# Mobile Robot Manipulation using Pure Object Detection

Brent Griffin
Agility Robotics
brent.griffin@agilityrobotics.com

## Abstract

*This paper addresses the problem of mobile robot manipulation using object detection. Our approach uses detection and control as complimentary functions that learn from real-world interactions. We develop an end-to-end manipulation method based solely on detection and introduce Task-focused Few-shot Object Detection (TFOD) to learn new objects and settings. Our robot collects its own training data and automatically determines when to retrain detection to improve performance across various subtasks (e.g., grasping). Notably, detection training is low-cost, and our robot learns to manipulate new objects using as few as four clicks of annotation. In physical experiments, our robot learns visual control from a single click of annotation and a novel update formulation, manipulates new objects in clutter and other mobile settings, and achieves state-of-the-art results on an existing visual servo control and depth estimation benchmark. Finally, we develop a TFOD Benchmark to support future object detection research for robotics:* `https://github.com/griffbr/TFOD`.


Figure 1. **Detection-based Manipulation**. Perceiving objects from camera motion and bounding boxes (right), our robot learns to grasp the Chips Can in four robot-collected training examples.

## 1. Introduction

Object detection, i.e., predicting bounding boxes and category labels for objects in an RGB image, has seen remarkable methodological advances [7, 50, 51] thanks to an abundance of annotated training and evaluation data in high-quality datasets [13, 19, 36]. Recently, *few-shot* object detection [10, 58, 65] has emerged as a critical innovation to detect new object classes using a limited subset of annotated examples from existing datasets [28, 56].

Detection also supports many downstream applications [45, 63, 70]. When changing objects, tasks, or environments, however, we find that off-the-shelf detectors are less reliable outside of their initial training setting, causing the subsequent application to fail. Thus, passive learning is not enough, especially for robots, where visual experience is dynamic and interactive [4]. 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 that end, this paper presents *ClickBot*, a robot that learns mobile manipulation for new objects and changing environments using pure object detection (i.e., learning and perceiving objects using only 2D bounding boxes). Our approach relies on two primary contributions.

1) *Detection-Based Manipulation*. We develop a novel set of detection-based tasks to complete mobile robot manipulation. Innovations include a novel update formulation to learn visual servo control, motion-based depth estimation that improves as ClickBot approaches objects, and active multi-view grasp selection. Using our approach, ClickBot manipulates unstructured objects without 3D models using a single RGB camera. To our knowledge, this is the first work to develop end-to-end object manipulation entirely from detection.

2) ***T**ask-Focused **F**ew-Shot **O**bject **D**etection (TFOD)*. We introduce TFOD to learn detection-based tasks for

new objects and settings. Using TFOD, ClickBot automatically performs tasks, collects data, determines if new few-shot examples are required, and, if so, directs annotation toward specific tasks. In practice, TFOD improves performance for difficult or evolving robot tasks while reducing overall annotation costs.

We validate our combined approach in a variety of robot experiments. First, ClickBot learns detection-based visual control on average in less than 14 s and reduces learning variability by 65-85% relative to prior visual servo approaches. Next, ClickBot achieves state-of-the-art results on the VOSVS Benchmark [16], increasing visual servo control and depth estimation performance by 16.7% and 25.0% respectively. Finally, ClickBot learns to grasp objects in clutter (see Figure 1) and cleans up scattered objects with moving placement locations at 124.6 picks-per-hour.

This paper builds a foundation to guide future research and innovation for using few-shot detection algorithms in robotics. However, many researchers do not have a robot or data to evaluate their algorithms in a robotics setting. Thus, as a final contribution, we develop a corresponding TFOD Benchmark. The TFOD Benchmark is configurable for a variety of few-shot object detection settings, includes evaluation across a diverse set of YCB Dataset objects [6] using standard MS-COCO AP metrics [36], and will guide future research toward increasingly reliable detection in this new task-focused setting for robot manipulation.

## 2. Related Work

**Object Detection** is a preliminary process for many methods in our community. Example detection-based methods include segmentation [21], 3D shape prediction [15], depth [17] and pose estimation [45, 63], and single-view metrology [70], to name but a few. In this paper, we introduce a novel detection-based method for mobile robot manipulation that similarly operates directly from object detection.

Learning object detection typically requires a large number of bounding box annotations from a labeled dataset for training and evaluation [13, 36], with some datasets additionally focusing on continuous recognition [37] or multi-view indoor environments [52]. However, static datasets do not account for new objects and settings in the wild.

**Few-Shot Object Detection** (FSOD) addresses part of this limitation by detecting new objects from only a few annotated examples [10]. For evaluation, the first FSOD benchmark [28] uses set splits of $k = 1, 2, 3, 5, 10$ annotated bounding boxes for 5 few-shot objects on the PASCAL VOC Dataset [13] and $k = 10, 30$ for 20 few-shot objects on the MS-COCO Dataset [36]. Subsequent work [56] revises this protocol by randomly selecting few-shot objects and annotated examples with an average evaluation over 40 trials with additional results on the LVIS Dataset [19].

Using these prior benchmarks, FSOD has seen rampant methodological advances. Initial finetuning methods treat FSOD as a transfer learning problem from a large source domain to few-shot objects [10, 56]. Other methods use meta-learning algorithms to learn from existing detectors and quickly adapt to few-shot objects, either by using feature rewieghting schemes [28, 65] or by using model parameter generation from base classes to efficiently learn few-shot objects [58]. Other FSOD approaches include a distance metric learning-based classifier [29], incremental few-shot learning to reduce training requirements [43, 47], *one*-shot detection by matching and aligning target-image-features with query-image-features [44], plug-and-play detectors to maintain known category performance while learning new concepts [67], and an attention-guided cosine margin to mitigate class imbalance [2], to name but a few.

However, generating few-shot examples for new objects or using FSOD to support other applications has drawn scant attention. One recent work collects new detection training data by teleoperating a UAV [3], but this approach uses substantially more training examples than current FSOD methods require ($k \gg 30$) and does not consider applications other than detection. On the other hand, one FSOD method [63] supports viewpoint estimation applications by developing a unified framework that uses arbitrary 3D models of few-shot objects, but this work only detects and estimates viewpoints for objects in existing datasets.

To that end, this paper extends FSOD by improving detection for specific application tasks and collecting new few-shot examples in the wild for new objects and settings using an approach we call ***T**ask-Focused* **F**ew-Shot **O**bject **D**etection (TFOD). Furthermore, rather than trying to predict the best set of few-shot examples *a priori*, we let the robot and difficulty of each task decide, thereby limiting annotation to a few relevant examples. Notably, we use a finetuning FSOD method in our TFOD experiments, but, as we will show, TFOD is generalizable across FSOD methods.

**Visual Servo Control** (VS) uses visual data in a servo loop for robot control. Closed-form VS methods typically relate image features to robot actuators using a *feature Jacobian* [8, 24, 59] with advanced methods learning the feature Jacobian directly on the robot [9, 22, 25]. Closed-form VS can position UAVs [18, 41] or wheeled robots [38, 40] and manipulate objects [25, 30, 57]. Although this early VS work demonstrates the utility of VS, these 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, 31, 48] or in simulation with innovative sim-to-real transfer techniques [27, 46, 71]. However, all of these end-to-end methods learn in a fixed workspace and do not address the challenges of *mobile* manipulation, which includes moving cameras,

changing environments, and dynamic grasp positioning.

To bridge the gap between learned VS and mobile robot applications, in recent work, we developed mobile VS with features based on pre-trained video object segmentation [16]. However, this approach does not learn new objects, tasks, or environments, which would require a substantial cost to annotate new segmentation masks [26]. To that end, this paper develops a novel detection-based approach to mobile VS, which, in comparative experiments, requires less than 5% the annotation cost while significantly improving performance. We also develop a new approach to learn VS using the *pseudoinverse* feature Jacobian, which, relative to prior work, learns VS faster and more consistently. Finally, coupling our mobile VS with TFOD and other tasks lets our robot learn to locate and manipulate new objects efficiently.

## 3. Task-Focused Few-Shot Object Detection

We develop an interactive approach we call **T**ask-Focused **F**ew-Shot **O**bject **D**etection (TFOD) to collect data and learn detection for new objects and applications.

### 3.1. Task and Detection Model

TFOD is generally applicable for any task $T$ and object detection model $D$ that satisfy the following criteria:

1. $T$ observes $n \geq 1$ images $\{I_1, I_2, \ldots, I_n\}$.
2. $D(I)$ outputs a set of bounding boxes with class labels.
3. There are one or more failure criteria $F$ based on $D$.
4. If $F$ occurs, $m \geq 1$ images $\{I_{F_1}, I_{F_2}, \ldots, I_{F_m}\}$ $\in \{I_1, I_2, \ldots, I_n\}$ are saved to log the failure.
5. $D$ can update its output predictions given a set of $p \geq 1$ annotated few-shot examples $E(I_{E_1}, I_{E_2}, \ldots, I_{E_p})$.

Using these definitions, the goal of TFOD is to update $D$ until $T$ is completed without $F$ using the minimum $p$.

In plain words, as our robot attempts difficult or evolving detection-based tasks, its detection model can fail. However, if our robot recognizes a failure ($F$), it is a meaningful opportunity for learning, and our robot saves image data of the failure for annotation ($I_F$). After we provide annotation ($E$), our robot updates its detection model. Notably, criteria for $F$ and selecting $I_F$ can change depending on the specific task, and we provide several examples in Section 4.

### 3.2. Task-Focused Data Collection

Given task $T$ and detection model $D$, our robot performs $T$ using $D(I_1), D(I_2), \ldots, D(I_n)$. If $F$ occurs during $T$, our robot selects one or more representative failure images $I_F \in \{I_1, \ldots, I_n\}$ for annotation. For each $I_F$ we annotate, our robot adds it to an aggregate set of annotated few-shot examples $E(I_{E_1}, I_{E_2}, \ldots, I_{E_p})$ that update $D$. In effect, these updates prevent $F$ from recurring and let our robot learn difficult or evolving tasks. Once $T$ completes without $F$, our robot goes on to complete others task.


Figure 2. **Task-Focused Annotation**. ClickBot collects task data and selects new few-shot examples for tasks requiring better detection, e.g., Find (left), Grasp (middle), and Placement (right).

In practice, unless objects or settings change, our robot rarely needs an update after $D$ is learned for $T$. We also find that sharing few-shot examples in $E$ across related tasks reduces the overall number of examples required.

### 3.3. Few-Shot Annotation

We provide annotation $E(I_{E_1}, I_{E_2}, \ldots, I_{E_p})$ using a custom GUI. After $F$, a user reviews new failure images $I_F$ saved by our robot. For each $I_F$ with task-relevant objects, the user can drag bounding boxes around each object and then add the new few-shot example $I_E$ to $E$ (see Figure 2).

In practice, annotating a bounding box takes about 7 seconds per object [26]. Also, $I_F$ without objects can *optionally* be added to $E$ as a true negative, which generally reduces false positives from $D$, e.g., for a task that searches for objects. In this work, we only annotate one $I_F$ with task-relevant objects per update, which gives our robot the opportunity to complete a task using the least annotation.

## 4. Detection-Based Manipulation

We perform mobile robot manipulation using a novel set of detection-based tasks. Notably, for objects learned *a priori*, our approach also works with standard object detection and tracking algorithms. To start, our robot needs to find task-relevant objects for manipulation, i.e., the *Find* Task.

### 4.1. Finding Objects for Manipulation

For the Find Task, we use a set of $n$ robot kinematic poses that moves a camera in the task space. Our robot collects an image at each pose $(I_1, I_2, \ldots)$ until an object is found using detection model $D$, which completes the Find Task. Our failure criterion $F$ is if our robot collects all $n$ images without detection, in which case, each image is saved for few-shot annotation ($I_F$). Using this process, the Find Task is typically how our robot first learns new objects.

In practice, after learning and manipulating new objects, we discontinue the Find Task's failure criterion $F$ to initiate *Sentry Mode*. In Sentry Mode, our robot intermittently uses the Find Task to search for objects but no longer assumes that the absence of detections indicates a false negative. Thus, our robot finds objects if they are in the task space without generating unnecessary few-shot examples if they are absent.

### 4.2. Learning Visual Servo Control from Detection

A key innovation for mobile manipulation is our learned visual servo controller (VS), which enables our robot to position itself relative to found objects, i.e., the *Servo* Task.

**Image Features from Detection**. To start, we use detection model $D$, input image $I$, and a target object class label $l$ to define image features $\mathbf{s} \in \mathbb{R}^2$ as

$$\mathbf{s}\big(D(I), l, \mathbf{s}_{t-1}\big) := \big[s_x, s_y\big]^\intercal, \tag{1}$$

where bounding boxes with class labels other than $l$ are ignored, $\mathbf{s}_{t-1}$ represents $\mathbf{s}$ from the previous time step, and $s_x, s_y$ denote the two image coordinates of the target object's bounding box center. We use $\mathbf{s}_{t-1}$ in (1) for two reasons. First, if there are multiple boxes with label $l$, we select the closest match to $\mathbf{s}_{t-1}$ for stability. Second, we use $\|\mathbf{s} - \mathbf{s}_{t-1}\|_{L_1}$ to check if $\mathbf{s}$ indicates a physically improbable discontinuity in object position. Finally, if detection of $l$ is absent at any time step, we temporarily use $\mathbf{s} = \mathbf{s}_{t-1}$.

Using (1), our failure criteria $F$ for the Servo Task are:

1. A discontinuity $\|\mathbf{s} - \mathbf{s}_{t-1}\|_{L_1} > 150$ pixels.
2. Detection of $l$ is absent for 20 consecutive time steps.

If either of these occur, our robot stops the Servo Task and saves the last input image for few-shot annotation ($I_F$).

**Visual Servo Feedback Control**. We use image features $\mathbf{s}$ (1) for our VS feedback error $\mathbf{e}$, which is defined as

$$\mathbf{e} = \mathbf{s} - \mathbf{s}^* = \big[s_x - s_x^*,\ s_y - s_y^*\big]^\intercal, \tag{2}$$

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

Typical VS [8] relates image features $\mathbf{s}$ to six-degrees-of-freedom (6DOF) camera velocity $\mathbf{v}$ using $\dot{\mathbf{s}} = \mathbf{L_s}\mathbf{v}$, where $\mathbf{L_s} \in \mathbb{R}^{2\times 6}$ is called the *feature Jacobian*. In this work, we use a constant $\mathbf{s}^*$ (i.e., $\dot{\mathbf{s}}^* = 0$), which, from (2), implies that $\dot{\mathbf{e}} = \dot{\mathbf{s}}$ and $\dot{\mathbf{s}} = \mathbf{L_s}\mathbf{v} = \dot{\mathbf{e}}$. Using this relationship, we find our control input $\mathbf{v}$ to minimize $\mathbf{e}$ using

$$\mathbf{v} = \text{-}\widehat{\mathbf{L}_\mathbf{s}^+}\mathbf{e}, \tag{3}$$

where $\widehat{\mathbf{L}_\mathbf{s}^+} \in \mathbb{R}^{6\times 2}$ is the *pseudoinverse* feature Jacobian.

When our robot controls $\mathbf{e}$ (3) below a threshold, it is accurately positioned relative to an object and the Servo Task is complete. In experiments, we use a threshold of 10 pixels before depth estimation and 5 pixels before grasping.

**A New Update Formulation to Learn Visual Control**. It is impossible to know the exact feature Jacobian $\mathbf{L_s}$ on real VS systems [8]. Instead, some VS work estimates $\mathbf{L_s}$ [22, 25] or $\widehat{\mathbf{L}_\mathbf{s}^+}$ [16] from observations using a Broyden update. Inspired by [5, (4.12)] in Broyden's original paper, we introduce a new update formulation to estimate $\widehat{\mathbf{L}_\mathbf{s}^+}$ in (3) as

$$\widehat{\mathbf{L}_\mathbf{s}^+}_{t+1} := \widehat{\mathbf{L}_\mathbf{s}^+}_{t} + \alpha\left(\frac{\big(\Delta\mathbf{x} - \widehat{\mathbf{L}_\mathbf{s}^+}_{t}\Delta\mathbf{e}\big)\Delta\mathbf{e}^\intercal}{\Delta\mathbf{e}^\intercal\Delta\mathbf{e}}\right) \circ \mathbf{H}, \tag{4}$$

where $\alpha \in \mathbb{R}$ determines the update speed, $\Delta\mathbf{x} = \mathbf{x}_t - \mathbf{x}_{t-1}$ is the change in 6DOF camera position since the last update, $\Delta\mathbf{e} = \mathbf{e}_t - \mathbf{e}_{t-1}$ is the change in error, and the element-wise product with logical matrix $\mathbf{H} \in \mathbb{R}^{6\times 2}$ determines which $\widehat{\mathbf{L}_\mathbf{s}^+}$ elements can update. We add $\mathbf{H}$ to prevent association of unrelated elements in $\mathbf{v}$ and $\mathbf{e}$ (3), which leads to more consistent and faster learning in comparative experiments.

Our robot actively learns to relate detection to camera motion for visual control (3) using (4). In plain words, our robot moves a camera ($\Delta\mathbf{x}$), observes corresponding changes in detection-based error ($\Delta\mathbf{e}$), then updates its learned motion-detection model ($\widehat{\mathbf{L}_\mathbf{s}^+}$) based on the difference between the actual ($\Delta\mathbf{x}$) and predicted ($\widehat{\mathbf{L}_\mathbf{s}^+}_{t}\Delta\mathbf{e}$) change in camera position.

In our experiments, we initiate (4) with $\widehat{\mathbf{L}_\mathbf{s}^+}_{t=0} = 0_{6\times 2}$, $\alpha = 0.5$, and $\mathbf{H} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^\intercal$. This choice of $\mathbf{H}$ couples image features $s_x$ and $s_y$ in $\mathbf{e}$ (2) with the $x$- and $y$-axis camera velocities in $\mathbf{v}$ (3).

### 4.3. Estimating Depth from Motion and Detection

After centering the camera on an object using the Servo Task, our robot estimates the object's depth using active perception with detection, i.e., the *Depth* Task.

In recent work, we estimated object depth by comparing changes in camera pose to changes in detection bounding box size [17, (9)]. In this work, we improve this estimate by actively advancing the robot's camera toward an object while recalculating the object's depth using every available detection $D(I_1), \ldots, D(I_n)$ with its corresponding kinematic camera pose. Once our robot estimates the object is within 0.2 m, the Depth Task is complete. Our failure criterion $F$ for the Depth Task is if detection of the object's label $l$ disappears, in which case, our robot stops the Depth Task and saves the last input image for annotation ($I_F$).

For depth-based grasping, our robot uses the median of an aggregate set of depth estimates, which consists of the latest estimate at every 0.05 m of camera motion. Basically, this approach mitigates any proximity-based detection errors that can occur when the camera is close to an object.

### 4.4. Grasping with Active Perception and Detection

After estimating an object's depth, our robot grasps the object using detection, i.e., the *Grasp* Task. Similar to other work, we use a simple visual representation that generalizes grasping across many novel objects [55] but also use multiple views to improve grasp selection [42].


Figure 3. **Grasping from Detection**. ClickBot rotates its camera (left) to find the narrowest detection-based parallel grasp points (middle) then uses a force-based grasp and lifts the Drill (right).

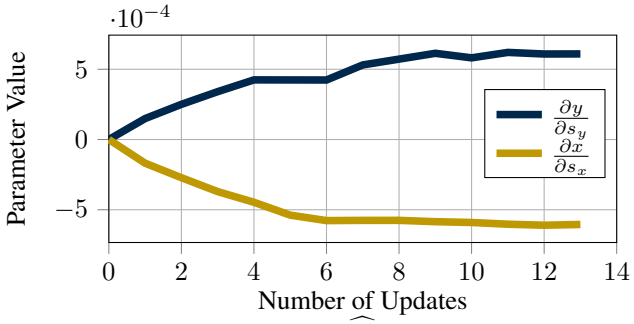


Figure 4. **Learned Visual Control $\widehat{\mathbf{L}_\mathbf{s}^+}$ Parameter Convergence**.

For active grasp planning, our robot moves its camera 0.16 m above the object's estimated depth then uses VS to center the object underneath its gripper. Next, our robot rotates its camera to find the best fit between the object and detection bounding boxes. Bounding boxes are rectangular, so our robot only rotates the camera $\frac{\pi}{2}$ radians because 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. As in the Depth Task, our Grasp Task failure criterion $F$ is if detection of the object disappears, which causes our robot to stop and save the last input image for annotation ($I_F$). After rotation and detection, our robot uses the box with the overall minimum height or width to plan its grasp.

Our grasp plan uses an antipodal grasp (i.e., a parallel grasp closing on two points). Basically, our robot uses the narrowest set of detection-based parallel grasp points and grasps at the object's center for balance (see Figure 3). After rotating its open gripper to align with the minimum height or width, our robot lowers its gripper to the object's estimated depth and applies a force-based parallel grasp. Our robot then lifts the object while continuing to apply force. If the gripper fingers remain separated by the object, the grasp is a success, and our robot releases the object at a goal location before returning to the Find Task for other objects.

## 5. Experimental Results

We validate TFOD and detection-based manipulation (ClickBot) in a variety of robot experiments with videos available at `https://youtu.be/Bby4Unw7HrI`.

### 5.1. Experimental Setup

**Robot and Camera Hardware**. We use a Toyota Human Support Robot (HSR) for our experiments [64]. We detect objects using HSR's end effector-mounted wide-angle grasp camera, which streams 640×480 RGB images at 25 Hz. We grasp detected objects using HSR's end effector-mounted parallel gripper with series elastic fingertips, which have a 135 mm maximum width. HSR's end effector moves on a 4DOF manipulator arm mounted on a torso with prismatic and revolute joints, but the relative pose between the grasp camera and gripper are constant. We typically point the end effector at the ground for detection and grasping (see Figures 1 and 3). For mobility, HSR uses a differential drive base. HSR's base also has a torso revolute joint directly above it, so we can control HSR as an omnidirectional robot (i.e., 3DOF ground-plane translation and rotation). We use quadratic programming [53] to command camera velocities $\mathbf{v}$ (3), but any velocity controller is applicable.

**Detection Model**. For our baseline model, we use Faster R-CNN [51], which runs in real time and has improved since its original publication. For reproducibility, we use the same Faster R-CNN configuration as Detectron2 [61] with ResNet 50 pre-trained on ImageNet and a FPN backbone trained on MS-COCO [61]. In our experiments, we update our detection model using annotated few-shot examples $E$ (Section 3), which consists of fine-tuning from the baseline model for 1,000 training iterations and takes less than four minutes using a standard workstation and GPU (GTX 1080 Ti). We also use a relatively high 0.9 confidence score threshold for detection, which significantly decreases false positives at the cost of increasing false negatives.

### 5.2. Learning Visual Servo Control from One Click

ClickBot learns visual servo control (VS) from camera motion and detection using our new update formulation (4). For each VS learning experiment, ClickBot starts a motion sequence, tracks detection changes, and updates $\widehat{\mathbf{L}_\mathbf{s}^+}$ after each motion. Each learning experiments ends when $\widehat{\mathbf{L}_\mathbf{s}^+}$ converges, i.e., $\left\| \widehat{\mathbf{L}_\mathbf{s}^+}_{t+1} - \widehat{\mathbf{L}_\mathbf{s}^+}_{t} \right\|_{L_1} < 10^{-6}$.

For camera motion ($\Delta\mathbf{x}$), ClickBot repeats eight motions comprising the permutations of {-5, 0, 5} cm across the $x$ and $y$ axes (e.g., $x$ =-5, $y$ =5). These motions are varied yet cycle through the initial camera pose for continued learning.

For detection, we use the racquetball from the YCB Object Dataset [6]. The racquetball is placed underneath ClickBot's grasp camera and our detection model learns from a single bounding box (i.e., one click of annotation). Notably, ClickBot learns VS from detection error *changes* $\Delta\mathbf{e}$ (4), so

Table 1. **Visual Servo Learning Results** are from a single consecutive set of 10 trials for each update formulation.

| $\widehat{\mathbf{L}_\mathbf{s}^+}$ Update Equation | Updates Required | | Range of $\widehat{\mathbf{L}_\mathbf{s}^+}$ Parameter Values Learned ($\cdot 10^{-4}$) | | | |
|---|---|---|---|---|---|---|
| | Mean | Range | $\frac{\partial x}{\partial s_x}$ | $\frac{\partial x}{\partial s_y}$ | $\frac{\partial y}{\partial s_x}$ | $\frac{\partial y}{\partial s_y}$ |
| ClickBot (4) | **13.6** | **9–21** | -6.2– -5.6 | **0.0–0.0** | **0.0–0.0** | **5.4–6.1** |
| VOSVS [16, (11)] | 22.5 | 15–30 | **-6.1– -5.6** | **0.0–0.0** | **0.0–0.0** | 5.4–8.3 |
| Broyden [5, (4.12)] | 31.7 | 15–76 | -6.4– -5.8 | -0.2–0.3 | -1.0–1.5 | 5.4–7.2 |
| Broyden [5, (4.5)] | 45.8 | 21–101 | -6.5– -5.6 | -0.4–0.8 | -1.1–2.2 | 5.5–8.3 |


Figure 5. **Experiment Objects from YCB Dataset**. Object sets left to right are Kitchen, Food, Tool, and Shape. Dimensions span between 4–470 mm and many objects exhibit specular reflection.

the *constant* desired values $s^*$ in $\mathbf{e}$ (2) are arbitrary.

In addition to learning VS for our remaining ClickBot experiments, we perform a single consecutive set of trials to compare (4) against existing $\widehat{\mathbf{L}_\mathbf{s}^+}$ update formulations. Notably, two formulas use $\Delta\mathbf{x}^\intercal\widehat{\mathbf{L}_\mathbf{s}^+}_t\Delta\mathbf{e}$ in the denominator and are undefined for $\widehat{\mathbf{L}_\mathbf{s}^+}_{t=0} = 0_{6\times 2}$. Thus, for VOSVS [16, (11)] and Broyden [5, (4.5)], we use VOSVS's convention and initiate with $\widehat{\mathbf{L}_\mathbf{s}^+}_{t=0} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^\intercal \cdot 10^{-3}$.

**Results**. ClickBot learns the VS model that we use in our remaining experiments in 13.29 s with 13 Broyden updates. Immediately afterward, we push the racquetball and ClickBot follows it, confirming that the learned visual controller (3) is a success (see video in supplementary material). We plot the learned $\widehat{\mathbf{L}_\mathbf{s}^+}$ values at each update in Figure 4.

We provide comparative VS learning results in Table 1. Relative to prior formulations, ClickBot requires 30-60% the updates and has 15-35% as much overall learned parameter variation. Thus, our new update formulation (4) learns VS faster and more reliably than the prior formulations.

### 5.3. VOSVS Benchmark

We evaluate ClickBot's VS and active depth estimation using the VOSVS Benchmark [16]. This benchmark consists of eight consecutive trials of VS and depth estimation (DE) on the YCB objects [6] shown in Figure 5. Each trial starts with three objects supported at 0.0, 0.125, and 0.25 m above the ground within camera view. VS is a success if a robot locates and servos to an object for DE. DE is a success if a robot advances without collision then closes it's gripper on the object without hitting the underlying surface.

Table 2. **VOSVS Benchmark Results** use a single consecutive set of trials. Visual Servo (VS) is a success (✓) if a robot moves to an object for depth estimation (DE). DE is a success if a robot's gripper closes on an object without collision.

| Object Set | Support Height (m) | YCB Object [6] | Method | | | |
|---|---|---|---|---|---|---|
| | | | ClickBot | | VOSVS [16] | |
| | | | VS | DE | VS | DE |
| Tool | 0.25 | Power Drill | ✓ | | ✓ | ✓ |
| Tool | 0.125 | Marker | ✓ | | ✓ | |
| Tool | 0.0 | Padlock | ✓ | ✓ | ✓ | |
| Tool | 0.25 | Wood | ✓ | ✓ | ✓ | |
| Tool | 0.125 | Spring Clamp | ✓ | | ✓ | |
| Tool | 0.0 | Screwdriver | ✓ | ✓ | ✓ | |
| Food | 0.25 | Chips Can | ✓ | ✓ | ✓ | ✓ |
| Food | 0.125 | Potted Meat | ✓ | ✓ | ✓ | ✓ |
| Food | 0.0 | Plastic Banana | ✓ | ✓ | ✓ | ✓ |
| Food | 0.25 | Box of Sugar | ✓ | ✓ | ✓ | ✓ |
| Food | 0.125 | Tuna | ✓ | ✓ | ✓ | |
| Food | 0.0 | Gelatin | ✓ | ✓ | ✓ | ✓ |
| Kitchen | 0.25 | Mug | ✓ | ✓ | ✓ | ✓ |
| Kitchen | 0.125 | Softscrub | ✓ | | | |
| Kitchen | 0.0 | Skillet with Lid | ✓ | | | |
| Kitchen | 0.25 | Plate | ✓ | ✓ | ✓ | ✓ |
| Kitchen | 0.125 | Spatula | ✓ | | | |
| Kitchen | 0.0 | Knife | ✓ | ✓ | ✓ | |
| Shape | 0.25 | Baseball | ✓ | ✓ | ✓ | |
| Shape | 0.125 | Plastic Chain | ✓ | ✓ | ✓ | |
| Shape | 0.0 | Washer | ✓ | | ✓ | |
| Shape | 0.25 | Stacking Cup | ✓ | ✓ | ✓ | ✓ |
| Shape | 0.125 | Dice | ✓ | | | |
| Shape | 0.0 | Foam Brick | ✓ | ✓ | ✓ | ✓ |
| Success Rate (%) | | | **100** | **66.7** | 83.3 | 41.7 |
| Annotations Per Object | | | **3.7** | | 10 | |
| Annotation Time Per Object | | | **26 s** | | 540 s | |

We also use the VOSVS Benchmark to evaluate TFOD-based learning. ClickBot learns new objects for each trial using the Find, Servo, and Depth tasks from Section 4. Starting without any annotation, ClickBot's first few-shot example $E(I_{E_1})$ is from an initial Find pose, and ClickBot returns to the Find Task after any other vision updates.

For each trial object, ClickBot finds it (Find), servos to it until $\mathbf{e} < 10$ pixels (3) (Servo), descends within an estimated 0.2 m (Depth), then closes it's gripper at the estimated depth. Each object is removed after its first full attempt, i.e., Find, Servo, Depth, and grasp closure without an update.

**Results**. We provide comparative VOSVS Benchmark results in Table 2. ClickBot achieves a perfect VS score and improves the prior DE success rate 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 also compare annotation time in Table 2. A segmentation mask takes about 54 s to annotate [26], which equates

Table 3. **Task-Focused Few-Shot Annotation Results** are averaged across corresponding trials (individual results in supplementary material). Clicks are the number of annotated bounding boxes, which each require 7 s (see user study [26]). CPU refers to training time.

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

to VOSVS using 540 s of annotation per object. On the other hand, ClickBot uses a simpler bounding box-based representation with task-focused annotation, which equates to 26 s of annotation per object, a 95% reduction.

We provide detailed TFOD results in Table 3. ClickBot averages 5 updates per trial with more few-shot examples for Depth than Find and Servo combined. A primary goal of TFOD is to focus annotation on difficult tasks, so we are encouraged that ClickBot automatically identifies and directs annotation to the task that requires the most improvement.

## 5.4. Pick-and-Place in Cluttered Environments

We evaluate ClickBot's end-to-end manipulation by adding pick-and-place to the VOSVS Benchmark for a new set of consecutive trials. First, we add the full Grasp Task (Section 4.4) after VS and DE for the Tool and Food sets. Notably, HSR cannot physically grasp some Kitchen and Shape objects, e.g., because they are too heavy (Skillet with Lid) or too low to the ground (Washer). After grasping, ClickBot also attempts to place objects in a bin. For evaluation, Grasp is only considered a success if ClickBot moves the object without dropping it and then releases it in the bin. Finally, as an added challenge, we repeat all of the consecutive pick-and-place trials in a cluttered environment.

We also use these pick-and-place trials to test two ablative TFOD configurations. For the first ablative configuration, we modify ClickBot to start with prior annotation from Section 5.3 for the non-cluttered pick-and-place trials. Subsequently, any new annotation is also included when learning pick-and-place in clutter. For a second ablative configuration, we remove TFOD and ClickBot *only* uses the prior annotation. For this configuration, ClickBot also uses a 0.1 confidence score threshold to increase detection likelihood.

**Results**. We provide ablative pick-and-place results in Table 4. The standard configuration achieves the best cluttered Grasp and pick-and-place rate of 88% (see two results in Figures 1 and 6). Considering the learning results in Table 3, we attribute the performance difference of the standard configuration over its ablative counterparts to having the most few-shot examples in clutter, which improves task performance for that particular setting. Nonetheless, the standard configuration uses less than a minute of annotation per object, which is approximately the same amount of time required to annotate a single segmentation mask and much less than the time required to generate a 3D model.

Table 4. **Pick-and-Place Results** with the Tool and Food Sets from the VOSVS Benchmark. All results use a single RGB camera.

| | Annotation | | Success Rate (%) | | |
|---|---|---|---|---|---|
| Method | Prior | TFOD | VS | DE | Grasp |
| VOSVS Benchmark for Tool and Food Sets | | | | | |
| VOSVS [16] | Yes | No | **100** | 50 | N/A |
| ClickBot | No | Yes | **100** | **75** | N/A |
| Tool and Food Sets *with* Pick-and-Place Added | | | | | |
| ClickBot without TFOD | Yes | No | 92 | 75 | 50 |
| ClickBot with Prior | Yes | Yes | **100** | **100** | **75** |
| ClickBot | No | Yes | **100** | **100** | **75** |
| Tool and Food Sets *with* Pick-and-Place in Clutter | | | | | |
| ClickBot without TFOD | Yes | No | 75 | 67 | 58 |
| ClickBot with Prior | Yes | Yes | **100** | **100** | 69 |
| ClickBot | No | Yes | **100** | **100** | **88** |

Across all tasks and setting in Table 4, using TFOD improves performance. Both ClickBot configurations using TFOD were perfect for VS and DE regardless of clutter. As in Section 5.3, ClickBot primarily requests annotation for tasks that require improvement, particularly when using prior annotation, which focuses most new annotation on grasping. Notably, Grasp-focused annotation can also improve detection performance in other tasks, such as DE.

## 5.5. Pick-and-Place with Dynamic Locations

We perform qualitative experiments to evaluate ClickBot's mobile manipulation with dynamic placement locations. For dynamic placement, ClickBot performs a second Find Task using a new set of placement class labels (e.g., Bin or Person). Once a placement location is detected, ClickBot releases the grasped object at that location.

We also use these experiments to demonstrate the modularity of ClickBot tasks. Using detection with HSR's RGBD head camera, ClickBot now creates a map for grasp and placement objects during the Find Task. This map effectively replaces the Depth Task while all other tasks remain.

**Results**. For our first dynamic pick-and-place experiment, we scatter cups for grasping and bins for placement across the floor. ClickBot learns to grasp a cup after two few-shot examples and learns to place it in a bin after two more (we

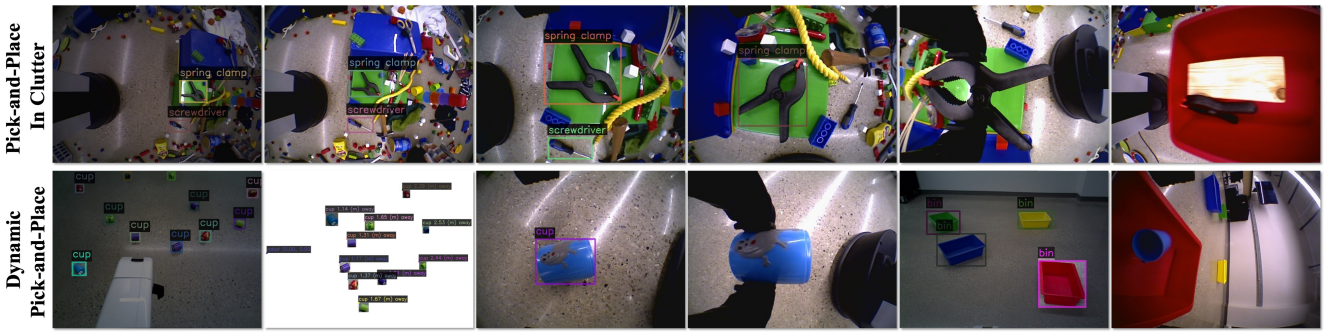


Figure 6. **Experimental Results**. For pick-and-place in clutter (top), ClickBot uses motion and detection to estimate the spring clamp's depth (left) and active detection-based grasping to place it in a bin (right). In dynamic pick-and-place (bottom), ClickBot uses detection with its head camera to map and grasp scattered objects (left) and then similarly discovers a suitable placement location (right).

Table 5. **Task-Focused Few-Shot Object Detection Benchmark** evaluation uses MS-COCO AP metrics and $k$ few-shot examples.

| Method | $k$ | AP | AP50 | AP75 | APs | APm | APl |
|---|---|---|---|---|---|---|---|
| ClickBot | 1 | 14.1 | 19.9 | 17.2 | 0.0 | 32.9 | 22.8 |
| | 2 | 18.3 | 24.3 | 22.5 | 0.0 | 32.1 | 27.7 |
| | 4 | 35.0 | 46.0 | 42.0 | 1.7 | 57.4 | 39.0 |

show this result in Figure 6, bottom). We attribute four-shot dynamic pick-and-place to removing the Depth Task, which offsets annotation on the new placement-based Find Task.

For our second experiment, ClickBot learns to retrieve thrown cups and return them to a moving person using eight more few-shot examples (see Figure 2 right).

As a final demonstration, ClickBot places scattered cups in specific bins that match colors. As an added challenge, we move the bins after each placement. ClickBot places all nine cups in their correct bin at a rate of 124.6 picks-per-hour. To our knowledge, there is no precedent for this rate of vision-based mobile robot manipulation in the literature.

### 5.6. Task-Focused Few-Shot Detection Benchmark

ClickBot's performance will improve with future few-shot object detection methods. Thus, we are introducing the **T**ask-Focused **F**ew-Shot **O**bject **D**etection (TFOD) Benchmark to help guide innovation. The TFOD Benchmark is 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 makes robot-collected data and corresponding annotations publicly available for research, which enables object detection researchers to evaluate their methods in this new task-focused setting for robot manipulation.

**Results**. We provide baseline TFOD results in Table 5, which averages our fine-tuning approach (Section 5.1) across ten consecutive trials (per-object baseline results in supplementary material). We see opportunity for future object detection innovation across all settings, especially for small objects (APs) and one- or two-shot detection.

## 6. Conclusions

We develop a new method of mobile manipulation based on object detection. To our knowledge, our robot is the first to manipulate objects using detection alone. Furthermore, our robot collects data as it performs tasks and, if it recognizes a detection error, automatically selects a new few-shot example for annotation to improve performance. In this way, our robot avoids many vision-based errors while adapting to changing objects, tasks, and environments.

We evaluate our approach using a variety of experiments. First, our robot learns a novel visual servo controller from detection in 13.3 s. Furthermore, we show in repeat trials that our visual servo formulation learns faster and more reliably than alternative approaches. Using learned visual control with detection-based depth estimation, our robot also 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 using a single RGB camera with as few as four few-shot examples, achieving an overall pick-and-place rate of 88%. This result is on par or better than recent state-of-the-art methods [42, 46, 55, 66], which all use an RGBD camera in a fixed workspace. Notably, we can optionally configure our approach for an RGBD input, which our robot uses to clean up scattered objects with moving placement locations at over 120 picks-per-hour.

In conclusion, our experiments show that our RGB-based approach to mobile manipulation works if few-shot annotation is acceptable to learn new objects and settings. In addition, our approach can supplement RGBD-based approaches or substitute when full 3D sensing is unavailable.

In future work, we will expand our approach to accommodate new challenging tasks (e.g., manipulation across multiple cluttered rooms). Future innovations in object detection will help us achieve these results. Thus, we are releasing a new object detection benchmark that enables future detection work to evaluate and improve performance in a challenging robotics setting. We also plan to release future additions for this benchmark in new application areas.

## 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.

[2] Ashutosh Agarwal, Anay Majee, Anbumani Subramanian, and Chetan Arora. Attention guided cosine margin to overcome class-imbalance in few-shot road object detection. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) Workshops*, 2022.

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

[4] 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.

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

[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*, 2015.

[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.

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

[9] F. Chaumette and S. Hutchinson. Visual servo control. ii. advanced approaches [tutorial]. *IEEE Robotics Automation Magazine*, 2007.

[10] 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.

[11] X. Deng, Y. Xiang, A. Mousavian, C. Eppner, T. Bretl, and D. Fox. Self-supervised 6d object pose estimation for robot manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[12] Guoguang Du, Kai Wang, Shiguo Lian, and Kaiyong Zhao. Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review. *Artificial Intelligence Review*, 2021.

[13] 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.

[14] 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.

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

[16] 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.

[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.

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

[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.

[20] Guangxing Han, Yicheng He, Shiyuan 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.

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

[22] K. Hosoda and M. Asada. Versatile visual servoing without knowledge of true jacobian. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1994.

[23] 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.

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

[25] 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.

[26] Suyog Dutt Jain and Kristen Grauman. Predicting sufficient annotation strength for interactive foreground segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, 2013.

[27] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[28] Bingyi Kang, Zhuang Liu, Xin Wang, Fisher Yu, Jiashi Feng, and Trevor Darrell. Few-shot object detection via feature reweighting. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[29] Leonid Karlinsky, Joseph Shtok, Sivan Harary, Eli Schwartz, Amit Aides, Rogerio Feris, Raja Giryes, and Alex M. Bronstein. Repmet: Representative-based metric learning for

classification and few-shot object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[30] Suseong Kim, Hoseong Seo, Seungwon Choi, and H. Jin Kim. Vision-guided aerial manipulation using a multirotor with a robotic arm. *IEEE/ASME Transactions on Mechatronics*, 2016.

[31] Thomas Lampe and Martin Riedmiller. Acquiring visual servoing reaching and grasping skills using neural reinforcement learning. In *International Joint Conference on Neural Networks (IJCNN)*, 2013.

[32] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research (IJRR)*, 2018.

[33] Aoxue Li and Zhenguo Li. Transformation invariant few-shot object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[34] Bohao Li, Boyu Yang, Chang Liu, Feng Liu, Rongrong Ji, and Qixiang Ye. Beyond max-margin: Class margin equilibrium for few-shot object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[35] Yiting Li, Haiyue Zhu, Yu Cheng, Wenxin Wang, Chek Sing Teo, Cheng Xiang, Prahlad Vadakkepat, and Tong Heng Lee. Few-shot object detection via classification refinement and distractor retreatment. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[36] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2014.

[37] Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In *Proceedings of the 1st Annual Conference on Robot Learning (CoRL)*, 2017.

[38] Alessandro De Luca, Giuseppe Oriolo, and Paolo Robuffo Giordano. Feature depth observation for image-based visual servoing: Theory and experiments. *The International Journal of Robotics Research (IJRR)*, 2008.

[39] Pat Marion, Peter R. Florence, Lucas Manuelli, and Russ Tedrake. Label fusion: A pipeline for generating ground truth labels for real rgbd data of cluttered scenes. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[40] G. L. Mariottini, G. Oriolo, and D. Prattichizzo. Image-based visual servoing for nonholonomic mobile robots using epipolar geometry. *IEEE Transactions on Robotics (TRO)*, 2007.

[41] A. McFadyen, M. Jabeur, and P. Corke. Image-based visual servoing with unknown point feature correspondence. *IEEE Robotics and Automation Letters (RA-L)*, 2017.

[42] Douglas Morrison, Peter Corke, and Jürgen Leitner. Learning robust, real-time, reactive robotic grasping. *The International Journal of Robotics Research (IJRR)*, 2020.

[43] A. Opelt, A. Pinz, and A. Zisserman. Incremental learning of object detectors using a visual shape alphabet. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[44] Anton Osokin, Denis Sumin, and Vasily Lomakin. Os2d: One-stage one-shot object detection by matching anchor features. In *European Conference on Computer Vision (ECCV)*, 2020.

[45] Kiru Park, Timothy Patten, and Markus Vincze. Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[46] Ole-Magnus Pedersen, Ekrem Misimi, and François Chaumette. Grasping unknown objects by coupling deep reinforcement learning, generative adversarial networks, and visual servoing. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[47] Juan-Manuel Perez-Rua, Xiatian Zhu, Timothy M. Hospedales, and Tao Xiang. Incremental few-shot object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[48] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[49] Limeng Qiao, Yuxuan Zhao, Zhiyuan Li, Xi Qiu, Jianan Wu, and Chi Zhang. Defrcn: Decoupled faster r-cnn for few-shot object detection. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[50] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[51] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems 28 (NIPS)*, 2015.

[52] Mohammad Reza Loghmani, Barbara Caputo, and Markus Vincze. Recognizing objects in-the-wild: Where do we stand? In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[53] 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.

[54] 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.

[55] Mohit Vohra, Ravi Prakash, and Laxmidhar Behera. Real-time grasp pose estimation for novel objects in densely cluttered environment. In *IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 2019.

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

[57] Y. Wang, H. Lang, and C. W. de Silva. A hybrid visual servo controller for robust grasping by wheeled mobile robots. *IEEE/ASME Transactions on Mechatronics*, 2010.

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

[59] L. Weiss, A. Sanderson, and C. Neuman. Dynamic sensor-based control of robots with visual feedback. *IEEE Journal on Robotics and Automation*, 1987.

[60] 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.

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

[62] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[63] 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.

[64] 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.

[65] 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.

[66] Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics (TRO)*, 2020.

[67] 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.

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

[69] Chenchen Zhu, Fangyi Chen, Uzair Ahmed, Zhiqiang Shen, and Marios Savvides. Semantic relation reasoning for shot-stable few-shot object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[70] Rui Zhu, Xingyi Yang, Yannick Hold-Geoffroy, Federico Perazzi, Jonathan Eisenmann, Kalyan Sunkavalli, and Manmohan Chandraker. Single view metrology in the wild. In *European Conference on Computer Vision (ECCV)*, 2020.

[71] Yiming Zuo, Weichao Qiu, Lingxi Xie, Fangwei Zhong, Yizhou Wang, and Alan L. Yuille. Craves: Controlling robotic arm with a vision-based economic system. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.