SN Computer Science (2023) 4:543
https://doi.org/10.1007/s42979-023-01881-3

ORIGINAL RESEARCH

Real-Time 3D Object Detection, Recognition and Presentation Using a Mobile Device for Assistive Navigation

Jin Chen[1,2] · Zhigang Zhu[1,3]

Received: 28 September 2022 / Accepted: 1 May 2023

Abstract

This paper presents an integrated solution for 3D object detection, recognition, and presentation to increase accessibility for various user groups in indoor areas through a mobile application. The system has three major components: a 3D object detection module, an object tracking and update module, and a voice and AR-enhanced interface. The 3D object detection module consists of pre-trained 2D object detectors and 3D bounding box estimation methods to detect the 3D poses and sizes of the objects in each camera frame. This module can easily adapt to various 2D object detectors (e.g., YOLO, SSD, mask RCNN) based on the requested task and requirements of the run time and details for the 3D detection result. It can run on a cloud server or mobile application. The object tracking and update module minimizes the computational power for long-term environment scanning by converting 2D tracking results into 3D results. The voice and AR-enhanced interface integrates ARKit and SiriKit to provide voice interaction and AR visualization to improve information delivery for different user groups. The system can be integrated with existing applications, especially assistive navigation, to increase travel safety for people who are blind or have low vision and improve social interaction for individuals with autism spectrum disorder. In addition, it can potentially be used for 3D reconstruction of the environment for other applications. Our preliminary test results for the object detection evaluation and real-time system performance are provided to validate the proposed system.

Keywords 3D object detection · Assistive technology · ARKit · Blind or low vision · Voice assistance

Introduction

Background and Problem Statements

Visual impairment refers to the loss of vision, whether partial or complete, that cannot be effectively corrected with glasses, contact lenses, medication, or surgery. The number of individuals who are blind or have low vision (BLV) has been steadily increasing over the past three decades, and is expected to continue to rise as the population ages [1]. According to the IAPB Vision Atlas [2], in 2020, there were 43 million people who were blind and 295 million people with moderate or severe visual impairment (i.e., with visual acuity worse than 6/12 to 6/18) globally.

Individuals who have BLV encounter a multitude of challenges in their daily lives, from navigating indoor spaces to completing tasks. These difficulties are greatly amplified in unfamiliar environments where they only have limited assistance tools, such as long canes, guide dogs, hand touches, and potential help from others. Familiarizing themselves with their surroundings is not only time-consuming but also raises safety concerns, particularly in indoor environments that are constantly changing. The limited space for movement in indoor areas makes it easy for individuals with BLV to bump into obstacles. According to a survey of 300 legally blind individuals [3], over 40% of them experienced accidents at head level at least once a month, even with the help

This article is part of the topical collection "Advances on Image Processing and Vision Engineering" guest edited by Sebastiano Battiato, Francisco Imai and Cosimo Distante.

✉ Jin Chen
jin@nearabl.com

Zhigang Zhu
zzhu@ccny.cuny.edu

1 Visual Computing Laboratory, Computer Science Department, The City College-CUNY, New York, NY 10031, USA

2 Nearabl Inc., New York 10023, NY, USA

3 PhD Program in Computer Science, The Graduate Center-CUNY, New York, NY 10016, USA

Published online: 29 July 2023

of a long cane or guide dog. In addition to safety concerns, the ability to recognize people and objects would significantly improve their understanding of their surroundings, leading to better interaction with others and a higher quality of life.

Studies indicate that individuals with autism spectrum disorder (ASD) have difficulty adjusting their visual attention, particularly when it comes to dynamic moving objects or shifting their attention [4, 5]. Recognizing objects can aid people with ASD in their cognitive development and enhance their visual attention in complex environments [4, 6]. Additionally, object recognition using AR visualization can assist individuals with ASD or low vision in performing tasks, such as providing instructions or information on corresponding objects and helping them locate target objects.

Object detection is a fundamental topic in computer vision, with many techniques developed and applied in real-world applications, especially in the fields of autonomous vehicles and robotics. While real-time 2D object detection techniques are well-established, they do not provide adequate information (e.g., location and size of an object) that is crucial for individuals to obtain a more complete understanding of their environment. On the other hand, real-time 3D object detection and recognition often require expensive high-end devices (e.g., LiDAR sensors and depth cameras) and deep learning models that demand significant computational power to ensure real-time performance. These methods are not suitable for indoor environments and are not affordable for individuals with BLV or ASD, who typically have lower incomes. For instance, data shows that 89% of visually impaired individuals reside in low- or middle-income countries [7].

Choosing relevant information to report to users is a significant challenge when multiple obstacles and objects are detected, particularly with real-time detection, where information constantly evolves over time. This can be especially challenging for individuals with BLV or ASD who may become overwhelmed and distracted by the abundance of data, potentially causing them to miss critical information.

Overview of the Solution

This paper presents a real-time 3D object detection, recognition, and presentation system that is both low-cost and efficient, designed to enhance accessibility for people with BLV or ASD, as well as others who require assistance in indoor spaces. The system is designed to operate on a mobile device, such as an iPhone or iPad, and can identify a variety of *objects*, including *obstacles* along a user's path, *items* the user is searching for, or *persons/humans* who may be moving around. Therefore object detection/recognition includes obstacle detection, important item detection/recognition, and human detection/recognition.

This system is an extension and continuation of the earlier work reported in our IMPROVE conference paper [8]. The early system is expanded to the current system to (1) accommodate different data types from various mobile devices (e.g., single images, RGB-D images, and 3D point clouds); (2) allows for the use of customized object detection models, depending on the desired objects to be detected, levels of 3D detection details, and run-time requirements as well as specific tasks at hand (i.e., object search or obstacle detection); and (3) add a voice and AR-enhanced user interface to facilitate communication with users. Importantly, this system can be integrated with other systems, such as our previously developed indoor navigation system [9], to improve travel safety for people with travel challenges.

The key components and features of the current system are:

- A portable system for real-time 3D object detection, recognition, and presentation that works on iOS-based mobile devices, integrating ARKit, SiriKit, object detection cloud server, and object tracking. No additional hardware is needed from the users.
- An object detection server with an adaptive 3D object detection and tracking method based on input data types and request tasks. The 3D detection and tracking method can work with any 2D object detectors.
- A voice and AR-enhanced interface that improves information delivery to people with BLV or other disabilities. This will lead to hand-free operations for the users.
- A data collection and annotation toolkit for constructing a mobile device based object detection dataset, which can be used for 3D reconstruction when integrated with the previously developed modeling app [9]. This will greatly facilitate modeling and testing by system developers.

The structure of this paper is as follows. "Related Work" provides a brief overview of the state-of-the-art 2D and 3D object detection methods. In "Method", we present the proposed 3D object detection, recognition, and presentation system. "Experiments" contains the system requirements and experimental results of the proposed method. Finally, we conclude the paper by discussing the proposed system and future work in "Conclusions and Discussions".

Related Work

2D Object Detection

2D object detection predicts the class labels and 2D bounding boxes of objects in an input image. There are two main types of object detectors: one-stage detectors and two-stage detectors. Two-stage object detectors first use a regional

proposal network (RPN) to extract the region of interest (ROI) for each object in the image and then refine the 2D bounding boxes of the objects using regression while classifying them into different classes. Examples of two-stage object detectors include R-CNN families, such as Fast R-CNN [10] and Mask R-CNN [11]. On the other hand, one-stage object detectors (e.g., YOLO families [12–14], SSD [15], and RetinaNet [16]) directly detect the 2D bounding boxes of objects from the image with dense sampling of areas, without performing ROI extraction. Although two-stage detectors generally offer higher accuracy, they usually require a longer processing time compared to one-stage detectors. In contrast, one-stage detectors are faster but may have lower accuracy.

2D object detection techniques are sufficiently mature to support real-time performance. However, 2D bounding boxes with object labels do not provide sufficient information (e.g., location and size of each object in the 3D space) for people with BLV to gain a comprehensive understanding of their indoor environment and provide them the confidence to walk around and interact with others. Nevertheless, 3D object detection can be built on top of 2D detectors.

3D Object Detection

3D object detection is typically performed by estimating the 3D bounding boxes of objects in a scene, which provides more detailed spatial information compared to 2D object detection techniques that use only RGB images. To achieve this, 3D object detection methods typically incorporate additional data sources such as depth images, stereo images, or point cloud data in order to obtain the necessary 3D information.

Image Based 3D Object Detection

Several studies have applied variations of region segmentation algorithms for obstacle detection in depth images [17, 18]. Huang et al. [17] utilized morphological closing (dilation and erosion) to remove depth map noise and estimated the ground curves and height thresholds for obstacles using the standard least-squares method and V-disparity method [19]. Stair edges were determined by the difference in the depth values, and a connected component region growth method was applied to distinguish different objects. Cheng et al. [18] used a seeded region growth method with Sobel edges to detect ground and obstacles at a refreshing frequency of 10 frames per second (fps). In addition to the region growth method, a deep learning model with semantic segmentation was applied for obstacle detection in RGB-D images [20]. This approach achieved real-time performance at 22Hz with an Nvidia GTX2080Ti GPU.

Obstacle detection using stereo images has also been explored in research. Chen [21] introduced the deep stereo geometry network (DSGN) that leverages 2D image features at both pixel and semantic levels to construct a plane-sweep volume (PSV) with depth estimation. The PSV is then transformed into 3D space to construct a 3D geometric volume, which is subsequently used to detect objects with a 3D neural network. Chen's method achieved an average processing time of 0.682 s and an average depth error of 0.55 ms when detecting objects in an image pair with an NVIDIA Tesla V100.

Although these methods work in many scenarios, different lighting conditions can affect the estimated depth values and decrease the accuracy of object locations. Additionally, they are unsuitable for real-time systems combined with navigation due to longer computation times for complex scenes and difficulties in detecting smaller objects.

Point Cloud Based 3D Object Detection

Object detection using point cloud data is more popular than depth or stereo images as it contains more information and has been widely used for scene reconstruction. Pham et al. [22] utilized color-depth images along with accelerometer data to reconstruct the point cloud of a scene. They then applied voxelization and a pass-through filter to remove noise. The random sample consensus (RANSAC) algorithm [23] was employed to segment planes and detect the ground plane of the scene. Different algorithms were used to detect doors, stairs, walls, and other loose obstacles after the ground plane was removed. Domenech et al. [24] used a voxel grid clustering approach to cluster the point cloud. They then computed the fractal dimension for each cluster and combined them to form a voxelized fractal descriptor. The fractal dimension was obtained from the slope of the decrease in the box size value over the number of iterations. The support vector machine was used with the voxelized fractal descriptor to classify detected objects, achieving 92.84% accuracy on ModelNet10 [25].

LiDAR (light detection and ranging) utilizes near-infrared light to create a 3D map of the environment. Due to its low wavelength of light signals (in the nanometer range), it can detect smaller objects and provides a more accurate and precise point cloud than most sensors, making it increasingly popular for object detection [26, 27]. He et al. [26] used the American Velodyne-16 line LiDAR to generate 300K points per second. As the point cloud becomes sparser with increasing distance, they utilized a pass-through filter and voxel mesh method to filter out noise data. Similar to [22], the RANSAC method was used for plane segmentation, followed by the use of K-D trees to cluster the remaining points and group them based on thresholding. Garnett et al. [27] proposed a unified deep convolutional network with LiDAR

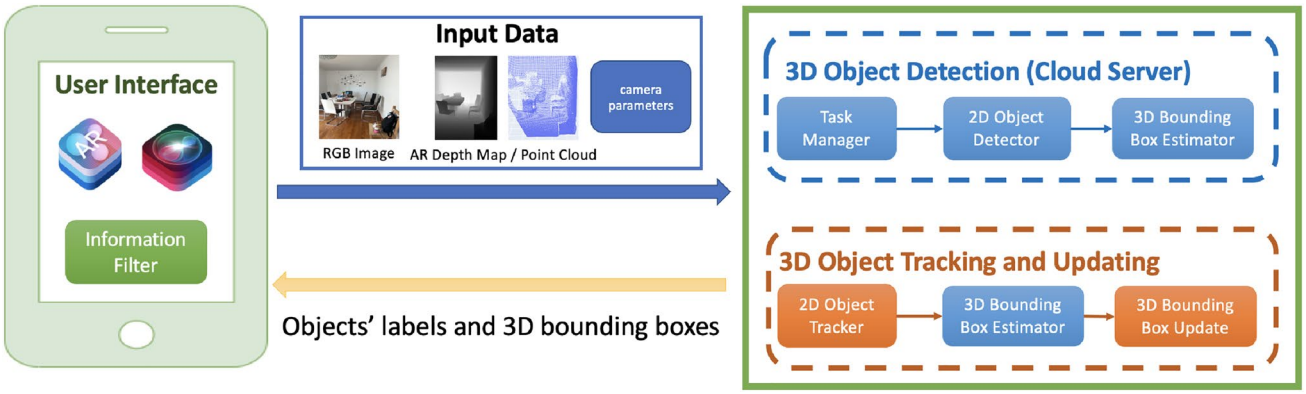

Fig. 1 The system architecture of the 3D object detection, recognition, and presentation system works on mobile devices

point cloud for real-time (30 fps) categorical-based and general obstacle detection in outdoor environments. For general obstacle detection, they used a column-based approach with StixelNet [28] as the base, where the ground truth was determined by the bottom contour of each obstacle and the clear columns detected. However, these solutions are primarily designed for autonomous driving and are not readily applicable for indoor environments.

ARKit is Apple's augmented reality (AR) platform [29], which utilizes visual-inertial odometry (VIO) to comprehend the relationship between an iOS device and the surrounding environment and track real-world objects. By analyzing captured 2D video frames and utilizing motion sensors to determine the camera's motion, ARKit detects and tracks visual feature points (i.e., distinctive markers) and estimates their 3D positions in the world coordinate system. Additionally, recent versions of iPhones now include LiDAR sensors, which allow for obtaining a more precise point cloud and enhancing detection accuracy. Our proposed 3D object detection system will leverage ARKit's capabilities and utilize ARKit data as its foundation.

Method

The 3D object detection, recognition, and presentation system detects and tracks the objects and people in the user's camera views and determines their 3D poses and sizes in real time, and then provides voice interaction and AR visualization to enhance the users' understanding of the surroundings based on the requested task.

The system consists of the following three modules (Fig. 1):

3D object detection: The 3D object detection module utilizes an adaptive method based on the required task at hand and the available data (i.e., RGB image, AR depth map or point cloud, camera intrinsic and extrinsic parameters) collected from mobile devices to determine the 2D/3D bounding boxes of the objects and/or people in each camera frame. This module operates mostly on the cloud server but can also run on users' mobile devices in regions with limited network connectivity.

3D object tracking and updating: The 3D object tracking and updating module tracks the 2D bounding boxes of the detected objects in consecutive camera frames and obtains new estimated 3D bounding boxes through communication with the 3D object detection module. Next, it updates the 3D bounding boxes of objects based on the collected data types.

Voice and AR-enhanced user interface: A mobile application is designed to allow users to search for objects or explore their surroundings using voice interaction. The application offers voice guidance and AR visualization to educate users based on the search or filtered detection outcomes.

3D Object Detection

The iOS-based ARKit platform utilizes a powerful technique called world tracking[1] to track the position and motion of the device through scene analysis across camera frames and motion sensing data comparison. Consequently, it can detect the horizontal and vertical planes in the scene and retrieve the AR point cloud—a set of notable 2D/3D feature points of the corresponding camera frames. Additionally, recent iOS devices[2] with LiDAR sensors can capture 3D data of nearby scenes and generate a depth map of each corresponding camera frame, with a resolution of 256×192.

[1] https://developer.apple.com/documentation/arkit/arframe/2887449-rawfeaturepoints.

[2] https://www.cubi.casa/support/hardware/list-of-supported-lidar-and-tof-devices.

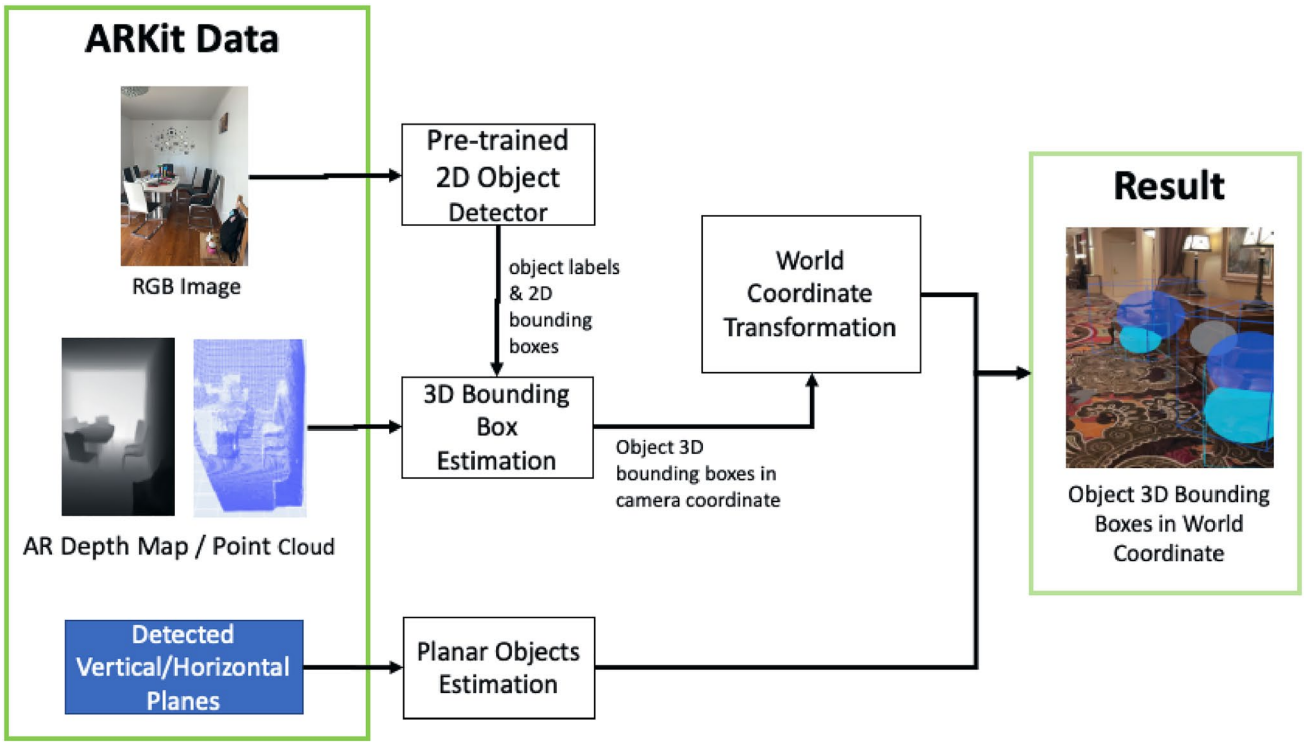

Fig. 2 The general workflow of the 3D object detection and 3D bounding box estimation

As ARKit has the capability to map real-world scenes, we utilize the data generated by ARKit in our 3D object detection module. Our proposed object detection method (Fig. 2) combines 2D bounding boxes and labels estimated by a pre-trained 2D object detector from each input RGB image with a point cloud or depth map collected by ARKit. Additionally, we use the horizontal and vertical planes detected by ARKit to improve the object detection results.

In a typical situation of assistive navigation, the mobile application needs to obtain the device's real-time location to provide navigation instructions. Processing 3D object detection within the same mobile device can cause battery drainage due to the significant computational power required to support multiple functionalities simultaneously. This also limits the flexibility to customize object detection methods based on different situations or task requests. To address these issues, we have created a cloud server to perform object detection in the cloud. However, since communication with the cloud server requires a stable network connection to ensure real-time performance, the general object detection process is also available in the mobile application as a backup for areas with poor network connections.

Planar Object Detection

ARKit can detect both vertical and horizontal planes by collecting 3D feature points over time, which can be classified into various classes such as floor, wall, ceiling, and surfaces of rectangular objects such as tables. In our system, we consider any detected horizontal plane higher than the floor plane as an obstacle, as well as any vertical planes that could be walls or doors. However, the lack of length or width information makes it impossible for our system to determine 3D bounding boxes of detected vertical planes. Therefore, the system tracks the direction and distance between the user's camera location and the vertical plane instead.

To determine the 3D bounding box of an object on a horizontal plane (such as the table with the blue plane in Fig. 3), our system uses four endpoints (P1, P2, P3, and P4) of the plane and the y location (Fy) of the detected floor plane (the green plane in Fig. 3), which in the direction of gravity. The width and length of the object are computed as the distances between P1 and P2, and P2 and P3, respectively, ignoring the y values. The height of the object is calculated as the difference between the y location of the horizontal plane (Hy) and that of the floor plane (Fy). The width and length of the object are updated along with the updated size of the horizontal plane over time, based on the updated endpoints P1, P2, P3, and P4.

3D Bounding Box Estimation

Plane detection can only identify objects with large and flat surfaces. Therefore, we employed a pre-trained image object

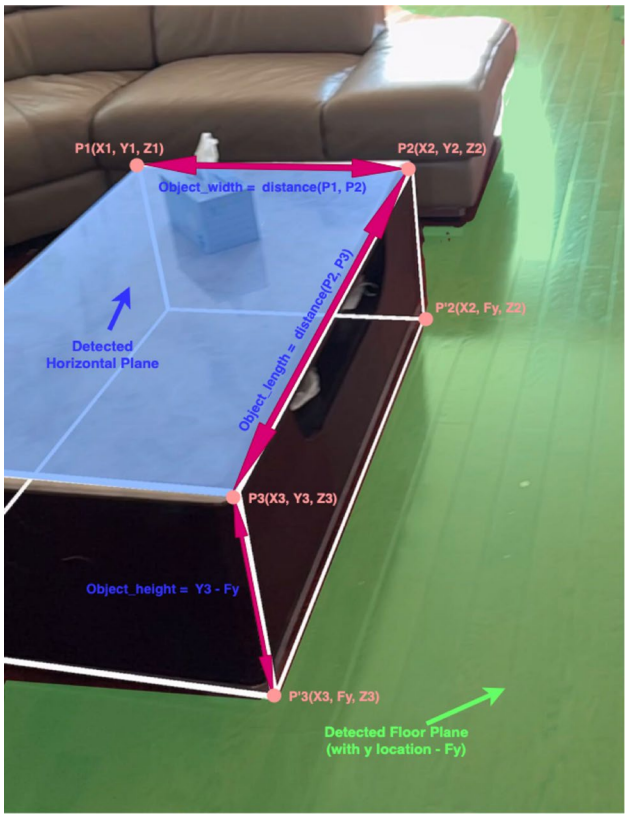

Fig. 3 Estimated 3D bounding box (white rectangular box) of an object with the detected horizontal plane (blue) using the floor plane's (green) y-location (Fy)

detection model to detect other objects that we could further classify as obstacles, humans, or targets. We then used our proposed method to determine the 3D bounding boxes of these objects. This paper tested several object detection models, including YOLOv3 [12] and YOLOv5 [13]. However, the system can be adapted to any pre-trained 2D object detector based on demand.

The proposed 3D bounding box estimation method used the point cloud or depth map collected by ARKit with the 2D bounding boxes and labels estimated by a pre-trained 2D object detector to determine the 3D bounding box of objects in the corresponding image frame.

Point cloud based 3D bounding box estimation: For mobile devices without LiDAR sensors, the system estimates 3D bounding boxes of objects using the corresponding frame's AR point cloud. Typically, each camera frame detects between 0 and 200 3D feature points. Due to the limited number of feature points, we only processed frame that contained the number of feature points above a threshold (i.e., 30) after removing the points belonging to the detected planar objects.

To convert the 2D bounding boxes of detected objects into 3D bounding boxes, first, the 3D raw feature points are projected into the 2D image coordinate system (Fig. 4). Second, the projected feature points are grouped based on the 2D bounding boxes of the detected objects (Fig. 4a, b). If the object group has a number of feature points below a certain threshold, the 3D bounding box for that object will not be computed for the current frame. For example, one

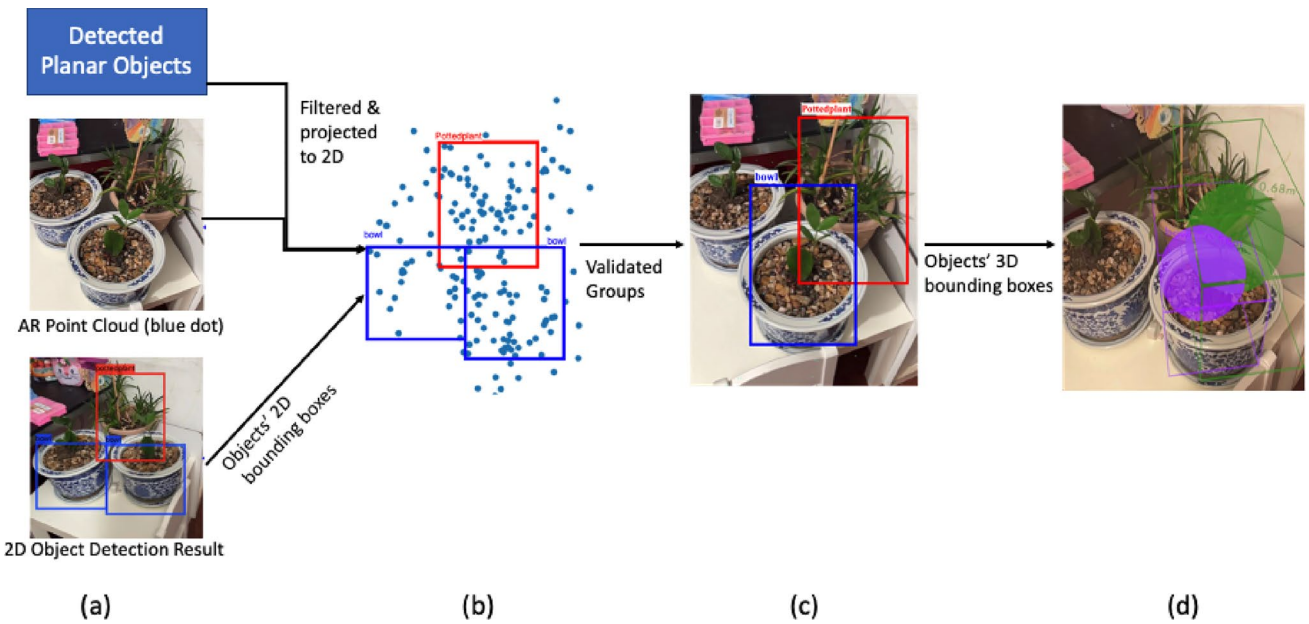

Fig. 4 3D bounding box estimation using the AR point cloud and 2D bounding boxes from 2D object detectors

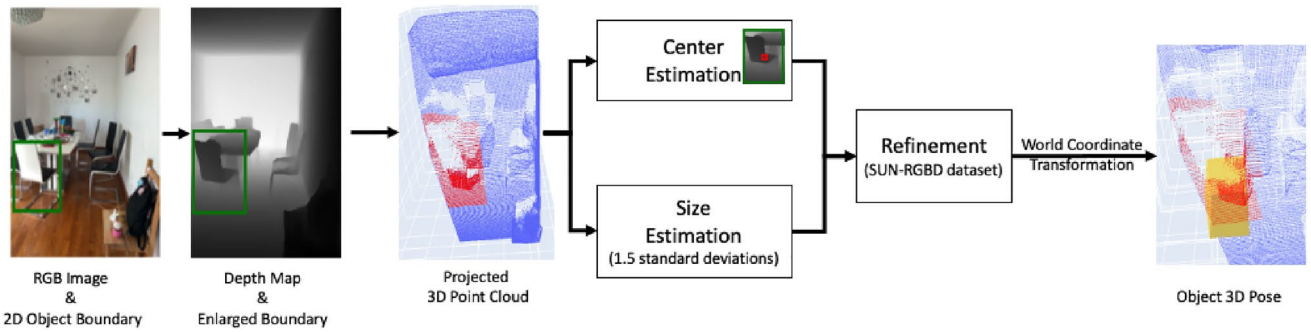

Fig. 5 3D bounding box estimation using the AR depth map and 2D bounding boxes from 2D object detectors

of the bowls is ignored in this step from Fig. 4b, c due to insufficient feature points. Next, the 3D bounding box of the object is calculated using the mean and standard deviation of the x, y, and z values of the corresponding feature point group (Fig. 4c, d). The resulting 3D bounding box and its label are then processed by the tracking and matching method to either update its 3D bounding box or consider it a new object. More details are discussed in "Point Cloud Based Tracking".

Depth map based 3D bounding box estimation: The general pipeline for object detection using a depth map is shown in Fig. 5. First, start with the 2D object bounding box detected by the 2D object detector from the RGB image, then convert the 2D bounding box into a depth map scale and enlarges by 10% of the detected 2D bounding box, as the 2D bounding box might not fully cover the object. Next, project the depth map onto the 3D point cloud using the camera's intrinsic parameters, where the points of the detected object are denoted in red (Fig. 5). Subsequently, we estimate the object's 3D size and center location based on the object's point cloud. A 5×5 mask was applied to the center of its 2D bounding box in the depth map to determine the depth of the object center, where the object's center x and y values are the average values in the object's point cloud. The object's 3D size is 1.5 standard deviations of the x, y, and z values of its 3D point cloud. After that, we refine the 3D size of the object with the average object's 3D size computed from the SUN RGB-D dataset [30] (the object category comes from the 2D object detector). The estimated 3D size is sorted by the average size of the precomputed object, and the length of each object side is updated if it is not within the deviation threshold of the corresponding average length. Finally, we convert the object's pose from the camera coordinates into world coordinates using the camera's extrinsic parameters.

Adaptive Detection in Various Environments

When operating in a noisy environment, a large number of objects could be detected, which can affect the system's performance in processing and identifying relevant information for users. For object search tasks, the presence of other objects does not have a significant impact, since the system can focus on the objects of interest with object recognition. For obstacle detection tasks, the goal is to help users avoid hitting obstacles. Therefore, the system only focuses on objects that are within a specific distance and exceed a certain size threshold. These thresholds are adaptive based on the number of distinct objects detected over time. User actions can also create noise in object detection. For instance, frequent changes in camera orientation can result in objects being repeatedly detected and can make it difficult to determine the user's orientation and the position of objects with respect to their pose. To mitigate this issue, the system monitors changes in the user's camera orientation and alerts the user when unstable movement is detected. Additionally, the system will decrease the frequency of information updates and resume them when the camera movement stabilizes.

In different environments, particularly in a workspace, such as a shopping mall or an industrial setting where a user with BLV works, specific objects are present, and it is crucial to identify those specific objects to follow work instructions accurately. A general object detection model is insufficient in such cases and may result in reduced detection accuracy when attempting to handle numerous objects with a single model. To address this issue, the system includes customized object detection models that correspond to specific objects or locations, in addition to a general object detection model. If a responsive model for the target object is available, it will be utilized for object search tasks. The system also continuously monitors the user's location using the GPS sensor of the mobile device. If the location encompasses several rooms, each having its own detection model, the system also detect the BLE beacon signal, which also used in our indoor navigation system [9], to identify the appropriate detection model to use. If there are no customized models that correspond to

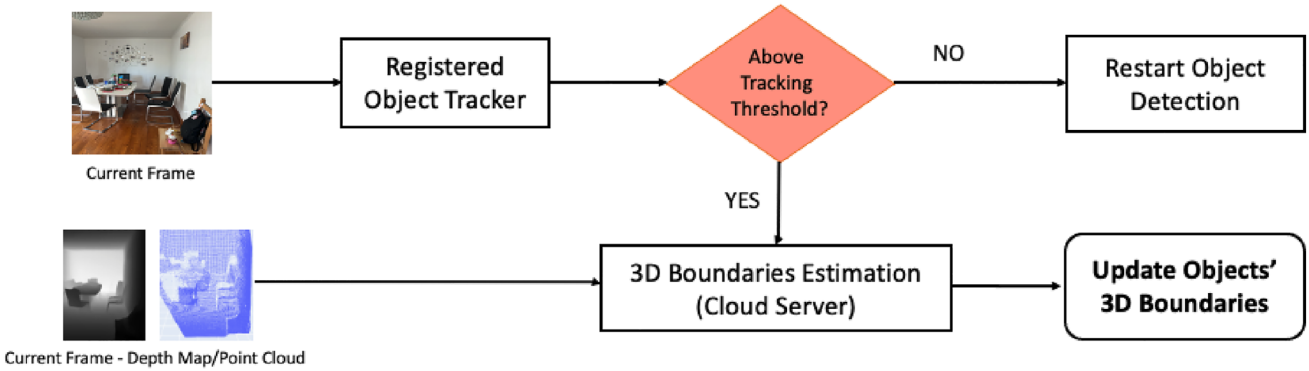

Fig. 6 General workflow of the 3D object tracking and updating process

the detected location, the system will use a general object detection model.

3D Object Tracking and Updating

Tracking an object's 2D bounding box across frames (Fig. 6) requires less computational power and is more effective than detecting the objects in each frame. By utilizing Swift's vision framework,[3] we can track the 2D bounding box of the detected object across frames. Each tracking result provided an updated 2D bounding box and confidence score. We considered the object lost tracking in the new frame if its tracking confidence score was below a threshold (i.e., 70%). To obtain the updated 3D bounding box of track objects, we use the 3D bounding box estimation method (described in "3D Bounding Box Estimation") with updated 2D bounding boxes, which can be either run on the mobile device or through the API call. The detection mode will be reactivated when any of the following requirements are satisfied:

- All tracked objects with their tracking confidence score below the threshold.
- The device moved more than a meter away from its latest detection location.
- The device orientation changed for more than 45° from the latest detection orientation.

Point Cloud Based Tracking

The ARKit point cloud is sparse and contains considerable noise due to various circumstances in the environment, such as lighting conditions, shadows, and colors. The sparsity within the ARKit point cloud increases the error in estimating the 3D bounding box of the object from a single camera frame. To obtain a more stable and accurate 3D bounding box (i.e., the validated bounding box), we used the non-maximum suppression (NMS) method [31] for objects that have five or more estimated 3D bounding boxes across multiple frames. The object with the validated bounding box then undergoes an information filtering process to determine whether it should be considered for voice notification and AR visualization.

3D matching: Previously detected/tracked objects might be detected again when the app switches from tracking mode to detection mode with the point cloud data. Therefore, we need to match the objects to avoid creating multiple instances for the same object. This object matching process compares the estimated 3D bounding box of a newly detected object with the 3D bounding boxes of previously detected/tracked objects with the same label. If the 3D bounding boxes overlap, they are considered the same object, and the object's 3D bounding box is updated. As discussed previously, the estimated 3D bounding box of an object from a single frame with point cloud data has low accuracy; therefore, there may not always be an overlap for the bounding boxes of the same object. More importantly, the object may move. We must also compare closely located objects with the same label. Suppose that the distance between the previous and newly estimated 3D bounding boxes is under a distance threshold percentage, it will be considered the same object and updated its 3D bounding box. Otherwise, it is considered a new object and created a new object instance.

Depth Map Based Tracking

Unlike point cloud data, depth map data provide much higher accuracy in estimating the 3D bounding boxes of objects. Therefore, it is not necessary to track all objects across frames. For the depth map data, we only track dynamic moving objects. In this case, we assume that only people or objects with their 3D bounding box overlapping a person's bounding box are dynamic moving objects. However, the list

[3] https://developer.apple.com/documentation/vision/vntrackobjectrequest.

Fig. 7 Examples of AR visualization of 3D object detection results. Blue represents the chair; red represents the person; purple represents the cup, bottle, and bowl; and gray represents the unknown object

of dynamic moving object types can expand if needed. The 2D bounding boxes of these objects are tracked using the vision framework and passed to the object detection module to obtain its updated 3D bounding box if its tracking confidence score is over the threshold.

Voice and AR-Enhanced User Interface

Our system is designed to serve different users, including people with BLV or ASD and others who have traveling challenges. The application utilizes ARKit and SiriKit to provide a voice and AR-enhanced user interface to adapt to each user's needs and preferences. Furthermore, to minimize the computational power required for our user interface application, the frame rate was set at 30 frames per second (fps), as it is difficult for the human eye to see differences above 30 fps. Note that the maximum speed of updating objects' 3D bounding boxes is 10 fps in our current implementation, depending on the object detection and tracking run time.

AR Visualization

The AR mobile application utilizes ARKit to display AR assets by aligning them with the 3D bounding boxes of detected objects, thereby providing visual assistance to individuals with low vision or ASD. Additionally, it can be used as digital signage for various scenarios, including task guidance and emergency evacuations.

The current design uses a 3D rectangular box to show the object's 3D pose and size, along with a sphere whose radius depends on the shortest side of the rectangular box. The color and transparency of display AR assets are determined by the object label and its confidence score based on the detection or tracking results. Figure 7 shows some examples of AR visualization. The detected person is represented in red, as we feel it is essential to identify the person, especially for users with the challenge of social interaction.

The AR visualization functionality can be enhanced to offer users additional assistance in task completion. By utilizing the object search feature in tandem with the application, task instructions can be displayed at the appropriate location based on the 3D pose of the target object. As illustrated

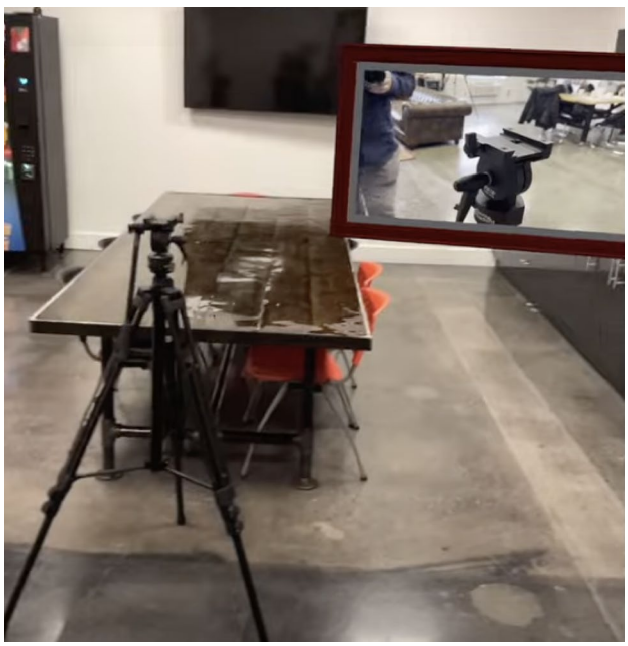

Fig. 8 AR instruction example. The AR frame displays a video showing the user how to attach the camera to a tripod

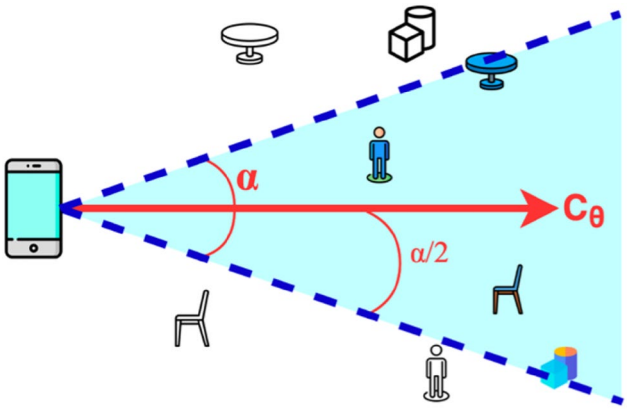

Fig. 9 Information filtering process. Filter the objects within the angular range (α) with respect to the camera orientation (C_θ)

in Fig. 8, the AR instruction is placed in response to the detected 3D location of the tripod. The availability of object detection eliminates the need to predetermine the location of the object or account for potential variations in its position over time. Additional use cases can be derived based on the capability of the AR visualization and user studies.

approach provides us with greater flexibility in creating in-app actions. Our voice interaction can confirm user actions, identify target objects that users are looking for, and provide information about detected objects. A video demonstration of the voice interaction is available in "Demo".

Information Filtering

With massive objects detected over time, information filtering is necessary to extract essential information for alerting individuals with BLV or ASD users based on their settings. First, we filter out objects that do not interfere with the camera-facing direction of the user's device for all detected objects (Fig. 9). Secondly, we sort the remaining objects by alert priority using Eq. 1. Alert priority is calculated based on the distance between the user's camera (C) and the object (O), as well as the object size, where the distance has more weight than the object size ($\alpha \gg \beta$). Lastly, a voice notification is provided only if the alert priority value exceeds a threshold.

$$\text{AlertPrority}(C, O) = \frac{\alpha}{\text{distance}}(C, O) + \beta \times \text{volume}_0 \tag{1}$$

$$\text{Distance}(C, O) = \sqrt{\left(C_X - \left(O_{\text{center}_x} - \frac{O_{\text{width}}}{2}\right)\right)^2 + (C_Z - (O_{\text{center}_Z} - \frac{O_{\text{length}}}{2}))^2} \tag{2}$$

Voice Interaction

Individuals with BLV may face challenges with touch-based interaction, and incorporating voice interaction is crucial to improve user-friendliness. The voice interaction in the proposed system consists of two primary components: one that receives action commands from the user and another that presents the required information based on the action command.

Continuous listening to a user's voice commands not only consumes additional computational power of a mobile device but may also result in inaccurate interpretation of commands due to environmental noise. To address this, we have opted to utilize the SiriKit library[4] to enable users to provide in-app commands for our application via Siri. This approach eliminates the need for users to learn a new way of interacting with the app and avoids the additional power consumption associated with continuous voice command processing.

In addition to utilizing SiriKit for initiating app activities, we also developed our in-app voice interaction to handle specific commands, reducing the need for users to say "Hi Siri" each time they want to communicate with our app. This

[4] https://developer.apple.com/documentation/sirikit.

The voice notification includes both the distance and direction of the alert object. The distance between the object and camera is calculated using Eq. 2, which considers only the differences in the x- and z-axes between the object and camera positions. The object direction (i.e., in front, left, or right) is determined based on the angle difference between the y-direction of the current camera orientation and the object's position with respect to the device's location. Whenever a new object is detected, or any object is within the dangerous range of the user's location, the information filtering process is triggered, and a new voice alert is issued if necessary.

Experiments

After introducing the hardware and software requirements of our system, we will discuss several experiments we conducted to evaluate the proposed system's object detection accuracy and real-time performance. These experiments were conducted in public and private residential areas with the consent of the subjects recorded in the videos. We have also planned a series of usability experiments, but due to the limited time and other constraints such as pandemics, they will be conducted in the future.

Table 1 Amazon elastic compute cloud cost [33]

Instance name	GPUs	# of vCPUs	Memory (GiB)	Clock speed (GHz)	Network performance (Gigabit)	On-demand hourly rate
T3 large	–	2	8	3.1	< 5	$0.0832
VT1 3xlarge	–	12	24	2.5	3.12	$0.6500
G4DN eight extra large	1 NVIDIA T4 Tensor Core (16GiB)	32	128	2.5	50	$2.1760
P3DN 24xlarge	8 NVIDIA Tesla V100 (256 GiB)	96	768	2.5	100	$31.2120

Hardware and Software Requirements and Cost

Our system is portable and doesn't have a fixed cost as it can be customized to suit various devices and setups. The two primary expenses are the user's devices and the cloud server, which can vary depending on the devices and processing power used for the cloud server. The key idea is that users will use their existing mobile devices, so no additional costs are needed, except for possible modest monthly cloud service fees.

In our system, we use ARKit for image-based localization and modeling. The ARKit-based approach can function on all iOS devices that meet certain specifications, namely iOS 11.0 or later and an A9 or newer processor. This includes the iPhone 6 s and subsequent models (priced between $100 and $1100) as well as the 5th generation iPad and newer versions (priced between $150 and $1100). Although optional, the iPhone LiDAR sensor can enhance the accuracy of 3D object detection, as discussed in the previous section. The iPhone's LiDAR sensor operates at a wavelength of approximately 800 nanometers (nm) and employs photon counting detectors and vertical cavity surface emitting lasers [32]. Additionally, its range capability is discussed in "Object Detection Accuracy Analysis".

According to [32], the LiDAR sensor on the iPhone operates at a wavelength of approximately 800 nanometers (nm) and employs photon counting detectors and vertical cavity surface emitting lasers. Additionally, its range capability is discussed in "System Performance Evaluation".

The system includes a cloud server to process 3D object detection, whose storage and processing power can vary based on the user's needs. The cloud server can be host by third party provider with two host options, one is pay per request and another one is run continuously with fixed cost. The cost will vary depending on factors such as storage capacity, processing power, and network performance. For our cloud server, the minimum requirements are 8GB memory and 2 virtual CPUs (vCPUs), which correspond to a t3.large instance in AWS, priced at $0.0832 per hour. "Real-Time System Performance Analysis" includes an overview of the real-time performance of our system, using the aforementioned cloud server settings.

As mentioned in "3D Object Detection", the 3D object detection approach relies on the output of pre-trained 2D object detectors and can incorporate several custom object detection models tailored to different tasks and settings. By fine-tuning these pre-trained models, the customized 2D detectors can be trained for new object categories with reduced training costs. The cost of training these models is largely dependent on the number of training samples and the desired accuracy. Table 1 provides information on the cost of several AWS Elastic compute cloud (EC2) instances [33] with different processing power specifications, serving as a reference for estimating the cost of training the 2D object detectors and hosting a more powerful cloud server that can scale beyond the minimum requirement.

System Performance Evaluation

Our system can easily be adapted to any 2D object detector and turn its detection results into 3D. Therefore, the performance of the 3D detection result highly depends on the 2D object detector used and the range of objects it can detect.

To evaluate object detection accuracy, we examined several datasets of indoor scenes that include 3D object annotation, particularly SUN RGB-D [30] and NYUDv2 [34]. These datasets typically have a depth map resolution of 640×480, which is approximately six times larger than the depth map collected by our mobile device (256×192) and much more substantial than the AR point cloud data (approximately 200 points per frame). Due to this significant difference, existing datasets cannot be used to evaluate the performance of our object detection method. Therefore, we developed a data collection and annotation toolkit to create our dataset using the mobile device sensors.

We used YOLOv3 as the 2D object detectors for our system with an image resolution of 640×480 to evaluate our system's object detection accuracy on 13 object categories

in our test dataset (Table 2). The YOLOv3 detector is trained on the COCO dataset [35] and can detect 80 different object categories, such as people, chairs, and table, among others.

We conducted several experiments to evaluate the real-time performance of our system using different combinations of three 2D object detectors (YOLOv3, YOLOv5 Nano, and YOLOv5 Medium) and two image resolutions (640×480 and 256×192). All three object detectors were trained on the COCO dataset, and we also assessed the impact of adapting the cloud server to the system's real-time performance. We used an iPhone 13 Pro Max as a local processor, while the object detection server was deployed on AWS Elastic Beanstalk with a t3 large instance that contained 2 vCPUs and 8GB of memory.

Data Collection and Annotation Toolkit

The data collection and annotation toolkit comprises an iOS app for data collection and a data annotation interface (Fig. 10) for annotating 3D bounding boxes of objects. The iOS app collects several data for each frame, including RGB images, AR point cloud data, depth maps, and intrinsic and extrinsic camera parameters.

The annotation interface was developed in Python and is designed to preprocess the data and estimate a set of 3D object bounding boxes, with the ability for annotators to make corrections. First, the RGB image is processed with the Mask R-CNN model to obtain the object segmentation mask, which is then aligned with the depth map to generate the object's 3D point cloud. Our proposed method is then used to pre-label the 3D bounding boxes of objects based on the object point cloud data. The interface then displays a 3D point cloud and RGB image of the corresponding frame along with the pre-labeled 2D and 3D bounding boxes of the object. The annotator checks each object in turn, correcting the bounding boxes and invalidating the object if necessary. This hybrid approach reduces the burden of the data labeling process.

This annotation tool enables us to create an object dataset based on mobile devices, which can be utilized to train deep learning models for more accurate predictions based on different purposes in the future.

Object Detection Accuracy Analysis

For the experiments, we collected data using our data collection application from various locations, including residential houses, public stores, and streets. We obtained the ground truth 3D bounding boxes of the objects using our annotation interface. Table 2 provides a breakdown of the collected objects.

The accuracy of the depth map is crucial for the performance of 3D object detection since the depth values play a critical role in estimating the 3D pose of an object from its 2D detection results. Incorrect depth values can lead to inaccurate 3D bounding boxes for the objects. Therefore, we evaluated the confidence level of the accuracy of depth data based on the collected experimental data.

Table 2 Number of objects collected based on the distance range

	< 3 m	3–5 m	5–7 m	> 7 m
Chair	34	4	0	0
Bottle	21	4	0	0
Laptop	11	0	0	0
Bags	8	0	0	0
Cup	5	1	0	0
Person	4	0	0	0
Plant	3	0	0	0
Bowl	2	1	0	0
Vase	2	1	0	0
Phone	2	0	0	0
Tv	2	0	0	0
Table	0	2	0	0
Car	0	0	0	8

The distance is calculated based on the camera location and the object center

The depth map generated by the built-in LiDAR scanner of the selected iOS devices is generally accurate within a range of 5 ms and can detect up to 15 ms, but the depth accuracy decreases as the distance increases. The ARKit framework uses three confidence levels to indicate the accuracy of the collected depth data: low, medium, and high. We divided the distances into four major ranges: less than 3 ms, between 3 and 5 ms, between 5 to 7 ms, and greater than 7 ms. Objects within 3 ms are particularly important for people with BLV, as they may bump into these objects if not notified. It is worth noting that the Microsoft Kinect sensors used in [22] have the limitation of providing inaccurate distance information beyond 3 ms.

Based on our preliminary study, we found that the built-in LiDAR sensor of iOS devices is usually accurate within 5 ms. Thus, we set 5 ms as another threshold for our distance range. From our experiments, we observed that the farthest distance detectable in the experimental residential house was within 7 ms. Therefore, we set 7 ms as the threshold for object detection between indoor and outdoor environments. Figure 11 shows the distributions (in percentages) of the three confidence levels of the depth value accuracy versus distances based on our experimental data. Table 3 lists the percentages of the number of points in the three confidence levels for the accuracy of the AR depth value based on distance ranges. At each distance, the three percentage values add up to 100%.

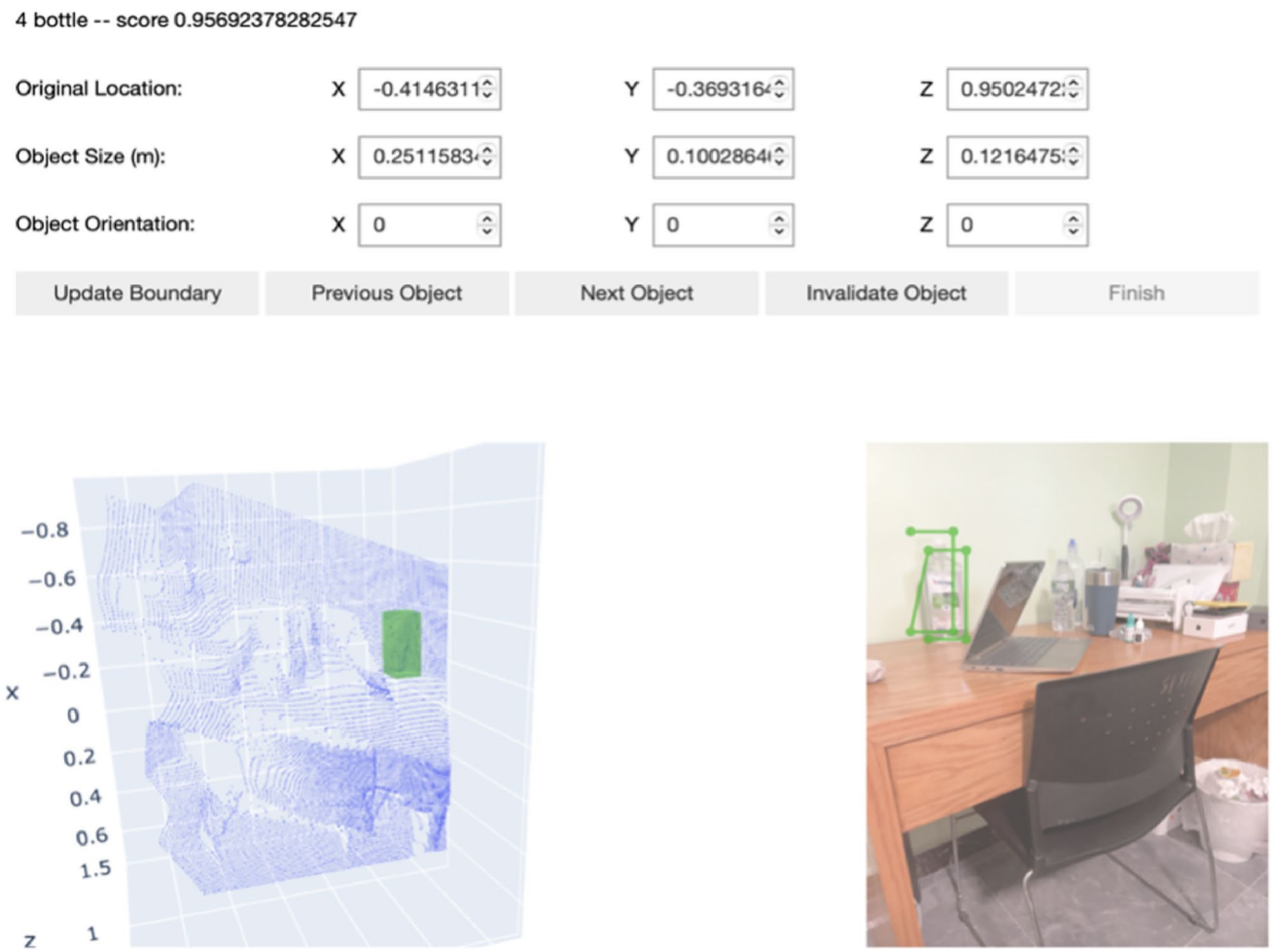

Fig. 10 The data annotation interface displays the current annotated object's 3D location, size, and orientation on the top, while the corresponding image frame's point cloud and RGB image are shown at the bottom. The green line segments and rectangular box indicate the 2D and 3D positions of the annotated object, respectively

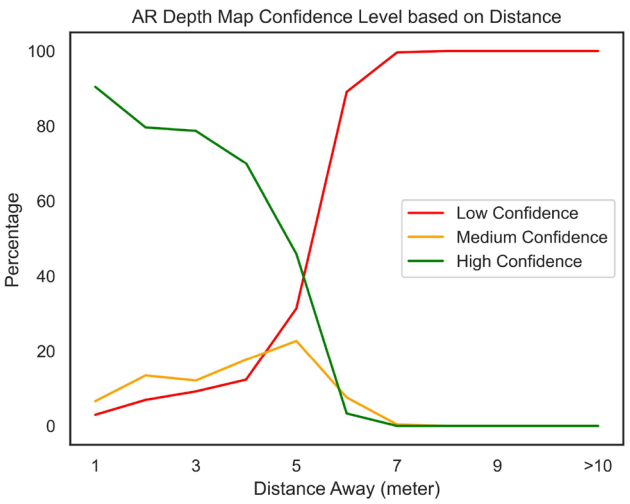

Fig. 11 Distributions of the three confidence levels for accuracy of the AR depth values versus distances. At each distance, the three percentage numbers add up to 100%

Table 3 Distributions of the three confidence levels for accuracy of the AR depth value based on distance ranges

	Distance range			
Confidence level	< 3 m	3–5 m	3–5 m	> 7 m
Low confidence	7.66%	18.22%	93.17%	100%
Medium confidence	12.14%	19.22%	4.79%	0%
High confidence	80.20%	62.55%	2.02%	0%

To evaluate our object detection method, we used using YOLOv3 as the 2D object detector for our system with an image resolution of 640×480. We measured the performance using the intersection over union (IoU) metric, which compares the predicted 3D object boundaries with the ground truth. We considered a prediction accurate if the IoU exceeded 25%. Additionally, we compared our results with Transferable3D [36], a state-of-the-art semi-supervised

Table 4 Object detection performance based on the distance range

Distance range	Number of objects	Our method (mAP)	Transferable3D (mAP)
< 3 m	94	63.88%	38.13%
3–5 m	13	50.00%	33.33%
5–7 m	0	–	–
> 7 m	8	37.50%	0.00%

3D object detection model capable of predicting untrained objects. However, since our annotated data was relatively small and insufficient for training Transferable3D, we used the SUN RGB-D dataset [30] for this comparison.

Table 2 presents the data captured from random scenes in the experimental areas to test the performance of our system in a real-world environment. Hence, the distribution of each object class is uneven in the annotated data, which can cause some objects to have a more significant influence than others. For example, a bad prediction for a vase can dramatically decrease the mAP value. Moreover, our data is much sparser than the SUN RGB-D dataset, so it is reasonable that Transferable3D does not perform as well as it did for the SUN RGB-D dataset (41.8% mAP). Additionally, the SUN RGB-D dataset contains only indoor objects and lacks any data for cars, resulting in poor performance of Transferable3D in predicting cars. However, with more data, we can refine Transferable3D to achieve better performance. Despite these limitations, our method achieves an mAP of 63.88% for objects within 3 m (Table 4), where the classes with low accuracy are smaller objects, particularly vases with IoU less than 10%.

In addition to the object detection performance based on distance ranges, we also tested the detection performance for objects captured from different perspectives. Specifically, we tested the laptop and chair because both objects have different shapes when viewed from different angles.

Based on Table 5, the average IoU for laptops and chairs does not vary significantly when viewed from different perspectives. Additionally, the difference in average precision (AP) is minimal, likely due to the relatively small sample size, where a single sample can have a significant impact on the AP value. Nevertheless, it is worth noting that the width and length of laptops and chairs are very similar, which means that the resulting 3D bounding boxes may still encompass the objects at different angles if they were placed in the nearby center.

Real-Time System Performance Analysis

We developed an iOS application to evaluate the real-time performance of our system, which can perform object detection on either a mobile device or a cloud server. We tested three 2D object detectors (YOLOv3, YOLOv5 Nano, and YOLOv5 Medium) at two image resolutions (640 × 480 and 256 × 192) using an iPhone 13 Pro Max. The object detection server was deployed on AWS Elastic Beanstalk with a t3 large instance containing 2 vCPUs and 8GB of memory.

The experiments were conducted in a residential house, where a data collector walked along a predetermined path while using the object detection application. The app and cloud server collected the run-time data during the experiments.

Table 6 displays both the run-time and accuracy in object detection of our system with different combinations of object detectors (YOLOv3, YOLOv5 Nano, and YOLOv5 Medium) and image resolutions (640x480 and 256x192), using depth map data. The performance with the point cloud data has similar processing times.

The experiments mainly compare the mobile device run time (MDT) and the cloud server run time (CST). For each combination of the model size and the image resolution, more than 200 camera frames have been processed for 3D object detection. The total cloud server run time (CST-T) has two components: data transfer time (CST-D), which is the time from when the API call was initiated in the mobile app until the cloud server received the API request and loaded the data, and data computation time (CST-C), which is the time taken to process the data in the cloud. The MDT represents the total process time of the object detection module in the iPhone 13 Pro Max, which uses the A15 Bionic processor with a 6-core CPU and 5-core GPU. In fact, this mobile device has more computational power than the AWS cloud server host, which has only two vCPUs and 8GB of memory. In real-world applications, the cloud server can have much more powerful hardware.

Table 5 Object detection performance for objects captured from different perspectives

Object class	Object perspective					
	Left		Center		Right	
	Avg IoU	AP	Avg IoU	AP	Avg IoU	AP
Laptop	24.31%	33.33%	28.30%	50.00%	23.04%	50.00%
Chair	27.53%	50.00%	24.14%	44.44%	23.12%	37.50%

Table 6 Comparison system performance on both a mobile device and a cloud server, included the mean and standard deviation of the number of objects detected in a single frame and the system's run time in milliseconds using various 2D object detectors and image resolutions

RES	YOLOv3 (248.4MB)		YOLOv5 Medium (85.1MB)		YOLOv5 Nano (7.9MB)	
	640 × 480	256 × 192	640 × 480	256 × 192	640 × 480	256 × 192
#OBJ	7.4 ± 3.8	5.3 ± 2.1	5.8 ± 3.4	4.4 ± 2.4	3.6 ± 1.9	3.1 ± 1.9
MDT	85 ± 4	80 ± 4	68 ± 13	62 ± 4	54 ± 15	50 ± 7
CST-D	399 ± 130	257 ± 80	394 ± 96	249 ± 91	390 ± 88	262 ± 128
CST-C	2078 ± 48	2035 ± 48	791 ± 26	773 ± 23	188 ± 26	171 ± 22
CST-T	2478 ± 138	2291 ± 97	1185 ± 101	1022 ± 92	578 ± 96	433 ± 131

RES: image resolution; #OBJ: number of objects detected; MDT: mobile device time (ms); CST-D, CST-C, CST-T: cloud server time (ms)—data transfer, computation, total

However, data transfer may become a bottleneck unless the mobile device lacks the power and computational budget to process object detection/tracking while performing basic localization services. Additionally, the 2D object detector requires 2–3 s to initialize the model on the mobile device.

The computation run time was greatly influenced by the 2D object detectors, with a large model (YOLOv3) taking nearly twice as long to process on the mobile device and over ten times longer on the cloud server compared to a small model (YOLOv5 Nano). However, the larger model was able to detect more objects in the same scenario than the smaller one. As shown in Table 6, YOLOv3 produces approximately twice as many detections as YOLOv5 Nano for images of the same resolution.

The image resolution also affects the computation run time although it is minor compared to the model type. However, it becomes an important factor when aiming for real-time performance using the object detection server. The 256 × 192 image resolution is usually around 50 KB, whereas the 640 × 480 image resolution is around 300 KB. With around 250 KB difference in data size, it took more than 100 ms to transfer the data to the cloud server. The difference in the data transfer time for the same image resolution in Table 6 is due to the fluctuation of the internet speed. Faster internet and edge computing techniques could reduce data transfer time. Furthermore, a lower image resolution can also affect the performance of 2D object detectors, as shown in Table 6, where fewer objects are detected in the same image frame.

Our system can easily adapt to different 2D object detectors, image solutions, and the processing power of the object detector server to meet diverse user requirements in terms of run time and details for the 3D detection result. Moreover, the system can deploy multiple cloud servers to adapt to various user requirements.

System Usability Evaluation

Usability experiments were planned to collect user feedback and evaluate the user experience with our application. The main target user groups for the experiments were individuals with BLV or ASD.

The first objective of the experiment was to collect user feedback on voice interaction and the user interface with AR visualization. The baseline of the experiment will establish by asking participants to explore an unfamiliar environment using their current tools, such as a white cane or guide dog. Next, we had participants use our application to explore another similar environment. At the end of each experiment, we will evaluate participants' understanding of the environment and conduct a user survey to learn about their challenges during exploration and their suggestions for our application.

We also planned to test voice notifications for the detected objects, including alert frequency, content delivery, and voice speech. A set of metrics for different attributes will be tested for all participants in a similar environment. We will collect data for each combination of the metrics, including the self-evaluated anxiety level, the time needed to explore the scene, the number of times participants hit an obstacle, and time taken to reach the target object.

After conducting all the experiments, the experimental data and participants' feedback will be analyzed and used to improve the application.

Demo

A video demo can provide better visualization of our system's performance in mobile applications. It can be viewed by clicking on this link: https://youtu.be/4tj4fqiCVeE. The demo takes place in a residential house that shows two primary use cases:

- **Object search.** The user opens and activates the object search using Siri commands. The in-app voice interaction is initialized with an object search command that asks the user for the target object. In this demo, the user searches for a laptop. The app starts searching for the laptop after confirming it with the user. The application provides the updated location of the laptop as the user moves.
- **Object detection.** The user activates the object detection through Siri commands. The application continues to provide the detected object and people's locations as the user moves.

Conclusions and Discussions

A Few Concluding Remarks

In this paper, we propose a 3D object detection, recognition, and presentation system that operates on a mobile device. The system efficiently works in real-time, does not require specialized high-end sensors, and contains voice interaction and AR visualization to assist in delivering information to individuals with BLV, ASD, or other challenges. Additionally, the system is low cost, with the only requirement being an iPhone 6 s or subsequent models that a user may already have, and the real-time performance is adjustable based on the available budget for hosting the cloud server.

To increase flexibility in adapting to different data input types and object detectors based on run time requirements and details for the 3D detection result, we introduced a 3D object detection server. The processing power of the object detection server can be adjusted by user based on their budget and desired information update frequency. It also reduces the computational requirements for the mobile device and enables integration with other applications, such as navigation applications. Additionally, the object detection module is available in the mobile app to handle areas with poor network connections.

The system's ability to adapt to any 2D object detector and turn the 2D detection result into 3D allows for customization based on the user's needs. The system can easily incorporate multiple object detectors and determine which one to use based on the tasks the user wants to perform or the locations that the user are in, through GPS sensor or BLE beacon signal, as each model could correspond to a specific set of tasks or specific locations. For example, one detector can be used for detecting doors and knobs to guide users to reach and open doors, while another one can be used for general obstacle detection, and a third one for location-based detection of landmarks or industrial equipment. By using target-oriented object detectors, the system can have more focused training data and adaptive computational times specific to each task, thereby improving its performance. Furthermore, the location-based object detection models can be deployed on local machines in their respective locations, which can reduce the cost of hosting on the cloud server and decrease data transfer time and cost. Additionally, by utilizing cloud servers, these models can be employed alternately based on the actions the user wants to perform.

We developed a data collection and annotation toolkit to create an object dataset based on mobile device sensors. This dataset can then be used to train a deep learning model to improve the accuracy of object detection. Additionally, the data collection application can be integrated with the modeling application [9] that we previously developed for indoor navigation. This integration will enable the creation of a 3D semantic reconstruction of the scanning environment that includes static objects' locations and sizes on top of the 2D floorplan and AR model: During environment scanning, the modeling application collects data for object detection. The object detection module will then processes this data to obtain each object's 3D pose and size for each frame. Afterward, a merge module is created to match and refine the 3D bounding box of the same object using the detected object's location in the AR coordinate system. Finally, the 3D location and size of the object can be projected onto the 2D floorplan using the transformation matrices obtained in [9].

Limiations and Future Work

There is much room for future improvement to the current system. The first and most crucial step was to conduct the user experiments, as discussed in "System Usability Evaluation", to enhance the mobile application's usability. This step is critical in validating whether the system can meet user needs and identifying its weaknesses as a real-world application. The primary motivation of this paper was to integrate this system with our indoor navigation system ASSIST [9], which has successfully transitioned from a research prototype at CUNY to a real system by Nearabl. This integration will improve the situation awareness of users with BLV, ASD, or other challenges in indoor area exploration and navigation. Additionally, integrating the data collection application with the modeling application would be ideal for constructing a 3D map of the scanned environment and expanding our dataset. Ultimately, the dataset can be used to develop a deep learning model to improve object detection accuracy.

The current system for 3D object detection has a distance limitations, as the accuracy of depth values captured by existing mobile devices decreases with increasing distances. Therefore, the system is only adaptable to indoor scenes. Additionally, the detection of dynamic moving objects or persons increases the challenge of accurately detecting an object's 3D pose, making the current system unsuitable for crowded environments such as shopping malls. However,

we are planning to incorporate several methods into the current system to mitigate noise in the future. First, the system will monitor detected objects and the user's device motion to determine if the user is in a crowd environment. If so, the application will alert the user by asking them to move slower. Second, incoming sensor data (i.e., point cloud and depth map) can be filtered using previous timestamp data and the user's motion data to handle sensor noise. Finally, to handle dynamic moving objects, especially humans, a Kalman-filter-based approach can be used to predict their movements and prevent collisions with the user.

Based on the user's actions, the system will handle object search, obstacle avoidance, and understanding the surrounding environment differently. For object search, a target-oriented model will be applied. For obstacle avoidance, the system will focus on detecting the closest obstacle without the need for obstacle recognition. This can be further integrated with sensors installed on the user's cane, such as ultrasonic sensors.

Furthermore, to help the user to have a full understanding of the environment, the system can take a longer process time to capture the scene in 360° as the user stands in one location. The captured data can then be integrated to create a rough 3D construction and provide information to the user.

Funding This study was funded by US National Science Foundation (#2131186, #2118006, #1827505 and #1737533), ODNI Intelligence Community Center for Academic Excellence (IC CAE) at Rutgers (#HHM402-19-1-0003 and #HHM402-18-1-0007) and the US Air Force Office for Scientific Research (#FA9550-21-1-0082).

Data availability The data that support the findings of this study are available on request from the corresponding author, JC. The data are not publicly available due to the potential compromise of personal privacy.

Declarations

Conflict of interest Jin Chen is the CTO from Nearabl Inc. and owns stock in Nearabl Inc. Zhigang Zhu declares no conflict of interest.

Ethical approval This article does not contain any studies with human participants performed by any of the authors.

References

1. Bourne R, Steinmetz JD, Flaxman S, Briant PS, Taylor HR, Resnikoff S, Casson RJ, Abdoli A, Abu-Gharbieh E, Afshin A, Ahmadieh H. Trends in prevalence of blindness and distance and near vision impairment over 30 years: an analysis for the global burden of disease study. Lancet Glob Health. 2021. https://doi.org/10.1016/s2214-109x(20)30425-3.
2. Vision Atlas. The International Agency for the Prevention of Blindness. https://www.iapb.org/learn/vision-atlas. Accessed 15 Aug 2022.
3. Manduchi R, Kurniawan S. Watch your head, mind your step: mobility-related accidents experienced by people with visual impairment. Department of Computer Engineering, University of California, Santa Cruz, Technical Report 2010;1.
4. Koldewyn K, Weigelt S, Kanwisher N, Jiang Y. Multiple object tracking in autism spectrum disorders. J Autism Dev Disord. 2013;43:1394–405.
5. van der Geest JN, Kemner C, Camfferman G, Verbaten MN, van Engeland H. Eye movements, visual attention, and autism: a saccadic reaction time study using the gap and overlap paradigm. Biol Psychiatry. 2001;50(8):614–9.
6. Quintana E, Ibarra C, Escobedo L, Tentori M, Object Favela J, gesture recognition to assist children with autism during the discrimination training. In: Progress in pattern recognition, image analysis, computer vision, and applications: 17th Iberoamerican congress, CIARP, Buenos Aires. Argentina. 2012. p. 877–84.
7. Laser Eye Surgery Hub. Visual impairment and blindness global data and statistics. https://www.lasereyesurgeryhub.co.uk/data/visual-impairment-blindness-data-statistics. Accessed 15 Aug 2022.
8. Chen J, Zhu Z. Real-time 3D object detection and recognition using a Smartphone. In: Proceedings of the 2nd international conference on image processing and vision engineering. 2022. p. 158–65. https://doi.org/10.5220/0011060600003209.
9. Zhu Z, Chen J, Zhang L, Chang Y, Franklin T, Tang H, Ruci A. iassist: an iphone-based multimedia information system for indoor assistive navigation. Int J Multimed Data Eng Manag (IJMDEM). 2020;11(4):38–59. https://doi.org/10.4018/IJMDEM.2020100103.
10. Ren S, He K, Girshick R, Sun J. Faster r-cnn: towards real-time object detection with region proposal networks. Advances in neural information processing systems. 2015. p. 28. https://doi.org/10.1109/TPAMI.2016.2577031.
11. He K, Gkioxari G, Dollár P, Girshick R. Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. 2017. p. 2961–9. https://doi.org/10.1109/ICCV.2017.322.
12. Redmon J, Farhadi A. Yolov3: an incremental improvement. 2018. arXiv preprint arXiv:1804.02767.
13. Glenn J, Ayush C, Alex S, Jirka B, et al. ultralytics/yolov5: v7.0—YOLOv5 SOTA realtime instance segmentation. 2022. https://doi.org/10.5281/zenodo.7347926.
14. Wang CY, Bochkovskiy A, Liao HY. YOLOv7: trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. 2022. arXiv preprint arXiv:2207.02696.
15. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC, Ssd: Single shot multibox detector. In: Computer vision-ECCV, 14th European conference, Amsterdam, The Netherlands. 2016. p. 21–37. https://doi.org/10.1007/978-3-319-46448-0_2.
16. Lin TY, Goyal P, Girshick R, He K, Dollár P. Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision. 2017. p. 2980–8. https://doi.org/10.1109/ICCV.2017.324.
17. Huang HC, Hsieh CT, Yeh CH. An indoor obstacle detection system using depth information and region growth. Sensors. 2015;15(10):27116–41. https://doi.org/10.3390/s151027116.
18. Cheng R, Wang K, Yang K, Zhao X. A ground and obstacle detection algorithm for the visually impaired. In: IET international conference on biomedical image and signal processing. 2015. p. 1–6.
19. Soquet N, Aubert D, Hautiere N. Road segmentation supervised by an extended v-disparity algorithm for autonomous navigation. In: 2007 IEEE intelligent vehicles symposium. 2007. p. 160–5. https://doi.org/10.1109/IVS.2007.4290108
20. Sun L, Yang K, Hu X, Hu W, Wang K. Real-time fusion network for RGB-D semantic segmentation incorporating unexpected obstacle detection for road-driving images. IEEE Robot Autom

Lett. 2020;5(4):5558–65. https://doi.org/10.1109/LRA.2020.3007457.
21. Chen Y, Liu S, Shen X, Jia J. Dsgn: Deep stereo geometry network for 3d object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020. p. 12536–45. https://doi.org/10.1109/CVPR42600.2020.01255
22. Pham HH, Thi-Lan L, Vuillerme N. Real-time obstacle detection system in indoor environment for the visually impaired using microsoft kinect sensor. J Sens. 2016. https://doi.org/10.1155/2016/3754918.
23. Fischler MA, Bolles RC. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun ACM. 1981;24(6):381–95. https://doi.org/10.1145/358669.358692.
24. Domenech JF, Escalona F, Gomez-Donoso F, Cazorla M. A voxelized fractal descriptor for 3D object recognition. IEEE Access. 2020;8:161958–68. https://doi.org/10.1109/ACCESS.2020.3021455.
25. Wu Z, Song S, Khosla A, Yu F, Zhang L, Tang X, Xiao J. 3d shapenets: a deep representation for volumetric shapes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2015. p. 1912–20.
26. He C, Gong J, Yang Y, Bi D, Lan J, Qie L. Real-time track obstacle detection from 3D LIDAR point cloud. J Phys Conf Ser. 2021;1910(1):012002. https://doi.org/10.1088/1742-6596/1910/1/012002.
27. Garnett N, Silberstein S, Oron S, Fetaya E, Verner U, Ayash A, Goldner V, Cohen R, Horn K, Levi D. Real-time category-based and general obstacle detection for autonomous driving. In: Proceedings of the IEEE international conference on computer vision workshops. 2017. p. 198–205. https://doi.org/10.1109/ICCVW.2017.32
28. Levi D, Garnett N, Fetaya E, Herzlyia I. Stixelnet: a deep convolutional network for obstacle detection and road segmentation. Br Mach Vis Conf. 2015;1(2):4. https://doi.org/10.5244/C.29.109.
29. Apple Inc. Arkit—augmented reality. 2022. https://developer.apple.com/augmented-reality. Accessed 15 Aug 2022.
30. Song S, Lichtenberg SP, Xiao J. Sun rgb-d: a rgb-d scene understanding benchmark suite. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2015. p. 567–76. https://doi.org/10.1109/CVPR.2015.7298655.
31. Rothe R, Guillaumin M, Van Gool L. Non-maximum suppression for object detection by passing messages between windows. In: Computer vision-ACCV 2014: 12th Asian conference on computer vision, Singapore. 2015. pp 290–306. https://doi.org/10.1007/978-3-319-16865-4_19.
32. Sabbir R. The iphone 12—LIDAR AT YOUR FINGERTIPS. In Forbes. 2020. https://www.forbes.com/sites/sabbirrangwala/2020/11/12/the-iphone-12lidar-at-your-fingertips/?sh=3c3b72493e28. Accessed 25 Apr 2023.
33. Amazon Inc. Amazon EC2 instance types—Amazon Web Services. https://aws.amazon.com/ec2/instance-types/. Accessed 25 Apr 2023.
34. Silberman N, Hoiem D, Kohli P, Fergus R. Indoor segmentation and support inference from rgbd images. Eur Conf Comput Vis. 2012;7576:746–60. https://doi.org/10.1007/978-3-642-33715-4_54.
35. Lin TY, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL, Microsoft coco: Common objects in context. In: Computer Vision-ECCV, 13th European Conference, Zurich, Switzerland. 2014. p. 740–55. https://doi.org/10.1007/978-3-319-10602-1_48.
36. Tang YS, Lee GH. Transferable semi-supervised 3d object detection from rgb-d data. In: Proceedings of the IEEE/CVF international conference on computer vision. 2019. p. 1931–40. https://doi.org/10.1109/ICCV.2019.00202.