DeepMix: Mobility-aware, Lightweight, and Hybrid 3D Object Detection for Headsets

Yongjie Guan and Xueyu Hou
New Jersey Institute of Technology
{yg274,xh29}@njit.edu

Nan Wu and Bo Han
George Mason University
{nwu5,bohan}@gmu.edu

Tao Han
New Jersey Institute of Technology
tao.han@njit.edu

ABSTRACT

Mobile headsets should be capable of understanding 3D physical environments to offer a truly immersive experience for augmented/mixed reality (AR/MR). However, their small form-factor and limited computation resources make it extremely challenging to execute in real-time 3D vision algorithms, which are known to be more compute-intensive than their 2D counterparts. In this paper, we propose DeepMix, a *mobility-aware, lightweight, and hybrid* 3D object detection framework for improving the user experience of AR/MR on mobile headsets. Motivated by our analysis and evaluation of state-of-the-art 3D object detection models, DeepMix intelligently combines *edge-assisted 2D object detection* and novel, *on-device 3D bounding box estimations* that leverage depth data captured by headsets. This leads to low end-to-end latency and significantly boosts detection accuracy in mobile scenarios. A unique feature of DeepMix is that it fully *exploits the mobility of headsets to fine-tune detection results and boost detection accuracy*. To the best of our knowledge, DeepMix is the first 3D object detection that achieves 30 FPS (*i.e.,* an end-to-end latency much lower than the 100 ms stringent requirement of interactive AR/MR). We implement a prototype of DeepMix on Microsoft HoloLens and evaluate its performance via both extensive controlled experiments and a user study with 30+ participants. DeepMix not only improves detection accuracy by 9.1–37.3% but also reduces end-to-end latency by 2.68–9.15×, compared to the baseline that uses existing 3D object detection models.

CCS CONCEPTS

• **Human-centered computing** → **Ubiquitous and mobile computing systems and tools**; Systems and tools for interaction design.

KEYWORDS

3D Object Detection, Hybrid Mobile Vision, Augmented and Mixed Reality, Mobile Headsets

ACM Reference Format:
Yongjie Guan and Xueyu Hou, Nan Wu and Bo Han, and Tao Han. 2022. DeepMix: Mobility-aware, Lightweight, and Hybrid 3D Object Detection for Headsets. In *The 20th Annual International Conference on Mobile Systems, Applications and Services (MobiSys '22), June 25–July 1, 2022, Portland, OR, USA.* 14 pages. https://doi.org/10.1145/3498361.3538945

MobiSys '22, June 25–July 1, 2022, Portland, OR, USA

ACM ISBN 978-1-4503-9185-6/22/06…$15.00
https://doi.org/10.1145/3498361.3538945

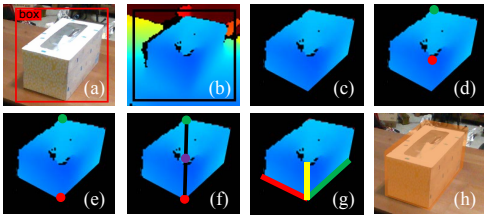

Figure 1: Workflow of DeepMix**: (a) 2D bounding box on an image, (b) Bounding box alignment on a depth frame, (c) Background removal, (d) Key points detection, (e) Key point projection, (f) Central point calculation, (g) Dimension and orientation estimation, (h) 3D bounding box visualization.**

1 INTRODUCTION

Mobile headsets such as Microsoft HoloLens [44] and Magic Leap One [42] bring numerous opportunities to enable truly immersive augmented/mixed reality (AR/MR). To offer the best quality of experience (QoE), real-time, interactive AR/MR should be able to perceive and understand the surrounding environment in 3D for seamlessly blending virtual and physical objects [10, 24]. With recent advances in 3D data capturing devices such as LiDAR and depth cameras, the computer vision (CV) community has developed several 3D object detection algorithms [12, 35, 51, 58, 63, 68, 73] by leveraging deep neural networks (DNNs). Due to the huge amount of data to process, 3D object detection is more computation-intensive than its 2D counterpart [15]. Moreover, the performance of 3D vision algorithms heavily depends on the quality of input data (*e.g.,* point cloud density or depth image resolution) [71]. Thus, existing AR/MR systems [3, 6, 38, 72] mainly focus on 2D object detection.

Even for the 2D case, it is well-known that the high latency caused by DNN inference negatively impacts the quality of user experience [3, 25]. A widely-used acceleration technique is to offload the compute-heavy tasks to cloud/edge servers [26, 38, 72], which is also a promising solution to speed up 3D object detection. However, we find that even with the help of a powerful GPU, the inference time of 3D object detection ranges from 72 to 283 ms (§2.3). By considering the network latency for offloading and local processing time on headsets, the end-to-end latency of AR/MR systems that integrate existing 3D object detection models would be higher than 100 ms, the threshold required by interactive AR/MR [6, 38], hindering providing an immersive experience to users.

In this paper, we propose DeepMix, a mobility-aware, lightweight, and accurate 3D object detection framework that can offer 30 frames per second (FPS) processing rate on Microsoft HoloLens 2, a commodity mobile headset. Our key insight is that instead of utilizing

Methods	Input	E2E Lat.	Accuracy	Mobility
MLCVNet [68]	Point Cloud	High	Medium	–
Trans3D [63]	RGB-D	Med. High	Low	–
D^{4}LCN [12]	RGB	Med. Low	Medium	–
DeepMix	Hybrid	Low	High	+

Table 1: Comparison of DeepMix **and existing DNN-based 3D object detection models. By exploiting headset mobility (+),** DeepMix **achieves low end-to-end (E2E) latency and high detection accuracy.**

DNN-based 3D object detection models to learn object class and infer bounding box, we can *decouple* the whole process and measure/estimate the 3D bounding box of an object by processing depth data on headsets. The key challenge of designing DeepMix is again the huge amount of 3D data to handle, given the limited computation resources on the headset. Also, while it is feasible, although not trivial, to measure the size and the 6DoF (six degrees of freedom) pose of an object (*i.e.,* its position and orientation), we still need to label the object of interest.

To address the above challenges, we design a *hybrid mechanism* that combines the mature DNN-based 2D object detection, which is fast by offloading it to the edge, and our lightweight and intelligent on-device depth data processing. More specifically, DeepMix offloads only 2D RGB images to the edge for object detection (*i.e.,* getting the label) and benefits from the returned 2D bounding box to drastically reduce the amount of to-be-processed 3D data. By doing this, DeepMix achieves accurate 3D object detection at line-rate (*i.e.,* 30 FPS). A unique feature of DeepMix is that it can fully exploit the movement of users to further fine-tune the measured bounding box and boost object-detection accuracy. To the best of our knowledge, DeepMix is the first 3D object detection framework that can bring about both low latency and high accuracy. We compare DeepMix and existing DNN-based models in Table 1.

Our detailed study of DeepMix consists of the following:

Performance Dissection of Existing 3D Object Detection Methods (§2). To understand the feasibility of applying existing DNN-based 3D object detection to interactive AR/MR, we investigate the detection accuracy and computation latency of eight state-of-the-art algorithms. We find that existing methods are not ready for real-time AR/MR applications due to the high computation latency.

Novel System Design of DeepMix **(§4).** As shown in Figure 1, DeepMix starts by offloading only RGB images to the edge that executes 2D object detection models for labeling objects of interest and generating their 2D bounding boxes (Figure 1 (a)). After aligning the bounding box on the depth image, it extracts depth data of only the target object (Figure 1 (b)–(c)). It then detects two key points on the 3D bounding box and projects one of them to the ground for determining the center point of the box (Figure 1 (d)–(f)). Finally, after inferring the dimension of the object and measuring its orientation, DeepMix renders the 3D bounding box on the display of the headset (Figure 1 (g)–(h)).

Effective Performance Optimization of DeepMix **(§5).** To further improve detection accuracy and end-to-end latency, we propose a few optimizations for DeepMix. Our key optimization is to leverage device mobility to refine the estimated bounding box. By doing this, we dramatically enhance the detection accuracy of DeepMix in dynamic environments. This feature makes DeepMix competitive for mobile AR/MR.

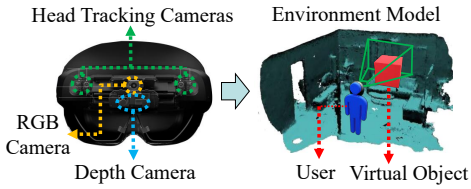

Figure 2: Configuration of mobile headsets (*i.e.,* Microsoft HoloLens 2) and a typical application scenario.

Implementation of DeepMix **and Performance Evaluation (§7).** We build a prototype implementation of DeepMix and thoroughly evaluate its performance via repeatable, controlled (live) experiments and a user study with more than 30 participants. We highlight our evaluation results as follows.

- On a high-throughput WiFi network, the end-to-end latency of DeepMix is only 34 ms (§7.2), much lower than that of existing DNN-based models (ranging from 91 to 311 ms).
- Compared to the besting performing existing model (D^{4}LCN [12]), the accuracy improvement of DeepMix increases from 3.5% for the static scenarios to up to 11.5% for the mobile scenarios.
- The experimental results from our user study demonstrate that the accuracy of DeepMix is 12.5%, 5.1%, and 9.6% higher than that of the most accurate existing model (D^{4}LCN [12]) for three pre-defined mobility patterns, leading to a better QoE (§7.8).

Overall, DeepMix is a first-of-its-kind practical 3D object detection framework for mobile headsets. We make the following contributions in this paper: (1) performance dissection of DNN-based 3D object detection models in the context of real-time, interactive AR/MR on mobile headsets; (2) system design of DeepMix, a full-fledged, ready-to-deploy 3D object detection framework for commodity mobile headsets that fully exploits device mobility to boost detection accuracy; and (3) prototype implementation and evaluation of DeepMix, including dataset-driven repeatable experiments, controlled live experiments, and an IRB-approved user study. We plan to release the implementation of DeepMix.

2 BACKGROUND & MOTIVATION

2.1 Mobile Headsets for AR/MR

Different from smartphones that can support only video see-through AR by overlaying virtual content in the physical world that is displayed via the devices' camera view, headsets allow users to see the physical world through a transparent, optical see-through display that simultaneously imposes virtual objects into the user's view of the surrounding environment using optical combiners [22]. As a result, those headsets create a truly immersive AR/MR experience, compared to smartphones and tablets, by extending our perception of the environment from 2D images to the 3D real world and enabling interactions between users and virtual objects.

Take Microsoft HoloLens as an example [45]. As illustrated in Figure 2, it has an RGB camera, a time-of-flight (ToF) sensor for depth perception, four visible light cameras for head tracking, and two infrared cameras for eye tracking. It is also equipped with an inertial measurement unit (IMU) with an accelerometer, gyroscope, and magnetometer. With these sensors, HoloLens can perceive the surrounding environment by building a 3D model and blending the digital and physical worlds based on this 3D model of the environment. To accurately mix virtual content with physical objects,

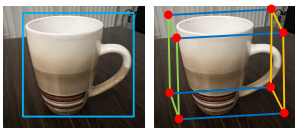

Figure 3: Comparison between 2D and 3D object detection.

HoloLens creates a spatial coordinate system of the physical world. This coordinate system uses the initial location where the HoloLens was turned on as the origin. Moreover, to guarantee an immersive experience, AR/MR applications running on HoloLens should be capable of detecting objects in 3D space (*i.e.,* conducting 3D object detection [35, 51, 58, 73]), instead of leveraging 2D object detection in traditional AR systems [3, 6, 38, 72].

Mobile headsets are usually lightweight and wearable. As a result, their hardware resources and computation capabilities are limited. For instance, Microsoft HoloLens has an Intel 1GHz 32-bit processor with a customized holographic processing unit (HPU) and only 2GB of memory [45]. Such limited computation resources make it challenging to support the real-time execution of deep neural networks for 3D object detection [35, 51, 58, 73]. Furthermore, headsets' batteries can usually last only 2-3 hours, and the heat generated from the headset can only be dissipated via passive cooling. Hence, considering the energy consumption, mobile headsets are unsuitable for executing heavy computation tasks.

2.2 A Primer of 3D Object Detection

In Figure 3, we visualize the difference between 2D and 3D object detection. The result of 2D object detection is a rectangular bounding box of the object in a 2D image. In contrast, the result of 3D object detection is a cubic bounding box of the object that provides three dimensional information of the object in the real world.

We can classify existing methods of 3D object detection into three categories based on their input-data format. The first one utilizes point clouds as the input and directly draws 3D bounding boxes on them [35, 48, 49, 54, 58, 68]. Point clouds can be either captured by LiDAR devices or generated by processing the RGB images and their corresponding depth images. The second category uses RGB images as the input of DNN models and learns 3D bounding boxes that will be drawn on 2D images [7, 8, 12, 41]. Some of the algorithms actually generate/estimate depth maps from RGB images to train the DNN models [12, 41]. Image-based 3D object detection is an active research area because its run-time inference relies on only RGB images that are much easier to capture at a low cost, compared to 3D data such as depth maps and point clouds. The third category benefits from 2.5D data (*i.e.,* RGB-D images) that combine 2D RGB images and depth maps [34, 50, 61, 63]. While both DeepMix and methods in this category use RGB images and depth data, the key difference is that DeepMix offloads only RGB images to the edge for 2D object detection and processes depth data on the headset, whereas models with RGB-D input offload both RGB and depth images for 3D object detection. We offer a detailed review of existing work on 3D object detection in §9.

2.3 Challenges of 3D Object Detection

There are several challenges when executing 3D object detection for AR/MR applications on mobile headsets. The first one is that the performance of 3D vision algorithms heavily depends on the quality of

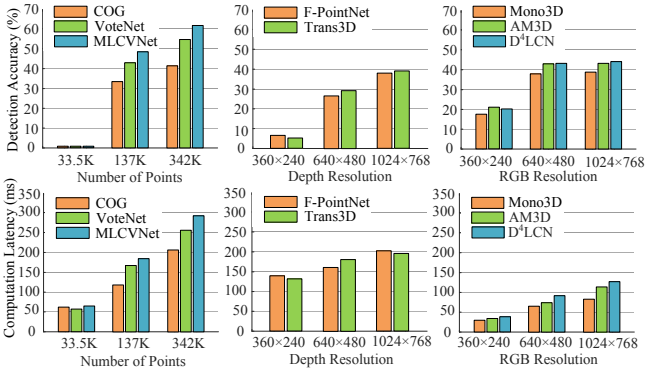

Figure 4: Accuracy (1st row) and computation latency (2nd row) of 3D object detection methods using point cloud (left column), RGB-D (middle column), and 2D image (right column) input-data formats. The point clouds with 33.5K, 137K, and 342K points are generated from depth images with 360×240, 640×480, and 1024×768 resolutions, respectively.

input data (*i.e.,* the resolution of depth images or the density of point clouds). For example, the accuracy of 3D semantic segmentation decreases when the point clouds become sparse [70]. However, due to the limited hardware resources on mobile headsets, the resolution of their captured depth images is usually low, for instance, 360×360 for Microsoft HoloLens 2 at 30+ FPS[1]. In contrast, standalone RGB-D cameras such as Intel RealSense and Microsoft Kinect DK can capture depth images with 1024×768 and 1024×1024 resolutions at 30+ FPS[2]. Thus, the density of point clouds generated from the depth images is also low (*e.g.,* around only 33.5K points when using 360×240 depth images).

In order to understand the impact of input-data quality on 3D object detection, we evaluate the accuracy of the following representative algorithms using point clouds with different densities that are generated from depth images with different resolutions. We use 3D IoU that is defined in §7 as the evaluation metric. We select COG [54], VoteNet [49], and MLCVNet [68] for 3D object detection with point clouds as input and F-PointNet [50] and Trans3D [63] for models using RGB-D input. As the baseline, we evaluate the performance of Mono3D [7], AM3D [41], and D^{4}LCN [12] that take 2D images as input. We train the above models with the publicly available SUN RGB-D dataset [60]. The testing RGB and depth images with different resolutions are created by the Intel RealSense camera. As a motivating example, the captured object is a bottle under a given setting (*i.e.,* a specific viewing angle and distance, see Figure 10). We conduct an extensive evaluation for different objects under different settings in §7.

The main observations from the experimental results in Figure 4 are as follows. First, the detection accuracy is extremely low for point-cloud-based models (at most 1.6% in the upper-left subfigure) and models using RGB-D images as input data (5.1%-6.8% in the upper-middle subfigure), when the resolution of depth images is 360×240 (*i.e.,* the typical setup of HoloLens 2). Second, when the quality of input data is low, image-based models achieve the most

[1]While Microsoft HoloLens 2 can generate 1024×1024 depth images, the frames rate is only 1-5 FPS at this high resolution, which is too low for interactive AR/MR.
[2]Intel RealSense captures rectangle depth images, whereas Microsoft HoloLens and Kinect capture square ones.

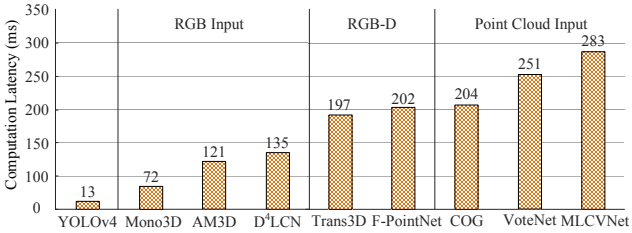

Figure 5: Comparison of the computation latency of 2D vs. 3D object detection algorithms. YOLOv4 is for 2D object detection, and the rest are all for 3D object detection with different input data formats.

accurate detection among the three categories, whereas point-cloud-based models are more accurate than the other two for high-quality input data. Third, with high-quality input, point-cloud-based models achieve the most accurate detection, but lead to the highest computation latency. Fourth, the computation latency of point-cloud-based and image-based models drastically increases for high-density point clouds and high-resolution images.

Another challenge of leveraging 3D object detection for mobile AR/MR applications is the high computation overhead and the resulting high latency of data processing. To better appreciate this issue, we compare the inference time of traditional 2D object detection models such as YOLOv4 [5] with the aforementioned representative 3D object detection models. The input RGB images of both 2D and 3D models have the same resolution of 1280×720, to make the comparison fair. The resolution of the input depth images is 1024×768 for 3D models. To follow the common practice of edge-based acceleration for 2D object detection/recognition in mobile AR [38, 72], we conduct the experiments on a machine with an NVIDIA RTX 2080S GPU and present the results in Figure 5.

We have the following three observations from Figure 5. First, the computation latency for most 3D object detection algorithms is higher than 100 ms, making them unsuitable for real-time, interactive AR/MR applications [6, 38]. Ideally, the latency should be at most 33–34 ms to achieve 30 FPS line-rate processing. While the latency of image-based models could be lower than 100 ms, as we will show in §7, by adding the extra network latency and local computation time, the end-to-end latency would still deteriorate the quality of user experience. Second, the computation latency of 3D object detection is much higher than its 2D counterpart. It takes only 13 ms for YOLOv4 [5] to detect objects on 2D images, whereas the computation latency could be as high as 283 ms for 3D models. Third, the computation latency of 3D object detection heavily relies on the complexity of input data.

The above large performance gap makes edge-side optimizations, such as DNN-model acceleration and better GPU support, challenging. Note that we assume point clouds will be created on the server to reduce network latency and computation overhead on the headsets. Generating high-fidelity point clouds, which is required to improve detection accuracy (Figure 4), also takes time and will further increase the latency of point-cloud-based models.

Summary: The state-of-the-art 3D object detection solutions are not suitable for supporting AR/MR applications on mobile headsets due to the following two reasons.

• Existing 3D object detection models achieve the most accurate result when using high-quality point clouds as input data, which *cannot be generated by commodity mobile headsets* due to their limited hardware resources.

• The computation latency of existing 3D object detection models, even with edge offloading, are *too high to guarantee a truly immersive experience* for real-time, interactive AR/MR that requires imperceptible latency (<100 ms).

The poor performance of existing 3D object detection models and the complex interplay among the input-data quality, detection accuracy, and computation latency motivate our design of DeepMix, which effectively combines edge-assisted 2D object detection and on-device lightweight 3D bounding box estimation with depth data.

3 OVERVIEW OF DEEPMIX

DeepMix is a generic 3D object detection framework that is designed for enhancing AR/MR experience on mobile headsets. It is mobility-aware by taking advantage of user movement to refine measured 3D bounding boxes, lightweight by avoiding heavyweight 3D object detection and resorting to the mature 2D counterpart, and hybrid by effectively splitting workload between the edge (*i.e.,* RGB-image-based 2D object detection) and the headset (*i.e.,* depth-image-based 3D bounding box estimation). We depict the system architecture of DeepMix in Figure 6.

The design of DeepMix is inspired by three key observations of existing solutions for object detection and the differences between smartphone-based and headset-based mobile AR/MR. First, while DNN-based 3D object detection leads to high computation latency even when assisted by the edge, its 2D counterpart is computation-efficient (*e.g.,* 13 ms latency in Figure 5). Second, although mobile headsets are equipped with various sensors to facilitate AR/MR applications, their hardware resources are typically constrained and the low-resolution depth images limit the performance of 3D object detection models. Third, headset-based AR/MR differs from smartphone-based one by rendering the bounding box using the physical location of the object, instead of its relative position in the camera view.

The overarching goal of DeepMix is to simultaneously reduce end-to-end latency and increase detection accuracy, improving QoE for next generation headset-based AR/MR. To achieve the above goal, we face the following challenges when designing DeepMix.

• How to jointly consider existing techniques to reduce end-to-end latency of 3D object detection?

• How to accurately and efficiently measure/estimate the 3D bounding box of an object from depth data on the headset?

• How to boost the performance of DeepMix under different scenarios for improving user experience?

Next, we present how we address these challenges.

4 SYSTEM DESIGN OF DEEPMIX

In this section, we introduce the basic design of DeepMix. We will explain how to improve its performance in §5.

4.1 Edge-assisted 2D Object Detection

As shown in Figure 6, the workflow of DeepMix begins with retrieving RGB images and offloading them to the edge for conducting 2D object detection. Given that DeepMix is a generic framework,

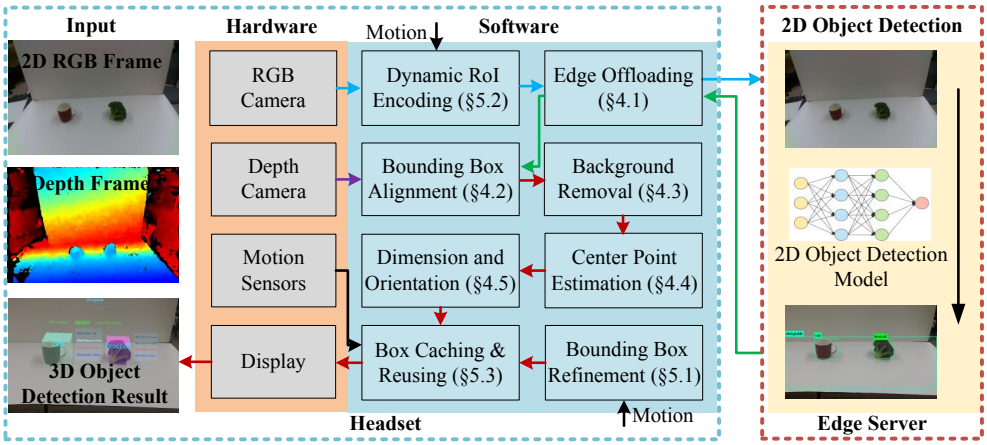

Figure 6: System architecture and workflow of DeepMix**.**

it can work with any DNN-based model that can accurately label objects and generate their 2D bounding boxes in real time [5, 19, 53]. The 2D bounding box drawn on the RGB image will be used as the starting point to derive the 3D bounding box using depth data. Since the main purpose of the 2D bounding box is to reduce the amount of to-be-processed depth data, the key requirement is that the object should completely fit into the returned bounding box, which could be larger than the object if doing this can speed up 2D object detection. We will describe how to optimize the offloading efficiency in terms of data usage in §5.2.

4.2 Bounding Box Alignment on Depth Frame

The next step is to align 2D bounding boxes from the RGB image onto the depth image. Since it takes time to get the results from the edge, during which the camera view may change due to movement, we need to first transform the returned 2D bounding boxes on the offloaded image to the current viewport. Otherwise, there will be a misalignment between 2D bounding boxes and objects, as shown in Figure 7. To solve this problem, DeepMix records the 6DoF pose of the frame when it is captured by the RGB camera, which is provided by the headset. Once users start the headset, its motion sensors (*e.g.,* gyroscope, accelerometer, visible light cameras, *etc.*) begin to track the headset's 6DoF pose during movement and make it available to applications. After receiving the detection results from the edge, it can transform the 2D bounding box to the current viewport based on the change of 6DoF pose (*i.e.,* θ and d in Figure 7) [55].

DeepMix then calculates the coordinate of the center pixel for a detected object using the updated 2D bounding box, based on its four vertices ($R_{Right}, R_{Left}, R_{Top}, R_{Bottom}$), as $R_{Ctr} = ((R_{Right} + R_{Left})/2, (R_{Top} + R_{Bottom})/2)$. Different from the setup of RGB-D cameras, most headsets are equipped with an RGB camera and a depth camera that are not synchronized with each other. As a result, both the center point and the resolution of the depth frame are different from those of the corresponding RGB image (captured at the same time). To determine the center point P'_{Ctr} on depth frame that is mapped to the center R_{Ctr} on RGB frame, we can utilize the pinhole camera principle [62].

Note that P'_{Ctr} is just a point on the surface of the object on the depth image, not the actual center point of the 3D bounding box. This point will be used for determining one of the surfaces of the 3D bounding box (§4.5). Similarly, we can get the corresponding points on the depth frame for the four vertices of the 2D bounding box, which will be used for background removal (§4.3). Note that the accuracy of 3D bounding box estimated by DeepMix is not determined by the accuracy of 2D bounding box generated by object detection algorithms. DeepMix uses the 2D bounding box mainly to reduce the amount of 3D data that should be processed on the headset when estimating the 3D bounding box.

4.3 Background Removal on Depth Image

After getting the bounding box of the object on the depth image, in this step, DeepMix removes the background in the bounding box to reduce computation overhead and improve the accuracy of our 3D bounding box estimation, by leveraging existing solutions developed by the computer graphics (CG)/CV communities [30]. An alternative solution is to perform semantic segmentation, which can label each pixel, instead of object detection on the edge. However, this will increase both data transmission overhead by sending per-pixel labels and computation overhead to match each pixel of the object onto the depth image. After removing the background, we can obtain depth data of mainly the detected object. Since the collected depth data may contain noises and undetected areas of the object, the depth information of the above vertices and the center point on the depth frame may be missing. To improve the quality of depth frames, we further apply an edge-preserving filter and Spatial Hole-filling algorithm [23] to the depth frame to make it smooth and complete.

4.4 Center Point Estimation

There are three parameters to determine the 3D bounding box of an object, the spatial position (*i.e.,* location), 3D dimensional size (*i.e.,* height, width, and thickness), and orientation. We first estimate the center point of the 3D bounding box (*i.e.,* the spatial position of the object). We then measure the dimension and orientation of the object in §4.5. With the depth data, DeepMix can get the spatial coordinates of the closest point P_{Min} and the furthest point P_{Max} to the headset, as shown in Figure 8. After that, it projects P_{Min} to the plane that the object is placed on, which could be detected by the headset, to get P'_{Min}. The center point of the 3D bounding box P_{Ctr} is estimated as the center of the line connecting P'_{Min} and P_{Max}. The estimation of the two key points may not always be accurate,

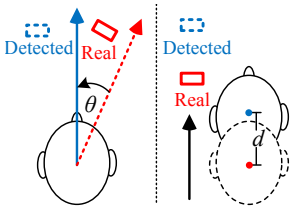

Figure 7: Misalignment caused by movement.

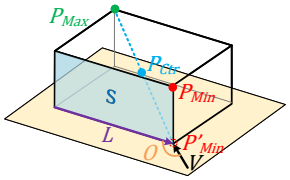

Figure 8: Geometric model of bounding box.

especially for P_{Max}, as the actual furthest point may be occluded and thus not visible. By exploiting the headset movement, we propose an optimization to further improve the accuracy when users move around the object to observe more details (§5.1), which is a typical use case for headset-based mobile AR/MR.

4.5 Dimension and Orientation Estimation

To get the dimension and orientation of the 3D bounding box, DeepMix first detects the surface S that P'_{Ctr} is on, as shown in Figure 8[3], with the following method. It uses P'_{Ctr} as a start pixel of a seed patch [16] on the depth frame. It then grows the patch to a certain size and utilizes the linear least-squares plane fitting [40] to identify the best fitting plane for this patch. This plane will be used to approximate the surface S in Figure 8. To improve the accuracy of this estimation, DeepMix can repeat the above process multiple times with different start pixel of the seed patch, for example, by using other points close to P'_{Ctr}, and then aggregate the calculated planes to approximate S.

With the surfaces S and the center point P_{Ctr} (§4.4), DeepMix can calculate the dimension of the 3D bounding box. If the distance between P_{Ctr} and the underlying plane is d_h, the object height H is $2 \times d_h$. Next, DeepMix calculates the distance d_t between P_{Ctr} and S. The thickness of the bounding box T will be $2 \times d_t$. With H and T, we can calculate the width of the bounding box W by using the right angle theorem: $W = \sqrt{d^2_{P'P} - H^2 - T^2}$, where $d_{P'P}$ is the distance between P'_{Min} and P_{Max}. After getting the spatial position and dimension of the object, DeepMix still needs to determine the orientation O of the 3D bounding box. It first calculates the intersecting line L of the surface S and underlying plane. From the 6DoF pose, DeepMix knows the viewing direction V of the user D. Thus, O can be calculated based on the angle between L and V, as shown in Figure 8.

5 PERFORMANCE OPTIMIZATIONS

5.1 Motion-aware Bounding Box Refinement

A unique feature of DeepMix is that it can keep refining estimated 3D bounding boxes when users move around an object of interest, for example, to investigate the details. As we will show in §7.4, existing DNN-based 3D object detection models cannot benefit from headset movement in their current form. This refinement mode is enabled only when users move around an object, which can be inferred from the 6DoF pose of the headset and the location of the object. For two consecutive 3D bounding boxes that are estimated by DeepMix, it first gets the spatial point that is the center of the line connecting the two center points of the two boxes. It then uses this point as the center of the updated bounding box and moves the two estimated boxes to this point. It finally uses the union of the two boxes as the updated box, which will be combined with the next estimated bounding box.

A key difference between video see-through based AR/MR on smartphones and optical see-through based one on headsets is that the latter does not need to continuously offload camera views to the edge even when users move. The location of an object displayed on the screen of smartphones changes if users move, which requires conducting object detection on the updated camera view. Optical flow tracking can alleviate this issue only to some extent, as the tracking error will accumulate as time goes on. On the other hand, the 3D bounding box of an object is determined by its actual physical location and orientation that will not change with headset movement. The underlying coordinate-system drift caused by movement will be fixed by the headset itself, and thus DeepMix can always get an accurate pose from the headset to update the rendered bounding box. As a result, headset-based AR/MR does not need to frequently perform (edge-assisted) object detection. To further optimize the overhead of DeepMix's bounding box refinement, we next introduce the motion-assisted dynamic region of interest (RoI) encoding to decrease the offloading overhead.

5.2 Motion-assisted Dynamic RoI Encoding

Dynamic RoI encoding selectively applies lossy compression to parts of the frame that are less likely to contain objects of interest and lossless compression to other areas for reducing the amount of encoded data. The RoIs on the current frame are determined by analyzing the microblocks of 2D images and checking whether they overlap with the identified RoIs in a previous frame. This scheme has been demonstrated to be helpful for AR on handheld smartphones with limited moving speed [38]. However, the camera view of the headset may drastically change with user movement. For example, the peak speed of head movement can reach 240 degrees per second [14], much higher than the moving speed of a smartphone when used for AR and making microblock-based scheme less effective for headsets.

DeepMix resorts to the 6DoF tracking offered by headsets to solve this problem. By recording the 6DoF pose of consecutive frames, it can determine whether they overlap with each other. If not, dynamic RoI encoding will not be applied. Otherwise, DeepMix checks whether there are known RoIs of a previous frame appearing on the current frame and (if they do) get their locations on the current frame through coordinate transformation. DeepMix compresses the identified RoIs and the area that is not overlapped with the previous frame losslessly and the remaining area in a lossy fashion. Note that dynamic RoI encoding is a generic design and can be applied to not only the bounding box refinement mode but also other scenarios.

5.3 3D Bounding Box Caching and Reusing

To better support mobile scenarios where users move around to explore the surrounding environment, we design a mechanism to cache and reuse 3D bounding boxes of detected objects, which avoids unnecessary detection of the same object multiple times. The

[3] S could be the other vertical surface shown in Figure 8, but this does not affect the estimation (width vs. thickness). S could also be the top horizontal surface. In this case, DeepMix keeps moving P'_{Ctr} on this surface toward a direction until it hits one of the two vertical surfaces.

goal is to display the 3D bounding box of a detected object as fast as possible, when it reappears, by reducing the initial rendering time, which can boost user experience and decrease computation resource utilization on both the edge and the headset. This optimization is helpful, especially under dynamic network conditions that increase end-to-end latency of object detection.

In the cache, we store the 6DoF pose and 3D dimension of detected objects. When users move, DeepMix keeps updating the viewing frustum (*i.e.,* 3D viewport) based on the 6DoF pose of the headset and checks whether there are cached items that should be in the current viewport by examining the 6DoF pose of cached bounding boxes. To further reduce the rendering time of 3D bounding boxes for cached items, DeepMix saves their translucent cubes in the memory. Based on the cached results, it can reshape and rotate the cubes and immediately render them on display. After that, DeepMix performs object detection on the current viewport, in case there are new objects in the scene, and updates the cache accordingly. Another benefit of our caching design is that if a cached item is further away from the user in the updated viewport and out of the range of the depth camera, DeepMix can still render its bounding box that is retrieved from the cache.

6 SYSTEM IMPLEMENTATION

We develop a prototype implementation of the DeepMix client on Microsoft HoloLens 2 and the DeepMix edge server on Linux. We implement the device-side functions with Windows SDK [46], DirectX [43], and Unity 3D engine [65]. We use multi-threading to simultaneously read data from both RGB and depth cameras. We collect the camera frame using libraries of Windows SDK. When receiving the 2D detection results from the edge server, we obtain the depth frame by enabling the Research Mode of HoloLens. We store depth images in bitmaps to improve the speed of data processing. To enable motion-based dynamic RoI encoding, we utilize the position and orientation of the headset, which are retrieved via a library in DirectX. After estimating the 3D bounding box, we render it on the screen with Unity. By adapting the detection results from the previous frame to the change of users' 6DoF pose position, we encode the RoI of the current frame using JPEG and send it to the edge. We implement the DeepMix edge server in the Darknet [52] open-source neural networks. The edge provides 2D object detection for DeepMix using YOLOv4 [5]. As a generic framework for 3D object detection, DeepMix can use any mature 2D object detection model on the edge.

In total, our implementation consists of 4,600+ lines of code (LoC): 3,000+ LoC in C# (rendering, device localization, and bounding box estimation) and 1,000+ LoC in C++ (gathering sensor data, image compression, and networking) for the client, and 600+ LoC in C++ (networking and multi-threading) for the server. We also build a prototype of DeepMix on HoloLens (1st gen), on which the performance of DeepMix is only slightly worse than that of HoloLens 2. Hence, we report the results for only HoloLens 2.

7 PERFORMANCE EVALUATION

In this section, we measure the performance of DeepMix through dataset-driven evaluations, controlled (live) experiments, and an IRB-approved user study.

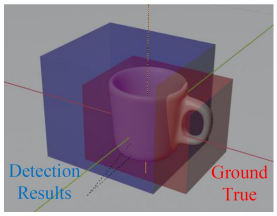

Figure 9: Visualization of 3D IoU.

7.1 Experimental Setup

We compare the performance of DeepMix with the following *eight* start-of-the-art 3D object detection models, COG [54], VoteNet [49], and MLCVNet [68] with point clouds as input, F-PointNet [50] and Trans3D [63] for models using RGB-D input, and Mono3D [7], AM3D [41], and D^{4}LCN [12] that take 2D images as input.

Testbed. The edge server is equipped with an Intel i9-9900k CPU, an NVIDIA RTX 2080S GPU, and 64GB DDR4 3200MHz RAM. The headset, Microsoft HoloLens 2, and the edge run the Universal Windows Platform (version 10.0.20346.0) and Ubuntu 16.04, respectively. For most experiments, we connect the headset and the edge with a Linksys AC1900 WiFi router that is attached to the same 1 Gbps Ethernet as the edge server. The normal throughput of this WiFi network is around 260 Mbps, and its round trip delay is less than 1 ms. We use this WiFi router exclusively for our experiments, by avoiding interference with other co-existing WiFi networks. For the experiments under dynamic network conditions, we attach an LTE modem to the headset, which connects to the edge server through our USRP-based LTE base station. The throughput of this LTE network ranges from 8.4 to 37.1 Mbps, and its typical round trip delay is about 14 ms.

Evaluation Metrics. We use the accuracy of 3D object detection and the end-to-end latency as the metrics to evaluate DeepMix. We measure the device power rate and other computation resource utilization (*e.g.,* CPU, GPU, and memory) on Microsoft HoloLens 2.

3D Intersection over Union. We evaluate the accuracy of the 3D bounding box using 3D Intersection over Union (3D IoU), as shown in Figure 9, which has been widely used in the literature [9, 33, 39, 51, 58, 59, 66]. By following the common practice in the computer vision community [9, 33, 51, 59, 66], we set the 3D IoU thresholds to be 0.25 and 0.5, respectively. That is to say, when the 3D IoU is larger than the threshold, we consider the detection result to be accurate. In the following, we report the percentage of accurate detections using the 3D IoU metric.

End-to-end Latency. The end-to-end latency is important for real-time, interactive AR/MR systems. We record the time t_i when the ith frame is captured by the camera and the time $\hat{t}_i$ when the 3D bounding boxes are rendered for it. The latency of the ith frame is defined as $\tau_i = \hat{t}_i - t_i$. Let n be the number of processed frames. The end-to-end latency can be expressed as $\Delta = \sum_{i=1}^{n} \tau_i / n$.

Battery Power Level. To monitor and analyze the battery power level of the headset, we disassemble it and remove its battery. In the experiment, a Keithley 2281S battery simulator [32] is used as the power supply, which can provide the headset with a stable DC input and monitor the current and power. In this way, we can have a systematic evaluation of the headset's battery power level.

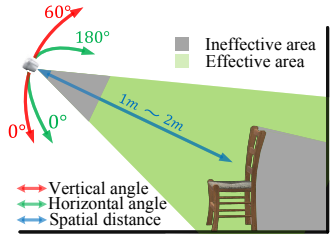

Figure 10: Different settings for constructing the dataset.

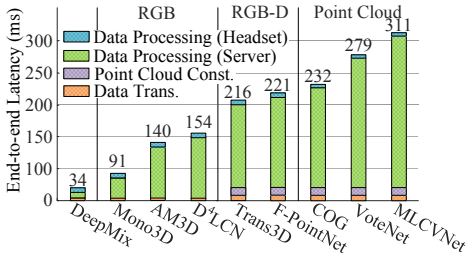

Figure 11: Comparison of end-to-end latency.

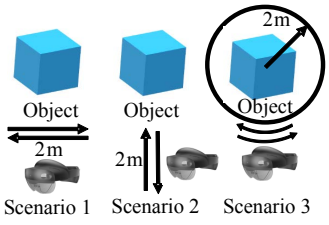

Figure 12: Three mobility patterns.

Evaluation Datasets. We train the state-of-the-art 3D object detection models with the well-known SUN RGB-D dataset [60], and train YOLOv4 [5] that is used in DeepMix for 2D object detection with the popular COCO [37] dataset. One unique challenge of comparing the performance of DeepMix with existing models is it needs extra information that is not included in regular RGB-D datasets such as SUN RGB-D. The reason is that to make DeepMix lightweight and suitable for mobile headsets, we leverage the 6DoF pose of the headset to estimate the 3D bounding box, which is missing from existing RGB-D datasets.

To address this challenge and conduct a fair comparison of DeepMix and existing models, we construct a test dataset that consists of not only RGB-D images but also the 6DoF pose of the camera (*i.e.,* an Intel RealSense D435i [27] camera) when capturing the images. Our test dataset has 2,184 RGB-D images captured with different viewing angles and distances, as shown in Figure 10. The resolution is 1280×720 for RGB images and 1024×768 for depth images, much higher than the 360×360 resolution of depth images generated by HoloLens 2 at line rate. This is the reason that we choose Intel RealSense D435i as the capturing device. This RGB-D dataset contains seven classes of objects, table, chair, bottle, box, desk, bag, and TV, which are common objects in the public datasets [37, 60]. Our dataset is diverse as it covers objects of different sizes, shapes, and materials (which affect the generation of depth images), and some objects (*e.g.,* chair, bottle, and bag) may have irregular shapes. The whole dataset is annotated with the ground truth 3D bounding boxes (*i.e.,* accurate object orientation and position). We plan to make our collected dataset publicly available, which may benefit future research on designing more efficient and accurate 3D object detection models for mobile headsets.

7.2 End-to-end Latency

We first compare the end-to-end latency of DeepMix with existing 3D object detection models by replaying the RGB-D images in our collected dataset on HoloLens 2 and plot the results in Figure 11.

We break down the end-to-end latency into four parts, data transmission, point cloud construction, server-side data processing, and client-side data processing. DeepMix and image-based models send only RGB images to the edge; whereas the others send RGB-D images to the edge. Note that instead of generating point clouds on the headset and sending them to the edge for object detection, we send RGB-D images and create point clouds on the edge. The reason is that doing this can not only reduce computation resource utilization and energy consumption, but also drastically save network data usage. For example, the size of a point cloud generated from an RGB image and a depth image (with a combined size of 1.68 MB) could be as large as 35.9 MB. The server-side data processing latency is mainly the inference time of DNN models for 2D/3D object detection. For DeepMix, the estimation of 3D bounding boxes happens on the headset, after receiving the returned 2D bounding box. The client-side data processing of existing models mainly includes the transformation of returned 3D bounding boxes to the coordinate system of the headset for rendering and display.

The key observation from Figure 11 is that the end-to-end latency of DeepMix is significantly lower than existing 3D object detection models. It takes only 34 ms for DeepMix to accurately detect 3D objects, among which the latency is 5 ms (14.7%) for data transmission, 13 ms (38.2%) for server-side data processing, and 16 ms (47.1%) for client-side data processing, respectively. Compared to image-based models, DeepMix's dynamic RoI encoding scheme (§5.2) effectively reduces the data transmission time by 34% (from 6.7 ms to 5 ms). As DeepMix estimates the 3D bounding box on the headset, its client-side data processing time (16 ms) is slightly longer than that of other schemes (13.8 ms for image-based models and models with RGB-D input, 11 ms for point-cloud-based models). Thus, thanks to its lightweight design (§4) and various optimizations (§5), the latency of DeepMix is far below the 100 ms requirement for interactive AR/MR.

For existing 3D object detection models, their end-to-end latency is dominated by the server-side processing time, especially for point-cloud-based solutions. Even on the edge server, it takes about 23 ms to generate the point clouds. Note that although the end-to-end latency of Mono3D [7] is also lower than 100 ms, as we will show next (§7.3 & §7.4), its detection accuracy is much lower than that of DeepMix, and its latency will be higher than 100 ms under dynamic network conditions (Figure 15). Figure 11 shows that the data transmission time of existing models with RGB-D and point cloud inputs are higher than that of DeepMix and image-based models, due to the additional depth images that are needed to learn the 3D bounding box. We conduct the above experiments on the WiFi network with high throughput (§7.1). When the network condition becomes worse, this data transmission time of RGB-D data will be more noticeable. We will evaluate the end-to-end latency on an LTE testbed in §7.7.

7.3 Detection Accuracy in Static Scenario

We compare the detection accuracy of DeepMix with existing models for the static scenario using the 3D IoU metric. To make the comparison fair, we replay on the headset the 2,184 RGB-D images in our collected dataset. We present the results for 3D IoU in Table 2. For the static scenario, remarkably, DeepMix achieves better overall

Methods	Input	3D IoU@0.25/0.5							Average
		Table	Chair	Bottle	Box	Desk	Bag	TV	
COG [54]	PC	47.3/2.6	39.3/-	46.4/1.8	57.3/15.2	49.5/13.3	45.1/10.3	39.5/5.7	47.9/7.2
VoteNet [49]	PC	56.2/23.6	45.5/22.2	67.2/18.3	52.4/22.6	48.6/14.2	54.3/18.5	41.9/19.2	51.6/21.7
MLCVNet [68]	PC	61.3/22.5	**74.2**/27.6	**68.2**/16.4	62.1/22.3	63.7/26.5	57.6/21.1	52.3/17.9	63.1/22.1
F-PointNet [50]	RGB-D	45.5/11.2	37.5/17.3	44.2/9.5	31.6/17.5	38.5/-	54.5/17.6	37.4/8.2	42.3/11.9
Trans3D [63]	RGB-D	52.3/13.5	38.4/-	47.2/13.2	39.5/8.4	47.2/18.4	49.1/-	37.2/12.9	45.8/10.1
Mono3D [7]	RGB	59.3/11.2	52.4/9.5	47.3/8.6	65.3/13.2	67.5/21.5	48.6/5.2	54.5/16.4	55.4/10.7
AM3D [41]	RGB	65.9/14.9	71.2/11.3	62.7/16.3	57.8/10.7	49.2/12.5	51.7/11.2	57.9/12.9	58.7/13.6
D^{4}LCN [12]	RGB	68.3/14.3	69.2/24.4	67.2/18.6	58.9/21.5	62.3/14.3	56.6/9.6	57.2/11.7	63.9 /15.9
DeepMix	Hybrid	**72.4/37.5**	67.4/**33.7**	63.1/**38.4**	**66.5/39.1**	**72.1/33.2**	**65.8/46.5**	**61.2/31.8**	**67.4/37.2**

Table 2: Class-wise comparison of DeepMix **and state-of-the-art 3D object detection models ('-': the method could not detect the object).**

performance than all other models across different object categories. On average, DeepMix has the highest 3D IoU for both the 0.25 and 0.5 thresholds. One possible reason is that instead of completely relying on learning-based models, DeepMix estimates the 3D bounding box using pixel-level depth information on the headset.

When the threshold is 0.5, DeepMix's accuracy is the highest for all object classes, and is 1.68× (on average) over the best existing model MLCVNet [68]. When the threshold changes to 0.25, the detection accuracy of DeepMix is still higher than the best state-of-the-art model D^{4}LCN [12] by 3.5% (on average). In this case, MLCVNet [68] performs the best for chairs and bottles. The reason is that the depth data will be missing when there is a reflection on the bottle, affecting the accuracy of DeepMix. We are exploring efficient computer graphics algorithms [29, 56] to address this problem[4]. For chairs with irregular shapes, the 3D bounding box estimation of DeepMix may not be accurate from some specific viewing angles. This issue could be alleviated in mobile scenarios (§7.4). For example, if users are interested in an object, they may move around it to inspect the details from different angles, which offers opportunities to improve detection accuracy (§5.1). DeepMix still has the most accurate results when the threshold is 0.5, 1.22× (on average) over the best existing model MLCVNet [68] (33.7% vs. 27.6%), for this challenging "chair" category. We also evaluate the detection accuracy with another widely-used metric, called mean spatial position accuracy (mSPA) [18]. The results (not shown due to the limited space) are qualitatively similar to those of 3D IoU.

7.4 Detection Accuracy in Mobile Scenarios

Since AR/MR headsets are usually used in dynamic environments, we design three mobility patterns, as shown in Figure 12, to further evaluate the accuracy of DeepMix.

• `Scenario 1`: The user moves along a 2-meter line, perpendicular to the line that connects its center and the object.

• `Scenario 2`: The user moves along a 2-meter line, away, or toward the object (2 meters between line center and object).

• `Scenario 3`: The user moves around the object in a circle with a diameter of 2 meters.

In the mobile scenarios, we cannot replay the collected RGB-D images in our dataset that were captured at fixed locations. Thus, we conduct controlled, live experiments with three moving speeds, 0.5, 1, and 2 m/s. The user always looks at the object when moving with different patterns.

[4]The scheme [23] adopted by DeepMix (§4.3) is *lightweight* and cannot handle it.

We first examine the 3D IoU results for `Scenario 3` that are presented in Table 3. For each setup with different movement patterns, moving speeds, 3D object detection methods, we run the experiments 20 times to measure detection accuracy. Due to the large parameter space (*i.e.,* 8 methods, 3 patterns, 3 speeds, and 20 times for each setup), we select 4 out 7 classes, chair, bottle, box, and bag. We only report the 3D IoU for the threshold of 0.25, since DeepMix does not achieve the most accurate detection for only the chair and bottle categories with that threshold for the static scenario (Table 2).

Table 3 demonstrates that DeepMix outperforms all existing 3D object detection models, for all four object classes, and under all three moving speeds. The reason is that when the user moves around the object, DeepMix can estimate and fine-tune the 3D bounding box from different viewing angles, significantly boosting the detection accuracy (§5.1). Another key observation from this table is that the end-to-end latency drastically affects the detection accuracy in mobile scenarios. While the best performing point-cloud-based model MLCVNet [68] achieves comparable detection accuracy as the best image-based model D^{4}LCN [12] for the static scenario (63.1% vs. 63.9% in Table 2), the detection accuracy of MLCVNet is much worse than D^{4}LCN for this mobile scenario (*e.g.,* 49.8% vs. 59.6 when the moving speed is 0.5 m/s). The worse performance of MLCVNet is mainly caused by its high end-to-end latency than D^{4}LCN (311 vs. 154 ms in Figure 11), which leads to the mismatch between objects and their bounding boxes due to accumulated tracking errors.

Table 3 shows that fast moving speeds reduce detection accuracy. For example, the accuracy is 71.2%, 61.8%, and 53.8% for the 0.5, 1, and 2 m/s moving speeds, respectively. After analyzing the captured traces of different speeds, we find that high moving speeds can result in a larger and faster change of the object in the user's viewport than low moving speeds, which reduces the detection accuracy.

We next examine 3D IoU results for `Scenario 1` in Table 4 and `Scenario 2` in Table 5, respectively. In these tables, we show results for only 0.5 m/s moving speed. For the other speeds, DeepMix still outperforms existing models. Moreover, we present results for only MLCVNet [68] (the best performing point-cloud-based method), AM3D [41], and D^{4}LCN [12]. D^{4}LCN performs better than DeepMix for boxes. One of the possible reasons is that some boxes in our collected dataset have uneven surfaces (*e.g.,* big holes, Figure 1) that affect the quality of captured depth images, which the lightweight scheme [23] in DeepMix cannot fix. This issue could potentially be addressed by leveraging computer graphics algorithms (*e.g.,* surface reconstruction [28]). The performance of AM3D is slightly better than DeepMix for the chair category for

Methods	Input	3D IoU@0.25 (`Scenario 3`: speed at 0.5, 1, and 2 m/s)				Average
		Chair	Bottle	Box	Bag	
COG [54]	PC	38.5/33.2/27.4	32.2/27.6/19.7	46.2/34.1/28.6	36.8/28.8/21.6	38.4/30.9/24.3
VoteNet [49]	PC	48.4/35.5/29.2	52.1/43.2/24.6	48.5/36.7/28.2	45.2/30.2/18.1	48.6/36.4/25.1
MLCVNet [68]	PC	55.7/42.2/28.4	53.7/43.2/33.9	45.2/30.2/19.8	44.6/25.1/17.2	49.8/35.1/24.8
F-PointNet [50]	RGB-D	32.8/21.3/17.6	35.2/25.9/21.2	25.6/14.1/8.9	41.8/32.4/19.6	33.9/23.4/16.8
Trans3D [63]	RGB-D	33.8/25.7/22.4	34.6/22.5/17.6	33.7/25.4/15.5	45.5/36.2/18.6	36.9/27.5/18.5
Mono3D [7]	RGB	59.6/49.1/41.6	45.2/37.6/23.9	66.5/51.1/36.2	41.7/34.1/27.7	53.2/42.9/32.3
AM3D [41]	RGB	64.5/54.2/47.3	59.4/47.2/32.9	63.3/54.9/46.2	61.4/53.2/42.5	62.1/52.4/42.2
D^4LCN [12]	RGB	65.3/49.2/41.7	54.8/47.3/39.2	64.7/56.8/48.9	53.6/48.6/39.5	59.6/50.5/42.3
DeepMix	Hybrid	**78.3/69.7/61.4**	**73.5/74.2/52.7**	**68.7/58.9/51.2**	**64.5/68.4/49.8**	**71.2/61.8/53.8**

Table 3: Class-wise 3D IoU@0.25 comparison of DeepMix and state-of-the-art 3D object detection models for `Scenario 3`.

Methods	Input	3D IoU@0.25: speed@0.5 m/s				Average
		Chair	Bottle	Box	Bag	
MLCVNet [68]	PC	54.2	56.1	47.1	37.5	48.7
AM3D [41]	RGB	61.7	53.6	51.7	57.2	58.2
D^4LCN [12]	RGB	59.7	51.2	**64.1**	50.6	56.4
DeepMix	Hybrid	**63.1**	**58.1**	59.4	**57.9**	**60.5**

Table 4: 3D IoU@0.25 comparison of DeepMix and state-of-the-art 3D object detection models for `Scenario 1`.

Methods	Input	3D IoU@0.25: speed@0.5 m/s				Average
		Chair	Bottle	Box	Bag	
MLCVNet [68]	PC	51.8	58.6	42.5	36.9	47.5
AM3D [41]	RGB	**67.2**	59.4	60.4	55.4	60.6
D^4LCN [12]	RGB	65.2	57.4	**68.2**	52.4	60.8
DeepMix	Hybrid	66.5	**77.5**	62.2	**58.3**	**62.1**

Table 5: 3D IoU@0.25 comparison of DeepMix and state-of-the-art 3D object detection models for `Scenario 2`.

only `Scenario 2` (67.2% vs. 66.5%). For `Scenario 1`, DeepMix still outperforms AM3D (63.1% vs. 61.7%). The comparison between Table 4 and Table 5 reveals that `Scenario 2` leads to better performance than `Scenario 1` (*e.g.,* 62.1% vs. 60.5% for DeepMix). The reason is that, similar to the case of different moving speeds, `Scenario 1` may lead to a larger and faster change of the object in the viewport than `Scenario 2`, affecting detection accuracy.

By comparing Table 3 with Tables 4 and 5, we find that the 3D object detection accuracy is better for `Scenario 3` than the other two scenarios (*e.g.,* 71.2% vs. 60.5% and 62.1% for DeepMix at 0.5 m/s moving speed.). One possible reason is that moving around the object leads to more opportunities to view it from different directions than the other two mobility patterns, which refine the detection accuracy of DeepMix (§5.1). Note that existing 3D object detection models cannot benefit from the movement of users. For example, the detection accuracy is 59.6%, 60.8%, and 56.4% for D^4LCN [12] for scenarios 3, 2, and 1, respectively. On the other hand, their accuracy may drop when users are moving around, due to their high end-to-end latency of 3D object detection.

7.5 Motion-aware Bounding Box Refinement

We conduct experiments to evaluate the performance of bounding box refinement in DeepMix when the user moves around an object. We choose two objects, a box (regular) and a chair (irregular), as the test objects and measure the 3D IoU of the estimated bounding box and the ground truth after moving a certain degree, 0°, 30°, 60°, and 90°. Figure 13 shows that when the user moves around the objects, DeepMix can gradually refine the detection accuracy. The 3D IoU increases from 0.578 to 0.822 for the chair and from 0.781 to 0.845

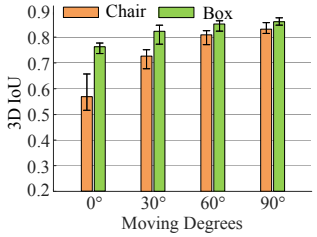

Figure 13: Performance of motion-aware bounding box refinement.

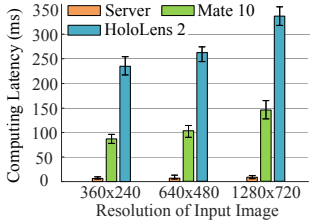

Figure 14: Computing Latency of YOLOv4-tiny on different devices.

for the box after the user moves for 90°. The reason is that DeepMix may not be able to accurately measure the dimension and orientation of irregular objects from a single estimation based on only one depth image (§4.5). However, when the user moves around the object, the inaccurate estimation of dimension and orientation can be corrected with more depth images from different viewing angles.

7.6 Executing DeepMix on Headsets without Offloading

To demonstrate the benefits of offloading 2D object detection in DeepMix to the remote server, we compare the performance of executing the same model on the server (§7.1), the HoloLens 2 headset, and a smartphone (HUAWEI Mate 10 Pro), in terms of computing latency. Since HoloLens 2 and smartphones do not support the vanilla YOLOv4 model [5], we utilize YOLOv4-tiny [1] as the object detection model for all three devices. The deep learning frameworks are TensorFlow [2] for the server, TensorFlow Lite [20] for the smartphone, and Barracuda [64] for HoloLens 2. The resolutions of the input images are 360×240, 640×480, and 1280×720.

Figure 14 shows that the computing latency of executing YOLOv4-tiny on HoloLens 2 is much higher than that on the server. The latency is 234.6ms (6.4ms), 262.1ms (8.7ms), and 337.2ms (11.5ms) on HoloLens 2 (the server) for the three resolutions. While the chipset of HoloLens 2 (Snapdragon 850) is slightly better than the HUAWEI Mate 10 Pro smartphone (Kirin 970), the computing latency of HoloLens 2 is even 147.9ms (183.8ms) higher than the smartphone when processing 360×240 (1280×720) images. The reason is that the performance of these models depends on not only the deep learning framework but also the hardware platform [57]. Our experimental results demonstrate that the Barracuda framework and HoloLens 2 cannot achieve real-time processing of input images for 2D object detection, which justifies the design decision of DeepMix that offloads it to the remote server.

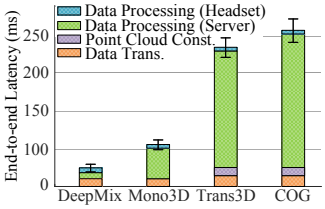

Figure 15: End-to-end latency on LTE network.

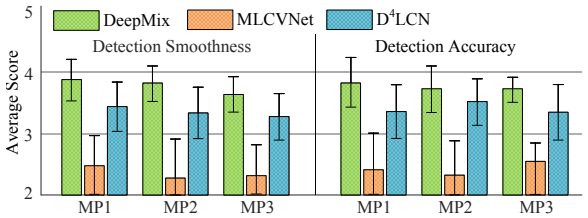

Figure 16: Evaluation of the smoothness and accuracy of 3D object detection through a user study with 33 participants.

7.7 Impact of Dynamic Network Conditions

In addition to the high-throughput WiFi network, we evaluate the performance of DeepMix on our USRP-based LTE testbed with fluctuating network bandwidth (8.4–37.1 Mbps vs. 260 Mbps for WiFi). The latency of LTE is also higher than that of WiFi (14 vs. 1 ms). The dynamic network conditions affect the end-to-end latency of DeepMix (*i.e.,* longer data transmission time and higher network latency). We compare the performance of DeepMix with three models Mono3D [7], Trans3D [63], and COG [54], which have the lowest end-to-end latency in their own categories (Figure 11).

We plot the end-to-end latency on the LTE network in Figure 15. In this figure, the latency of data transmission on the LTE network is more visible, compared to the results in Figure 11 for the WiFi network. The end-to-end latency of DeepMix increases from 34 ms to 47–62 ms (51 ms on average) and is dominated by data transmission (on average 22 ms, 43.1%), whereas the latency of Mono3D [7], which has the lowest latency among existing methods, increases from 91 ms to 105–118 ms, higher than the 100 ms latency requirement of interactive AR/MR [6, 38]. The data transmission time increases from 5 ms to 18–33 ms for RGB images, and from 8 ms to 24–39 ms for RGB-D images. The headset is static during the experiments, and thus the dynamic network conditions have a limited impact on detection accuracy. On this USRP-based LTE network, moving the headset attached with an LTE dongle makes network connection unstable. Hence, we do not conduct mobile experiments.

7.8 User Study

Besides the above evaluations on datasets and controlled experiments, we assess the performance of DeepMix through an IRB-approved user study to understand how the smoothness and accuracy of 3D object detection affect user experience. We define smoothness as the update frequency of 3D bounding boxes in dynamic environments, which reflects the end-to-end latency of object detection.

We conduct the user study with 33 diverse participants with age 18 to 45. Among them, 7 are female, 21 are familiar with AR/MR, and 3 used mobile headsets before. We ask each participant to experience three 3D object detection schemes, DeepMix, MLCVNet [68] (point-cloud-based), and D^{4}LCN [12] (image-based). The last two have better accuracy than other existing solutions (Table 2, Table 4, and Table 5). We randomly order the three schemes, and thus participants do not know which one is DeepMix. We ask the participants to compare the smoothness and accuracy of the three solutions by providing their mean opinion scores (MOS), from 1 to 5 (1: bad, 2: poor, 3: fair, 4: good, 5: excellent). Participants experience the detection of a chair and a bottle by following the three mobility patterns in Figure 12 for 30–60 seconds.

We observe the following from the results of our user study in Figure 16. First, DeepMix leads to the best user experience in terms of both smoothness and accuracy, thanks to its lightweight and accurate 3D object detection. For the average score, the smoothness of DeepMix is 44.3% (10.4%), 59.4% (16.6%), and 56.9% (11.7%) higher than that of MLCVNet [68] (D^{4}LCN [12]) for the three mobility patterns; whereas the accuracy of DeepMix is 48.2% (12.5%), 65.4% (5.1%), and 52.3% (9.6%) higher than that of MLCVNet [68] (D^{4}LCN [12]) for the three patterns. Second, the point-cloud-based model MLCVNet [68] has the worse performance among the three, mainly caused by its high end-to-end latency (due to the huge amount of 3D data to process). This demonstrates the importance of end-to-end latency for mobile AR/MR. Third, for both smoothness and accuracy, the average score of DeepMix is still slightly lower than 4 (good experience), which shows that there is room to improve for achieving immersive experience on mobile headsets.

7.9 Effectiveness of Bounding Box Caching

We conduct controlled experiments to evaluate the 3D bounding box caching and reusing scheme (§5.3). We mount two headsets on a gimbal that can rotate at a fixed angular velocity. Both headsets are equipped with DeepMix, but only one of them has caching enabled. We randomly place 3 objects around the gimbal for testing and rotate both headsets 720° at the same time with the same speed. The result shows that the time to render the bounding box of a cached item is only 2.6–3.2 ms. Without caching, DeepMix needs to execute the entire workflow, which takes at least 34 ms. Moreover, the number of offloaded frames decreases from 113 to 12 when caching is enabled and almost does not change without caching. Thus, our bounding box caching optimization can drastically improve user experience, reduce the amount of offloaded data, and decrease computation overhead on both the edge and the headset.

7.10 Power and Computation Resources

To demonstrate its lightweight feature, we finally compare the on-device battery power level and computation resource utilization of DeepMix with MLCVNet [68] (point-cloud-based), and D^{4}LCN [12] (image-based). Existing schemes in the same category have almost the same performance, because their heavy-lifting jobs are all offloaded to the edge. We generate point clouds for MLCVNet on mobile headsets to demonstrate its overhead. Otherwise, the performance of MLCVNet is close to that of D^{4}LCN. Due to the high cost of generating point clouds, after using the headset for 40 minutes, the average battery power rate level of DeepMix is 8.2 W, which is 0.3 W higher than D^{4}LCN and 1.8 W lower than MLCVNet. The average CPU (memory) usage of DeepMix is only 24.3% (11.3%), which is 2.3% (0.4%) higher than D^{4}LCN and 7.6% (6.4%) lower than MLCVNet. There is no significant difference in GPU usage among these methods. Unfortunately, we do not find a method to measure the HPU utilization of Microsoft HoloLens 2.

8 DISCUSSION AND FUTURE WORK

Image-based 3D Object Detection. As shown in Table 2, image-based 3D object detection such as D^{4}LCN [12] achieves high accuracy for the static scenario (*e.g.,* only 3.5% lower than DeepMix for 3D IoU@0.25). Its poor performance for the dynamic scenario (11.6% lower than DeepMix in Table 3 for speed@0.5 m/s) is mainly caused by the high end-to-end latency. However, it can potentially handle more use cases than DeepMix, as we will discuss next. We plan to optimize the runtime inference performance of image-based 3D object detection to make it practical.

Limitations. As the first-of-its-kind accurate 3D object detection that is suitable for mobile headsets, DeepMix has a few limitations of its current design. For example, it can detect mainly objects that are placed on a plain surface, and it is challenging to detect, for instance, a TV that is hung on a wall. Also, DeepMix could not handle the case that the shape of the object changes during the user movement, and its caching and reusing mechanism may not work for deformable objects. However, we argue that the target scenarios of DeepMix (*e.g.,* objects on a plane and indeformable objects) are the common use cases for indoor mobile AR/MR. Another issue is that the range of depth cameras is typically limited (*e.g.,* from 0.5 to 5.5 m), which our caching scheme can help only to some extent. Hence, DeepMix cannot detect objects that are far away from users. We are extending DeepMix to address these limitations. One possible solution is to design a hybrid scheme that dynamically switches between DeepMix and image-based 3D object detection [67], given that plane detection is a solved problem [4, 16, 21] and the range of RGB cameras is longer than that of depth ones.

Supporting Interactive AR/MR. The 3D object detection offered by DeepMix lays the foundation for enabling real-time, interactive AR/MR on mobile headsets. We plan to build various immersive applications by leveraging this key capability of DeepMix.

Light-weighted and Low-priced AR/MR Headsets. The current-generation of AR/MR headsets are responsible for executing computation-intensive tasks locally. In the future, with the emerging network technologies such as 5G and beyond, we expect that the majority of tasks with heavy computation can be offloaded to remote cloud/edge servers. As a result, the weight and cost of future AR/MR headsets may be greatly reduced.

9 RELATED WORK

3D Object Detection. *Point-cloud-based 3D Object Detection:* With the development of deep learning models on point clouds [51], many 3D object detection models have emerged [13, 35, 49, 51, 68, 73]. For example, approaches such as VoteNet [49], COG [54], and MLCVNet [68] can directly take raw point cloud as input. Thanks to the available depth information and the underlying DNN networks, those methods can achieve high detection accuracy. *Image-based 3D Object Detection:* Instead of processing point clouds, some existing schemes [7, 8, 12, 41] utilize 2D detectors to achieve 3D object detection. For example, D^{4}LCN [12] estimates the depth information from monocular images and fuses RGB and depth using improved 2D convolutions to generate 3D bounding boxes. *3D Object Detection with RGB-D Input:* This category utilizes both RGB images and depth data for 3D object detection [34, 50, 61, 63]. For example, F-PointNet [50] narrows down the 3D space by leveraging 2D object detection and further performs segmentation on selected 3D frustums with PointNet [51] to help estimate the 3D bounding box. Although these approaches can reduce the amount of to-be-processed 3D data, their accuracy is usually not as good as point-cloud-based schemes. Different from the above work, DeepMix benefits from 2D object detection models that have low computation latency. By utilizing real-time depth information from sensors, it can achieve *high 3D object detection accuracy with low end-to-end latency*.

Mobile AR/MR. There is a rich literature on building mobile AR/MR systems [3, 6, 10, 31, 36, 47, 69]. For example, HomeMeld [31] enables the telepresence between remote living areas through robot agents as avatars, by finding an equivalent functional place in rooms and predicting real-time paths to prevent lagging caused by the robot's slow movement. LpGL [10] is a device-independent graphics library that reduces energy consumption for mobile headset applications, which dynamically selects frame rate and object shape complexity and leverages user movements to extend the battery life. Heimdall [69] coordinates concurrent GPU usage for multi-tasking in mobile AR applications, by splitting the DNNs into small units and executing them between rendering frames. Different from the above work, DeepMix offers a real-time, accurate 3D object detection framework, which is *missing in existing mobile AR/MR systems*, to support better interaction between and seamless integration of the digital and 3D physical worlds and provide a truly immersive user experience for headset-based applications.

10 CONCLUSION

In this paper, we present the design, implementation, and evaluation of DeepMix, a mobility-aware, lightweight, and accurate 3D object detection system for improving the quality of user experience of AR/MR applications running on mobile headsets. Instead of directly leveraging/accelerating existing 3D object detection models that are computation-intensive, DeepMix benefits from mature 2D object detection algorithms to derive a bounding box for the object of interest. It then utilizes this 2D bounding box to extract depth data from depth images captured by the headset and estimates the 3D bounding box by effectively exploring 3D geometry and data processing. By doing this, DeepMix not only reduces the end-to-end latency of AR/MR applications but also drastically increases the detection accuracy in dynamic environments, by exploiting the mobility of headsets. We implement DeepMix on a commodity mobile headset and compare its performance with several state-of-the-art 3D object detection models. Our extensive experiments, including a user study, demonstrate the efficacy of DeepMix in terms of both end-to-end latency and detection accuracy.

ACKNOWLEDGMENTS

We thank the anonymous reviewers and our shepherd Mahadev Satyanarayanan for their insightful comments. The research of Tao Han, Yongjie Guan and Xueyu Hou is partially supported by the US National Science Foundation under Grant No. 2147821, No. 2147623, No. 2047655, and No. 2049875. The research of Bo Han and Nan Wu was funded in part by 4-VA, a collaborative partnership for advancing the Commonwealth of Virginia.

REFERENCES

[1] YOLOv4-Tiny. https://github.com/HirataYurina/yolov4-tiny-keras.

[2] TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[3] K. Apicharttrisorn, X. Ran, J. Chen, S. V. Krishnamurthy, and A. K. Roy-Chowdhury. Frugal Following: Power Thrifty Object Detection and Tracking for Mobile Augmented Reality. In *Proceedings of ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2019.

[4] Apple Inc. ARKit (initial release on June 2017). https://developer.apple.com/arkit/, 2017.

[5] A. Bochkovskiy, C. Wang, and H. M. Liao. YOLOv4: Optimal Speed and Accuracy of Object Detection. *CoRR*, abs/2004.10934, 2020.

[6] K. Chen, T. Li, H.-S. Kim, D. E. Culler, and R. H. Katz. MARVEL: Enabling Mobile Augmented Reality with Low Energy and Low Latency. In *Proceedings of ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2018.

[7] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun. Monocular 3D Object Detection for Autonomous Driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[8] X. Chen, K. Kundu, Y. Zhu, H. Ma, S. Fidler, and R. Urtasun. 3D Object Proposals Using Stereo Imagery for Accurate Object Class Detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(5):1259–1272, 2018.

[9] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-View 3D Object Detection Network for Autonomous Driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[10] J. Choi, H. Park, J. Paek, R. K. Balan, and J. Ko. LpGL: Low-power Graphics Library for Mobile AR Headsets. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2019.

[11] Cytron. LM35 temperature sensor. https://tutorial.cytron.io/2017/07/13/getting-started-temperature-sensor-celsius-sn-lm35dz/.

[12] M. Ding, Y. Huo, H. Yi, Z. Wang, J. Shi, Z. Lu, and P. Luo. Learning Depth-Guided Convolutions for Monocular 3D Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[13] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner. Vote3deep: Fast Object Detection in 3D Point Clouds Using Efficient Convolutional Neural Networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[14] Y. Fang, R. Nakashima, K. Matsumiya, I. Kuriki, and S. Shioiri. Eye-Head Coordination for Visual Cognitive Processing. *PloS one*, 10(3):e0121035, 2015.

[15] Y. Feng, B. Tian, T. Xu, P. Whatmough, and Y. Zhu. Mesorasi: Architecture Support for Point Cloud Analytics via Delayed-Aggregation. In *Proceedings of IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2020.

[16] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[17] FLIR. FLIR E8-XT Infrared Camera with Extended Temperature Range. https://www.flir.com/products/e8-xt/.

[18] A. Frisoli, M. Solazzi, D. Pellegrinetti, and M. Bergamasco. A New Screw Theory Method for the Estimation of Position Accuracy in Spatial Parallel Manipulators with Revolute Joint Clearances. *Mechanism and Machine Theory*, 46(12):1929–1949, 2011.

[19] R. B. Girshick. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[20] Google. TensorFlow Lite. https://www.tensorflow.org/lite/api_docs.

[21] Google. ARCore (initial release on March 2018). https://developers.google.com/ar/, 2018.

[22] J. Grubert, Y. Itoh, K. Moser, and J. E. Swan. A Survey of Calibration Methods for Optical See-Through Head-Mounted Displays. *IEEE Transactions on Visualization and Computer Graphics*, 24(9):2649–2662, Sep. 2018.

[23] A. Grunnet-Jepsen and D. Tong. Depth Post-Processing for Intel Realsense™ D400 Depth Cameras. *New Technologies Group, Intel Corporation*, 2018.

[24] Y. Guan, X. Hou, T. Han, and S. Zhang. Deepmix: A real-time adaptive virtual content registration system with intelligent detection. In *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–2, 2021.

[25] X. Hou, Y. Guan, T. Han, and N. Zhang. Distredge: Speeding up convolutional neural network inference on distributed edge devices. In *36th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2022.

[26] X. Hou and T. Han. Trustserving: A quality inspection sampling approach for remote dnn services. In *2020 17th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–9, 2020.

[27] Intel. Intel RealSense D435. https://www.intelrealsense.com/depth-camera-d435/.

[28] S. Izadi, R. A. Newcombe, D. Kim, O. Hilliges, D. Molyneaux, S. Hodges, P. Kohli, J. Shotton, A. J. Davison, and A. W. Fitzgibbon. KinectFusion: Real-Time Dynamic 3D Surface Reconstruction and Interaction. In *Proceedings of International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 2011.

[29] Y. Ji, Q. Xia, and Z. Zhang. Fusing Depth and Silhouette for Scanning Transparent Object with RGB-D Sensor. *International Journal of Optics*, 2017:1–11, 2017.

[30] P. Kainiemi and I. Salento. kinect-bits (including "Simple Background Removal and ROI Estimation" and "Floor Determination and Removal"). https://github.com/kainiemi/kinect-bits/.

[31] B. Kang, I. Hwang, J. Lee, S. Lee, T. Lee, Y. Chang, and M. K. Lee. My Being to Your Place, Your Being to My Place: Co-present Robotic Avatars Create Illusion of Living Together. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2018.

[32] Keithley. Keithley Series 2281S Battery Simulator. https://www.tek.com/tektronix-and-keithley-dc-power-supplies/2281s.

[33] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander. Joint 3D Proposal Generation and Object Detection from View Aggregation. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[34] J. Lahoud and B. Ghanem. 2D-Driven 3D Object Detection in RGB-D Images. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[35] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. PointPillars: Fast Encoders for Object Detection From Point Clouds. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[36] Z. Li, M. Annett, K. Hinckley, K. Singh, and D. Wigdor. HoloDoc: Enabling Mixed Reality Workspaces that Harness Physical and Digital Content. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019.

[37] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common Objects in Context. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2014.

[38] L. Liu, H. Li, and M. Gruteser. Edge Assisted Real-time Object Detection for Mobile Augmented Reality. In *Proceedings of the 25th Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2019.

[39] L. Liu, J. Lu, C. Xu, Q. Tian, and J. Zhou. Deep Fitting Degree Scoring Network for Monocular 3D Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[40] G. Lukács, R. Martin, and D. Marshall. Faithful Least-Squares Fitting of Spheres, Cylinders, Cones and Tori for Reliable Segmentation. In *Proceedings of European Conference on Computer Vision (ECCV)*, 1998.

[41] X. Ma, Z. Wang, H. Li, P. Zhang, W. Ouyang, and X. Fan. Accurate Monocular 3D Object Detection via Color-Embedded 3D Reconstruction for Autonomous Driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[42] Magic Leap Inc. Magic Leap One. https://www.magicleap.com/magic-leap-one.

[43] Microsoft Corporation. DirectX. https://docs.microsoft.com/en-us/windows/win32/directx-sdk--august-2009-.

[44] Microsoft Corporation. Microsoft HoloLens 2. https://www.microsoft.com/en-us/hololens.

[45] Microsoft Corporation. Mixed Reality Documentation. https://docs.microsoft.com/en-us/windows/mixed-reality.

[46] Microsoft Corporation. Windows SDK. https://docs.microsoft.com/en-us/windows/win32/api/.

[47] T. Park, M. Zhang, and Y. Lee. When Mixed Reality Meets Internet of Things: Toward the Realization of Ubiquitous Mixed Reality. *GetMobile: Mobile Computing and Communications*, 22(1):10–14, 2018.

[48] C. R. Qi, X. Chen, O. Litany, and L. J. Guibas. ImVoteNet: Boosting 3D Object Detection in Point Clouds With Image Votes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[49] C. R. Qi, O. Litany, K. He, and L. J. Guibas. Deep Hough Voting for 3D Object Detection in Point Clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[50] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum PointNets for 3D Object Detection from RGB-D Data. *arXiv preprint arXiv:1711.08488*, 2017.

[51] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[52] J. Redmon. Darknet: Open Source Neural Networks in C. http://pjreddie.com/darknet/, 2013–2016.

[53] J. Redmon and A. Farhadi. YOLOv3: An Incremental Improvement. *ArXiv*, abs/1804.02767, 2018.

[54] Z. Ren and E. B. Sudderth. Three-Dimensional Object Detection and Layout Prediction Using Clouds of Oriented Gradients. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[55] D. Roetenberg, H. Luinge, and P. Slycke. Xsens MVN: Full 6DOF Human Motion Tracking Using Miniature Inertial Sensors. *Xsens Motion Technologies BV, Tech. Rep*, 1, 2009.

[56] S. Sajjan, M. Moore, M. Pan, G. Nagaraja, J. Lee, A. Zeng, and S. Song. ClearGrasp: 3D Shape Estimation of Transparent Objects for Manipulation. In *Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[57] S. Shams, R. Platania, K. Lee, and S.-J. Park. Evaluation of deep learning frameworks over different hpc architectures. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 1389–1396, 2017.

[58] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li. PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[59] S. Shi, X. Wang, and H. Li. PointRCNN: 3D Object Proposal Generation and Detection From Point Cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[60] S. Song, S. P. Lichtenberg, and J. Xiao. SUN RGB-D: A RGB-D Scene Understanding Benchmark Suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[61] S. Song and J. Xiao. Deep Sliding Shapes for Amodal 3D Object Detection in RGB-D Images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[62] K. H. Strobl and G. Hirzinger. More Accurate Pinhole Camera Calibration with Imperfect Planar Target. In *Proceedings of IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, 2011.

[63] Y. S. Tang and G. H. Lee. Transferable Semi-Supervised 3D Object Detection From RGB-D Data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[64] Unity. Barracuda. https://docs.unity3d.com/Packages/com.unity.barracuda@1.0/manual/index.html.

[65] Unity Technologies. Unity Real-Time Development Platform. https://unity3d.com/.

[66] H. Wang, Y. Cong, O. Litany, Y. Gao, and L. J. Guibas. 3DIoUMatch: Leveraging IoU Prediction for Semi-Supervised 3D Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[67] N. Wu, F. X. Lin, F. Qian, and B. Han. Hybrid Mobile Vision for Emerging Applications. In *Proceedings of the 23nd ACM Workshop on Mobile Computing Systems and Applications (HotMobile)*, 2022.

[68] Q. Xie, Y. Lai, J. Wu, Z. Wang, Y. Z. andvote Kai Xu, and J. Wang. MLCVNet: Multi-Level Context VoteNet for 3D Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[69] J. Yi and Y. Lee. Heimdall: Mobile GPU Coordination Platform for Augmented Reality Applications. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2020.

[70] H. Zhang, B. Han, C. Y. Ip, and P. Mohapatra. Slimmer: Accelerating 3D Semantic Segmentation for Mobile Augmented Reality. In *Proceedings of IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, 2020.

[71] H. Zhang, B. Han, and P. Mohapatra. Toward Mobile 3D Vision. In *Proceedings of IEEE International Conference on Computer Communications and Networks (ICCCN)*, 2020.

[72] W. Zhang, B. Han, and P. Hui. Jaguar: Low Latency Mobile Augmented Reality with Flexible Tracking. In *Proceedings of the 26th ACM international conference on Multimedia (MM)*, 2018.

[73] Y. Zhou and O. Tuzel. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.