Object Detection in the Context of Mobile Augmented Reality

Xiang Li*   Yuan Tian†   Fuyao Zhang‡   Shuxue Quan§   Yi Xu¶
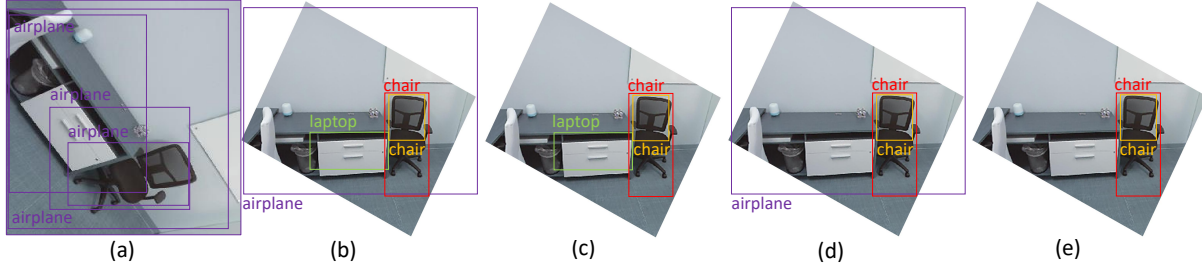
OPPO US Research Center

Figure 1: These images show the results of different object detection methods applied on the same image: (a) results of original DNN (SSD Mobilenet model), (b) DNN with image orientation correction, (c) DNN with image correction and scale-based filtering, (d) DNN with image correction and online semantic mapping, and (e) results of our proposed method.

ABSTRACT

In the past few years, numerous Deep Neural Network (DNN) models and frameworks have been developed to tackle the problem of real-time object detection from RGB images. Ordinary object detection approaches process information from the images only, and they are oblivious to the camera pose with regard to the environment and the scale of the environment. On the other hand, mobile Augmented Reality (AR) frameworks can continuously track a camera's pose within the scene and can estimate the correct scale of the environment by using Visual-Inertial Odometry (VIO). In this paper, we propose a novel approach that combines the geometric information from VIO with semantic information from object detectors to improve the performance of object detection on mobile devices. Our approach includes three components: (1) an image orientation correction method, (2) a scale-based filtering approach, and (3) an online semantic map. Each component takes advantage of the different characteristics of the VIO-based AR framework. We implemented the AR-enhanced features using ARCore and the SSD Mobilenet model [26] on Android phones. To validate our approach, we manually labeled objects in image sequences taken from 12 room-scale AR sessions. The results show that our approach can improve on the accuracy of generic object detectors by 12% on our dataset.

Index Terms: Computer Graphics—Graphics systems and interfaces—Mixed / Augmented Reality; Artificial intelligence—Computer vision—Computer vision problems—Object detection;

1 INTRODUCTION

Object detection plays an important role in Augmented Reality (AR), where virtual content can be associated with real-world objects to enable applications such as product recommendations for e-commerce,

---

*e-mail: xiang.li@oppo.com

†e-mail: yuan.tian@oppo.com

‡email: fuyao.zhang@oppo.com

§email: shuxue.quan@oppo.com

¶e-mail: yi.xu@oppo.com

and textual and graphical informational displays for educational purposes. In addition, the semantic information obtained by an object detector can be used to help VIO. Recently, tremendous progress has been made on object detection using Deep Neural Networks (DNNs) including SSD [26], YOLO [30], faster R-CNN [32], etc. With advancements in mobile computing power and lightweight DNN frameworks designed for mobile platforms (e.g., TensorFlow Lite), more and more object detection DNNs can run on smartphones in real-time [16, 43]. However, most of these general-purpose object detectors are not designed specifically for AR applications. First, most of them are trained with the objects oriented upright in the images, which presents challenges when the camera is rotated as seen in Fig. 1(a). This type of camera rotation frequently occurs when a user views virtual content during an AR session. For example, if a virtual object is large and wide, the user has a tendency to rotate the phone from a portrait to landscape orientation. Second, object detectors are oblivious to scale information due to the scale ambiguity of a single input image. Third, during an AR session, an object may be viewed from many different viewpoints. Since most object detectors do not memorize specific locations that the device has visited before, this can potentially result in inconsistent detection over a large range of viewpoints.

Although data augmentation and data synthesis can be used to ease the above-mentioned problems of scale ambiguity and viewpoint variance, it requires more training data and time. Another approach is to use the additional geometry information provided by a depth camera [42]. However, depth cameras on mobile devices are not widely available. Video-based methods can also alleviate the viewpoint variance problem (e.g. [12, 24]), but they are either limited to temporally-close frames or rely on intricate network designs. Our key insight is that we leverage the information generated by an AR framework, especially by a VIO method, to solve the aforementioned problems and improve the performance of object detectors in the context of mobile AR applications.

Mobile AR frameworks have become widely available for application developers, including Apple Inc.s ARKit and Google Inc.s ARCore. These AR frameworks use VIO to track a six degrees of freedom (6DoF) camera pose continuously during an AR session. These frameworks also provide sparse 3D points with real-world scales. In addition, AR frameworks have the ability to memorize places by tracking the camera within a map. Even when tracking is lost, re-localization can be used to re-estimate the camera pose

with respect to the map. Intermittent re-localization is typically performed to detect revisited places in order to mitigate the drift accumulated in the camera trajectory over time.

With recent advancements in mobile computing power, both AR frameworks and object detection DNNs can run simultaneously on a mobile device using a single camera as the input source. In this paper, a new object detection pipeline is proposed in the context of mobile AR. This pipeline takes advantage of the output and features of AR frameworks to increase the overall object detection accuracy of DNNs. Specifically, we use camera pose information from VIO to correct image orientation, use scale information to reject false positives, and build an online semantic map to fuse detection results from various viewpoints and distances on AR-capable mobile devices. We show that our approach can improve the accuracy of off-the-shelf object detection DNN models.

2 RELATED WORK

In this section, we discuss related work on object detection, including object detection methods that handle scale and viewpoint variance, as well as SLAM methods that utilize semantic information. Finally, we show examples of how VIO output can benefit other computer vision algorithms.

2.1 Object Detection

Generic object detection is one of the most fundamental computer vision problems. It provides a semantic understanding of the real world and is related to many applications. DNN-based object detection can be separated into two main types. The first type is region proposal followed by classification, such as Fast R-CNN [10], Faster R-CNN [32], R-FCN [7], FPN [22] and Mask R-CNN [13]. The second type uses DNNs as direct regressors to produce the final categories and locations simultaneously. Examples of this second method include MultiBox [8], YOLO [30], SSD [26], YOLOv2 [31], DSSD [9], etc. R-CNN [11] was the first work that applied CNN-based feature extraction in the traditional object detection pipeline. Later, it was improved by adding spatial pyramid matching [14], by introducing multi-task loss on classification and bounding box regression [10], by adding a Region Proposal Network (RPN) [32], and by adapting to a fully-convolutional architecture [7]. Mask R-CNN [13] adds an additional branch to predict the segmentation mask to enable instance segmentation. Many researchers have also investigated end-to-end frameworks for object detection. Redmon et al. proposed YOLO (you only look once) [30] which uses the same feature map to predict both the bounding boxes and the confidence scores for categories. Another important work is SSD [26], which uses anchor boxes with different aspect ratios and scales instead of fixed grids. YOLO and SSD are two very popular frameworks and have continued to evolve in recent years [9, 21, 31, 43]. Recently, there has been research on 6D detection from RGB images [37, 39, 44]. This method can predict the pose of 3D objects from images but requires a lot of training data to do so. On mobile devices, Ahmadyan et al. [1] proposed a system that tracks the pose of an object in real-time for AR applications. Similar to early attempts at template-based object detection in AR [40] and map-based relocalization [25], this 6D detection approach relies on a known category and/or geometry. Based on the tracking results, AR effects can be superimposed onto certain objects. Our work differs from previous methods in that we combine generic object detection with SLAM, so no prior knowledge of the scene or object is required.

2.2 Scale Invariance and Consistency

Although current object detection methods can achieve promising results on public datasets, there are still many challenges such as scale invariance, detection under large viewpoint changes, and detection consistency over time. To achieve scale invariance, image pyramids are used to extract multi-scale features [14]. However, this requires a significant amount of additional training time and higher memory consumption. As a result, some DNNs only use pyramids in the testing phase [10, 32]. Feature hierarchy inside the DNNs has also been investigated for scale invariance [26]. Multi-scale CNNs [5] generate multiple output layers so that receptive fields match objects of different scales. Similarly, TridentNet [20] generates scale-specific feature maps with a uniform representational power. HyperNet [18] aggregates feature maps from different resolutions into a uniform space. FPN [22] proposes a bottom-up and top-down architecture for the feature hierarchy. YOLO v2 [31] and SNIPER [35] also introduce multi-scale training. To improve object detection, researchers have also utilized multi-task learning [3, 6] to boost object detection. Contextual information is also introduced to improve object detection performance using Markov Random Field [48] and LSTMs [4]. Our work uses information from VIO to improve accuracy using off-the-shelf models without any additional training.

2.3 Semantic SLAM

With the development of deep learning, using semantic information for SLAM has drawn a great deal of attention. Many researchers [27, 36, 46] have proposed methods to combine semantic segmentation and SLAM together, where the semantic labels of objects are fused into the SLAM map to create a semantic map. MaskFusion [33] proposed segmenting different objects in the scene while tracking and reconstructing them even as they move independently from the camera. SLAM++ [34] reconstructed the object-level environment with a 3D object database based on RGBD SLAM in real-time. Bowman et al. [2] built an object-level map and also utilized object association tightly coupled with SLAM to improve accuracy. Mu et al. [28] treated all object landmarks with discrete distribution to be coupled with SLAM. This was later improved by Zhang et al. [47] by using a hierarchical Dirichlet process for data association. To the best of our knowledge, our work is the first to use VIO information to improve DNN-based object detection and provide a semantic map in real-time on AR capable mobile devices.

2.4 Improving Algorithms using VIO

AR capable mobile devices are often equipped with a camera and Inertial Measurement Unit (IMU) and use VIO to estimate camera pose. The additional sensor data and output of VIO can help solve challenging computer vision problems. Kurz et. al. [19] proposed gravity-aligned local feature descriptors for image classification. The idea is similar to our image orientation correction described in Sect. 3.1. For 3D reconstruction, Chisel [17] utilized RGBD data and camera pose information from VIO to reconstruct indoor scenes using an implicit Truncated Signed Distance Function data structure. For collision detection, Xu et. al. [45] proposed a real-time multi-scale voxelization method using camera pose information and sparse point clouds from mobile AR frameworks. Visual occlusion can also be implemented by depth densification [15, 38] or by depth from motion [41]. Phalak et. al. [29] utilized a sequence of posed RGB images as input to a DNN to estimate the boundaries of the indoor environment. Our method takes advantage of VIO to improve object detection.

3 METHODOLOGY

Fig. 2 shows an overview of our pipeline. During a live AR session, a user moves around a scene to view virtual objects from different viewing angles while holding the phone using different orientations. The image sequence is consumed by the mobile AR framework (e.g., ARCore or ARKit), which uses VIO to track the pose of the mobile device in real-time. The pose is used to correct the image orientation so that the objects in the image appear to be in an upright orientation. The corrected images are passed to a DNN for object detection.
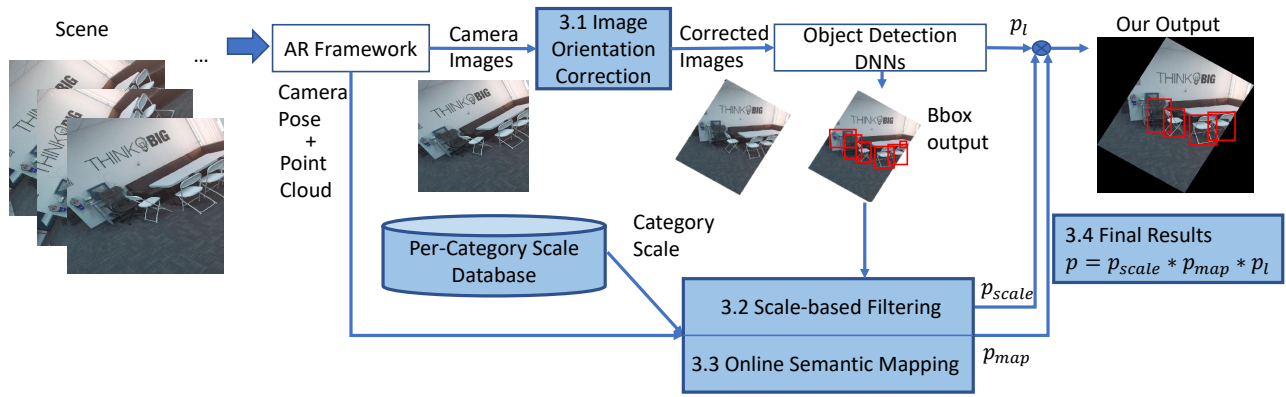
Figure 2: Overview pipeline of our approach. The different modules in our method are shown as blue boxes.

Figure 3: Most real-world objects are in an upright orientation, such as the desk, chair, and cup in this scene from the COCO dataset [23].
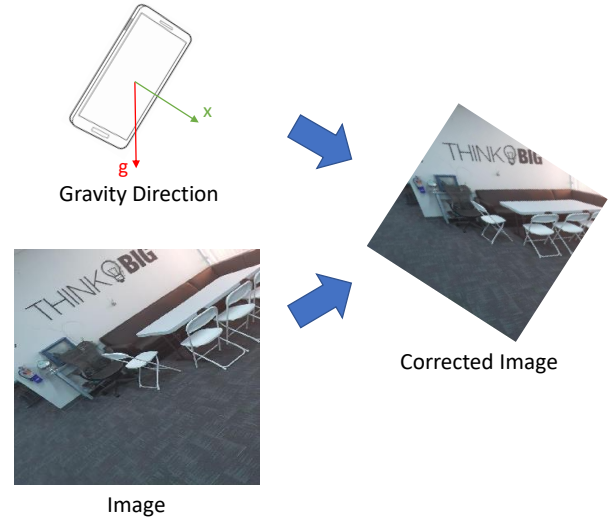

Figure 4: Image orientation correction using gravity direction.

Most DNNs used for object detection, such as YOLO, SSD, and Faster R-CNN, can output detection bounding boxes and corresponding labels. Although our method works with any of these pre-trained DNNs, we chose to focus on the efficient models that can be integrated with the AR framework on mobile devices. Moreover, since we are interested in indoor environments, we chose to use the SSD-mobilenetV1coco model from the Tensorflow detection model zoo. The model is pre-trained on the COCO dataset [23], which covers popular object categories in indoor environments.

In addition to camera pose information, sparse point cloud data is typically provided by the AR framework as well. The point clouds and camera pose are combined with object detection output from the DNN into our scale-based filtering module and semantic map module. Our method improves object detection accuracy by taking advantage of the geometric information with real-world scale. The final detection result for the current frame is obtained by adjusting the DNN output using information from the two modules.

3.1 Image Orientation Correction

If an out-of-the-box DNN-based object detection model is not trained with rotation augmentation, its detection capability will be impaired when the orientation of an object is not aligned with that of the camera. In this case, even if a detector has a certain level of built-in rotational invariance, the output bounding box of the object is still aligned with the camera axes and likely to include unwanted pixels from the background.

We take advantage of the fact that most objects in real-world scenes are placed on horizontal surfaces and are in their upright orientations, as shown in Fig. 3. One important feature of mobile devices is that the direction of gravity can be estimated by using the IMU sensor. For example, the Android platform provides the gravity sensor values that include the direction and magnitude of gravity. For an input image, it is straightforward to compute the angle to rotate the image around its center so that the x-axis of the image is perpendicular to the direction of gravity. After this correction, objects that are placed on horizontal surfaces will appear in their upright orientations. In our pipeline, different from [19], orientation correction works as a preprocessing step for the DNN model.

This orientation correction can greatly increase the detection accuracy of certain object categories such as furniture and appliances. Furthermore, the object bounding boxes can be used to estimate the real-world vertical and horizontal size of an object in 3D space. This information is used in our scale-based filtering in the next step.

3.2 Scale-based Filtering

The detection results from the object detector sometimes include false positives, many of which can be easily filtered if scale information is available (e.g., a book that is 5 m high is likely a false

Figure 5: Scale estimation. The 3D sparse points are projected onto the image shown as dots. Blue dots are points that are inside a 2D bounding box generated by the detector and the red dot is the projection of the median of all blue points.

Table 1: Example Data Entries in the Per-Category Scale Database

Category	min_w(m)	max_w(m)	min_h(m)	max_h(m)
chair	0.8	1.7	0.3	1.0
dining table	1.0	1.5	0.5	2.0
oven	0.3	1.4	0.4	1.2
refrigerator	0.5	2.0	0.5	1.4

Scale-based filtering cannot distinguish between categories that have a similar scale range. A more general approach, semantic mapping, is proposed in the next subsection to take advantage of the multi-viewpoint data of an object typically available in an AR session.

3.3 Online Semantic Mapping

There are many challenges associated with object detection such as variance in scale and viewpoint. An object detector must detect objects with different scales on the images and from different viewpoints. Moreover, the object detector might produce inconsistent detection results over time. To overcome these problems, we build a semantic map using output from both the AR framework and the object detector. A probabilistic model is then used to maintain and update object category, viewpoint, and scale information within the semantic map. Finally, we extract information from the updated 3D semantic map to adjust the object label probability from the object detector for the current frame as shown in Fig. 2. The idea is if an object can be correctly labeled by the detector most of the time over a range of different viewpoints, our approach should predict the correct label with higher confidence.

Our semantic mapping and updating process consists of three steps: (1) given the output of both the AR framework and the object detector, we create object point representations from the detection results on the current frame; (2) the new object points are fused with the superpoints that were already stored in the semantic map; and (3) the probability output from the object detector is updated. The entire algorithm is shown in algorithm 1.

positive). Fortunately, most mobile AR frameworks provide a per-frame sparse 3D point cloud in a coordinate system with a real-world scale. In this section, we propose a method to estimate the scale of the objects within the scene and filter out false positives.

In each frame, the detector predicts bounding boxes around objects. For each box, we compute two values to represent an object's real scale in 3D using the following equation:

$$D_w = w/f_x * d$$
$$D_h = h/f_y * d \qquad (1)$$

where D_w and D_h are the estimated horizontal and vertical lengths of the object, f_x and f_y are focal lengths of the camera, w and h are the width and height of the 2D detection bounding box, and d is the distance from the camera to the object. To compute d, we compute the median of the 3D sparse points produced by the AR framework whose 2D projections are within the detection bounding box (e.g., shown as blue dots in Fig. 5). Then, d is computed as the Euclidean distance between the camera position and the median.

To filter out false positives based on the estimated scale, we introduce a per-category scale database including 80 categories defined in COCO. For each category, we manually define a minimum and maximum scale for the horizontal and vertical dimensions, resulting in four values for each category. Table 1 provides a number of examples. Different object orientations may lead to varying widths of the 2D bounding box on an image (e.g., the width of a thin slab on an image will be very different from two different sides). Therefore, we use both the smallest and the largest possible values for the horizontal dimension when populating our database. Since the main goal of scale-based filtering is to reject false positives, using conservative values can ensure that true positives will be preserved during filtering.

The final step of scale-based filtering is to determine if the currently estimated scale matches the corresponding scale data from the database. Since both D_w and D_h are coarse estimations, we do not use the estimated dimension value to adjust the final detection probability. Instead, we only use these two values for filtering the false-positive results p_{scale} as follows:

$$p_{scale} = \begin{cases} 1 & \text{if } min_w \leq D_w \leq max_w \text{ and } min_h \leq D_h \leq max_h \\ 0.5 & \text{otherwise} \end{cases}$$

$$(2)$$

Algorithm 1: Online Semantic Mapping.

1: **while** *object detection is active* **do**
2:     **for** *each object detected on the current frame* **do**
3:         create one object point with the representation $(loc^{in}, l^{in}, v^{in}, s^{in})$;
4:         find all superpoints whose distance to the incoming object is within the maximum scale of object category l^{in} and insert them into a set S^{in}.
5:         **if** S^{in} *is not empty* **then**
6:             **for** *each superpoint* sp_i *in* S^{in} **do**
7:                 update the score E_l^{sp} using Equation 7
8:                 $list\_view \leftarrow v^{in}$ if $v_{diff} \geq 45°$
9:                 $list\_scale \leftarrow s^{in}$ if $s_{diff} \geq 1$
10:         **if** *no superpoints within the minimum scale distance of the incoming object point* **then**
11:             Add a new superpoint with the representation
12:             $(loc \leftarrow loc^{in}, list\_score(E_l^{in}), list\_view(v^{in}), list\_scale(s^{in}))$
13:         update the probability of current object using Equation 9

Step 1: For each frame, the object detector outputs a list of N object categories with associated bounding boxes and probabilities. For each object, we create one Object Point with the representation $(loc, label, view, scale)$, where loc is the 3D coordinates of the median of all sparse points within the bounding box (same as in the scale estimation step of Sect. 3.2), $label$ is the object label, $view$ is the view direction from the camera to loc and is a normalized unit vector, and $scale$ is the scale information that depends on the distance d from camera position to loc. The scale information s is

an integer defined as:

$$s = \lfloor log_2 d \rceil \qquad (3)$$

Step 2: The second step is to fuse the estimated object with the information already in the semantic map, which stores a set of superpoints for objects in the scene. Different from the estimated object point data structure, the object superpoints are represented as $(loc, list\_score, list\_view, list\_scale)$. The three lists encode information from all previous frames in the same AR session.

- $list\_score(E_1, E_2, E_3, \cdots, E_l, \cdots)$ is a list of the scores E_l for each label l that has been detected at this superpoint location. The higher the score, the higher probability this superpoint is of category l. The initial score values for all labels are 0.

- $list\_view(v_1, v_2, v_3, \cdots)$ is a list of historical view directions from camera positions to the superpoint when an object of any category is detected at the superpoint.

- $list\_scale(s_1, s_2, s_3, \cdots)$ is a list of historical scales when an object of any category is detected at the superpoint.

It is worth noting that different median locations (loc) from different viewpoints during an AR session might be computed for the same object because each image only shows a partial surface of an object. Moreover, the same object might be given different labels at different time instances. To fuse each incoming estimated object point (loc^{in}, l^{in}, v^{in}, s^{in}) with the superpoints in the semantic map, we first find a set of superpoints S^{in} whose loc are within a threshold distance to the incoming object point loc^{in}. We use the maximum scale of category l^{in} from the scale database as the threshold when selecting set S^{in}.

For the incoming object point of category l, we compute its score E_l^{in} as follows:

$$E_l^{in} = \frac{w_v + w_s}{2} * p_l \qquad (4)$$

where p_l is the probability of the category l generated by the DNN and weights w_v and w_s are calculated as:

$$w_v = \begin{cases} 0 & if \ v_{diff} < 45° \\ (v_{diff} - 45) \ / \ 45 & if \ 45° \leq v_{diff} \leq 90° \\ 1 & otherwise \end{cases} \qquad (5)$$

$$w_s = \begin{cases} k_s * s_{diff} & if \ s_{diff} < 1/k_s \\ 1 & otherwise \end{cases} \qquad (6)$$

where v_{diff} is the minimum absolute angular difference between v^{in} and all view directions in the $list\_view$s of all points in S^{in}. The higher the v_{diff}, the higher the weight w_v is. In this way, we value information observed from different viewing angles. We also set the weight w_v to zero when the v_{diff} is smaller than 45 degrees to only update the semantic map intermittently. w_v is capped at 1 when the v_{diff} is larger than 90 degrees.

Similarly, s_{diff} is the minimum absolute scale difference between s^{in} and all scales in the $list\_scale$s of all points in S^{in}. The higher the s_{diff}, the higher the weight w_s is. k_s is used to normalize s_{diff} based on a value range. In our method, this range is set empirically to be $[0, 5]$. Therefore k_s is set to be 0.2.

If the set S^{in} is not empty, for each superpoint $sp \in S^{in}$, we update its detection score E_l^{sp} using the information of the new detection of category l. We first check if the score E_l^{sp} exists in $list\_score$. If not,

we initialize E_l^{sp} with 0 and add it into $list\_score$. Then we update E_l^{sp} as follows:

$$E_l^{sp} = \begin{cases} E_l^{sp} + E_l^{in} & if \ maxScale \geq D > minScale \\ E_l^{sp} + E_l^{in} + 1 & if \ D \leq minScale \end{cases} \qquad (7)$$

where D is the distance between the incoming point and sp. If sp is within the minimum scale of the incoming object, we add 1 to the score of the detected label as a reward.

We also add view direction of the current detection v^{in} into $list\_view$ of sp if $v_{diff} \geq 45°$. We add scale of current detection s^{in} into $list\_scale$ of sp if $s_{diff} \geq 1$.

If there is no superpoint within the minimum distance of the incoming object point, we initialize the three lists $list\_score$, $list\_view$ and $list\_scale$ with the values of E_l^{in}, v^{in} and s^{in}. Then we add a new superpoint with these three lists into the map.

Step 3: For each incoming object point p with label l, we update its label probability p_l using the nearby superpoints stored in the semantic map. We find a second set of superpoints whose loc are within a distance (minimum scale of category l^{in}) to the incoming object point loc^{in}. We call this S^{update}. We find the maximum score E_l^{max} among all the superpoints with label l in S^{update}. We then find the maximum score $E_{\bar{l}}^{max}$ among all the superpoints with any other label in S^{update}. The probability p_{map} of the semantic map at point p is defined by a modified sigmoid function as follows:

$$p_{map} = \begin{cases} \frac{1}{1+e^{E_{\bar{l}}^{max} - E_l^{max}}} & if E_l \geq E_{\bar{l}} \\ 0.5 & if E_l < E_{\bar{l}} \end{cases} \qquad (8)$$

p_{map} is at least 0.5 to guarantee that it does not decrease the output probability p_l from the detector dramatically.

3.4 Final Detection Results

Combining scale-based filtering and adjustment using semantic map, the final probability of an object is:

$$p = p_{scale} * p_{map} * p_l \qquad (9)$$

For each frame, the final output is a list of bounding boxes, each of which has the output (l, p, $bbox$), where the label and bounding box are the same as the original output from the DNN.

4 EVALUATION

In this section, we show the evaluation of the proposed method compared with detection results from the DNN only.

4.1 Datasets

Our approach does not need extra data for re-training the DNN. Unfortunately, none of the popular image datasets for object detection provide the 3D sparse point cloud and camera pose information acquired by an AR framework. For this paper, we collected a new indoor dataset for evaluation.

Our indoor dataset includes about 2,000 images with 20 object categories and 3,384 instances in total, primarily obtained from scanning home and office environments. It also includes per-frame camera pose and sparse point cloud information from 12 room-scale AR sessions. Fig. 6 shows the number of instances in the top 10 categories of our dataset. The remaining 10 categories in our dataset have a total of 167 instances. Chairs were the most commonly scanned object, and the chair category is the largest consisting of 945 instances. Some example images are shown in Fig. 7. We use a mobile phone app developed using ARCore for data acquisition. The app collects images at about 1 fps while
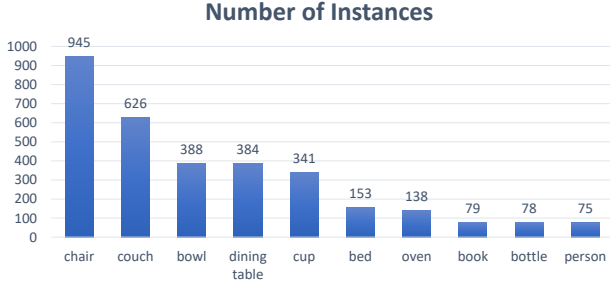
Figure 6: Number of instances for each of the top 10 categories of our dataset.

Figure 7: Examples of our testing dataset

ARCore is running continuously to track the camera pose of the device. Each session is about 1 to 2 minutes long. We encouraged the data collectors to capture objects from various viewpoints to simulate real-world AR use cases where users tend to observe virtual content from different viewing angles. The collected images are also rotation corrected by using the method introduced in Sect. 3.1 and used to compare our method with SSD (MobileNet) in an ablation study where rotation correction is removed. We labeled objects in the original and corrected images using the categories of COCO dataset [23].

4.2 Results

We tested all modules of our approach on our dataset using the SSD_mobilenet_V1_coco model from the Tensorflow detection model zoo. Fig. 1 shows the visualization of results on one uncorrected example image using different methods. The original DNN (SSD MobileNet) result in Fig. 1(a) has false positives because of the insufficient data augmentation for training. Fig. 1(b) shows that our image orientation correction process can ease this problem. Fig. 1(c) shows the result after applying scale-based filtering: the "airplane" label is filtered out as a false positive. Fig. 1(d) shows the result of online semantic mapping: the "laptop" label is also removed by the information from the semantic map. Fig. 1(e) shows the result using our proposed method, which combines all of the modules and generates a more accurate result.

To validate our online semantic mapping module, a semantic map is visualized in Fig. 8. The semantic superpoints are shown as spheres on two images from different viewpoints. The radius of a sphere reflects the minimum scale of the corresponding object category. Different colors represent different categories: white for the couch, red for the chair, and green for the dining table. As shown in Fig. 8, the superpoints maintain the correct semantic information. The visualization only shows the class label with the highest score for each superpoint, and the color saturation represents the score value. A blue circle highlights one superpoint which has a relatively low score for the current class label. It shows that using our semantic mapping, a superpoint that is in proximity to other superpoints with different labels tends to have a lower score. The semantic map is updated online, and it provides a more accurate probability for object detection.

Table 2 shows the evaluation results using metrics used by COCO. In Table 2, SSD represents the results of the original SSD MobileNet model, OC represents results using image orientation correction, SF represents results using scale-based filtering, and OSM represents results using online semantic mapping. As the results show, each of our modules has generated higher average precision compared with the original SSD MobileNet model. In particular, our entire pipeline ALL (OSM+SF+OC+SSD) has increased the average precision from 8.0% to 20.4%. Table 2 also shows that scale-based filtering and online semantic mapping improve detection results in different ways, since the average precision of our overall method

(20.4%) is significantly higher than those of SF+OC+SSD (14.6%) and OSM+OC+SSD (11.3%).

Table 3 shows that all individual modules improve the average recall, as well, although the average recall of the entire pipeline has decreased from the original model by 4%. This is because only the bounding boxes that can survive both modules will be accepted by our pipeline. This results in the missed detection of new objects or truncated objects. The definitions of superscripts of average precision and average recall metrics in both Table 2 and Table 3 are the same as those in the COCO dataset [23].

We have also tested Google MediaPipe 6D object detection [1] on our dataset. To compare our 2D detection bounding boxes with MediaPipe's 3D bounding boxes, we compute a 2D bounding box based on each output cuboid of 6D detection. We set the IOU threshold as 0.5 for evaluation purposes. Using a larger threshold could be seen as unfair to 6D detection since the 2D box always covers a larger area than the corresponding cuboid projected on the image. We tested their chair model since the chair model is one of the available models and accounts for almost one third of the instances in our dataset. Our results show that MediaPipe has a 14.4% average precision on our dataset. The results also indicate that our dataset, which is captured from real homes and offices, is very challenging.

All of the results were tested on a Pixel 3 smartphone. Our proposed method can run at ∼ 7 fps while ARCore tracks the pose of the camera in real-time. In comparison, the SSD inference alone runs at ∼ 7 fps while ARCore is running, confirming that our method is very efficient. We also expect that our method can build a semantic map for large scale environments since we only store a small number of superpoints for each detected object instead of a point cloud or voxels. The selected SSD model takes 300×300 image as input and outputs the top 10 bounding boxes with labels and properties.

5 CONCLUSION

There is a large amount of ongoing research on 2D object detection DNNs, and some of them have achieved high efficiency and great performance on mobile devices. Compared with 6D object detection [42], it is much easier to collect and label 2D object boxes. Instead
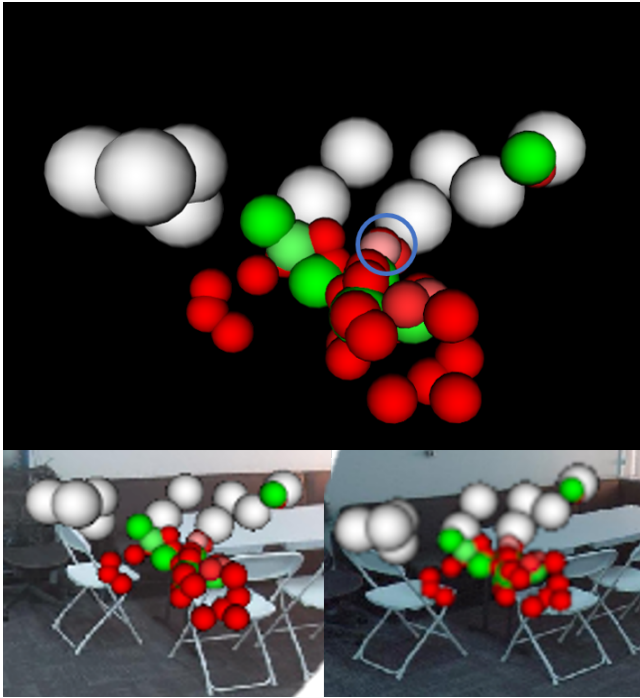
Figure 8: Visualization of a semantic map. Each sphere represents a superpoint. White is for the couch (note that there is a black couch behind the table), red is for the chairs, and green is for the dining table. The bottom two images visualize superpoints on two different frames.

Table 2: AP Metrics

Data	AP	$AP^{.5}$	$AP^{.75}$	AP^s	AP^m	AP^l
SSD	8.0	10.3	7.4	0.28	5.6	14.7
OC+SSD	10.7	13.8	10.0	0.75	11.0	17.6
SF+OC+SSD	14.6	18.9	13.5	0.60	14.6	19.1
OSM+OC+SSD	11.3	14.7	10.6	0.63	12.8	20.3
ALL	20.4	26.3	18.9	0.78	19.0	26.0

of 6D object detection, we take a hybrid approach where we utilize the 3D perception capabilities of the AR framework to improve 2D object detection. In this paper, we proposed three modules to achieve this purpose: we introduced an image orientation correction method to improve the input data to the object detector, we also introduced a per-category scale database as *a prior* knowledge to filter 2D detection results based on their estimated real-world scales, and finally we proposed an online semantic mapping approach to further improve detection accuracy. Our approach works with any object detector as long as the output includes object bounding boxes, labels, and probabilities. However, it is challenging to integrate other DNN models with ARCore or ARkit while maintaining real-time performance, with the exception of the SSD models maintained by Google. We plan to keep working on such integration in the future.

Table 3: AR Metrics

Data	AR^{10}	AR^s	AR^m	AR^l
SSD	24.0	0.1	6.3	17.6
OC+SSD	30.0	0.5	11.1	18.3
SF+OC+SSD	24.6	0.0	9.6	15.0
OSM+OC+SSD	26.0	0.4	9.6	16.0
ALL	20.0	0.0	7.4	12.5

6 LIMITATIONS AND FUTURE WORK

One limitation of our approach is that a newly detected truncated object might get rejected due to bad scale estimation and the fact that our map has no prior information about the detection. This can happen when the object is occluded by other objects or is only partially imaged by the camera. However, our comprehensive approach may still work if the online semantic map is continuously being updated using the output from the object detector. Another limitation is that the 3D sparse point clouds generated by the AR framework do not have correspondence between frames. Moreover, the 3D points do not have temporal consistency due to optimization during the SLAM process, which reduces the quality of our semantic map. In the future, we would like implement our own VIO algorithm so that we can further leverage the output of the VIO. Finally, our work may fail when the object being scanned moves.

There are a few future work directions that we can pursue. In this paper, we only use the sparse point cloud provided by the AR framework and bounding boxes from the object detector. In the future, we would like to explore how our pipeline can benefit from instance segmentation and dense depth map input. We would also like to evaluate how the accuracy of the VIO system under various conditions (e.g., scene depth, illumination, etc.) affects our object detection method. Moreover, we would like to extend our per-category scale database to make our approach more generic. For example, we would like to explore how to handle dynamic objects better by adding movement information to the database. Another idea is to customize the database specifically for the known environment in order to make the scale-based filtering more accurate.

REFERENCES

[1] A. Ahmadyan, T. Hou, J. Wei, L. Zhang, A. Ablavatski, and M. Grundmann. Instant 3D object tracking with applications in augmented reality. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Fourth Workshop on Computer Vision for AR/VR*, 2020.

[2] S. L. Bowman, N. Atanasov, K. Daniilidis, and G. J. Pappas. Probabilistic data association for semantic SLAM. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1722–1729, 2017.

[3] S. Brahmbhatt, H. I. Christensen, and J. Hays. StuffNet: Using stuffto improve object detection. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 934–943, 2017.

[4] W. Byeon, T. M. Breuel, F. Raue, and M. Liwicki. Scene labeling with LSTM recurrent neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3547–3555, 2015.

[5] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos. A unified multiscale deep convolutional neural network for fast object detection. In *European conference on computer vision (ECCV)*, pp. 354–370, 2016.

[6] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3150–3158, 2016.

[7] J. Dai, Y. Li, K. He, and J. Sun. R-FCN: object detection via region-based fully convolutional networks. In *30th International Conference on Neural Information Processing Systems*, pp. 379–387, 2016.

[8] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2147–2154, 2014.

[9] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. DSSD: deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*, 2017.

[10] R. Girshick. Fast R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448, 2015.

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 580–587, 2014.

[12] W. Han, P. Khorrami, T. L. Paine, P. Ramachandran, M. Babaeizadeh, H. Shi, J. Li, S. Yan, and T. S. Huang. Seq-NMS for video object detection. *arXiv preprint arXiv:1602.08465*, 2016.

[13] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988, 2017.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916, 2015.

[15] A. Holynski and J. Kopf. Fast depth densification for occlusion-aware augmented reality. *ACM Transactions on Graphics (TOG)*, 37(6):1–11, 2018.

[16] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[17] M. Klingensmith, I. Dryanovski, S. Srinivasa, and J. Xiao. Chisel: Real time large scale 3d reconstruction onboard a mobile device using spatially hashed signed distance fields. In *Robotics: Science and Systems*, vol. 4, p. 1, 2015.

[18] T. Kong, A. Yao, Y. Chen, and F. Sun. HyperNet: Towards accurate region proposal generation and joint object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 845–853, 2016.

[19] D. Kurz and S. B. Himane. Inertial sensor-aligned visual feature descriptors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 161–166, 2011.

[20] Y. Li, Y. Chen, N. Wang, and Z. Zhang. Scale-aware trident networks for object detection. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6053–6062, 2019.

[21] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollr. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):318–327, 2020.

[22] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 936–944, 2017.

[23] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pp. 740–755. Springer, 2014.

[24] M. Liu and M. Zhu. Mobile video object detection with temporally-aware feature maps. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5686–5695, 2018.

[25] R. Liu, J. Zhang, S. Chen, and C. Arth. Towards SLAM-based outdoor localization using poor GPS and 2.5D building models. In *2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 1–7, 2019.

[26] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *European Conference on Computer Vision (ECCV)*, pp. 21–37. Springer, 2016.

[27] J. McCormac, A. Handa, A. Davison, and S. Leutenegger. SemanticFusion: Dense 3D semantic mapping with convolutional neural networks. In *International Conference on Robotics and Automation (ICRA)*, pp. 4628–4635, 2017.

[28] B. Mu, S.-Y. Liu, L. Paull, J. Leonard, and J. P. How. SLAM with objects using a nonparametric pose graph. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4602–4609, 2016.

[29] A. Phalak, Z. Chen, D. Yi, K. Gupta, V. Badrinarayanan, and A. Rabinovich. DeepPerimeter: Indoor boundary estimation from posed monocular sequences. *arXiv preprint arXiv:1904.11595*, 2019.

[30] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016.

[31] J. Redmon and A. Farhadi. YOLO9000: better, faster, stronger. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6517–6525, 2017.

[32] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.

[33] M. Runz, M. Buffier, and L. Agapito. MaskFusion: Real-time recognition, tracking and reconstruction of multiple moving objects. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 10–20, 2018.

[34] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison. SLAM++: Simultaneous localisation and mapping at the level of objects. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1352–1359, 2013.

[35] B. Singh, M. Najibi, and L. S. Davis. SNIPER: Efficient multi-scale training. In *32nd Conference on Neural Information Processing Systems (NeurIPS)*, pp. 9310–9320, 2018.

[36] N. Sünderhauf, T. T. Pham, Y. Latif, M. Milford, and I. Reid. Meaningful maps with object-oriented semantic mapping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5079–5085, 2017.

[37] B. Tekin, S. N. Sinha, and P. Fua. Real-time seamless single shot 6D object pose prediction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 292–301, 2018.

[38] Y. Tian, Y. Ma, S. Quan, and Y. Xu. Occlusion and collision aware smartphone AR using time-of-flight camera. In *International Symposium on Visual Computing (ISVC)*, pp. 141–153. Springer, 2019.

[39] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. In *Conference on Robot Learning (CoRL)*, 2018.

[40] H. Uchiyama and E. Marchand. Object detection and pose tracking for augmented reality: recent approaches. In *18th Korea-Japan Joint Workshop on Frontiers of Computer Vision*, 2012.

[41] J. Valentin, A. Kowdle, J. T. Barron, N. Wadhwa, M. Dzitsiuk, M. Schoenberg, V. Verma, A. Csaszar, E. Turner, I. Dryanovski, J. Afonso, J. Pascoal, K. Tsotsos, M. Leung, M. Schmidt, O. Guleryuz, S. Khamis, V. Tankovitch, S. Fanello, S. Izadi, and C. Rhemann. Depth from motion for smartphone AR. *ACM Transactions on Graphics (TOG)*, 37(6), Dec. 2018.

[42] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese. DenseFusion: 6D object pose estimation by iterative dense fusion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3338–3347, 2019.

[43] A. Wong, M. Famuori, M. J. Shafiee, F. Li, B. Chwyl, and J. Chung. YOLO Nano: a highly compact you only look once convolutional neural network for object detection. *arXiv preprint arXiv:1910.01271*, 2019.

[44] Y. Xiang, W. Choi, Y. Lin, and S. Savarese. Data-driven 3D voxel patterns for object category recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1903–1911, 2015.

[45] Y. Xu, Y. Wu, and H. Zhou. Multi-scale voxel hashing and efficient 3D representation for mobile augmented reality. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1618–1625, 2018.

[46] H. Yu, J. Moon, and B. Lee. A variational observation model of 3D object for probabilistic semantic SLAM. In *International Conference on Robotics and Automation (ICRA)*, pp. 5866–5872, 2019.

[47] J. Zhang, M. Gui, Q. Wang, R. Liu, J. Xu, and S. Chen. Hierarchical topic model based object association for semantic SLAM. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 25(11):3052–3062, 2019.

[48] Y. Zhu, R. Urtasun, R. Salakhutdinov, and S. Fidler. segDeepM: Exploiting segmentation and context in deep neural networks for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4703–4711, 2015.