Continual Learning about Objects in the Wild: An Interactive Approach

Dan Bohus
Microsoft Research
Redmond, WA, USA
dbohus@microsoft.com

Sean Andrist
Microsoft Research
Redmond, WA, USA
sandrist@microsoft.com

Ashley Feniello
Microsoft Research
Redmond, WA, USA
ashleyf@microsoft.com

Nick Saw
Microsoft Research
Redmond, WA, USA
nick.saw@microsoft.com

Eric Horvitz
Microsoft
Redmond, WA, USA
horvitz@microsoft.com

ABSTRACT

We introduce a mixed-reality, interactive approach for continually learning to recognize an open-ended set of objects in a user's surrounding environment. The proposed approach leverages the multimodal sensing, interaction, and rendering affordances of a mixed-reality headset, and enables users to label nearby objects via speech, gaze, and gestures. Image views of each labeled object are automatically captured from varying viewpoints over time, as the user goes about their everyday tasks. The labels provided by the user can be propagated forward and backwards in time and paired with the collected views to update an object recognition model, in order to continually adapt it to the user's specific objects and environment. We review key challenges for the proposed interactive continual learning approach, present details of an end-to-end system implementation, and report on results and lessons learned from an initial, exploratory case study using the system.

CCS CONCEPTS

• **Computing methodologies** → **Object recognition**; • **Human-centered computing** → **Mixed / augmented reality**.

KEYWORDS

Mixed Reality; Augmented Reality; Object Identification; Object Teaching; Multimodal

ACM Reference Format:
Dan Bohus, Sean Andrist, Ashley Feniello, Nick Saw, and Eric Horvitz. 2022. Continual Learning about Objects in the Wild: An Interactive Approach. In *INTERNATIONAL CONFERENCE ON MULTIMODAL INTERACTION (ICMI '22), November 7–11, 2022, Bengaluru, India.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3536221.3556567

ICMI '22, November 7–11, 2022, Bengaluru, India

ACM ISBN 978-1-4503-9390-4/22/11…$15.00
https://doi.org/10.1145/3536221.3556567

1 INTRODUCTION

The work we report in this paper is part of a broader research effort on interactive methods that can enable computing systems to learn continually about physical objects in the world. Our long-term goals are two-fold. First, we seek to build intelligent systems that can fluidly collaborate with people in physically situated settings. Second, we seek fundamental principles for learning continually from multimodal streaming data.

Recognizing objects in images has been a long-term, canonical challenge in computer vision. In the last decade, fast-paced progress on object detection and recognition has been fueled by large datasets and deep neural network methods. More recently, interest has grown within the research community in egocentric vision tasks [6], as well as in zero-shot [17], few-shot [12], and continual learning [23]. These settings and capabilities frame important theoretical and practical research directions and promise to support a broad spectrum of applications, including robotic manipulation, situated interaction, and mixed-reality systems for task guidance, training, healthcare, education, and entertainment.

However, despite advances on benchmark datasets, developing practical applications that can operate robustly in the open world remains a challenging task. One of the biggest barriers to robust object recognition in the wild is that the quality and distribution of images encountered in a real-world deployment often differs greatly from the distribution of images with which a system was trained. Object recognition models embedded in open-world applications must deal with dynamic and diverse viewpoints, including changes in proximity, illumination, resolution, motion blur, and shifting relationships and occlusions among fixed and movable objects. The typical development process for building such models involves starting with a pretrained platform model followed by a fine-tuning procedure that requires collecting data in the field and then labeling it. This iterative process is time-intensive and costly.

In this paper, we introduce a mixed-reality, interactive approach for *continually learning to recognize an open-ended set of objects in a user's surrounding environment.* We leverage the affordances of egocentric sensing, interaction, and rendering in mixed reality, and present a system implementation that works as follows: As a user engages in tasks while wearing a mixed-reality headset, objects in the user's field of view are detected and highlighted with a holographic bounding box (see Figure 1, top left). The user may provide labels for these objects by gazing toward an object and saying, e.g.,

Figure 1: A mixed-reality system for interactive continual learning. Top left: A mixed-reality interface with gaze, speech, and gesture recognition capabilities is used to label objects in the world. Top right: The system uses egocentric color and depth cameras to detect and track objects in 3D world coordinates. Bottom: The system continually captures views of objects over time from different angles and uses them to adapt an object recognition model.

"This is my cutting board." The system tracks the 3D world position and orientation of the objects and of the camera (Figure 1, top right) and continuously collects views of these objects from different angles as the user goes about their regular activities (Figure 1, bottom). These views are used to predict object identities, and also to adapt a recognition model in order to improve performance over time on the set of objects encountered by the user.

While the basic ideas behind the approach can be described in a few sentences, building a system that can perform this task well in the open world hinges on answering several challenging questions, including: How does the system track object locations over time? Which objects should the system highlight and ask the user to label? Which object views should be used for inference or model adaptation? How well do current state-of-the-art vision models perform in this setting? How can we boost recognition robustness?

We describe initial steps that we have taken towards addressing some of these challenges. Specifically, we make the following contributions: (1) we introduce a multimodal, interactive methodology for continuously learning to recognize objects online and onsite, in a mixed-reality setting, (2) we construct and describe a concrete implementation of the proposed approach, (3) we perform an initial, exploratory case study with this system, and (4) we discuss lessons learned and insights gathered.

2 RELATED WORK

We begin with a brief review of relevant related work in object recognition and object teaching.

2.1 Object Recognition

The computer vision community has introduced methods for detecting, tracking, and classifying objects in 3D, extending from the larger body of research on 2D methods. Several approaches make use of prespecified CAD models with known meshes and labels. Such work includes DenseFusion [22], a deep learning approach that combines RGB and depth to predict the 6D pose of objects. Other studies have employed *voxel-based* representations for 3D shapes with density distributions computed on a 3D grid. Wu et al. [24] train a convolutional DNN over such a representation, while Maturana and Scherer [13] directly train a 3D convolutional neural network to classify objects from voxels.

Rather than working from a voxelized representation, PointNet applies DNNs to operate directly on 3D point clouds, and can be used for object classification, segmentation, and semantic parsing [15]. This work was later extended to VoteNet, which combines a Hough voting strategy with a DNN [14]. SE(3)-Transformer [5] also performs object classification from 3D point clouds over a closed set of known objects, using an equivariant attention network to gain robustness in the presence of rotations and translations. In general, researchers have found that directly extending 2D CNNs into three dimensions is challenging. Instead, they have had greater success for recognition with multiview 2D CNNs [16, 20].

Important algorithmic developments have also been recently made in the areas of *few-shot*, *class-incremental* and *continual learning* [1, 3, 18, 21, 26]. However, most work to date in this space is evaluated on pre-existing, static datasets (CIFAR-100, CUB-200, miniImageNet, etc.), by partitioning them in various configurations. Recent work [23] has raised the important challenge of devising new metrics for evaluating continual learning with the goal of higher validity for real scenarios, focusing on questions such as overall model performance, the degree to which the model can transfer new knowledge, and its ability to resist catastrophic forgetting. Other efforts are starting to bring attention to other real-world challenges that often do not appear in existing datasets, such as the need to handle motion blur, occlusions, clutter, and so on [11, 19].

Finally, another important recent development is the construction of large, multimodal platform models like Florence [25], CLIP

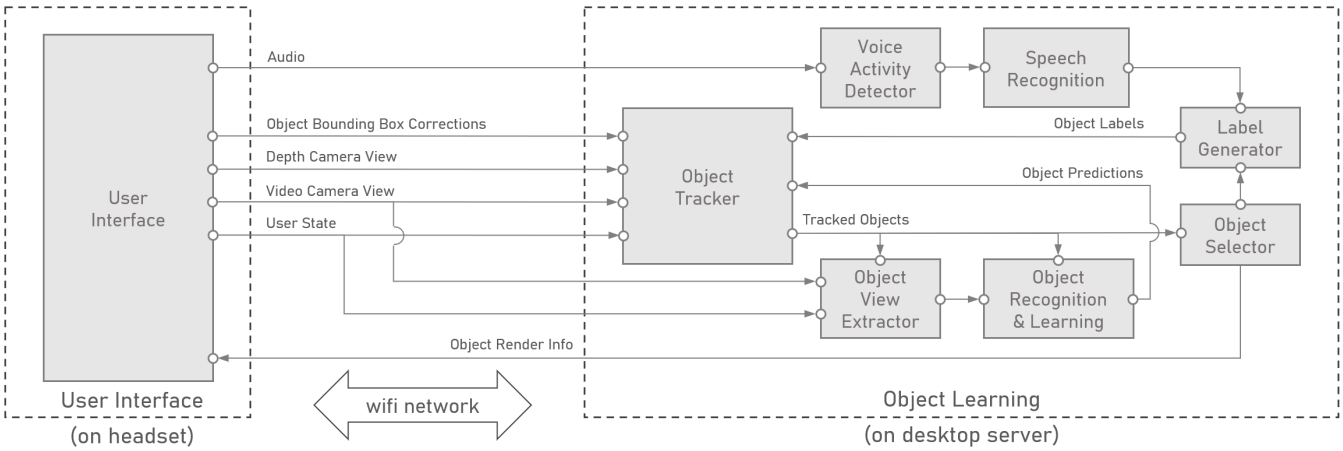

Figure 2: Architectural overview of the implemented system.

[17] and GLIP [10] that jointly reason over images and language, and are adaptable to many downstream tasks related to object recognition. These models are trained on massive datasets of visual concepts via supervision signals from natural language text.

In the work we report below, we take a multimodal, integrative approach, and build and experiment with data from an end-to-end, deployed system that leverages the CLIP model [17] to learn about objects online and onsite. While we do not introduce novel algorithmic contributions in object recognition or continual learning, the end-to-end system developed and the experiments conducted highlight several important lessons and outline real-world challenges, including 3D segmentation, handling motion blur and occlusions from hands and cluttered scenes, detection of new versus previously seen objects, and evaluation metrics.

2.2 Object Teaching

Teachable object recognition is a relatively recent area of work focused on enabling people to customize an object recognizer by capturing training examples of relevant objects, and training a recognizer on the fly to recognize the objects in new scenarios. The approach has been applied to accessibility solutions for blind and low vision people [7, 9]. The ORBIT dataset [12] of videos taken by blind and low-vision users is aimed at building models used in few-shot learning. The training data includes examples representing imbalance and quality observed in real-world settings, including problems with framing and image blur. LabelAR [8] is a similar effort on teaching recognizers about new objects geared towards augmented reality applications.

Our work is different from previous efforts in that we pursue a methodology that enables a user to teach the system about objects in-stream with other activities, and enables it to continually learn in the background, as the users go about their regular tasks.

3 OVERVIEW

Our high-level goal is to enable an object recognition system to continually learn to recognize an open-ended set of objects onsite, in a user's environment. The key idea behind the approach is to obtain training data—including *object views* and *object labels*—continually, in stream with other user activities, and to use the data to adapt a recognition model online. We investigate this approach in a mixed-reality setting and leverage the multimodal sensing and interaction affordances of a mixed-reality headset to create an object teaching interface that allows the end-user to provide object labels in stream with their other activities.

The proposed approach relies on three core capabilities: (1) 3D object detection and tracking, (2) interactive object teaching, and (3) online learning over extracted object views. We use the depth sensor from the mixed-reality headset to detect and track the poses and locations of objects in the user's environment. The system can automatically highlight detected objects via a holographic 3D bounding box representing the 3D segmentation of the object, which the user can manually correct via hand gestures if necessary. The 3D bounding boxes are projected into the device's color camera image to obtain 2D cropped views of each object. In addition, the user can provide object labels by gazing toward a particular object and speaking its name, e.g., *"That's my salt shaker."* As we shall discuss later, we leverage inferences about object permanence to *propagate the user-provided labels forwards and backwards in time*. The labels are therefore linked to multiple object views collected over time and can be used to improve the recognition model over time.

We implemented the proposed approach as a distributed mixed-reality application on the HoloLens 2, augmented with a dedicated desktop computer to bypass current computational limitations. The implementation is built on Platform for Situated Intelligence [4], an open-source framework for multimodal integrative-AI systems, which also provides infrastructure for accessing streaming sensor data and rendering holograms on HoloLens 2 [2].

Our application consists of two processes that communicate over a network connection, as shown in Figure 2. The *User Interface* process runs on the headset and transmits a wide array of multimodal sensor streams from the headset to the desktop server, including depth images (5Hz, 320×288 pixels), color images (depth-synchronized at 5Hz, 896×504 pixels), camera poses (5Hz), camera intrinsics (5Hz), audio, and user state data such as the 3D gaze direction (~30Hz) and hand tracking information (~30Hz, 26 joint poses).

Figure 3: Interactive approach to defining tracking regions. Left: Tracking region is created with a double-pinch gesture; region is displayed in mixed reality as a rectangular yellow hologram. Center: Size of the tracking region modified by pinching and moving one of its corners. Right: Position of the tracking region confirmed by push of virtual [+] button.

The User Interface also emits streams of information about the user's manual corrections to the object bounding-box holograms. A consistent world coordinate system is achieved by persisting a spatial anchor[1] to the device when the application runs for the first time. This anchor allows for all spatial information to be expressed in the same global frame of reference.

The *Object Learning* process runs on the desktop server and includes components for 3D object tracking, for extracting 2D object views, and for object recognition and learning. An Object Selector component computes which of the tracked objects to highlight based on the user's gaze, and streams back the relevant information to the *User Interface*, which shows the hologram bounding box around the object. Finally, audio is processed via a voice activity detector and a cloud speech recognizer to extract object labels and associate them with the currently selected object.

In the next three sections, we describe the major components of the system in more detail: object detection and tracking, interactive object teaching, and object recognition and learning.

4 OBJECT DETECTION AND TRACKING

We rely on a simple depth-based approach for detecting and tracking objects in a number of predefined tracking regions and use interactive techniques to enable users to perform onsite corrections of the object segmentations. The detection and tracking algorithm is based on the captured depth images. These images are converted to point-cloud representations, which are used to update a voxel occupancy grid in each predefined tracking region. Objects are detected based on a connected-components algorithm that runs over the voxel occupancy grid. We present more details on each of these subcomponents in the remainder of this section.

4.1 Tracking Regions

For computational efficiency, we only track objects in a set of cuboid *tracking regions*, defined by the user via hand gestures. Specifically, when the user performs a double-pinch gesture (see Figure 3 left), the system creates a new rectangular tracking region in the horizontal plane, centered on the location of the gesture, and extending upwards for 0.5 meters. The base of the tracking region is displayed as a hologram, visible as a yellow rectangle in Figure 3. The user can change the dimensions and orientation of the region by pinching and moving its corners to resize (see Figure 3 center) or its edges to rotate. The user confirms the desired size and placement of the tracking region by pressing the [+] button in the center of the region (see Figure 3 right). Multiple tracking regions can be defined in this fashion.

[1]https://docs.microsoft.com/en-us/windows/mixed-reality/design/spatial-anchors

4.2 Voxel Occupancy Grid

To construct the voxel occupancy grid, we begin by converting the depth images into a point-cloud representation in world coordinates. We use the hand-tracking information provided by the device to exclude the points corresponding to the user's hands and arms. Specifically, we project the 3D hand joint positions returned by the hand tracker into the depth image and mask the hands by performing a neighbor traversal of the depth image pixels. We start at the pixel position of the closest joint (lowest depth value), and traverse the depth image in all directions until we find a significant increase (>10 cm) in the depth of the neighboring pixel, which typically corresponds to the edge of the user's hand or arm.

We use the computed point clouds to maintain and update a 3D voxel occupancy grid in each predefined tracking region. At each voxel, we maintain two different counters: a *weight counter*, which is used to determine voxel occupancy, and a *delete counter*, which is used to determine when the voxel will be deleted.

The update to the occupancy grid is done in two steps. First, the system uses the incoming point cloud at time t to compute a new, instant occupancy grid, and then fuses this instant grid with the previous occupancy grid from time $t-1$. When estimating the instant occupancy grid, the weight counter for a voxel is set to 1 if the point cloud contains at least a point inside that voxel, and the angle between the mesh surface normal and the direction looking towards the voxel center is less than a specified threshold (60°)—when this angle is larger, the angle from which the voxel is being viewed is grazing the surface, and the likelihood of errors in the point cloud is increased. When fusing the information from the previous and instant occupancy grids, if a voxel has positive weight in both the previous and instant grids, the weight counters are added. If a voxel previously present is not found in the instant occupancy grid, and the voxel center is between the camera and the depth mesh, the delete counter is incremented. When the delete counter reaches a threshold (3), the voxel is removed.

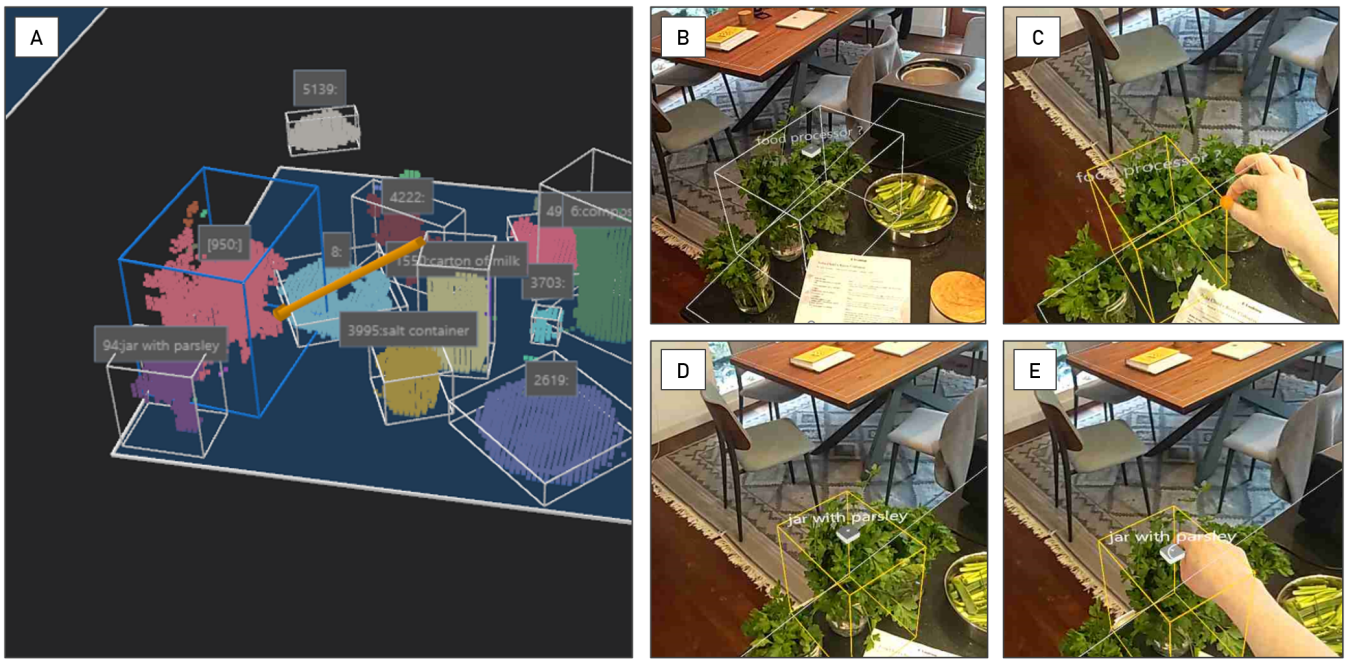

Figure 4: User interaction for teaching the location and label of a new object. A: System's view of the world with voxel-based object segmentation and gaze direction (orange ray). B: Mixed-reality view with (incorrect) object prediction. C: User edits the object's bounding box. D: User labels the object via speech *"This is a jar with parsley."* E: User confirms the label and position.

4.3 Segmentation and Tracking

The computed voxel occupancy grid is used to perform object segmentation. First, we determine the connected components in the occupied voxel space (a voxel is considered occupied if its weight counter >3). For each connected component, we assign an object ID based on the largest overlap with known objects from the previous time step. If no overlap is found, the voxels in the new component are assigned a new object ID. In this assignment process, we exclude all voxels in bounding boxes that have been manually edited and confirmed by the user (see Section 5)—all occupied voxels inside a manually edited box are assigned to its corresponding object ID.

Finally, to determine the position of objects, we compute bounding boxes by determining the minimum rectangular hull that covers all voxels with a given ID in the horizontal plane. In the vertical dimension, the bounding box limits are set based on the vertical span of these voxels. To avoid jitter, we also compute an updated bounding box based on the object orientation from the previous frame. If the size difference between the minimum-rectangular-hull bounding box and the previous-orientation bounding box is not large, we use the bounding box based on the previous orientation.

The method described above is able to detect new objects added to the scene. To construct valid object views over time, we also need to reason about when objects are removed from a given location. We detect removals with a simple heuristic that counts the number of occupied voxels in the object bounding box and compares that number against the maximum value observed so far. If the ratio drops below 0.5, the object and its bounding box are removed from the tracking state.

5 INTERACTIVE OBJECT TEACHING

We now turn to the interactive approach for teaching the system about objects. A core aspect of the approach is to enable the user to teach the system about new objects as they go about everyday activities. The teaching process should therefore require minimal effort and should not significantly distract the user from their primary tasks. An important consideration in the design space of interactive teaching is whether the interactions are *system-initiated*, *user-initiated*, or *mixed-initiative*: Does the system ask the user about specific objects, is the user in control of which objects to teach the system about, or should a mix of both strategies be employed? In the longer-term, we envision system-initiated and mixed-initiative solutions, guided by active learning and utility considerations, considering factors such as the estimated expected value of the label, the user's attention, and cost of engagement. For now, the current implementation employs a gaze-based, user-initiated approach which keeps the user in control and minimizes visual distractions.

Specifically, we implemented a mixed-reality interface that uses gaze, gesture, and speech to allow the user to provide object labels through interaction, and to manually correct the 3D object bounding boxes generated by the object detector (described in the previous section). The system computes which object the user is gazing towards by intersecting the gaze direction with the computed bounding boxes (see Figure 4 A). The bounding box for the object deemed in focus is rendered as a wire-frame hologram (see Figure 4 B). If a prediction from the recognition model is available for this object, the prediction is rendered as a text label followed by

Figure 5: Different image views for a "salt container." Left to right: image view affected by motion blur; image view affected by occlusion from another object; image view affected by hand occlusion; clear image view.

a question mark, hovering above the bounding box and oriented towards the user (see Figure 4 B).

The interface enables the user to correct the detected object bounding box with simple gestures when necessary. Pinching and moving one of the corners resizes the bounding box (see Figure 4 C), and pinching and moving the middle of an edge rotates the bounding box around its vertical axis. The corrected location of the bounding box is continuously streamed back to the automatic object detector and tracker (see Figure 2) and is leveraged in the tracking process, as described in subsection 4.3.

If a prediction about the object is not yet available or if the prediction is incorrect, the user can provide a label for the object via a simple spoken phrase like *"This is a/an/my <object name>."* The system performs speech recognition using a cloud-based speech service and the resulting object label is extracted and displayed above the object without a question mark (see Figure 4 D). The visual feedback allows the user to also detect potential speech recognition errors. If a speech recognition error occurs, the user can repeat the label to correct it.

Once a prediction or a user-generated label is available, a [+] button appears on top of the object (see Figure 4 D). When the user pushes this button (see Figure 4 E), the object location and identity is confirmed, and this explicit signal is used to generate labels for both continual learning and performance evaluation.

While the gaze-based selection mechanism is natural to use, the approach can become problematic when the user is actively engaged in another task, as the holograms that appear on the object (wireframe box and text label) can distract from the task at hand. To alleviate potential distractions, we implemented a "teaching mode" which can be enabled or disabled by the user via simple speech commands: *"Turn on/off teaching mode."* The object bounding boxes and predicted labels, along with the ability to teach new labels, are available only when the teaching mode is active. However, even when the teaching mode is disabled, the system continues to collect images in the background for previously labeled objects that are in view at any given time.

6 OBJECT RECOGNITION AND LEARNING

To learn about objects, the system pairs labels taught by the user with a set of views of each object that are continuously collected by the system in the background. We begin by discussing the mechanism for generating these object views in the next subsection. Then, in Subsection 6.2, we review the object recognition models and learning algorithms explored in the current study.

6.1 Generating Object Views

Object views are generated by cropping the color camera image around the location of the object. Specifically, the corners of the object's 3D bounding box are first projected into the pixel space of the color camera. We then compute the rectangle that encompasses these points and expand its dimensions by 10%. If the resulting rectangle overlaps with the camera's field-of-view by at least 80%, it is used to crop an object view from the full camera image.

The system continuously collects object views while the user is going about their tasks in the world. As a result, the quality of such views can vary significantly. During early development and testing, we identified two important issues with captured images. First, because the user's head—and by extension the head-mounted camera—is often in motion, many images suffer from motion blur, as illustrated in the first image in Figure 5. Second, views can be obstructed by other objects in cluttered scenes or by the user's hands, as shown the second and third image in Figure 5.

To rectify these issues, we filter the object views based on thresholds tuned from initial prototyping. To alleviate blur, we compute the linear and angular head speed, and select only object views acquired when these values are below 0.25 m/s and 0.25 rad/s respectively. To address occlusions, we eliminate an object's view if it is significantly overlapped by the view of another object in front of it, or by the user's hands. To compute hand occlusions, we compute a rectangle from the projection of the hand joints in the color camera space. For the right hand, this rectangle spans from the top-leftmost hand joint to the bottom-right corner of the image, and for the left hand it spans from the top-rightmost joint to the bottom-left corner of the image.

6.2 Models for Object Recognition

The object views collected by the system can be used both for object recognition and for the continual improvement of recognition models. We experiment with two methods, both of which leverage CLIP [17]—a pretrained, multimodal (vision-language) model. CLIP is trained on a large dataset of $(image, text)$ pairs [17], and supports *zero-shot* recognition.

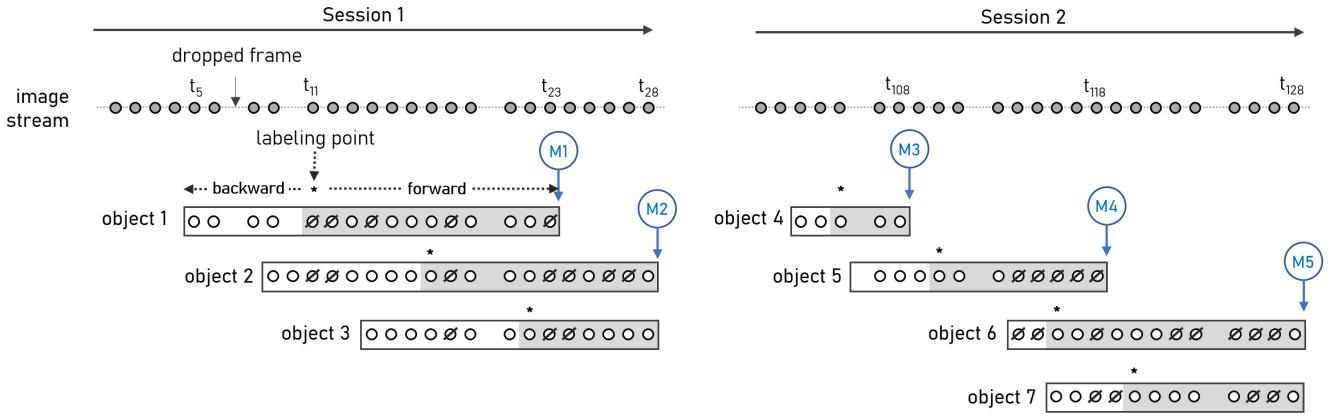

Figure 6: View of events, objects, and label propagation in interactive continual learning. Cropped frames (open circles) are segmented from the raw color image stream (filled circles). Detected objects (rectangles) are labeled by users at labeling points (asterisks). Labels are propagated forward and backward in time. Model update time-points are highlighted (M1-M5).

In zero-shot mode, CLIP supports queries of the form (*image*, $[s_1, s_2, ..., s_n]$), where s_i can be any English-language string, and it produces an output probability distribution over the given set of strings. We use the model in this zero-shot mode, and query it with the names of all known object types taught by the user so far, plus an empty string that we use to model any previously unseen object type. When the system runs for the first time, the set of object types contains only the empty string token. When a new label is introduced by the user, the set of object types is expanded accordingly. The number of object types that the system knows about (and uses to query the CLIP model with) therefore increases over time. The zero-shot capabilities of CLIP enable the construction of a system that can recognize increasing numbers of objects over time. In the experiments reported below, we explore an initial assessment of the robustness of the zero-shot model approach in an open-world, streaming, egocentric computer vision scenario.

Although the use of the fixed, pretrained CLIP model in zero-shot mode is amenable to a growing set of class labels, it does not involve any actual learning from new data collected by the user. As a second approach, we also implemented a learning-based technique on top of CLIP. Specifically, we use CLIP's visual encoder to generate encodings for the object views and use the encodings as input to a multinomial logistic regression model trained to classify the set of known object types. We note that this learning technique does not refine the CLIP model itself; rather, it employs a logistic regression layer on top of the model.

7 EXPERIMENTS

We conducted an exploratory case study with the implemented system, aimed at investigating the feasibility of the overall proposed approach, and at identifying challenges and areas of improvement for the current implementation.

In the case study, two of the paper's authors used the system over a period spanning 19 days at Site 1 and 8 days at Site 2 (total durations of 9h 32min and 4h 28min respectively), while performing routine tasks in their kitchens, such as cooking, food preparation, cleaning, organizing, and unloading grocery bags. Each user first defined five tracking regions in different parts of their kitchens. In subsequent sessions, the users interspersed object-teaching episodes with other activities in the kitchen at will—the set of objects and activities were not predefined or scripted, and the system was used as part of daily activities. Throughout the case study, the system used the `ViT-B/16` version of the CLIP model in zero-shot mode to produce and render predictions based on the object views that were filtered as described in Section 6.1. While this exploratory study has some clear limitations (i.e., only two participants, who both have knowledge of the system's internal functioning and limitations), we believe it is sufficient to provide an initial assessment of the proposed approach, reveal important challenges to be resolved, and highlight interesting research opportunities for the community.

There were 82 object types introduced at Site 1, and 47 at Site 2. However, analysis of the collected dataset revealed that some of these object types were *synonyms*, i.e., the user provided two or more different labels for the same object at different points in time. For example, an object that was labeled as "box of tomato sauce" was labeled at a later time as "box of tomato paste." Label synonyms can arise from the natural ambiguity in language and object taxonomies, as well as from speech recognition errors (i.e., the user at Site 1 mistakenly confirmed an object misrecognized as "open gloves" rather than "oven gloves") and represent just one of many challenges for object recognition in the real-world. The data from Site 1 included 8 synonyms (resulting in 74 unique types), and no synonyms were identified at Site 2.

We used the collected data to assess the performance of CLIP in zero-shot mode and with the basic learning technique described above. Before reporting the results in Sections 7.2 and 7.3 below, we begin with some preliminaries regarding the mechanics of training and incremental evaluation for interactive continual learning.

7.1 Evaluating Interactive Continual Learning

Well-established metrics for object detection and recognition have played an important role in evolving the state-of-the-art in the

Table 1: Evaluation results for CLIP in zero-shot mode, with and without filtering views for motion blur and occlusion.

Dataset	# Object Types Site 1 / Site 2	# Object Instances Site 1 / Site 2	# Object Views Site 1 / Site 2	Frame Accuracy Site 1 / Site 2	Aggregate Accuracy Site 1 / Site 2
Filtered	69 / 47	350 / 282	26,402 / 15,095	49.3% / 50.4%	51.4% / 57.8%
Unfiltered	74 / 47	427 / 332	121,421 / 55,657	36.5% / 41.7%	38.7% / 46.3%

computer vision community. The continual, interactive learning setting brings new challenges, not only in terms of principles and models, but also in the design of metrics and evaluation [23]. Useful metrics should not only provide the means for reporting progress, but also reflect the experience a user has with the system.

To relay the complexities involved in training and evaluation for continual interactive learning scenarios, we direct the reader's attention to Figure 6, which illustrates data captured by our system over time, across two hypothetical consecutive runs of the application, denoted as Session 1 and Session 2. The gray dots at the top represent the color image frames for which object tracking information is available. Note that frames may be missing, as the object tracker component may drop frames if it is not able to keep up with the incoming streams.

The rectangles in Figure 6 represent instances of *situated-objects*, by which we mean an object existing at a given location, as detected by the system. For instance, the system first detects object 1 at time t_5 at a given location, and detects that object 1 is no longer at that location after t_{23}. Note that this representation only reflects the system's automatic detection result—the object may have in reality existed at that location prior to t_5, and may have moved to a new location before or after t_{23}. The circles inside the situated-object rectangles represent image views of the objects, cropped from the corresponding color image frame at that time point. As discussed in Section 6.1, some of these views (represented with a crossed circle) are filtered out because of motion blur or detected occlusions. Another event of importance is the arrival of labeled data via interaction, in this case when the user presses the virtual [+] button on top of the object. We refer to this action as the *labeling point*, marked by the asterisk above each object in Figure 6. Given a user-provided label, the system can propagate the label backwards and forward in time, increasing the quantity of labeled training data for continual learning.

In the next two sections, we assess the performance of the zero-shot use of CLIP and of the learning approach based on CLIP encodings *purely on the task of object recognition*, factoring out potential errors introduced by the 3D object detector, i.e., assuming correct object bounding-box information is available. Specifically, we use as an evaluation set the data for each object from the labeling point forward, which we refer to as the *confirmed region*—shown in gray in Figure 6. Throughout the confirmed region, we trust that the object bounding box is accurate, as the user confirmed it explicitly at the labeling point (perhaps after editing it). We consider the label provided by the user as ground truth for this confirmed region[2].

[2]During initial experimentation, we found that the system could robustly identify when objects disappear from a location. However, a formal study of the reliability of these automatically generated labels is left for future work.

We report two types of performance metrics: frame accuracy and aggregate accuracy. The first metric is computed by measuring the percentage of correctly classified object views (frames). For the second metric, we aggregate the predictions into object instances to generate an instance accuracy, which is then averaged across all instances of a given class into a class accuracy, which in turn is averaged into the final aggregate accuracy metric. We believe the second metric is less affected by the distribution and duration of objects in the data, i.e., frame accuracy might be influenced (up or down) by a single object whose existence spans a large duration.

When we run zero-shot inference or train the logistic regression model with CLIP encodings, we keep all the object classes introduced at run time (including the synonym variants), as this set reflects what the live system encountered. However, we do not penalize the models in evaluation when they predict a synonym for a given class. In addition, two object instances (one at each site) where inadvertently confirmed by the users incorrectly. We keep their data for training—again reflecting the noisy nature of data collected by a live system—but eliminate them from the evaluation since these labels do not accurately reflect the ground truth.

7.2 Results for CLIP in Zero-Shot Mode

We now turn our attention to the performance of the CLIP zero-shot approach. Recall that with this approach, there is no actual adaptation of the CLIP model itself. Nonetheless, the predictions are computed incrementally, and the set of classes evolves over time: if a label provided by the user at a labeling point has not been previously encountered, it is added to the set of classes from that point forward. The evaluation is conducted by comparing all of the incrementally obtained predictions against labels for all object views in the confirmed regions.

The first row in Table 1 shows the total number of object types, instances, and views (data points) available in the evaluation set, and the model performance in terms of the frame and aggregate accuracy, at each site. The results indicate that, while the model is able to often recognize the objects introduced by the users in a zero-shot fashion, a significant gap in accuracy remains. To assess the importance of the proposed object view filtering approach, we conducted an experiment in which we did not filter out the views with motion blur and occlusion. As the second row in Table 1 shows, without filtering, the number of object views increases significantly and performance decreases by roughly 10% (note also that five object types at Site 1 had been completely removed due to filtering). This result highlights the importance of selecting high quality views. In Section 8, we return to this topic and discuss future directions for the selection of object views.

Table 2: Recognition accuracy for learning with CLIP encodings, propagating labels forward in time vs both forward and backward, and considering all object instances vs instances for which the object type has been seen at least once before.

Label Propagation	Instances	Frame Accuracy		Aggregate Accuracy	
		Site 1	Site 2	Site 1	Site 2
Forward	All	78.4%	75.2%	44.8%	51.6%
Forward	Seen	95.8%	96.4%	85.7%	90.9%
Forward and Backward	All	77.7%	74.7%	46.2%	53.0%
Forward and Backward	Seen	94.9%	95.9%	88.2%	94.1%

7.3 Results for Learning with CLIP Encodings

In a second evaluation experiment, we assessed the learning method described in Section 6.2. We simulate the construction of an adapted model by traversing the data chronologically and training a multinomial logistic regression layer on top of the image encodings generated by the CLIP model. Specifically, a new multinomial logistic regression model is trained upon reaching the endpoint of each *situated object* instance. In the illustrative example shown in Figure 6, the model M1 is constructed at time t_{23} (with the views from object 1), M2 at time t_{28} (with the views from objects 1, 2 and 3), M3 at t_{108} (with the views from object 1, 2, 3, and 4), and so forth. When evaluating, the latest model is used going forward for all incoming object views, until a new model is generated at the next update point. For example, model M1 from Figure 6 is used for the image views for objects 2 and 3 between times t_{24} and t_{28}. This process simulates in essence an experience with the model at runtime, where the current model is used on upcoming object views, while ensuring that we do not evaluate a model on views belonging to the same *situated object* instance on which we trained.

The first row in Table 2 shows the frame and aggregate accuracy of the learned model at each site. While the frame accuracy has increased compared to the zero-shot setting, the aggregate accuracy metric is significantly lower in this case than the frame accuracy, i.e., 44.8% vs 78.4% at Site 1, and 51.6% vs 75.2% at Site 2. We believe the low aggregate accuracies are explained in part by the fact that the adapted model is a closed-world model. Because the model is updated at the end of situated object instances, it often encounters object types during evaluation that it has not seen before. We determined that 19 out of the 69 object types from Site 1 (after filtering and collapsing synonyms), and 13 out of 47 at Site 2, were singletons, i.e., they occurred in only one instance. For these objects, the performance of the logistic regression model is always zero—as the model does not even know that the class exists when it encounters the datapoints, significantly lowering the aggregate measure.

To assess how well the model performs after it has seen an object type at least once, i.e., after it has a chance to learn about it, we eliminate from the evaluation the first encounter with each object type. Evaluation results on instances for which the object type has been previously seen at least once are shown in the second row in Table 2. Indeed, in this case, both the frame and aggregate accuracies are significantly higher.

Finally, we conducted another experiment in which we propagated the labels for training not just forward over the confirmed region (where the user has confirmed the bounding box for the object), but also moving backwards in time, all the way to the detection point for each object, as illustrated schematically for object 1 in Figure 6. By projecting the labels back in time, we gain additional training data for each instance, although the additional training views may come from periods when the segmentation was still poor (before the user had finished editing the bounding box). The results are shown in the third and fourth row from Table 2. We observe a modest increase in aggregate accuracy over object types that have been previously seen.

8 DISCUSSION AND FUTURE WORK

The evaluation reported above indicates that, if the problem of accurate object detection and tracking can be resolved, an off-the-shelf platform model like CLIP [17] holds promise for constructing a system that learns about objects online and onsite. Besides the evaluation of the CLIP model reported above, the case study revealed several important issues that need to be addressed on the path to developing a system that can continually learn to recognize objects in the open world. We briefly discuss these issues below.

The proposed approach relies on the ability to detect and track objects in 3D space. In the current implementation, we used a heuristic voxel-based detection and tracking algorithm. We found that this algorithm works well when objects are large and clearly separated. However, it often fails when the scene is cluttered and includes many objects in close proximity. In these cases, multiple objects are often grouped together and identified as one single large object. Additional challenging situations arise with small objects, and with shiny, reflective, or transparent objects (which often go undetected in the depth camera). This makes it difficult to detect and label a wide spectrum of objects, for instance in the kitchen settings: reflective utensils, transparent bottles, shiny pans, etc.

The ability of the user to correct the automated object segmentations by resizing and rotating the bounding boxes compensates for some of the deficiencies noted above. In most cases[3], it is relatively easy for users to correct the object boundaries. The users adjusted the bounding boxes for 82% (Site 1) and 80% (Site 2) of the confirmed objects. We believe that improvements in the 3D object detection and tracking (e.g., using techniques that combine both depth and color-space information) can significantly improve the end-to-end performance and usability of the proposed approach.

[3]Sometimes the poor segmentation interferes with the gaze-based object selection algorithm, leading to rapid alternation between two incorrect bounding boxes when the user's gaze is on a given object; in these cases it becomes difficult to pinch a corner or edge grasp-point

As the results from Table 1 indicate, carefully selecting which object views to use for recognition matters. The current system selects views based on a pretuned motion blur threshold and occlusion parameters. In future work, we plan to more closely investigate the relationship between different image properties, such as motion blur, occlusion, and framing on the recognition performance of platform models like CLIP. Given the many challenges posed by real-world deployments, and the inevitable train-test distribution shifts, finding ways to characterize and filter object views that lead to better recognition, or to better models when used in training, are important directions for future research.

Finally, a number of challenges and opportunities were identified with respect to the mixed-reality object teaching interface. The current implementation does not allow the user to change the label for an object or to edit its bounding box after the object had been confirmed by pushing the [+] button. This is problematic in several ways. For example, we identified at least one case where the confirmation button was pressed accidentally as the user was trying to perform a pinching gesture to manipulate the object's bounding box. In another case, a speech recognition error in the label went unnoticed by the user until after the object was confirmed and introduced an incorrect class name in the set of classes for the object recognition model. Future work should consider evolving the affordances of the teaching interface to allow for corrections, as well as dealing with synonyms for the labels assigned to objects. These real-world issues, in turn, raise interesting algorithmic questions for continual learning, such as how to construct incremental model updates not just in situations where new data arrives in stream, but also where old data and previous model updates need to be revised.

9 CONCLUSION

We focused on opportunities to leverage the multimodal sensing and interaction capabilities of mixed-reality devices to develop systems that continually learn about physical objects in the open world. We proposed an approach in which users can label objects in stream, over the course of their regular activities, and views of the objects can be collected in the background and paired with the labels to continually learn how to better recognize those same objects. We developed an end-to-end implementation of the proposed approach using the HoloLens 2 device, and reported results and lessons learned from an initial, exploratory case study.

Overall, the results indicate that, in the presence of good segmentations, and with careful selection of which object views to account for problems like motion blur and occlusion, platform computer vision models like CLIP [17] hold promise for developing systems that learn continually to recognize objects in a given environment. Developing and experimenting with a concrete system brought to the fore important questions about the definition of training and test cases, and different methods for evaluating recognition performance in interactive continual learning settings. Directions for future work include propagation of user-specified object segmentations backward and forward in time from labeling events, reasoning about occlusions due to changing viewpoints and object configurations, online algorithms for selecting image views for inference and training, and the handling of unknown object types.

ACKNOWLEDGMENTS

We would like to thank Xin Wang, Ishani Chakraborty, Felipe Vieira Frujeri, Mahdi Rad, as well as the anonymous reviewers for useful discussions and feedback.

REFERENCES

[1] Touqeer Ahmad, Akshay Raj Dhamija, Mohsen Jafarzadeh, Steve Cruz, Ryan Rabinowitz, Chunchun Li, and Terrance E Boult. 2022. Variable Few Shot Class Incremental and Open World Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3688–3699.

[2] Sean Andrist, Dan Bohus, Ashley Feniello, and Nick Saw. 2022. Developing Mixed Reality Applications with Platform for Situated Intelligence. In *IEEE Conference on Virtual Reality and 3D User Interfaces – Abstracts and Workshops (VRW)*. IEEE, 48–50.

[3] Ali Ayub and Alan R Wagner. 2020. Cognitively-inspired model for incremental learning using a few examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 222–223.

[4] Dan Bohus, Sean Andrist, Ashley Feniello, Nick Saw, Mihai Jalobeanu, Patrick Sweeney, Anne Loomis Thompson, and Eric Horvitz. 2021. Platform for Situated Intelligence. arXiv:2103.15975 [cs.AI]

[5] Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. 2020. Se (3)-transformers: 3d roto-translation equivariant attention networks. *Advances in Neural Information Processing Systems* 33 (2020), 1970–1981.

[6] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, Miguel Martin, Tushar Nagarajan, Ilija Radosavovic, Santhosh Kumar Ramakrishnan, Fiona Ryan, Jayant Sharma, Michael Wray, Mengmeng Xu, Eric Zhongcong Xu, Chen Zhao, Siddhant Bansal, Dhruv Batra, Vincent Cartillier, Sean Crane, Tien Do, Morrie Doulaty, Akshay Erapalli, Christoph Feichtenhofer, Adriano Fragomeni, Qichen Fu, Christian Fuegen, Abrham Gebreselasie, Cristina Gonzalez, James Hillis, Xuhua Huang, Yifei Huang, Wenqi Jia, Weslie Khoo, Jachym Kolar, Satwik Kottur, Anurag Kumar, Federico Landini, Chao Li, Yanghao Li, Zhenqiang Li, Karttikeya Mangalam, Raghava Modhugu, Jonathan Munro, Tullie Murrell, Takumi Nishiyasu, Will Price, Paola Ruiz Puentes, Merey Ramazanova, Leda Sari, Kiran Somasundaram, Audrey Southerland, Yusuke Sugano, Ruijie Tao, Minh Vo, Yuchen Wang, Xindi Wu, Takuma Yagi, Yunyi Zhu, Pablo Arbelaez, David Crandall, Dima Damen, Giovanni Maria Farinella, Bernard Ghanem, Vamsi Krishna Ithapu, C. V. Jawahar, Hanbyul Joo, Kris Kitani, Haizhou Li, Richard Newcombe, Aude Oliva, Hyun Soo Park, James M. Rehg, Yoichi Sato, Jianbo Shi, Mike Zheng Shou, Antonio Torralba, Lorenzo Torresani, Mingfei Yan, and Jitendra Malik. 2021. Ego4D: Around the World in 3,000 Hours of Egocentric Video. *arXiv preprint arXiv:2110.07058* (2021).

[7] Hernisa Kacorri. 2017. Teachable machines for accessibility. *ACM SIGACCESS Accessibility and Computing* 119 (2017), 10–18.

[8] Michael Laielli, James Smith, Giscard Biamby, Trevor Darrell, and Bjoern Hartmann. 2019. Labelar: a spatial guidance interface for fast computer vision image collection. In *Proceedings of the 32nd annual ACM symposium on user interface software and technology*. 987–998.

[9] Kyungjun Lee and Hernisa Kacorri. 2019. Hands holding clues for object recognition in teachable machines. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–12.

[10] Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, Kai-Wei Chang, and Jianfeng Gao. 2021. Grounded Language-Image Pre-training. https://doi.org/10.48550/ARXIV.2112.03857

[11] Vincenzo Lomonaco and Davide Maltoni. 2017. Core50: a new dataset and benchmark for continuous object recognition. In *Conference on Robot Learning*. PMLR, 17–26.

[12] Daniela Massiceti, Luisa Zintgraf, John Bronskill, Lida Theodorou, Matthew Tobias Harris, Edward Cutrell, Cecily Morrison, Katja Hofmann, and Simone Stumpf. 2021. Orbit: A real-world few-shot dataset for teachable object recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 10818–10828.

[13] Daniel Maturana and Sebastian Scherer. 2015. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 922–928.

[14] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. 2019. Deep hough voting for 3d object detection in point clouds. In *proceedings of the IEEE/CVF International Conference on Computer Vision*. 9277–9286.

[15] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 652–660.

[16] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. 2016. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and*

pattern recognition. 5648–5656.

[17] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*. PMLR, 8748–8763.

[18] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2001–2010.

[19] Qi She, Fan Feng, Xinyue Hao, Qihan Yang, Chuanlin Lan, Vincenzo Lomonaco, Xuesong Shi, Zhengwei Wang, Yao Guo, Yimin Zhang, et al. 2020. OpenLORIS-Object: A robotic vision dataset and benchmark for lifelong deep learning. In *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, 4767–4773.

[20] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. 2015. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*. 945–953.

[21] Xiaoyu Tao, Xiaopeng Hong, Xinyuan Chang, Songlin Dong, Xing Wei, and Yihong Gong. 2020. Few-shot class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12183–12192.

[22] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. 2019. Densefusion: 6d object pose estimation by iterative dense fusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 3343–3352.

[23] Jianren Wang, Xin Wang, Yue Shang-Guan, and Abhinav Gupta. 2021. Wanderlust: Online Continual Object Detection in the Real World. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 10829–10838.

[24] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1912–1920.

[25] Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, et al. 2021. Florence: A New Foundation Model for Computer Vision. *arXiv preprint arXiv:2111.11432* (2021).

[26] Chi Zhang, Nan Song, Guosheng Lin, Yun Zheng, Pan Pan, and Yinghui Xu. 2021. Few-shot incremental learning with continually evolved classifiers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12455–12464.