

# LabelAR: A spatial guidance interface for fast computer vision image collection

*James Smith  
Michael Laielli  
Giscard Biamby  
Trevor Darrell  
Björn Hartmann*



Electrical Engineering and Computer Sciences  
University of California at Berkeley

Technical Report No. UCB/EECS-2019-58

<http://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-58.html>

May 17, 2019

Copyright © 2019, by the author(s).  
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

---

# LabelAR: A spatial guidance interface for fast computer vision image collection

James Smith

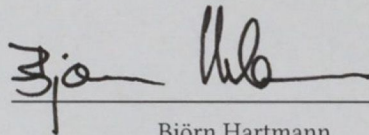
---

## Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

### Committee

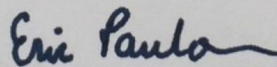


Björn Hartmann  
Research Advisor

5/13/2019

(Date)

\*\*\*\*\*



Eric Paulos  
Second Reader

17 MAY 2019

(Date)

## **Abstract**

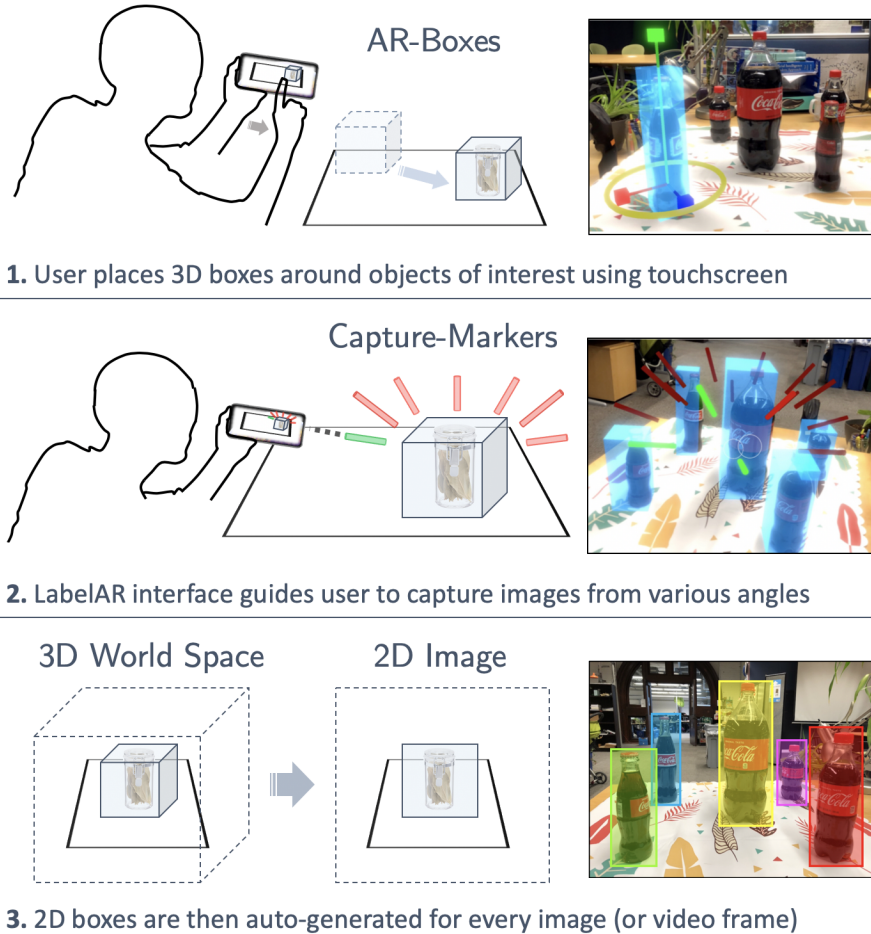
Computer vision is applied in an ever expanding range of applications, many of which require custom training data to perform well. We present a novel interface for rapid collection and labeling of training images to improve computer vision based object detectors. LabelAR leverages the spatial tracking capabilities of an AR-enabled camera, allowing users to place persistent bounding volumes that stay centered on real-world objects. The interface then guides the user to move the camera to cover a wide variety of viewpoints. We eliminate the need for post-hoc manual labeling of images by automatically projecting 2D bounding boxes around objects in the images as they are captured from AR-marked viewpoints. In a user study with 12 participants, LabelAR significantly outperforms existing approaches in terms of the trade-off between model performance and collection time.

# Contents

1	Introduction . . . . .	4
2	Related work . . . . .	7
2.1	Image collection . . . . .	7
2.2	Interfaces for Post-Hoc Labeling . . . . .	8
2.3	Spatial User Interfaces for Interacting with 3D Objects . . . . .	8
3	LabelAR Interaction Design . . . . .	9
3.1	Placing 3D Bounding Volumes . . . . .	9
3.2	Encouraging Diverse Image Perspectives . . . . .	11
4	Implementation . . . . .	11
5	Evaluation . . . . .	13
5.1	Variables . . . . .	14
6	Results . . . . .	19
6.1	Quantitative results . . . . .	19
6.2	Qualitative Results . . . . .	25
7	Discussion . . . . .	25
8	Limitations . . . . .	27
9	Future Work . . . . .	28
10	Conclusion . . . . .	28
	<b>Bibliography</b>	<b>30</b>

# 1 Introduction

Computer vision is being used in an increasing number of user-facing systems. Deep neural networks used in modern computer vision can require copious amounts of training data. Modern applications often rely on large datasets (10-200GB) to train these models. Often these datasets are collected by scraping the internet for existing images and then manually labeling them, for example through crowdsourcing [9].



**Figure 1:** We use an AR device to jointly collect and label training images. Typically this is performed separately by a camera and a web-based labeling tool. The resulting images captured using the LabelAR interface can be immediately used to train a computer vision model for multiple-object detection.

However, existing datasets produced in this fashion do not cover the “long tail” of use cases — where users have a need to detect specific classes of items that are not already labeled in existing datasets. An example would be that we may want to ask a robot to “bring me my jacket.” Although there is plenty of training data of jackets, there are no efficient ways to collect training data for a specific instance of a category (“my jacket”). Another example is needing a vision system to distinguish between different types of industrial hardware. Datasets for such fine grained categories can be difficult to source due to the effort needed in collecting them.

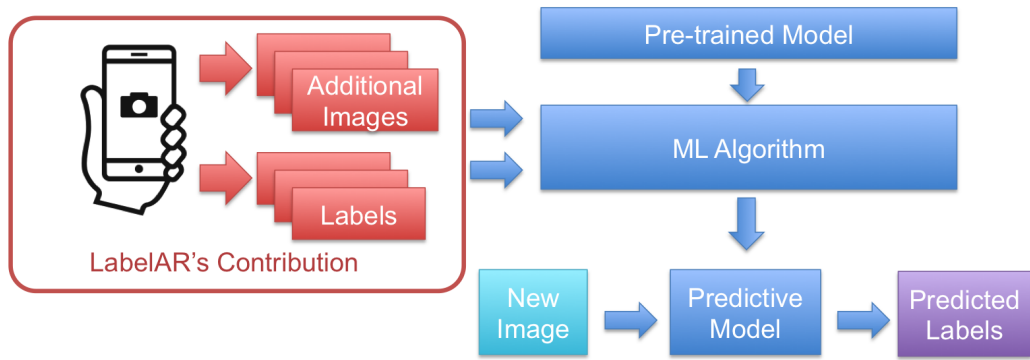
Existing approaches to overcome this bottleneck include parallelizing the labeling task with post-hoc annotation tools [36] or simple camera interfaces to guide image collection such as putting a bounding box in a viewfinder to avoid post-collection labeling [16]. The first approach can produce high quality data but is a very time intensive process that scales linearly with the number of objects or images that you wish to label. The second approach is significantly faster, but produces fairly low quality data, as the bounding box accuracy is compromised.

We propose LabelAR, an augmented reality interface that allows users to rapidly collect and label high quality training image datasets for computer vision (see Figure 1). LabelAR is applicable to any setting where a user needs to adapt a computer vision object detection model (see Figure 2). Transfer learning is a technique that can leverage a model pre-trained on large datasets to recognize new objects. Training data needed for transfer learning needs to be situated in context and diverse in viewpoint variety.

Two illustrative use cases where such adaptation is necessary are *augmented reality assembly* and *home robotics*. In *augmented reality assembly*, a worker employs a head-mounted AR device to project visual, step-by-step instructions that demonstrate how to assemble a collection of object parts. Object detection can be used for identifying and tracking particular parts throughout the assembly. A recent study concluded that better tracking capabilities are still needed for AR assembly to be sufficiently robust for industrial applications [11]. In *home robotics*, a robot owner would adapt a computer vision model for use on a robotic platform to recognize individual items in a household that may be significantly different from items in existing training sets.

Our interface design is based on the observation that computer vision model performance depends on the quality of the training data. Two important aspects of high quality training data are accurate bounding boxes for labels and a diversity of images (orientations, scales etc) of the objects to be recognized [5]. Our interface embodies the following two insights:

First, the spatial tracking algorithms used for AR can be harnessed to enable users to quickly and accurately place spatially stable 3D bounding volumes around objects of interest in their



**Figure 2:** LabelAR produces additional training data to adapt computer vision models to recognize additional objects in a user’s environment.

immediate environment, which can be used to automatically generate accurate bounding boxes at any camera angle (Figure 1).

Second, interactive guides in the AR interface can prompt the user to collect training data instances of the bounded objects from many viewpoints. We show that these appearance variations combined with accurate bounding box labels improve detection accuracy.

Through empirical experimentation, we show that our interface enables a better trade-off between time costs and model performance than existing baseline methods. We conduct a user study with 12 participants that compares collection times and model performance (when trained on collected images) between LabelAR and two alternative interfaces [36][16]. Compared to post-hoc labeling, collection times improved by over 2× on average with LabelAR ( $p < 0.05$ ), while model performance was similar. Compared to an existing rapid collection application, model performance increased by a factor of 4× on average ( $p < 0.003$ ), while collection times were similar. Additional empirical analysis shows that the equally spaced viewing angle intervals afforded by LabelAR are effective for low-sample learning and fast computer vision model training.

The main contributions of this work are the design, implementation and evaluation of an augmented reality interface for fast computer vision image collection. We demonstrate both AR-capable smartphone or head-mounted device implementations and show gains in collection time and model performance over existing approaches.



## 2 Related work

Prior work falls in the areas of image collection methods, post-hoc labeling interfaces, and other spatial user interfaces for interacting with 3D content. We review each area in turn.

### 2.1 Image collection

Several projects seek to shorten or eliminate post-hoc labeling time through novel capture-time interfaces and techniques. Raptor [16] modifies the camera interface by overlaying a pre-sized bounding box. The user points it to a new object, ensuring it is displayed within the pre-defined box area. Since the object is fit to the box with known image coordinates, there is no subsequent labeling task required. However, this results in images that are all collected at the same scale. The Doubleshot technique [35] asks users to take two images, one with the object, one without the object (my manually removing it) to automatically calculate labels. Sermanet et al. [28] use a two-person collection strategy for capturing multiple views of an object simultaneously. This allows training to be “self-supervised”. Our work differs in that it uses spatial tracking and user-placed 3D bounding volumes to create labeled images.

Another set of collection approaches use head-mounted cameras to collect video along with separate audio recording devices to allow the user to speak the labels verbally, either simultaneously [31] or immediately after video collection [7]. Both of these approaches rely on speech recognition APIs to extract functional labels from the user recordings. While these prove effective for categorization labeling, they do not aid in the placement of bounding box labels.

Other approaches employ robots to perform the image collection [22, 25, 32] The Amazon Robot Picking Challenge shows joint collection and labeling applied to a challenging real-world task [37]. They use a robot arm to capture multiple angles of a single object and, with knowledge of the background, automatically obtain segmentation labels by foreground masking. The idea behind LabelAR is similar, but of course the collector is a human. Furthermore, LabelAR is applicable to more than one object at a time and does not rely on prior knowledge of the background appearance.

Recent work at the intersection of cognitive development and computer vision shows deep neural networks categorization performance can be improved by increasing the variety of viewing angles. [5]. We seek to exploit AR to guide the user to capture such a variety of images.

Our work is also related to applications for building 3D models from images. For example, the Vuforia AR toolkit supports creation of a 3D model of an object by first scanning the object with an Android phone [19]. Our work seeks to improve object detection performance without relying on 3D model creation.

## 2.2 Interfaces for Post-Hoc Labeling

Image and video annotation for computer vision model training and adaptation is typically done via web interfaces where a human annotator sits at a terminal and uses a mouse to draw bounding boxes or segmentation boundaries on various objects of interest. [20].

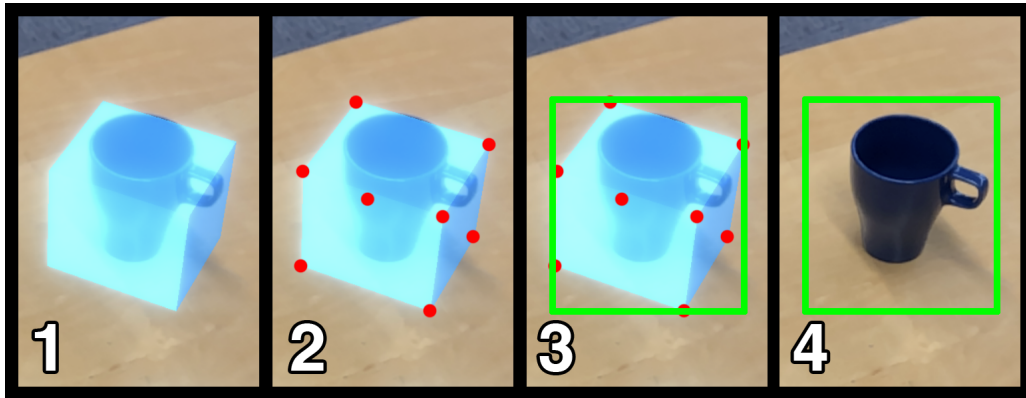
Web-based 2D bounding box labeling interfaces such as LabelMe [27] have been integral in constructing some of the most influential computer vision data-sets to date [9, 12], allowing labeling tasks to be distributed in the form of global crowdsourcing campaigns.

There are a few works that use 3D labeling tools to label a 3D scene, then leverage the 3D labels to generate large amounts of 2D labels. [6, 33] Our work makes use of 3D to 2D label transfer, but for real-time collection of interactive objects rather than post-hoc passive labeling of 3D scenes.

Several works leverage interactivity between the learner and predictive model to reduce human labeling time and effort by having the model predict labels that can then be approved or improved by the user [3]. *Crayons* [13] uses a simple interactive painting metaphor to reduce classifier creation time. A more recent approach incorporates interactivity into web-based crowdsourcing tools showing that interactive modes can reduce the number of annotator mouse-clicks by as much as 50 percent [1]. *CueFlik* [2] presents a design and evaluation of new approaches to guiding users in selected training examples interactively based on model predictions. Our work does not incorporate interactivity in this sense, rather it focuses on the interaction between the user and real-world objects of interest. *Eye-patch* [24] is a tool for designing camera-based interactions. The authors identify a need to accelerate the example-collecting process as a result of their deployment, which is aligned with LabelAR’s goals.

## 2.3 Spatial User Interfaces for Interacting with 3D Objects

Technologies and interaction techniques for spatially tracked screens and near-eye displays have been a focus of HCI research since pioneering efforts by Sutherland [29], Fitzmaurice [14] and others. A number of spatial interaction techniques can now be found in the literature — e.g. in surveys by Hinckley [17] and Argelaguet Sanz [4]. Early interfaces such as Peephole Displays [34] and the Boom Chameleon [30] used translational and rotational tracking of a hand-held display to navigate, view and annotate with large virtual maps and 3D models, respectively. Most relevant to LabelAR are interfaces for *Situated Modeling* where real-world context is used to create and place 3D geometry such as our bounding volumes [18, 21]. Our contribution differs in that our created 3D geometry is a means towards the end of collecting image sets and we study the benefits of such an approach for computer vision.



**Figure 3:** Process for extracting two dimensional bounding boxes from virtual bounding volumes. 1 - a virtual bounding volume is placed over a real object, 2 - the corners (red) of the volume are projected into camera space, 3 - a min/max is taken over those points to find a two dimensional bounding box (green), 4 - annotations are saved for the object.

### 3 LabelAR Interaction Design

At a high level, LabelAR seeks to lower the time that a user needs to spend collecting and annotating images of objects for training neural networks. We identified two key areas for improvement in existing workflows.

**Hand annotating bounding boxes:** hand annotated bounding boxes are often pixel perfect and very high quality, but take a long time to author. Today, this process is sometimes parallelized through crowdsourcing, which reduces time but not the total amount of labor required. We hypothesize that AR technology can automate the process of producing bounding boxes to a high degree, given an initial 3D bounding volume of an object.

**Capturing a large variety of training examples:** when users need to train a model to recognize instances of an object, they need to provide training data. We hypothesize that offering a guided experience with feedback to the user about how much of a variety they have collected can help them produce training sets that will result in better model performance.

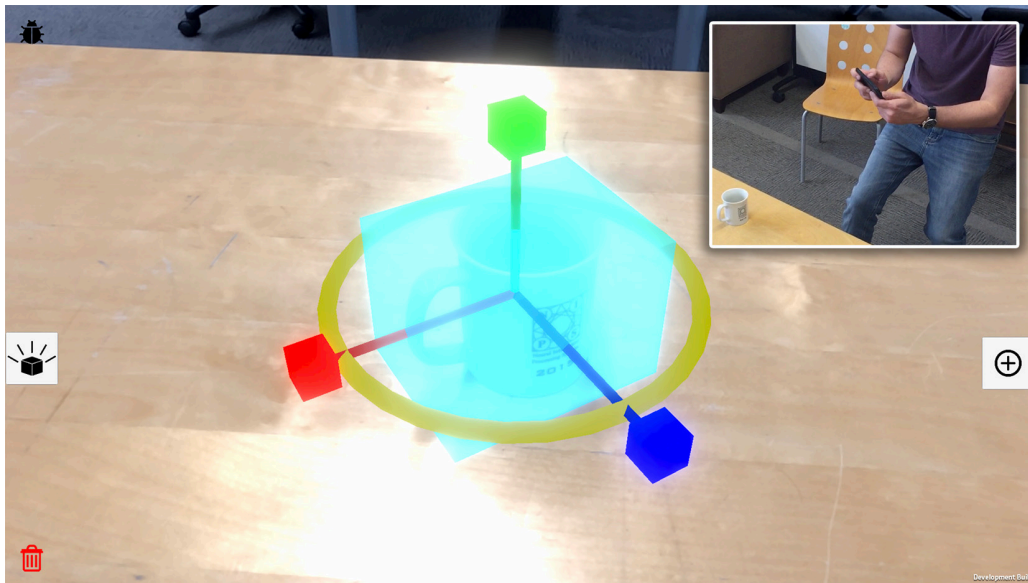
#### 3.1 Placing 3D Bounding Volumes

In existing workflows, when annotating a series of images that contain the same objects, users have to annotate a given object multiple times. LabelAR speeds up this process by asking the user to place a bounding volume around an object once, and then tracks it in all subsequent images.

To this end, we utilize AR technology. One of the important technologies in AR devices is the ability for them to self-localize, e.g. using SLAM [8]. This allows virtual objects to be placed in the real world environment, and for their positions and orientations to remain coherent.

LabelAR allows users to place virtual bounding boxes (holocubes) over objects in the environment so their positions can be tracked. When an image is taken in our interface, the two-dimensional bounding box can be computed for every object in the scene by projecting the holocube into the video frame and finding the bounding box of its vertices (Figure 3). It is thus important that the holocubes fit the size and shape of the objects if interest as closely as possible. To this end, we provide users translation, rotation, and scaling (TRS) widgets tools to manipulate the 3D position and size, as well as rotation around the axis normal to ground plane of the holocubes. The interaction design for the TRS manipulators mimics conventional widgets in 3D graphics software (Figure 4).

Users can also place cubes over multiple objects. In traditional approaches, the labeling effort grows as the product of  $images \times objects$ , which becomes prohibitive for long sequences containing many objects. In LabelAR, each additional object to be captured only incurs the one-time effort of placing and adjusting another bounding volume.



**Figure 4:** Users can move, re-size, and rotate bounding volumes by interacting with the TRS widget.

### 3.2 Encouraging Diverse Image Perspectives

LabelAR helps users collect a wide variety of images of given objects. In particular, it may direct users to capture objects from many azimuth and altitude angles. To accomplish this, we provide an interface to assist users in taking pictures of their objects. LabelAR will automatically take a picture if the user has a significantly different viewpoint than all previous pictures taken. This encourages the user to move the camera around the objects they wish to capture to ensure they get a variety of angles.

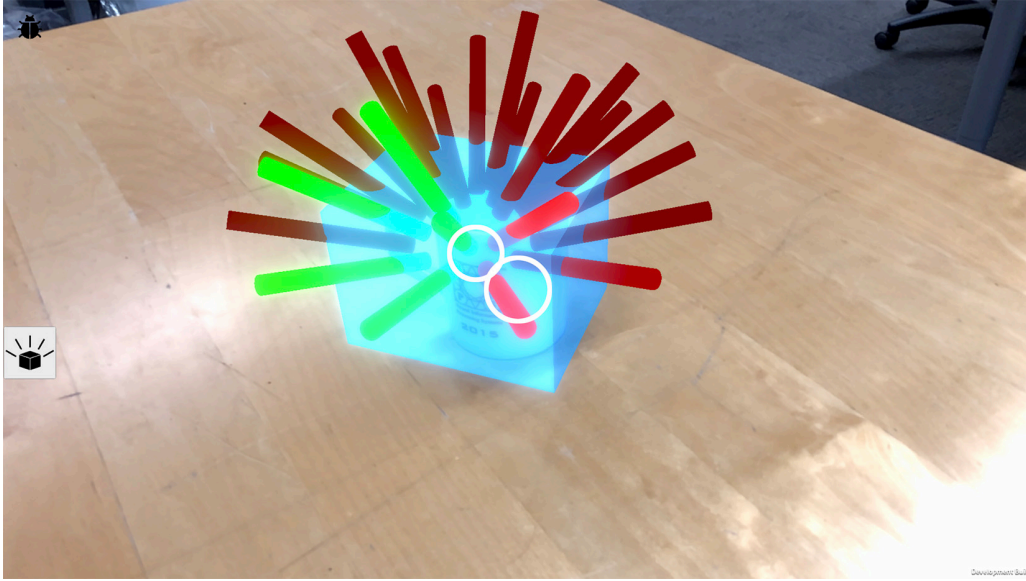
To facilitate this movement, our interface shows the user where they have already taken pictures and suggests new positions at which to take pictures while also requiring them to keep the holocubes in frame. This constraint means that there is a ray to which the user must move their camera and orient it such that they capture all of the holocubes.

We visualize this constraint to the user as a series of line segments that we refer to as markers (Figure 5). Markers change their color as users approach the correct position, providing them with real-time feedback about their progress. There is also a targeting cursor to direct the camera's orientation (Figure 5). A target for the cursor appears at the location in space that the user should point the camera to. Images are taken for the user automatically as soon as they are in position and the targeting cursors are aligned.

## 4 Implementation

We implemented LabelAR on two different AR platforms: hand-held, video-see-through on iOS devices, and head-mounted AR using Microsoft's HoloLens. Both versions were developed in the Unity game engine. A custom application was written to perform the function of LabelAR, and platform APIs were used to tie into hardware specific features.

The iOS version of LabelAR uses Apple's ARKit library, in particular the camera localization and plane detection functionality. As surfaces in the environment are detected, they are converted into planes that can be utilized in a Unity application. When the user places holocubes, they are automatically snapped to surfaces by performing raycasts onto these planes. To create new cubes, the user can press and hold a "new object" button that hovers a cube in front of them, which they can drop onto a table. Interaction with the holocubes and interface happen with the device's touch screen. When running ARKit apps on iOS, the camera is put into a special video mode that has a different field of view than the standard camera mode. Because of this, images are saved directly from the Unity rendering pipeline to make sure that the projected bounding boxes are guaranteed to line up. We down sample these images before saving to improve serialization times,

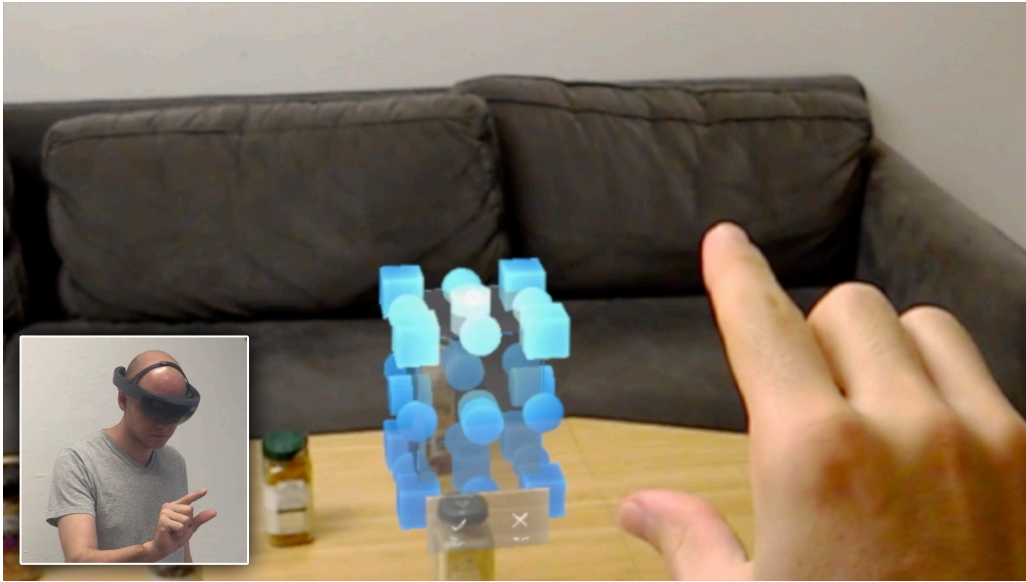


**Figure 5:** Markers indicate to the user where good next potential images should be taken from, and change color to indicate the user's proximity to the correct location (red) or if they have already captured an image at that location (green). Images are automatically captured when the user is in position and the two targeting circles align.

and because many cv model training pipelines down sample training data. Images are saved at  $960 \times 540$  resolution, and annotations are saved in the COCO json format. Images and annotations are automatically saved to the persistent storage of the application, which can be downloaded through XCode. This was chosen instead of saving images to the iOS Photos app so that images and annotations could be co-located.

The HoloLens version (Figure 6) uses Microsoft's Mixed Reality Toolkit (MRTK), which contains a set of assets for building native HoloLens applications within Unity. Interactions on the HoloLens are performed by using the standard gaze plus thumb-and-index-finger pinch gesture. MRTK contains a set of widgets for manipulating the size and orientation of virtual objects, so these were used in lieu of our own TRS widget. Objects are moved by using the built in hand tracking detection native to the hardware. Interacting with screen space interface elements has some issues on head mounted AR devices because the interactions are gaze driven, so we opted to use the phrase detection and dictation engines available on the hardware to allow the user to place new holocubes and initiate or stop the capture process. Images are saved at  $1280 \times 720$  pixels. Captured

images and bounding boxes are sent over a network connection to a server for collection, and bounding box data is converted in the COCO json format.



**Figure 6:** The translate, rotate, scale widget in the HoloLens version of the interface uses platform specific widgets. Sub pictures depicts the interface in use.

## 5 Evaluation

To evaluate the ability of our interface to meet our stated goals, we conducted a user study along with supporting ablation experiments to answering the following questions:

1. **Is AR based data collection faster?** How much time does it take to collect and annotate images compared to existing baseline methods?
2. **Does our AR interface result in better object detection models?** Does training on data collected via our interface result in a better model than training on data captured with other baseline collection-and-labeling methods?
3. **How accurate are the labels produced by AR based image collection?**
4. **How sensitive is detection performance to design choices such as the number of angles presented in the guidance interface?**

Our goal in defining an experimental setup was to portray a challenging object detection task in an environment that is both realistic and representative of plausible use-cases. We constrain the problem to the computer vision task of multiple object detection: That is the joint task of categorizing and localizing (via bounding boxes) any instances of a predefined set of objects. This is the type of problem that would need to be solved for our *in home robotics* and *augmented reality assembly* motivations.

## 5.1 Variables

The primary **independent variable** in our study is the interface used to capture and label images: We compare LabelAR to two baseline methods: image capture with post-hoc annotation using the Scalabel annotation tool [36]; and 2D guided capture using an overlaid 2D bounding box collection tool, such as the one in the Raptor project. Thus we have one independent variable with three levels: 1) LabelAR; 2) Post-hoc annotation; and 3) overlaid bounding box interface (referred to henceforth as “overlaid interface”).

The two primary **dependent variables** in our comparative study are *collection time* for an image set and object detection *model performance* when trained on collected images. We define collection time as the total time required for a participant to capture and completely label a set of images for 5 objects, each from multiple angles.

We define object detection performance as average precision (AP) of a trained object detection model at a given intersection-over-union (IOU) threshold between predicted and gold-standard bounding boxes. IOU is the area of intersection of two boxes (participant-generated vs. gold-standard) divided by the area of their union. See Figure 11 for a visual example of different IOU values. We used AP as our performance metric because it is a standard metric for object detection in popular computer vision benchmarks [23]. We chose to investigate IOU thresholds of 0.25 and 0.5. Data collected during our experiment was used to train a Fast R-CNN model [15].

We also investigate bounding box accuracy on *collected* images by calculating IOU between sets of randomly chosen participant-collected images and gold-standard annotations performed by the authors on those images.

We also collect qualitative feedback through a post-study survey.

*Participants:* 12 participants were recruited through email invitations sent to the departmental list-serves of the EE and CS departments at our university. The mean age is 26.4 years. 9 of the participants are male and 3 are female. Half of the participants had at least some prior experience with machine learning, computer vision, or both. All participants had taken classes in computer science. Although our study audience all have a technical background relative to the general



population, we believe that this makes them well suited to perform better in using the non-LabelAR interfaces as they likely have more prior understanding of data-driven how algorithms work. We also recognize that our study participants have high levels of technical literacy that will likely make all tested interfaces perform better than the general population average.

*Apparatus:* Each study took place in the same office space with one participant and one researcher administering the experiment. Each participant performed three separate collection-plus-labeling tasks using a dedicated app for each task on an iPhone 8plus. Each task asked the user to capture a series of images of multiple objects placed on a round table in the center of the room. The object sets were switched out in-between each task so that no user had the same set twice. The table was covered with a patterned table cloth to ensure the ARKit low-level functions had a sufficient amount of visual features for stable plane detection and tracking. All three apps were built with Unity and ARKit 2.0, and were deployed to the same iPhone 8plus running iOS 12.2 to ensure that all images were recorded in the same encoding, resolution, and aspect ratio (sRGB, 960 x 540) as the evaluation set. The Scalabel interface was run on a 2017 15-inch MacBook Pro running macOS Mojave version 10.14.2. The participants were given the choice of using a mouse or the laptop track pad for drawing bounding box labels.

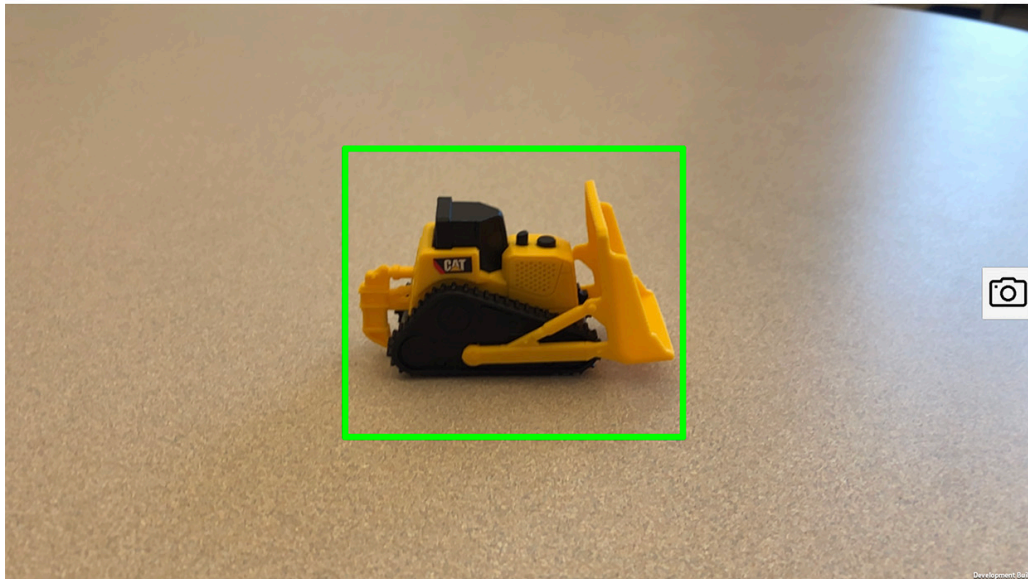
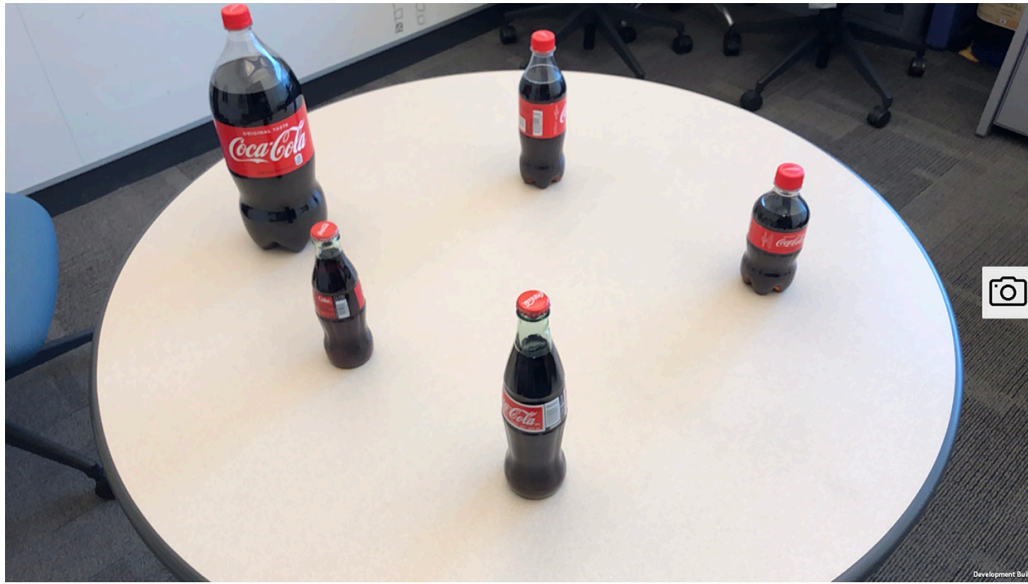
*Procedure:* Each study proceeded as follows. Upon receiving the user’s consent for participation, the researcher gave a 2-minute introduction of the LabelAR project starting with the following description:

Let’s say you just bought a robot that cleans up your living room. You might want it to recognize your personal items so it knows where to put them. LabelAR would help you teach the robot to recognize those things. So what you’re going to do is basically take a bunch of pictures of (these) items we have laid out on the table with a few different apps.

The purpose of the study and a procedural overview were then explained to the user.

The first task was to use the conventional camera app to collect images of an initial set of objects (Figure 7). A 1-minute tutorial was given on a single practice object to ensure the participant knew how to work the app. Once complete, the initial set of five objects were introduced and the participant was given guidance on how to proceed. A few simple suggestions, consistent with common computer vision knowledge and best practices, were made to each user:

1. *Take images from varied viewpoints.*
2. *Balance the number of times each object appears among the collected images and make sure each object appears at least a few times.*



**Figure 7:** Top: screenshot of the traditional camera application used to collect images for post-hoc annotation. Bottom: Picture of the overlaid interface, the onscreen bounding box has been highlighted in green for visibility, but is actually white. This box remains constant in size and the user positions the camera such that the subject fills the bounding box.

Additionally, the users were advised that they have a time limit of seven minutes to take as many pictures as they wish, but that they can stop early if they felt they'd captured enough data. They

were reminded that the hypothetical robotic assistant would need to recognize objects from various angles around a room and the training images they collect should reflect that. They were also told that they can move the objects around the table if they want, but that they might not want to bother flipping the object on their side or upside down since those views are not captured in the evaluation data.

The second task proceeded much like the first but instead with the overlaid interface, which is essentially a traditional camera app but with a 2D box guide on the screen so that the user can fit the object to the box (Figure 7). In the overlaid interface tutorial, the participant was again given a practice object, but this time was advised to take images of only one object at a time and to ensure the object fit in the box on the screen without exceeding the boundaries and without appearing too small within the box. The researcher was careful to explain that to maximize performance, the other objects on the table should not appear within the box or elsewhere in the image, or else it could confuse the robot during training. The same guidance on viewpoints, time limit, and number of images was given as in the first task. A new set of 5 objects was placed on the table and the participant was given seven minutes to take as many images as they wanted.

The third task used our interface. This time the researcher opened and initialized the app before handing it over by scanning the table for a few seconds to let ARKit find low-level visual features to establish a spatial plane on the table. The participant was advised to keep the phone generally pointed towards the table so that the app doesn't lose track of the table position. The participant was guided to place and fit an AR-box over a practice object, first fitting the sides of the object by positioning for a top-down view, then adjusting box-height from a side view. Then, the participant was told how to activate the capture marker system and how to capture an individual marker. Similar to the other tasks, a new set of five objects was placed in a rough circle on the table. Again, the participant was told they could move the objects, but that they should not do so once the capture markers were activated. For this task, we did not give advice on viewpoints or number of images, rather just to ensure each capture marker turns green.

The fourth task was to use a post-hoc annotation tool to label the images collected with the conventional camera app (the first task). A 5-minute tutorial was given on how to categorize and draw accurate 2D boxes around objects in the images with Scalabel in a time-efficient manner. The participant was advised to label overlapping objects to the extents of their respective visual features. Similarly, if an object was truncated by the image boundary, the participant was advised to label it only if 30% or more of the object was visible.

Our evaluation was always conducted in this order as to not introduce any bias in results for a user using LabelAR first. We believe that the user's natural instinct is to take a relatively small

number of pictures from a non-diverse set of angles (confirmed in our results), and that by using an interface that guides them to take a large, diverse set of images might pre-load that as a strategy for all interfaces, precluding counterbalancing the order.

#### *Design & Analysis*

**Computer vision task.** In our experiment, we focus on the computer vision task of object detection where the goal is to categorize and locate objects by drawing bounding boxes around them. For each set of training images collected by the user study participants, we train a Faster R-CNN [15] detection model until convergence and test its performance on a hold-out set of 360 images (each with 5 object instances) collected by scattering objects on floors and table throughout several rooms in each of 3 different buildings. Each object instance in the test set was meticulously labeled with a 2d bounding box. Some example test images are shown in figure 8

**Object categories.** We ran our experiments with three sets of objects: Coke bottles, toys, and industrial hardware. Each set consist of five distinct instances of its respective object category. Each set was chosen for its fine level of categorical granularity - each is finer than a typical category in the ImageNet-1000 set [26], thus a detector that is only pre-trained on the ImageNet or COCO dataset would not suffice without additional training images. In order to fairly evaluate the collection capabilities of the three interfaces, the objects were also chosen to ensure a variety of sizes and shapes: the bottles are tall, the hardware is flat, and the toys are small



**Figure 8:** We use three sets of objects in our experiments: Coke bottles, toys, and industrial hardware, each with five distinct instances.

**Environmental constraints.** We constrained our experiment environment to static objects placed on a table. This *multiple-objects-on-a-table* setup is common across cognitive development literature, robot learning (e.g. visual pick and place tasks), and fits with a common AR use case of multi-object assembly.

## 6 Results

In this section, we present and discuss the results of our user study.

### 6.1 Quantitative results

Figure 9 shows collection times and average precision (AP) at 0.25 IOU for all users and the three tested interfaces. Figure 10 shows average precision when the IOU threshold is raised to 0.5. These figures allow us to investigate the trade-off between collection time and object detection performance. Table .1 summarizes the mean values for collection time and average precision across all users.

#### Collection Time

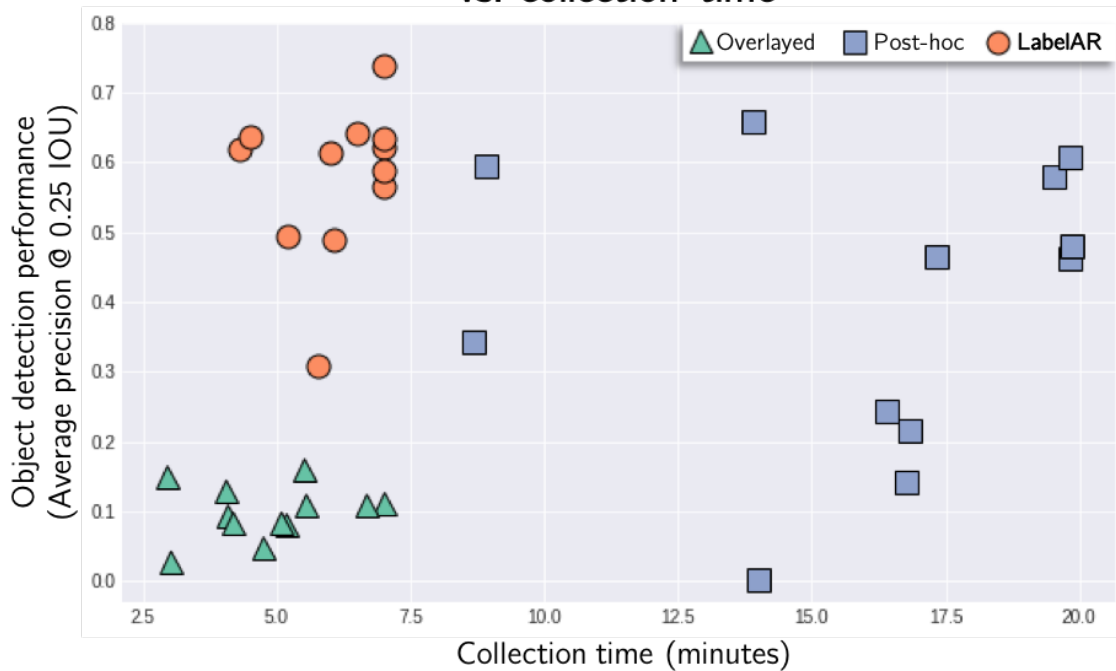
Collecting labeled images with LabelAR ( $\mu = 6.11$  min) is significantly faster (by 9.9 min or  $2.6\times$  faster) than using post-hoc annotation ( $\mu = 15.97$  min),  $t(22) = 8.39, p < .001$ . LabelAR is significantly slower (by 1.3 min or  $1.27\times$ ) than the overlaid interface ( $\mu = 4.82$  min)  $t(22) = 2.72, p < .05$ .

#### Average Precision

At an IOU threshold of 0.25, LabelAR ( $\mu = 0.58$ ) has significantly higher detection performance than both the post-hoc annotation ( $\mu = 0.40$ )  $t(22) = 2.63, p < 0.05$  and the overlaid interface ( $\mu = 0.10$ )  $t(22) = 14.48, p < 0.001$ . At an IOU threshold of 0.5, LabelAR ( $\mu = 0.22$ ) has a lower detection performance than post-hoc annotation ( $\mu = 0.32$ ), but the difference is not statistically significant  $t(22) = 1.28, p = 0.21$ . LabelAR remains significantly better than the overlaid interface ( $\mu = 0.02$ )  $t(22) = 4.34, p < 0.001$ .

To summarize, LabelAR was more than twice as fast in collecting and labeling images than post-hoc annotation, while its associated object detection performance either approaches or outperforms post-hoc annotation, depending on the chosen IOU threshold for detection. LabelAR is somewhat slower to use than 2D overlaid bounding boxes (by 1.3 minutes on average). That difference is much smaller than the difference to post-hoc annotation. LabelAR significantly outperforms overlaid bounding boxes in object detection by at least a factor of  $4\times$  at both investigated IOU levels.

## Object detection performance (AP@0.25 IOU) vs. collection time

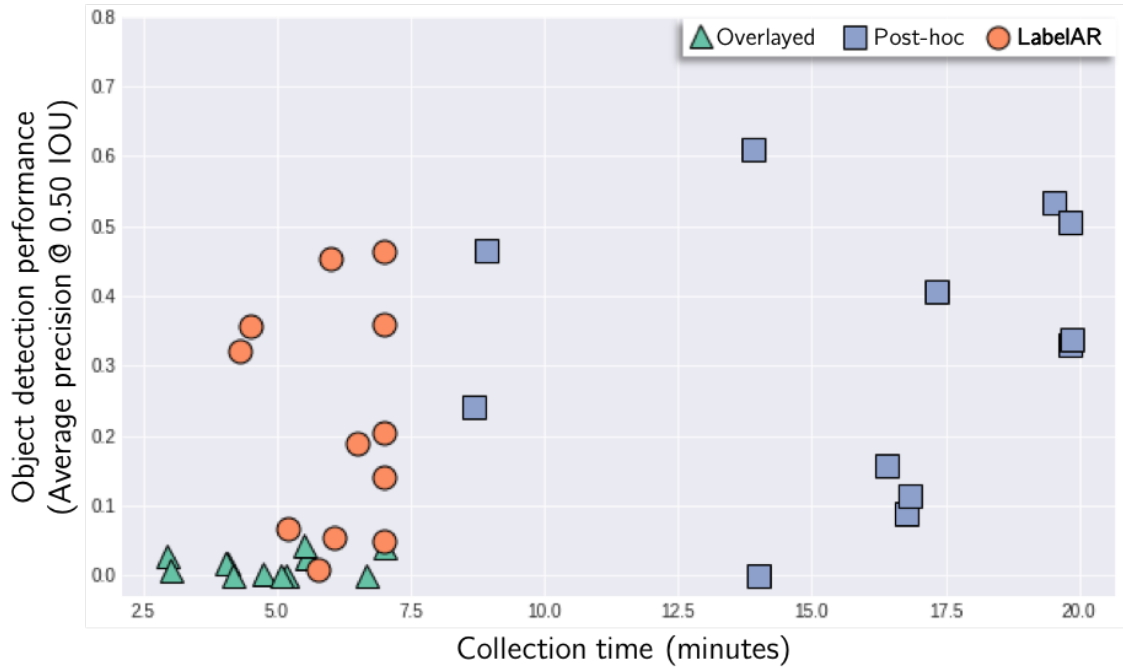


**Figure 9:** For each dot in the figure, a user collected and labeled a set of training images using post-hoc annotation, overlaid 2D bounding box interface, or LabelAR that were then used to train an object detection model. The resulting model performance (y-axis) was obtained by evaluating each trained model on a hold-out test set of images of the respective object sets.

### Bounding Box Accuracy of Collected Images

We compared the accuracy of bounding boxes on objects in collected images produced by participants against a meticulously labeled “gold-standard” set for randomly chosen participant-collected images in terms of intersection-over-union (IOU). 100 images for each condition were sampled at random, and a single annotation per image was hand annotated and then analyzed against the annotation produced during the study. The prior section used IOU of *predicted* versus gold-standard boxes on holdout *test* images, while this analysis focuses on *collected bounding boxes* versus gold-standard boxes on participant-collected *training images*.

## Object detection performance (AP@0.50 IOU) vs. collection time



**Figure 10:** While post-hoc annotation precision holds up the best to increasing the minimum bounding box accuracy of the AP metric to 0.5 intersection-over-union (IOU), the superior precision/time tradeoff of LabelAR remains.

Mean IOU for the post-hoc interface ( $\mu = 0.89$ ) was significantly higher than for LabelAR ( $\mu = 0.50$ ),  $t(214) = 18.54, p < 0.001$ . The IOU for LabelAR was slightly higher than for the 2D overlay condition ( $\mu = 0.45$ ), but the difference is statistically significant  $t(214) = 2.29, p < 0.05$ . We show some example images for each condition in Figure 11.

### Number of Images Taken

The results we obtained are dependent on the number and variety of images participants take in different conditions. While LabelAR guides users, the other two conditions do not. We investigate how many images participants took organically across three conditions here, then investigate how sensitive LabelAR performance is to the number of images and angles captured in the next section.

<i>Interface</i>	<i>Collect time (min)</i>	<b>Object detection performance</b>	
		<i>mAP .25IOU</i>	<i>mAP .5IOU</i>
Post-hoc	15.97	0.40	0.32
Overlaid	4.82	0.10	0.02
LabelAR	6.11	0.58	0.22

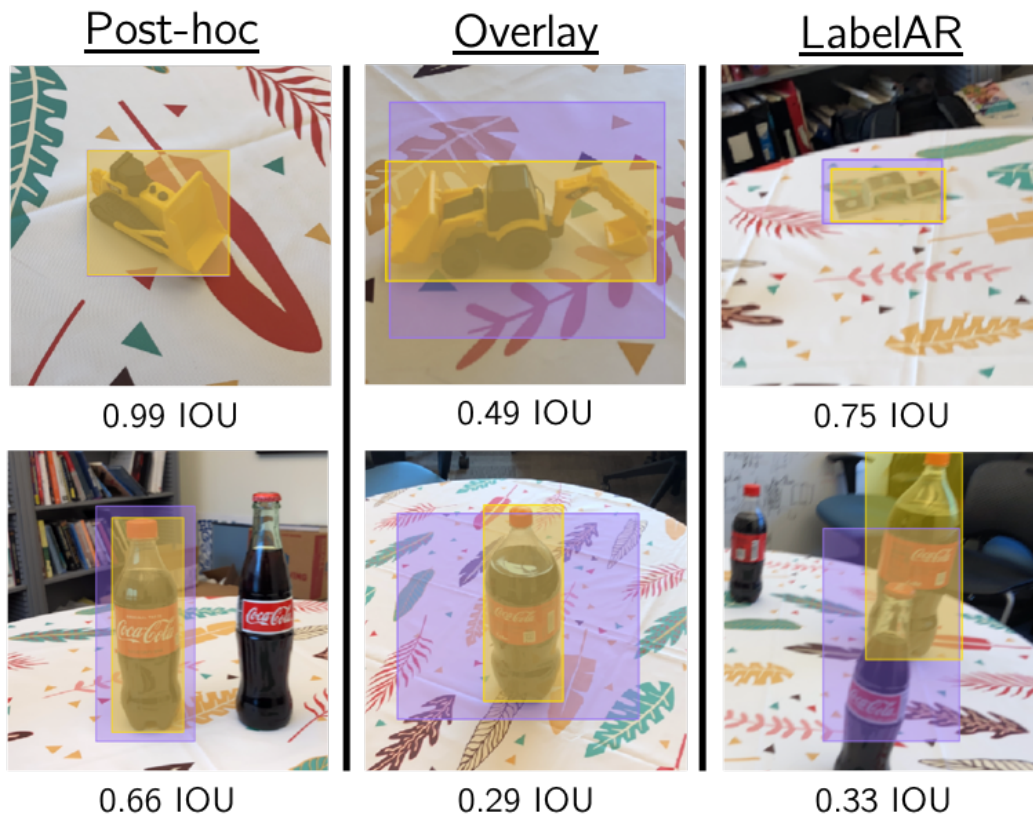
Table .1: Performance is measured in terms of mean-average-precision (mAP) which is the mean across participants’ resulting average precisions of the detectors trained on their respective data. This is computed at two different minimum bounding box accuracy thresholds: 0.25 and 0.5 intersection-over-union (IOU). The *Time* column shows the mean time across participants in minutes taken to capture and label the training images.

<i>Interface</i>	<b>Bounding box accuracy</b>
	<i>Intersection-over-union (IOU)</i>
Post-hoc	0.89
Overlaid	0.45
LabelAR	0.50

Table .2: Bounding box accuracy is measured against a meticulously labeled “gold-standard” set of randomly chosen participant-collected images in terms of intersection-over-union (IOU). IOU is the area of intersection of two boxes (participant-generated vs. gold-standard) divided by the area of their union.



## Examples of participant labels vs. 'gold-standard'

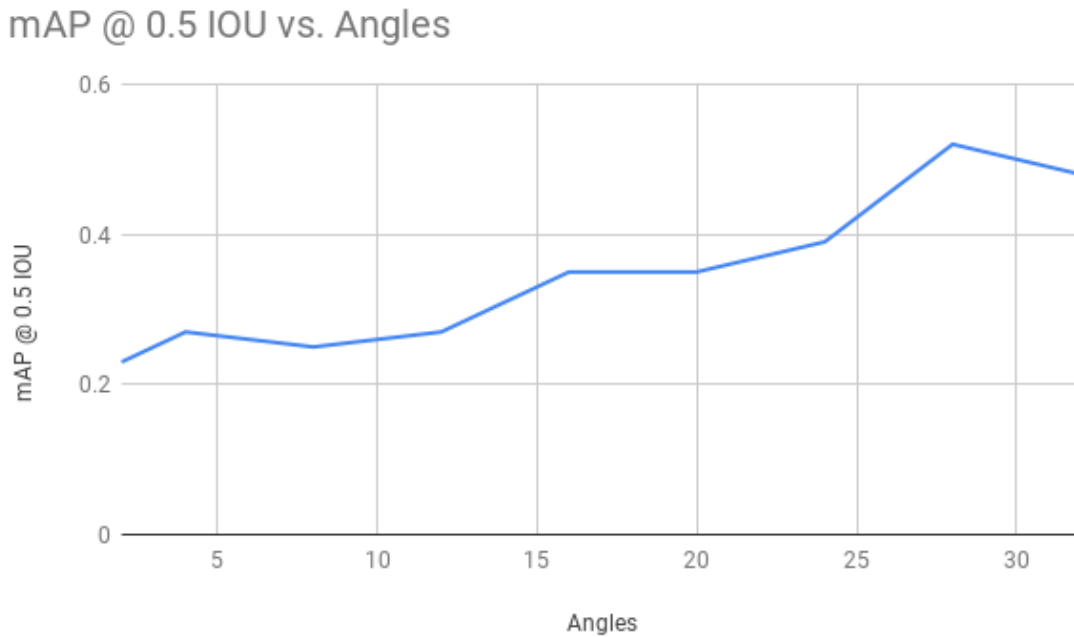


**Figure 11:** To measure the intersection-over-union (IOU) accuracy of the 2D bounding box labels produced by participants in the study (purple boxes), we had a researcher create meticulous 'gold-standard' labels to compare against (gold boxes). The best and worst IOU examples for each technique are shown.

On average, users took 29.8 ( $\sigma = 14.8$ ) images using the traditional camera app and annotated anywhere from 1-5 objects per image in the post-hoc annotation tool. Users took an average of 37.0 ( $\sigma = 24.8$ ) images using the overlaid interface, with at most 1 annotation per image. Users took exactly 24 images in LabelAR, as that is the number of images suggested by our guided interface, each image having 5 annotations.

## Angle Ablation for LabelAR

How many different angles should LabelAR ask a user to collect? To drive design decisions of the LabelAR interface, we performed an angle ablation on the coke bottles object set, starting with 32 angles and decreasing to 2 by increments of 4 (and by 2 on the last one). Figure 12 shows a linearly increasing relationship between the number of angles and detection performance, up to 32 equally space horizontal angles. That is, there is no “knee” in the curve at which point adding angles has a diminishing return on investment in terms of object detection performance. For this evaluation set, the more angles we capture of objects in our training data the better the performance, so a trade-off must be made in terms of desired collection time. The slight dips along the linear trend, including the one at the end are likely due to the pattern at which angles were dropped out during the ablation. We would expect an more linear relationship if every number of angles were perfectly spaced apart.



**Figure 12:** Object detection performance in terms of average precision (AP) steadily increases as images at various angles are added to the training set.

## 6.2 Qualitative Results

In a post-study survey, we elicited qualitative responses by participants on their experience using LabelAR as well as the comparison interfaces.

Ten of twelve participants stated they would prefer to use LabelAR over the other interfaces. One found the overlaid interface easier to use, and one stated that their answer depended on LabelAR producing better results (our analysis was performed offline after the study so participants could not see the performance of models trained on their images).

Participants appreciated that they were able to place spatially stable boxes around real-world objects (four of twelve mentioned this explicitly). One commented that it was particularly useful to track multiple objects simultaneously. Suggestions for improvement largely centered around the efficiency of the widgets for initial box placement and sizing, which four participants felt could be improved. These difficulties did not prevent users from completing their tasks: *“I was able to get to a high degree of precision (regarding the boxes being drawn around the objects)”*. These difficulties could be overcome with an additional round of refining the interaction design of the manipulation widgets. Also, several participants suggested dynamically changing the coloring of interface widgets based on the underlying image pixels to ensure legibility in a variety of settings.

Six of twelve participants commented positively how the capture markers provided a very engaging way to capture a variety of viewpoints: *“I really liked that it felt like a game. I loved turning the red vantage point bars green”*; *“I also liked how the red rods guided you on where to point the camera”*; *“The angle targets were fun to use”*. However, some orientations were harder to capture than others for two participants: *“I did think it was difficult to capture the rods/ handles that were nearly perpendicular to the object”*; *“In certain capturing angles (only at the top) it was sometimes difficult to line up two circles”*.

Finally, one participant noted they experienced temporary issues with the underlying tracking technology (ARKit), which in some instances lost the table plane and subsequently recomputed it several inches above or below the prior plane. As we discuss in the Limitations section, LabelAR is fundamentally dependent on the accuracy of the underlying AR tracking.

## 7 Discussion

**Object detection performance.** We have shown LabelAR can be over 4× more precise on average than overlaid bounding box tools, and comparable or slightly better in performance to post-hoc annotation tools. This suggests that AR-based image collection tools can have a significant impact on training CV models. For this result, average precision was assessed at an IOU of 0.25, which is

an acceptable evaluation criteria depending on the application. For example, in a hypothetical key-finder app, a detector does not need highly accurate boxes to indicate to a user where the keys are located. Conversely, in robotic-grasping applications, highly accurate IOU is very important since the robot needs to know exactly where the extents of the objects are to place a grabber. None of the collection apps resulted in desirable AP's at an IOU of 0.5, the best being post-hoc annotation ( $AP_{0.5IOU} = 0.32$ ). In other words, 68% or more of the predictions made by any of the apps at this IOU-level would be false-positives. This suggests that more training images or more efficient learning algorithms are needed to have quickly-created object detectors perform well enough to create actual value for end-users in high IOU applications. Future work that can improve the bounding box accuracy of LabelAR would be highly valued.

**Collection time.** The collection time results from the user study show that, on average, LabelAR is over  $2\times$  faster than post-hoc annotation and competitive with the overlaid interface. The significantly larger time cost with post-hoc annotation is due to the need to label every image individually by drawing 2D boxes after the capturing process. With LabelAR, there is a relatively small upfront time cost of placing one 3D box for each object. The time it takes to move the camera around the objects and capture the angle-markers does not depend on the number of objects and would stay constant if the number of objects were increased. The overlaid interface is the fastest method since there are no annotation tasks required by the user. However, since the user has no control over the size or ratio of the 2D box, a lot of bounding box accuracy is sacrificed for speed. It is worth noting that the time advantage of the overlaid interface over LabelAR would potentially disappear as more objects were added. The time needed to fully annotate an additional object with the overlaid interface scales with the number of images desired, whereas the time needed to annotate an additional object with LabelAR is the fixed up-front cost.

**2D bounding box accuracy.** LabelAR performed similarly to the overlaid interface in terms of box accuracy, but post-hoc annotation had the best accuracy over all, due to the fine-level of control the user can exercise in the Scalabel interface to place and fit nearly pixel-perfect boxes around objects in each image. Some causes of IOU degradation include poor box placement by the user (affects all three apps), fixed box ratio (overlay interface only), and projection error from 3D to 2D boxes due to spatial tracking errors (LabelAR only). Underlying AR spatial tracking errors might have significantly affected at least one of the collections in our study, where the 2D box results are all shifted down by about 40 pixels relative to the object positions. We are unable to confidently diagnose whether this was spatial tracking, user-related errors or both, so we included this data in the final results.

**Diverse image perspectives.** While the results show that the post-hoc annotation tool produces superior IOU numbers, the average precisions results show that LabelAR performs comparably well. We believe this is due to the diversity in image perspectives produced by LabelAR’s guided capture interface. While in theory it is possible to produce a similarly diverse set of data using post-hoc annotation, we believe that the results of the user study show that people’s natural instinct on how many distinct views are needed to train a cv model are significantly lower than reality, and that AR guided interfaces should be considered a good way to ensure that users can produce quality training datasets.

**Angle ablation.** The angle ablation results inform the design choice of maximizing the number of angle-markers within the constraints of usability and time (too many angle markers might be overwhelming in terms of user experience, or take too long to capture all of them.) Future work could try to establish an upper bound on performance gains from number of angles and find ways to collect more angles from the user without increasing time or decreasing usability.

## 8 Limitations

Some limitations of the interface we present are inherent to AR devices, while others result from assumptions that we make, such as static objects.

**Spatial tracking reliance** The quality of the training data produced by LabelAR is influenced heavily by the AR device’s ability to maintain spatial tracking of the environment. All AR devices we’ve tested our interface on were subject to some amount of drift, which manifests in holocubes no longer being physically on top of the real-world objects. This can cause problems leading to a decreased IOU.

**Static objects assumption.** An obvious limitation arises from our assumption of static objects, that is the objects must remain in-place or else the 3D bounding volumes do not track if those objects are displaced in the scene. In other words, if a wearer were to pick up and move an object of interest, the video annotation capabilities will be lost. We see this as a major limitation for machine learning research since humans tend to pick things up and manipulate them for closer examination when confronted with a novel object (especially toddlers, who are constantly engaging in interactive visual learning).

Also, our approach only works for objects of small enough size where it is easy for a user to capture different viewpoints efficiently — for example, one couldn’t efficiently get a lot of different viewpoints of a large building.

**Cuboid bounding volumes.** Although we chose the cuboid as our bounding volume shape for easy scaling and fitting, it does not always allow for a perfect fit. In particular, there is significant space around spherical and cylindrical objects that, when projected into image space, can contribute to inaccuracies when calculating IOU.

## 9 Future Work

In addition to addressing the limitations already discussed, there are a few particularly interesting research directions we would like to pursue:

**Real-time iterative model training.** We are excited about possible research directions involving interactive adaptation for object recognition. LabelAR’s ability to collect high quality training data in a short period unlocks a promising research direction to explore user interfaces and computer vision methods for iterative in-situ re-training that leverages user interaction. In this work, we have explored interactive collection followed by a single re-training step. In future work, we’d like to explore what the user can do with the training results and how additional in-situ re-training could benefit both the user and model.

**Crowd-sourcing with LabelAR** We think that LabelAR is a first step in creating a crowd-sourced image database along the lines of ImageNet [10] where collections of objects annotated from a large variety of angles can be uploaded and shared. There are many significant engineering challenges in building such a system, and interface improvements would need to be built on top of LabelAR to support a hierarchical labeling structure to the data.

**Extensions to robotic-platform-cameras.** Our data collection interfaces need not stay limited to hand-held or head-mounted devices. Conceivably, a wearer of a VR headset could control a robotic video platform such as a drone to overcome accessibility constraints. For example, an engineer could use our system to quickly train a computer vision model to recognize particular types of cracks or visible damage on buildings or bridges that would be difficult to collect otherwise.

## 10 Conclusion

We introduce LabelAR, an augmented reality interface for collecting computer vision model training data. LabelAR utilizes the spatial tracking technology in AR devices to localize bounding volumes over physical objects in the environment, and can use these volumes to automatically label 2D bounding boxes for these objects. It also provides a guided interface to assist users in collecting a variety of training data.

In a user study, we show that LabelAR is able to collect training data significantly faster than baseline post-hoc annotation tools, such as Scalabel, while producing comparable quality or better output. We also compared LabelAR against an overlaid 2D bounding box interface, a tool designed to collect training data very quickly, to which LabelAR was able to produce significantly better results. In short, LabelAR combines the speed of a tool like the overlaid interface with the output quality of post-hoc annotation tools.

We believe our work opens up many avenues for future joint computer vision and HCI research projects with real-time iterative model training feedback or interesting crowdsourcing opportunities. We also believe our tool can serve as a new benchmark in the computer vision community as a tool for collecting training data for instance recognition problems.

# Bibliography

1. D. Acuna, H. Ling, A. Kar, and S. Fidler. “Efficient Interactive Annotation of Segmentation Datasets with Polygon-RNN++”. In: *CoRR* abs/1803.09693, 2018. arXiv: 1803.09693. URL: <http://arxiv.org/abs/1803.09693>.
2. S. Amershi, J. Fogarty, A. Kapoor, and D. Tan. “Overview-Based Example Selection in End-User Interactive Concept Learning”. In: ACM Press, 2009, pp. 247–256. ISBN: 978-1-60558-745-5. URL: <https://doi.org/10.1145/1622176.1622222>.
3. M. Andriluka, J. R. R. Uijlings, and V. Ferrari. “Fluid Annotation: a human-machine collaboration interface for full image annotation”. In: *CoRR* abs/1806.07527, 2018. arXiv: 1806.07527. URL: <http://arxiv.org/abs/1806.07527>.
4. F. Argelaguet Sanz and C. Andujar. “A Survey of 3D Object Selection Techniques for Virtual Environments”. In: *Computers and Graphics* 37:3, 2013, pp. 121–136. DOI: 10.1016/j.cag.2012.12.003. URL: <https://hal.archives-ouvertes.fr/hal-00907787>.
5. S. Bambach, D.J. Crandall, L. B. Smith, and C. Yu. “Toddler-Inspired Visual Object Learning”. In: *NeurIPS*. 2018.
6. L.-C. Chen, S. Fidler, and R. Urtasun. “Beat the MTurkers: Automatic Image Labeling from Weak 3D Supervision”. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3198–3205.
7. D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray. “Scaling Egocentric Vision: The EPIC-KITCHENS Dataset”. In: *CoRR* abs/1804.02748, 2018. arXiv: 1804.02748. URL: <http://arxiv.org/abs/1804.02748>.
8. A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. “MonoSLAM: Real-time single camera SLAM”. In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 6, 2007, pp. 1052–1067.
9. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. “ImageNet: A Large-Scale Hierarchical Image Database”. In: *CVPR09*. 2009.
10. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. “ImageNet: A Large-Scale Hierarchical Image Database”. In: *CVPR09*. 2009.
11. G. Evans, J. Miller, M. I. Pena, A. MacAllister, and E. Winer. *Evaluating the Microsoft HoloLens through an augmented reality assembly application*. 2017. DOI: 10.1117/12.2262626. URL: <https://doi.org/10.1117/12.2262626>.



12. M. Everingham, L. Gool, C.K. Williams, J. Winn, and A. Zisserman. “The Pascal Visual Object Classes (VOC) Challenge”. In: *Int. J. Comput. Vision* 88:2, 2010, pp. 303–338. ISSN: 0920-5691. DOI: 10.1007/s11263-009-0275-4. URL: <http://dx.doi.org/10.1007/s11263-009-0275-4>.
13. J. Fails and D. Olsen. “A Design Tool for Camera-based Interaction”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’03. ACM, Ft. Lauderdale, Florida, USA, 2003, pp. 449–456. ISBN: 1-58113-630-7. DOI: 10.1145/642611.642690. URL: <http://doi.acm.org/10.1145/642611.642690>.
14. G. W. Fitzmaurice. “Situated Information Spaces and Spatially Aware Palmtop Computers”. In: *Commun. ACM* 36:7, 1993, pp. 39–49. ISSN: 0001-0782. DOI: 10.1145/159544.159566. URL: <http://doi.acm.org/10.1145/159544.159566>.
15. R. Girshick. “Fast R-CNN”. In: *International Conference on Computer Vision (ICCV)*. 2015.
16. D. Göhrling, J. Hoffman, E. Rodner, K. Saenko, and T. Darrell. “Interactive adaptation of real-time object detectors”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 1282–1289.
17. K. Hinckley, R. Pausch, J. C. Goble, and N. F. Kassell. “A Survey of Design Issues in Spatial Input”. In: *Proceedings of the 7th Annual ACM Symposium on User Interface Software and Technology*. UIST ’94. ACM, Marina del Rey, California, USA, 1994, pp. 213–222. ISBN: 0-89791-657-3. DOI: 10.1145/192426.192501. URL: <http://doi.acm.org/10.1145/192426.192501>.
18. K. Huo, Vinayak, and K. Ramani. “Window-Shaping: 3D Design Ideation by Creating on, Borrowing from, and Looking at the Physical World”. In: *Proceedings of the Eleventh International Conference on Tangible, Embedded, and Embodied Interaction*. TEI ’17. ACM, Yokohama, Japan, 2017, pp. 37–45. ISBN: 978-1-4503-4676-4. DOI: 10.1145/3024969.3024995. URL: <http://doi.acm.org/10.1145/3024969.3024995>.
19. P. Inc. *Vuforia Engine*. <https://developer.vuforia.com/>. 2011–2018.
20. A. Kovashka, O. Russakovsky, L. Fei-Fei, and K. Grauman. “Crowdsourcing in Computer Vision”. In: *CoRR* abs/1611.02145, 2016. arXiv: 1611.02145. URL: <http://arxiv.org/abs/1611.02145>.
21. M. Lau, M. Hirose, A. Ohgawara, J. Mitani, and T. Igarashi. “Situated Modeling: A Shape-stamping Interface with Tangible Primitives”. In: *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction*. TEI ’12. ACM, Kingston, Ontario, Canada, 2012, pp. 275–282. ISBN: 978-1-4503-1174-8. DOI: 10.1145/2148131.2148190. URL: <http://doi.acm.org/10.1145/2148131.2148190>.
22. J. Leitner, A. Förster, and J. Schmidhuber. “Improving robot vision models for object detection through interaction”. In: *2014 International Joint Conference on Neural Networks (IJCNN)*, 2014, pp. 3355–3362.
23. T. Lin, M. Maire, S.J. Belongie, L.D. Bourdev, R.B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. “Microsoft COCO: Common Objects in Context”. In: *CoRR* abs/1405.0312, 2014. arXiv: 1405.0312. URL: <http://arxiv.org/abs/1405.0312>.

24. D. Maynes-Aminzade, T. Winograd, and T. Igarashi. "Eyepatch: Prototyping Camera-based Interaction Through Examples". In: *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*. UIST '07. ACM, Newport, Rhode Island, USA, 2007, pp. 33–42. ISBN: 978-1-59593-679-0. DOI: 10.1145/1294211.1294219. URL: <http://doi.acm.org/10.1145/1294211.1294219>.
25. D. Pathak, Y. Shentu, D. Chen, P. Agrawal, T. Darrell, S. Levine, and J. Malik. "Learning Instance Segmentation by Interaction". In: *CVPR Workshop on Benchmarks for Deep Learning in Robotic Vision*. 2018.
26. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115:3, 2015, pp. 211–252. DOI: 10.1007/s11263-015-0816-y.
27. B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. "LabelMe: a database and web-based tool for image annotation". In: *International journal of computer vision* 77:1-3, 2008, pp. 157–173.
28. P. Sermanet, C. Lynch, J. Hsu, and S. Levine. "Time-Contrastive Networks: Self-Supervised Learning from Multi-View Observation". In: *CoRR abs/1704.06888*, 2017. arXiv: 1704.06888. URL: <http://arxiv.org/abs/1704.06888>.
29. I. E. Sutherland. "A Head-mounted Three Dimensional Display". In: *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I. AFIPS '68 (Fall, part I)*. ACM, San Francisco, California, 1968, pp. 757–764. DOI: 10.1145/1476589.1476686. URL: <http://doi.acm.org/10.1145/1476589.1476686>.
30. M. Tsang, G. W. Fitzmaurice, G. Kurtenbach, A. Khan, and B. Buxton. "Boom Chameleon: Simultaneous Capture of 3D Viewpoint, Voice and Gesture Annotations on a Spatially-aware Display". In: *Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology*. UIST '02. ACM, Paris, France, 2002, pp. 111–120. ISBN: 1-58113-488-6. DOI: 10.1145/571985.572001. URL: <http://doi.acm.org/10.1145/571985.572001>.
31. H. F. Tung, A. W. Harley, L. Huang, and K. Fragkiadaki. "Reward Learning from Narrated Demonstrations". In: *CoRR abs/1804.10692*, 2018. arXiv: 1804.10692. URL: <http://arxiv.org/abs/1804.10692>.
32. K. Welke, J. Issac, D. Schiebener, T. Asfour, and R. Dillmann. "Autonomous acquisition of visual multi-view object representations for object recognition on a humanoid robot". In: *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 2012–2019.
33. J. Xie, M. Kiefel, M. Sun, and A. Geiger. "Semantic Instance Annotation of Street Scenes by 3D to 2D Label Transfer". In: *CoRR abs/1511.03240*, 2015. arXiv: 1511.03240. URL: <http://arxiv.org/abs/1511.03240>.

34. K.-P. Yee. "Peephole Displays: Pen Interaction on Spatially Aware Handheld Computers". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '03. ACM, Ft. Lauderdale, Florida, USA, 2003, pp. 1–8. ISBN: 1-58113-630-7. DOI: 10.1145/642611.642613. URL: <http://doi.acm.org/10.1145/642611.642613>.
35. T. Yeh and T. Darrell. "Doubleshot: An Interactive User-aided Segmentation Tool". In: *Proceedings of the 10th International Conference on Intelligent User Interfaces*. IUI '05. ACM, San Diego, California, USA, 2005, pp. 287–289. ISBN: 1-58113-894-6. DOI: 10.1145/1040830.1040901. URL: <http://doi.acm.org/10.1145/1040830.1040901>.
36. F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell. "BDD100K: A Diverse Driving Video Database with Scalable Annotation Tooling". In: *CoRR abs/1805.04687*, 2018. arXiv: 1805.04687. URL: <http://arxiv.org/abs/1805.04687>.
37. A. Zeng, K.-T. Yu, S. Song, D. Suo, E. Walker, A. Rodriguez, and J. Xiao. "Multi-view self-supervised deep learning for 6D pose estimation in the Amazon Picking Challenge". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 1386–1383.