\( \hat{L}_a^+ \)) based on the difference between the actual (\( \Delta x \)) and predicted (\( \hat{L}_a^+ \Delta e \)) change in camera position.

As we will show in Section 4, ClickBot learns detection-based VS from a single annotated example and 13.3 seconds of motion, enabling other downstream manipulation tasks.

3.3. Final Detection-Based Tasks for Manipulation

The remaining detection-based tasks focus on completing the overall goal of mobile manipulation, namely, estimating object depth (Depth), grasping an object (Grasp), and placing a grasped object at a goal location (Place). As in (1), detection model \( D \) is filtered for a target object and, if detections start missing, ClickBot can request guidance.

Detection-Based Depth Estimation. After using visual control to center its camera on an object, ClickBot estimates that object’s depth. Recent work [17] introduces a method (BoXs) to estimate depth by comparing changes in bounding box height and width (i.e., optical expansion) to known kinematic changes in camera position. Motivated by prior segmentation-based work [16, Section 6.2], we adopt BoXs into an active perception framework, whereby ClickBot actively tracks the convergence of the depth estimate while collecting data and approaching the object. Furthermore, ClickBot increases the depth estimate’s reliability by
requesting guidance if detections are missing. Once ClickBot is within an estimated 0.2 m, it initiates grasping.

Detection-Based Grasping. After centering the camera on an object and estimating its depth, ClickBot uses detection-based grasping. Using detection, ClickBot approximates objects as cylinders [21], but increases the accuracy of this approximation by running detection while rotating its camera to find the best fit between the bounding box and object. Bounding boxes are rectangular, so ClickBot only needs \( \frac{\pi}{2} \) rad of camera rotation to find the best fit; note that 1) the height at any angle \( \theta \) is the same as the width at \( \theta + \frac{\pi}{2} \) and 2) the box dimensions at \( \theta \) and \( \theta + \pi \) are the same.

After rotation and detection, ClickBot uses the bounding box with the overall minimum height or width for grasp planning. ClickBot uses an antipodal grasp (a parallel grasp closing on two points) rotated to align with the minimum height or width at the box’s center (see Fig. 4). Basically, ClickBot uses the narrowest set of detection-based parallel grasp points and grasps at the object’s center for balance.

Using the detection-based grasp plan, ClickBot moves its gripper to the object’s estimated depth and applies a force-based parallel grasp. Next, ClickBot lifts the object while continuing to apply force. If ClickBot’s gripper fingers remain separated by the object, the grasp is complete, and ClickBot can now place the object at a goal location.

Detection-Based Placement of Objects. Manipulation methods typically use predetermined goal locations to place objects [44, 67]. On the other hand, ClickBot can use detection to place objects at flexible locations. Placement locations (e.g., bins) are distinct from our set of graspable objects \( O \), but ClickBot uses the same few-shot process to learn placement object classes \( O_p \). Notably, we designate each object in \( O \) to one or more locations in \( O_p \).

After grasping an object, ClickBot uses search poses to find a suitable placement location. If no location is found, ClickBot asks for guidance (see Find task in Section 3.1). If a placement location is found, ClickBot centers the grasped object over the detected placement location, then releases the object, completing the overall mobile manipulation task.

4. Experimental Results

4.1. Setup

Overview of Experiments. In Section 4.2, ClickBot learns detection-based visual servo control, which ClickBot then uses in all remaining experiments. In Section 4.3, we compare ClickBot to previous work using a mobile visual servo control and depth estimation benchmark. In Section 4.4, we evaluate task-focused learning and grasping in cluttered scenes. In Section 4.5, we evaluate mobile manipulation using random object and placement locations. Finally, in Section 4.6, we provide results for our new Task-Focused Few-Shot Object Detection (TFOD) benchmark.

Detection Model and Training. Similar to other few-shot object detection work [9, 59], we use a fine-tuning approach based on Faster R-CNN [53]. Faster R-CNN runs in real time, has improved since its original publication, and is particularly accurate for small objects [7]. For replicability, we use the same Faster R-CNN configuration as Detectron2 [63] with ResNet 50 [23] pre-trained on ImageNet [10] and a FPN [37] backbone trained on MS-COCO [39].

We use a relatively high 0.9 confidence score threshold, which significantly decreases false positives at the cost of increasing ClickBot requests for guidance after false negatives. Using robot-collected, task-focused few-shot annotation \( E \) (Section 3.1), we fine tune the baseline model for 1,000 training iterations, which takes less than four minutes using a standard workstation and GPU (GTX 1080 Ti).

Robot and Camera Hardware. For our robot experiments, we use a Toyota Human Support Robot (HSR) [65]. For detection, HSR uses an end effector-mounted wide-angle grasp camera, which moves with a 4DOF manipulator arm mounted on a torso with prismatic and revolute joints. As shown in Fig. 1 and Fig. 4, we typically point the grasp camera at the ground. In Section 4.5, HSR also uses a head-mounted Xtion RGBD camera, which moves using a 2DOF gimbal. Both cameras stream 640×480 RGB images.

For mobility, HSR uses a differential drive base. HSR has a torso revolute joint atop the base, so we control HSR as an omnidirectional robot (i.e., 3DOF ground-plane translation and rotation). To command camera velocities \( v \) (3), we use quadratic programming [55] with \( \lambda = 1 \) in (3), but any velocity controller is applicable. To grasp objects, HSR uses an end effector-mounted parallel gripper with series elastic fingertips, which have a 135 mm maximum width.

4.2. Learning Visual Servo Control from One Click

ClickBot learns detection-based visual servo control using camera motion and detection with our Broyden update (4). For camera motion (\( \Delta x \)), ClickBot repeats eight motions comprising the permutations of \((-5, 0, 5)\) cm across the \( x \) and \( y \) axes (e.g., \( x = -5, y = 5 \)). This motion sequence
Table 2. Task-Focused Few-Shot Annotation Results. All results are the mean of corresponding trials (individual results in supplementary material). Clicks are the number of annotated bounding boxes, which each require 7 s (see user study [28]). CPU refers to training time.

| Task-Focused Learning Experiment | Number of Task-Focused Annotated Examples Generated (E) | Requirements Per Object Class |
|---------------------------------|--------------------------------------------------------|-----------------------------|
|                                 | Find | Move | Depth | Grasp | Total | Annotation | Robot | CPU |
| Learning Visual Control         | 1.0  | 0.0  | N/A   | N/A   | 1.0   | 1.0        | 7.0   | 13.3 | 227 |
| Visual Servo and Depth Benchmark| 1.0  | 0.9  | 3.1   | N/A   | 5.0   | 3.7        | 26.0  | 20.2 | 383 |
| Grasping with Prior Annotation  | 0.3  | 0.3  | 1.3   | 2.8   | 4.5   | 3.4        | 23.9  | 29.1 | 343 |
| Grasping in Clutter with Prior Annotation | 0.5  | 0.8  | 0.0   | 2.3   | 3.5   | 2.7        | 18.7  | 23.2 | 287 |
| Grasping                        | 1.0  | 0.8  | 2.5   | 3.8   | 8.0   | 6.0        | 42.0  | 51.4 | 615 |
| Grasping in Clutter             | 1.0  | 2.0  | 4.3   | 3.3   | 10.5  | 7.5        | 52.5  | 67.3 | 811 |

Figure 5. Learned $\hat{L}_s^+$ Parameters for Visual Servo Control. ClickBot learns detection-based visual servo control in 13 updates. We also plot the camera motions in the supplementary material.

is varied yet cycles back through the initial camera position.

For detection, we use the racquetball from the YCB Object Dataset [6] as the target object. Initially, the object is unknown, so ClickBot immediately requests few-shot annotation (Section 3.1). After providing one bounding box (i.e., one click of annotation), ClickBot retrains its detection model $D$ (1) and requires no additional annotation. Notably, ClickBot learns from detection error changes ($\Delta e$ (4)), so constant desired values $s^*$ in $e$ (2) are arbitrary.

For our Broyden update, we initialize (4) with $\hat{L}_s^+ |_{t=0} = 0_{6 \times 2}$, $\alpha = 0.5$, and $H = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^\top$. This $H$ couples image features $s_x$ and $s_y$ ($e$ (2)) with $x$- and $y$-axis camera velocities ($v$ (3)) respectively. With the racquetball still in view, ClickBot starts the motion sequence, tracks detection changes, and updates $\hat{L}_s^+$ after each motion. If $\| \hat{L}_s^+ |_{t+1} - \hat{L}_s^+ |_t \|_{L_1} < 10^{-6}$, the learning task is complete.

Results. ClickBot completes this active learning task in a single experiment, requiring only 13.29 s with 13 Broyden updates to learn visual control. Immediately afterward, we push the racquetball and ClickBot follows it, confirming that the learned visual controller (3) is a success. We provide a video of this experiment in the supplementary material. Fig. 5 plots the learned $\hat{L}_s^+$ values for each update. Table 2 shows the overall learning costs for visual control.

4.3. Mobile Robot Visual Servo Benchmark

We evaluate ClickBot’s learned visual control and active depth estimation using an existing Visual Servo (VS) and Depth Estimation (DE) Benchmark [15]. The benchmark uses eight consecutive trials with the YCB objects [6] shown in Fig. 6. Each trial has three in-view objects supported at different heights: 0.0 m, 0.125 m, and 0.25 m above the ground (see Fig. 7, top). VS is a success if a robot locates and centers on an object for DE. DE is a success if a robot advances without collision then closes its gripper on the object without hitting the underlying surface.

In addition to the benchmark, we evaluate ClickBot’s task-focused learning. ClickBot learns new objects (O) for each trial using the Find, Move, and Depth tasks from Section 3. Starting without prior annotation, ClickBot initially requests guidance for Find. Subsequently, ClickBot returns to the Find pose after any further vision updates.

Using our approach, Clickbot centers VS on an object until $e$ (3) is below 10 pixels, approaches until DE is within 0.2 m, then closes it’s gripper at the estimated depth. Each object is removed after its first full attempt, i.e., VS, DE, and grasp closure without a guidance request.

Results. We provide the benchmark results in Table 3. ClickBot achieves the best result with a perfect VS score.
and a DE success rate increase over prior work from 42% to 67%. ClickBot is perfect on the Food set but leaves room to improve DE on the Tool and Kitchen sets by 50%.

We provide task-focused learning results in Table 2. ClickBot averages 5 requests per trial with more guidance for Depth than Find and Move combined. The task-focused learning goal is to improve performance on difficult tasks, so we are encouraged that ClickBot identifies and requests annotation primarily on the task that requires improvement.

We compare annotation costs in Table 3. Segmentation masks require 54 s of annotation [28], so we estimate that VOSVS uses 540 s of annotation per object. Alternatively, ClickBot uses 26 s of task-focused annotation per object.

### 4.4. Learning to Grasp Objects in Clutter

We evaluate ClickBot’s detection-based grasping (Grasp) by modifying the consecutive trials in Section 4.3. First, for the Tool and Food sets, we add the Grasp task after VS and DE. Notably, HSR cannot grasp some objects in the Kitchen and Shape sets, either because they are too heavy (Skillet with Lid) or too low to the ground (4 mm high RGB camera). Grasp is a success if ClickBot picks up the object and maintains its grasp until placing the object in a nearby bin. As an added challenge, we repeat the VS, DE, and Grasp trials in clutter (see Fig. 7, middle).

For the Grasp trials, we also test two ablative configurations for task-focused learning. First, we modify ClickBot to start with prior annotation from Section 4.3 when learning the Grasp task. Subsequently, when grasping in clutter, prior annotation also includes the non-cluttered Grasp trial. For a second configuration without task-focused learning, ClickBot only uses prior annotation with a 0.1 confidence threshold to increase the likelihood of detecting objects.

### Results

We provide Grasp trial results in Table 4. The standard configuration achieves the best cluttered grasp rate of 88%. From the learning results in Table 2, we attribute this to the standard configuration having the most annotation requests in clutter, which improves performance for that particular task and setting. Still, the standard configuration uses no prior annotation, so it is efficient overall.

Across all tasks and settings, using task-focused learning improves performance. Both ClickBot configurations using task-focused learning were perfect for VS and DE regardless of clutter. Similar to Section 4.3, ClickBot primarily requests annotation for tasks that require improvement, particularly when using prior annotation, which focuses almost all requests on grasping. Notably, this grasp annotation also improves performance for other tasks, such as DE.

### 4.5. Learning Pick-and-Place at Dynamic Locations

We use mobile experiments with scattered objects to test ClickBot’s detection-based object placement. First, we scatter cups for grasping and bins for placement across the floor. Next, ClickBot finds and grasps the closest cup then finds the closest bin for cup placement. As in previous experiments, we use the Find task. However, now ClickBot uses a head-mounted RGBD camera with detection to find and map object locations. In this way, ClickBot can...
Figure 7. Experimental Results. The first column shows the Find task for three rows of examples. In the servo benchmark (top), ClickBot centers on the plate then estimates its depth using camera motion and optical expansion. After estimating the spring clamp’s depth (middle), ClickBot uses active detection-based grasping to remove it from clutter. In dynamic pick-and-place (bottom), ClickBot uses detection with an RGBD camera to locate and grasp scattered objects and similarly uses detection to find a suitable placement location.

find a place for grasped objects while its grasp camera is blocked (see Fig. 7, bottom). Furthermore, we can completely remove the Depth task by using an RGBD map, which demonstrates the modularity of ClickBot tasks.

Results. ClickBot successfully grasps a scattered cup after two task-focused examples. After two more examples, ClickBot is able to place the cup in a bin (we show this result in Fig. 7). We attribute four-shot pick-and-place to removing the Depth task, which reduces annotation even with the Place task added. ClickBot does occasionally request more annotation as cups and bins are repeatedly scattered.

Subsequently, ClickBot also learns to retrieve thrown cups and return them to a moving person using eight more annotations (see Fig. 3 right). We provide an example RGBD map and videos of both types of dynamic pick-and-place experiments in the supplementary material.

4.6. Task-Focused Few-Shot Detection Benchmark

Admittedly, this work will improve with more advanced few-shot object detection (FSOD) algorithms. Accordingly, we are introducing the Task-Focused Few-Shot Object Detection (TFOD) benchmark. The TFOD benchmark is easily configurable for $k = 1, 2, 4$ annotated bounding boxes across 12 YCB [6] object classes, and our test set includes challenging examples in cluttered settings. The TFOD benchmark also makes robot-collected data and corresponding annotations publicly available for research, which will encourage FSOD innovation in this new task-focused detection setting for robot manipulation.

Results. We provide baseline TFOD results in Table 5. We see opportunity for innovation across all settings, especially for small objects (APs) and one- or two-shot detection.

Table 5. Task-Focused Few-Shot Object Detection Benchmark. We introduce a new benchmark with standard MS-COCO [39] AP metrics and $k$ task-focused annotations across 12 object classes.

| Method  | $k$ | AP | AP50 | AP75 | APs | APm | API |
|---------|-----|----|------|------|-----|-----|-----|
| ClickBot | 1   | 13.6 | 19.5 | 18.1 | 0.0 | 33.7 | 23.2 |
| ClickBot | 2   | 17.7 | 23.3 | 21.6 | 0.0 | 27.4 | 21.0 |
| ClickBot | 4   | 33.7 | 45.0 | 39.8 | 0.5 | 52.0 | 43.4 |

5. Conclusions

We develop a method of detection-based mobile robot manipulation that learns to perform tasks for new objects and settings using few-shot examples. Furthermore, our robot collects data while performing tasks and, if encountering errors, selects its own few-shot examples for annotation to improve performance for difficult tasks. In this way, our robot performs tasks but continues to learn and adapt to evolving tasks, objects, and environments.

We evaluate our approach using a mobile robot in a variety of settings. First, our robot learns detection-based visual servo control in 13.3 s using a single annotated example. Using this visual control, our robot achieves state-of-the-art results on an existing visual servo control and depth estimation benchmark. Next, our robot learns to grasp objects in clutter with as few as four few-shot examples. Finally, our robot learns to pick-and-place objects at dynamic locations.

Innovations in detection will improve our current results. Thus, we are releasing a new benchmark for few-shot detection to enable future work to evaluate and improve performance in this task-focused setting for robot manipulation.
Acknowledgements

Toyota Research Institute provided funds to support this work.

References

[1] Pooya Abolghasemi, Amir Mazaheri, Mubarak Shah, and Ladislau Boloni. Pay attention! - robustifying a deep visuomotor policy through task-focused visual attention. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019. 3

[2] Saif Alabachi, Gita Sukthankar, and Rahul Sukthankar. Customizing object detectors for indoor robots. In IEEE International Conference on Robotics and Automation (ICRA), 2019. 2, 3

[3] Jeannette Bohg, Karol Hausman, Bharath Sankaran, Oliver Brock, Danica Kragic, Stefan Schaal, and Gaurav S. Sukhatme. Interactive perception: Leveraging action in perception and perception in action. IEEE Transactions on Robotics (TRO), 2017. 1

[4] C. G. Broyden. A class of methods for solving nonlinear simultaneous equations. Mathematics of Computation, 19(92):577–593, 1965. 4

[5] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018. 2

[6] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar. Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set. IEEE Robotics Automation Magazine, 22(3):36–52, Sep. 2015. 6, 7, 8

[7] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In European Conference on Computer Vision (ECCV), 2020. 1, 2, 5

[8] F. Chaumette and S. Hutchinson. Visual servo control. i. basic approaches. IEEE Robotics Automation Magazine, 2006. 3, 4

[9] Hao Chen, Yali Wang, Guoyou Wang, and Yu Qiao. Lstd: A low-shot transfer detector for object detection. AAAI Conference on Artificial Intelligence (AAAI), 2018. 1, 2, 3, 5

[10] J. Deng, W. Dong, R. Socher, L. J. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009. 5

[11] Mark Everingham, S. M. Ali Eslami, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. International Journal of Computer Vision (IJCV), 2015. 1, 2

[12] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. International Journal of Computer Vision (IJCV), 2010. 1, 2

[13] Zhibo Fan, Yuchen Ma, Zeming Li, and Jian Sun. Generalized few-shot object detection without forgetting. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021. 2

[14] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn. In IEEE/CVF International Conference on Computer Vision (ICCV), 2019. 2

[15] Brent Griffin, Victoria Florence, and Jason J. Corso. Video object segmentation-based visual servo control and object depth estimation on a mobile robot. In IEEE Winter Conference on Applications of Computer Vision (WACV), 2020. 3, 4, 6, 7

[16] Brent A. Griffin and Jason J. Corso. Learning object depth from camera motion and video object segmentation. In European Conference on Computer Vision (ECCV), 2020. 4

[17] Brent A. Griffin and Jason J. Corso. Depth from camera motion and object detection. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021. 2, 4, 12

[18] N. Guenard, T. Hamel, and R. Mahony. A practical visual servo control for an unmanned aerial vehicle. IEEE Transactions on Robotics (TRO), 2008. 3

[19] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019. 1, 2

[20] Guangxing Han, Yicheng He, Shiuyuan Huang, Jiawei Ma, and Shih-Fu Chang. Query adaptive few-shot object detection with heterogeneous graph convolutional networks. In IEEE/CVF International Conference on Computer Vision (ICCV), 2021. 2

[21] Kensuke Harada, Kazuyuki Nagata, Tokuo Tsuji, Natsuki Yamanobe, Akira Nakamura, and Yoshihiro Kawai. Probabilistic approach for object bin picking approximated by cylinders. In IEEE International Conference on Robotics and Automation (ICRA), 2013. 5

[22] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In IEEE International Conference on Computer Vision (ICCV), 2017. 2

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. 5

[24] K. Hosoda and M. Asada. Versatile visual servoing with simultaneous equations. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 1994. 4

[25] Hanzhe Hu, Shuai Bai, Aoxue Li, Jinshi Cui, and Liwei Wang. Dense relation distillation with context-aware aggregation for few-shot object detection. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021. 2

[26] S. Hutchinson, G. D. Hager, and P. I. Corke. A tutorial on visual servo control. IEEE Transactions on Robotics and Automation (TRO), 1996. 3

[27] M. Jagersand, O. Fuentes, and R. Nelson. Experimental evaluation of uncalibrated visual servoing for precision manipulation. In International Conference on Robotics and Automation (ICRA), 1997. 3, 4

[28] Suyog Dutt Jain and Kristen Grauman. Predicting sufficient annotation strength for interactive foreground segmen-
Krishna Shankar, Joel W. Burdick, and Nicolas H. Hudson. A quadratic programming approach to quasi-static whole-body manipulation. In Algorithmic Foundations of Robotics XI, 2015.

Bo Sun, Banghuai Li, Shengcai Cai, Ye Yuan, and Chi Zhang. Fsce: Few-shot object detection via contrastive proposal encoding. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021.

Zhi Tian, Chanhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In IEEE/CVF International Conference on Computer Vision (ICCV), 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in Neural Information Processing Systems (NIPS), 2017.

Xin Wang, Thomas Huang, Joseph Gonzalez, Trevor Darrell, and Fisher Yu. Frustratingly simple few-shot object detection. In International Conference on Machine Learning (ICML), 2020.

Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Meta-learning to detect rare objects. In IEEE/CVF International Conference on Computer Vision (ICCV), 2019.

Aming Wu, Yahong Han, Linchao Zhu, and Yi Yang. Universal-prototype enhancing for few-shot object detection. In IEEE/CVF International Conference on Computer Vision (ICCV), 2021.

Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

Yang Xiao and Renaud Marlet. Few-shot object detection and viewpoint estimation for objects in the wild. In European Conference on Computer Vision (ECCV), 2020.

Takashi Yamamoto, Koji Terada, Akiyoshi Ochiai, Fuminori Saito, Yoshiaki Asahara, and Kazuto Murase. Development of human support robot as the research platform of a domestic mobile manipulator. ROBOMECH Journal, 2019.

Xiaopeng Yan, Ziliang Chen, Anni Xu, Xiaoxi Wang, Xiaodan Liang, and Liang Lin. Meta r-cnn: Towards general solver for instance-level low-shot learning. In IEEE/CVF International Conference on Computer Vision (ICCV), 2019.

Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas A. Funkouser. Tossingbot: Learning to throw arbitrary objects with residual physics. In Proceedings of Robotics: Science and Systems (RSS), 2019.

Gongjie Zhang, Kaiwen Cui, Rongliang Wu, Shijian Lu, and Yonghong Tian. Pnpdet: Efficient few-shot detection without forgetting via plug-and-play sub-networks. In IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2021.

Weilin Zhang and Yu-Xiong Wang. Hallucination improves few-shot object detection. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021.
**Supplementary Material:**

**Task-Focused Few-Shot Object Detection for Robot Manipulation**

**Motivation for TFOD Benchmark.** As we discuss in Sections 1 and 2, the current evaluation paradigm for few-shot object detection uses standard images from previous object detection datasets. However, after extensive testing of off-the-shelf detectors in the lab, we find that detection is not reliable outside of its initial training setting for many robot tasks. Maybe this result is obvious, but the question of how to apply few-shot detection to robotics and how to collect supplementary training data for a new setting has drawn scant attention.

While our paper primarily focuses on these two questions, we find that much more innovation is possible for few-shot detection algorithms in robotics. However, many researchers do not have a robot or even access to data to evaluate few-shot detection algorithms in a robotics setting, which provides varying image characteristics for consistent objects. Thus, we are introducing the TFOD Benchmark to provide this evaluation and guide our community toward increasingly reliable few-shot detection for robotics.

**Per-Object TFOD Benchmark Results.** We provide per-object Task-Focused Few-Shot Object Detection (TFOD) benchmark results in Fig. 8, which correspond to the ClickBot $k = 1, 2, 4$ few-shot example configurations in Table 5. As in Table 5, we find opportunities for innovation across all settings, especially one- or two-shot detection. The Wood, Box of Sugar, and Chips Can are particularly inaccurate for $k < 4$. Meanwhile, the $k = 4$ configuration has the best performance for all objects with the exception of Gelatin.

With future few-shot object detection research and evaluation in this new task-focused setting for robot manipulation, we expect performance to improve across all objects and few-shot configurations, which will improve robot task performance and reduce overall annotation requirements.

**Camera Movement and Learned Visual Servo Control.** We plot the camera movements for learning visual servo control in Fig. 9 with the corresponding learned parameters originally shown in Fig. 5.

For camera motion ($\Delta x$), ClickBot repeats eight movement commands comprising the permutations of $\{-5, 0, 5\}$ cm across the $x$ and $y$ axes (e.g., $x = -5$, $y = 5$ for the second Broyden update). However, ClickBot’s base movements are imprecise for small motions, so the actual measured movement distance we use for the update is slightly less (e.g., Base Forward = 2.7 cm, Base Lateral = 2.5 cm of actual motion for the second update). Nonetheless, the actual motion profile is sufficient to learn ClickBot’s visual control, which we use for all experiments in Section 4.

**Depth Estimate Convergence.** In Section 3.3, we introduce ClickBot’s active depth estimation, which continually processes incoming data while approaching objects for grasping. We provide an example depth convergence plot in Fig. 10, which corresponds to the Chips Can result in Fig. 1. ClickBot advances in 0.05 m increments, so the depth estimate generally completes with the object between 0.15 m to 0.2 m away. In this example, after the grasp camera moves 0.15 m, the Chips Can’s final estimated depth is 0.18 m, which leads to a successful grasp of the Chips Can.

As discussed in Section 3.3, ClickBot estimates object depth from detection by comparing changes in bounding box size (i.e., optical expansion) with the corresponding camera movement, which we obtain using robot kinematics. The Box$_{LS}$ solution [17] uses all available observations in a least-squares formulation, thus, our active depth estimate generally improves as more data are collected. Finally, the depth estimate’s accuracy significantly improves as the object gets closer and exhibits more rapid optical expansion.

**Individual Trial Results for Task-Focused Annotation.** We provide the task-focused few-shot annotation results for individual trials in Table 6. All Mean results are the same as those originally shown in Table 2. Remarkably, no experiment configuration uses more than a minute of human annotation time per object, which is approximately the same amount of time required to generate a single segmentation mask and much less than the time required for a 3D model.

We discuss a few notable individual trial results. For the Visual Servo and Depth Benchmark on the Food: Chips Can, Potted Meat, Plastic Banana trial, ClickBot learns the Find, Move, and Depth tasks for all objects without prior annotation using 3 task-focused examples. For Grasping in Clutter with Prior Annotation on the Food: Box of Sugar, Tuna, Gelatin trial, ClickBot requires only 1 task-focused Move example to transfer learning from the prior grasp task to learn grasping in clutter. Finally, for Grasping in Clutter on the Food: Chips Can, Potted Meat, Plastic Banana trial, ClickBot learns all tasks for all objects in a cluttered setting without prior annotation using 7 task-focused examples.

**ClickBot-Generated Map for Dynamic Pick-and-Place.** We provide an example ClickBot-generated map in Fig. 11, which corresponds to the dynamic pick-and-place result originally shown in Fig. 7.

ClickBot uses the same few-shot detection model with its head-mounted RGBD camera, which enables ClickBot to map any RGB-based bounding box to a median 3D point using the corresponding depth image. Using this map for the Find task, ClickBot quickly identifies the closest grasp object and subsequent placement location even after a grasped object is blocking ClickBot’s grasp camera.

**Experiment Videos** are provided at [https://youtu.be/rSMWf7osI4w](https://youtu.be/rSMWf7osI4w). These include an overview of
Figure 8. **Per-Object Task-Focused Few-Shot Object Detection (TFOD) Benchmark Results.** All TFOD test results correspond to the baseline ClickBot method in Table 5. There are many opportunities for future improvements, especially for $k = 1, 2$ few-shot examples.

Figure 9. **Learning $\hat{L}_k$ Parameters for Visual Servo Control with Camera Movement.** ClickBot learns detection-based visual servo control in 13.3 seconds after 13 camera movements (top) and Broyden updates (4) (bottom). Subsequently, ClickBot uses this learned visual control in all remaining experiments.

Detection-based manipulation, the learning visual servo control experiment from Section 4.2, and two example dynamic pick-and-place experiments from Section 4.5.

Figure 10. **Depth Convergence.** We plot the depth estimate corresponding to the Chips Can result in Fig. 1. ClickBot actively estimates an object’s depth as it approaches for grasping. The depth convergences as the camera moves closer and collects more data.

**Configuring ClickBot after Learning.** After ClickBot learns manipulation, it can be useful to reconfigure ClickBot into a “sentry” mode. In sentry mode, ClickBot intermittently uses the Find task to search for relevant objects but no longer assumes that the absence of detections indicates a false negative. Thus, ClickBot will complete tasks when needed without requesting unnecessary annotation.
Table 6. **Task-Focused Few-Shot Annotation Results (Individual Trials)**. All results are from a single consecutive set of trials. Clicks are the number of annotated bounding boxes, which each require 7 seconds (see user study [28]). Note that Clicks per Annotated Example varies with the number of in-view objects. CPU refers to training time. Mean results are the same as those originally shown in Table 2.

| Task-Focused Learning Experiment Trial | Number of Task-Focused Annotated Examples Generated ($E$) | Requirements Per Object Class |
|---------------------------------------|----------------------------------------------------------|--------------------------------|
|                                       | Find | Move | Depth | Grasp | Total | Clicks | Time (seconds) | CPU |
| Learning Visual Control (Section 4.2) |      |      |       |       |       | 1.0    | 7.0            | 13.3 | 227 |
| Tool: Power Drill, Marker, Padlock    | 1    | 1    | 3     | N/A   | 5     | 3.7    | 25.7          | 22.9 | 381 |
| Tool: Wood, Spring Clamp, Screwdriver | 1    | 1    | 2     | N/A   | 4     | 3.7    | 25.7          | 10.9 | 309 |
| Food: Chips Can, Potted Meat, Plastic Banana | 1 | 0    | 2     | N/A   | 3     | 2.7    | 18.7          | 9.3  | 233 |
| Tool: Box of Sugar, Tuna, Gelatin    | 1    | 1    | 4     | N/A   | 6     | 3.7    | 25.7          | 30.0 | 460 |
| Kitchen: Mug, Softscrub, Skillet with Lid | 1  | 1    | 5     | N/A   | 7     | 5.0    | 35.0          | 30.0 | 536 |
| Kitchen: Plate, Spatula, Knife       | 1    | 0    | 3     | N/A   | 4     | 2.7    | 18.7          | 18.0 | 304 |
| Shape: Baseball, Plastic Chain, Washer | 1  | 2    | 3     | N/A   | 6     | 4.7    | 32.7          | 18.6 | 457 |
| Shape: Stacking Cup, Dice, Foam Brick | 1  | 1    | 3     | N/A   | 5     | 3.7    | 25.7          | 21.5 | 387 |
| Mean                                 | 1.0  | 0.9  | 3.1   | N/A   | 5.0   | 3.7    | 26.0          | 20.2 | 383 |

Visual Servo and Depth Benchmark (Section 4.3)

| Tool: Power Drill, Marker, Padlock    | 0    | 0    | 0     | 4     | 4     | 3.0    | 21.0          | 27.3 | 307 |
| Tool: Wood, Spring Clamp, Screwdriver | 0    | 0    | 1     | 2     | 3     | 2.7    | 18.7          | 21.9 | 231 |
| Food: Chips Can, Potted Meat, Plastic Banana | 1  | 0    | 1    | 3     | 5     | 3.7    | 25.7          | 32.3 | 378 |
| Food: Box of Sugar, Tuna, Gelatin    | 0    | 1    | 3     | 2     | 6     | 4.3    | 30.3          | 35.0 | 457 |
| Mean                                 | 0.3  | 0.3  | 1.3   | 2.8   | 4.5   | 3.4    | 23.9          | 29.1 | 343 |

Grasping with Prior Annotation (Section 4.4)

| Tool: Power Drill, Marker, Padlock    | 0    | 0    | 0     | 3     | 4     | 2.7    | 18.7          | 32.6 | 374 |
| Tool: Wood, Spring Clamp, Screwdriver | 0    | 2    | 0     | 3     | 5     | 3.7    | 25.7          | 34.6 | 387 |
| Food: Chips Can, Potted Meat, Plastic Banana | 1  | 0    | 0    | 3     | 4     | 3.3    | 23.3          | 25.7 | 309 |
| Food: Box of Sugar, Tuna, Gelatin    | 0    | 1    | 0    | 0     | 1     | 1.0    | 7.0           | 0.2  | 76  |
| Mean                                 | 0.5  | 0.8  | 0.0   | 2.3   | 3.5   | 2.7    | 18.7          | 23.2 | 287 |

Grasping in Clutter with Prior Annotation (Section 4.4)

| Tool: Power Drill, Marker, Padlock    | 1    | 2    | 3     | 5     | 4     | 2.7    | 18.7          | 56.1 | 689 |
| Tool: Wood, Spring Clamp, Screwdriver | 1    | 1    | 2    | 3     | 7     | 6.0    | 42.0          | 38.3 | 543 |
| Food: Chips Can, Potted Meat, Plastic Banana | 1  | 0    | 2    | 3     | 6     | 5.3    | 37.3          | 33.4 | 457 |
| Food: Box of Sugar, Tuna, Gelatin    | 1    | 1    | 4    | 4     | 10    | 6.0    | 42.0          | 73.0 | 770 |
| Mean                                 | 1.0  | 0.8  | 2.5   | 3.8   | 8.0   | 6.0    | 42.0          | 51.4 | 615 |

Grasping (Section 4.4)

| Tool: Power Drill, Marker, Padlock    | 1    | 2    | 5     | 5     | 13    | 10.0   | 70.0          | 97.0 | 1,008 |
| Tool: Wood, Spring Clamp, Screwdriver | 1    | 0    | 4    | 3     | 8     | 5.7    | 39.7          | 60.2 | 614 |
| Food: Chips Can, Potted Meat, Plastic Banana | 1  | 2    | 2    | 7     | 6.0    | 42.0   | 33.0          | 340  | 540 |
| Food: Box of Sugar, Tuna, Gelatin    | 1    | 4    | 6    | 3     | 14    | 8.3    | 58.3          | 79.2 | 1,082 |
| Mean                                 | 1.0  | 2.0  | 4.3   | 3.3   | 10.5  | 7.5    | 52.5          | 67.3 | 811 |

Grasping in Clutter (Section 4.4)
Figure 11. **ClickBot-Generated Map for Pick-and-Place at Dynamic Locations.** In dynamic pick-and-place (bottom), Clickbot uses detection with an RGBD camera to locate and grasp scattered objects (left) and similarly uses detection to find a suitable placement location (right). Here, we show the ClickBot-generated map (top) corresponding to the pick-and-place result originally shown in Fig. 7.