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

Thesis

Submitted in partial fulfillment of the requirements for the degree

Master of Data Science and Engineering

At

The City College of the City University of New York

By

Jin Chen

September 2022

Approved by:

Professor Zhigang Zhu, Thesis Advisor

Professor Michael Grossberg Professor Zhigang Zhu

Co-Directors, Data Science and Engineering Program

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

Jin Chen

Data Science and Engineering Program

Thesis Advisor: Professor Zhigang Zhu

Abstract

This thesis proposes 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 (BLV) and improve social interaction for individuals with autism spectrum disorder (ASD). 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

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my thesis advisor, Dr. Zhigang Zhu, for his invaluable supervision, endless support, and motivation throughout my master's study. He has always provided professional guidance and encouragement to help me overcome numerous obstacles in my life, both academic and personal. From him, I am not only learning to become a better researcher but also to discover my life purpose. My research experience opens my mind and will have a long-lasting influence on the rest of my life.

My sincere thanks also go to my fellow lab members at the City College of Visual Computing Lab for their feedback, support, and friendships. Special thanks to Arber Ruci and Fani Maksakuli, who helped me better understand the users' needs and real-life use cases to which my research can be applied. I would also like to thank Professor Hao Tang for his valuable suggestions on my research.

Last but not least, I wish to express my gratitude to my family for their encouragement throughout my research. My grandmother, sister, brother, and cousin for all their corporation in my experiments.

The work is supported by US National Science Foundation (#2048498, #2131186, #2118006, #1827505, and #1737533), Intelligence Community Center for Academic Excellence (IC CAE) at Rutgers (#HHM402-19-1-0003 and #HHM402-18-1-0007) and AFOSR (#FA9550-21-1-0082).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Visual impairment is the loss of some or all vision perception and is not easily fixable with treatments such as glasses, contact lenses, medication, or surgery. The blind and low vision population has continuously grown over the past three decades and is expected to increase significantly as the population ages [2]. According to the IAPB Vision Atlas [29], in 2020, 43 million people 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.

People who are legally blind or have low vision (BLV) face numerous challenges throughout their daily lives, ranging from moving indoors to completing tasks. The difficulty dramatically increases in unfamiliar environments, where they only have limited assistance tools, such as long canes, guide dogs, hand touches, and potential assistance from people. It is not only time-consuming for them to get comfortable with their surroundings but also raises safety concerns. The indoor environment is full of dynamic changes, and individuals with BLV can easily bump into obstacles with limited space for movement in the indoor area. Based on the survey results of 300 individuals who are legally

blind[17], over 40% of them experienced head-level accidents at least once a month, even with the help of a long cane or guide dog. Safety is the number one issue; recognizing humans and other objects could enhance their understanding of their surroundings, thus significantly improving their interaction with others and the quality of their lives.

Several studies have shown that people with autism spectrum disorder (ASD) have difficulty adjusting their visual attention, especially for dynamic moving objects or when shifting their attention [12, 28]. Therefore, assistive object recognition is essential for people with ASD, particularly for cognitive development and increased visual attention in complex environments [12, 19]. Furthermore, object recognition with AR visualization can assist people with ASD and low vision in completing tasks, such as providing instructions or information for corresponding objects and even helping them with object searching tasks.

Object detection is a fundamental topic in computer vision. Numerous techniques have been developed and applied in real-life applications, particularly in autonomous vehicles and robotics. Real-time 2D object detection techniques are very mature. However, they do not provide sufficient information (e.g., location and size of an object) that can be essential for people to obtain a more comprehensive understanding of the indoor environment. Real-time 3D object detection and recognition often involves expensive high-end devices (e.g., LiDAR sensors and depth cameras) and deep learning models that would require enormous computational power to ensure real-time performance. These methods are unsuitable for indoor environments and are not affordable for individuals with BLV or ASD, who usually have relatively low incomes. For example, data show that 89% of visually impaired people live in low- or middle-income countries [13].

With multiple obstacles and objects detected, selecting essential information to report

to users is another challenge, especially with real-time detection, where information constantly changes with time. People with BLV or ASD will be confused and distracted by massive amounts of data and easily miss the necessary information.

1.2 Overview of the Solution

This thesis aims to develop a low-cost and efficient real-time 3D object detection, recognition, and presentation system that works on mobile devices to increase accessibility for people with BLV or ASD and others that need assistance in indoor areas. *Objects* could be *obstacles* along a user's walking path, particular *items* the user is looking for, and *humans* that usually move around. This system is designed to adapt to different data types (such as single images, RGB-D images, stereo images, and 3D point clouds) collected by various mobile devices, apply customized models based on the desired run time and level of 3D detection details, and request tasks (i.e., object search or obstacle detection). Importantly, this system can be integrated with other systems, such as the previously developed indoor navigation system [32], to improve travel safety for people with travel challenges.

The main contributions of this thesis are as follows:

- A real-time 3D object detection, recognition, and presentation system works on iOS-based mobile devices that integrate ARKit, SiriKit, object detection cloud server, and object tracking.
- An object detection server with an adaptive 3D object detection and tracking method for various data inputs and task requirements.

- A voice and AR-enhanced interface improves information delivery to people with BLV or other disabilities.

- A data collection and annotation toolkit for constructing the object detection dataset based on iOS devices and can eventually be used for 3D reconstruction when integrated with the previously developed modeling app [32].

1.3 Outline of the Thesis

The remainder of this thesis is organized as follows. Chapter 2 provides a brief overview of the state-of-the-art 2D and 3D object detection methods. Chapter 3 describes the proposed 3D object detection, recognition, and presentation system. The experimental results of the proposed method are discussed in Chapter 4. Finally, the conclusion of the proposed system and a discussion of future work are presented in Chapter 5.

Chapter 2

Related Work

2.1 2D Object Detection

2D object detection predicts the class labels and 2D bounding boxes of objects from an input image. There are two main types of object detectors: one-stage detectors and two-stage detectors. Two-stage object detectors first extract the region of interest (ROI) of each object in the image using a regional proposal network (RPN), and then refine the 2D bounding boxes of the objects using regression and classify them into different classes. R-CNN families are examples of two-stage object detectors, including Fast R-CNN [21] and Mask R-CNN [10]. One-stage object detectors (e.g., YOLO families [8, 20, 30], SSD [16], and RetinaNet [15]) do not perform the ROI extraction and directly detect the 2D bounding boxes of objects from the image with dense sampling of areas. Two-stage detectors usually require a longer process time than one-stage detectors, but have higher 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), which is essential for people with BLV to understand their surroundings, avoid obstacles along their path, and further obtain a more comprehensive understanding of the indoor environment for them to walk around with both confidence and interaction with other people. Nevertheless, 2D detectors can be used as the basis for 3D object detection.

2.2 3D Object Detection

3D object detection usually estimates three-dimensional (3D) bounding boxes of objects in a scene. Compared to 2D object detection methods that use only RGB images, 3D object detection methods typically use depth images, stereo images, or point cloud data to obtain 3D information.

2.2.1 Image based 3D Object Detection

Several studies have applied variations of region segmentation algorithms for obstacle detection in depth images [4, 11]. Huang et al. [11] removed the noise of depth maps by applying morphological closing (dilation and erosion) and then used the standard least-squares method and V-disparity [25] method to estimate the ground curves and height threshold for obstacles. The difference in the depth value used to determine the stair edges and the connected component region growth method applied afterward to distinguish different objects. Cheng et al. [4] used a seeded region growth method with Sobel edges to detect ground and obstacles with a refreshing frequency of 10 frames per second (fps). In addition to the region growth method, the semantic segmentation method with a deep learning model was also applied for obstacle detection in RGB-D images [26], which can support real-time performance at 22Hz with an Nvidia GTX2080Ti GPU.

These methods can work in many scenarios; however, different lighting conditions can affect the estimated depth values and decrease the accuracy of the object locations. Furthermore, it is difficult to detect smaller objects and is unsuitable for real-time systems combined with navigation, because it requires a longer computation time for complex scenes.

Stereo images have also been used in obstacle detection. Chen [3] proposed a deep stereo geometry network (DSGN) that uses the extracted 2D image features at both the pixel and semantic levels to construct the plane-sweep volume (PSV) with depth estimation. The PSV is transformed into the 3D space to construct the 3D geometric volume, which is used to detect objects using a 3D neural network. It detected the objects for an image pair with NVIDIA Tesla V100 at an average process time of 0.682 seconds and an average depth error of 0.55 meters.

2.2.2 Point Cloud based 3D Object Detection

Point cloud data are more popular for object detection because they contain more information than depth or stereo images and have been widely used to reconstruct scenes. Pham et al. [18] used color-depth images with the accelerometer data to reconstruct the point cloud of a scene and then applied voxelization and a pass-through filter to remove noise. The random sample consensus (RANSAC) algorithm [6] was used to segment planes and detect the ground plane of the scene. Different algorithms were used to detect doors, stairs, walls, and other loose obstacles with the ground plane removed. Domenech et al. [5] used a voxel grid clustering approach to cluster the point cloud. They then obtained and combined the fractal dimension for each cluster to form a voxelized fractal descriptor, where the fractal dimension is the slope of the decrease in the box size value over the number of iterations. The support vector machine was then used with the

voxelized fractal descriptor to classify detected objects and reached 92.84% accuracy on ModelNet10 [31].

LiDAR (light detection and ranging) uses near-infrared light to create a 3D map of the environment. It has a low wavelength (nanometer range) of light signals that can detect smaller objects and results in a more accurate and precise point cloud than most sensors. Object detection using LiDAR has been increasingly used in recent years [7, 9]. He et al.[9] used the American Velodyne-16 line LiDAR to generate 300K points per second. The point will be sparse with the increase in the distance; therefore, a pass-through filter and voxel mesh method are used to filter the noise data. Similar to [18], the RANSAC method is used for plane segmentation, and then the K-D tree is utilized to cluster the remaining points and groups based on thresholding. Garnett et al. [7] proposed a unified deep convolutional network with LiDAR point cloud to achieve real-time (30 fps) in both categorical-based and general obstacle detection in the outdoor environment. They used a column-based approach for general obstacle detection with StixelNet [14] as the base, where the ground truth was determined by the bottom contour of each obstacle and the clear columns detected. Nevertheless, these solutions are primarily workable for autonomous driving; unfortunately, they are not easily usable by users in indoor environments.

Apple's augmented reality (AR) platform, ARKit [1], uses the visual-inertial odometry (VIO) technique to understand the iOS device's relationship with the real-world environment and track real-world objects. With scene analysis across captured 2D video frames and the camera's motion that is determined by the motion sensors, it can detect and track the visual feature points (i.e., distinctive markers) and estimate their 3D positions in the world coordinate system. Moreover, with the availability of LiDAR sensors in the recent version of iPhones, a more accurate point cloud can be obtained to improve

the detection result. With the capabilities of ARKit, our proposed 3D object detection system will be built on top of ARKit data.

Chapter 3

Method

3.1 System Overview

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.

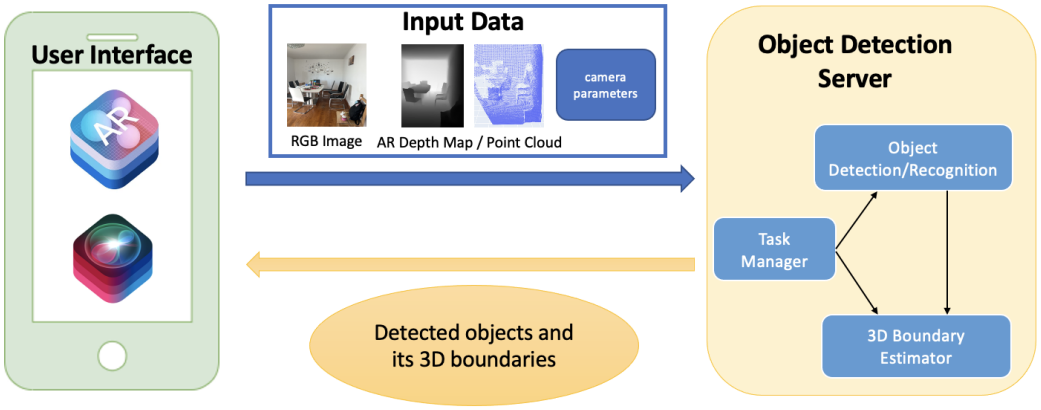

FIGURE 3.1: The system architecture of the 3D object detection, recognition, and presentation system works on mobile devices.

The system consisted of the following three modules (Figure 3.1):

3D Object Detection. This module processes the data (i.e., RGB image, AR depth map or point cloud, camera intrinsic and extrinsic parameters) collected from mobile devices. It utilizes an adaptive method based on the required task at hand and the available data types of the sensors to determine the 2D/3D bounding boxes of the objects and/or people in each camera frame. This module runs mainly on the cloud server; however, it can also run on users' mobile devices when they enter areas with poor network connections.

Object Tracking and Update. This 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 was created to enable users to search for an object or explore their surroundings through voice interaction. The application delivers voice information and AR visualization to inform users based on the search or filtered detection results.

3.2 3D Object Detection

The iOS-based ARKit platform consists of a powerful method, called world tracking [1], which tracks the device's position and motion through scene analysis across camera frames and comparing motion sensing data. As a result, it can detect the horizontal and vertical planes in the scene and obtain the AR point cloud - a set of notable 2D/3D feature points of the corresponding camera frames. Furthermore, LiDAR sensors in more

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

recent iOS devices[2] can capture 3D data of nearby scenes and generate a depth map of each corresponding camera frame with a resolution of 256x192.

As ARKit can map the world, our 3D object detection module is built on top of the data generated by ARKit. The proposed object detection method (Figure 3.2) integrates 2D bounding boxes and labels estimated by a pre-trained 2D object detector from each input RGB image and a point cloud or depth map collected by ARKit. In addition, the horizontal and vertical planes detected by ARKit were used to enhance the object detection results.

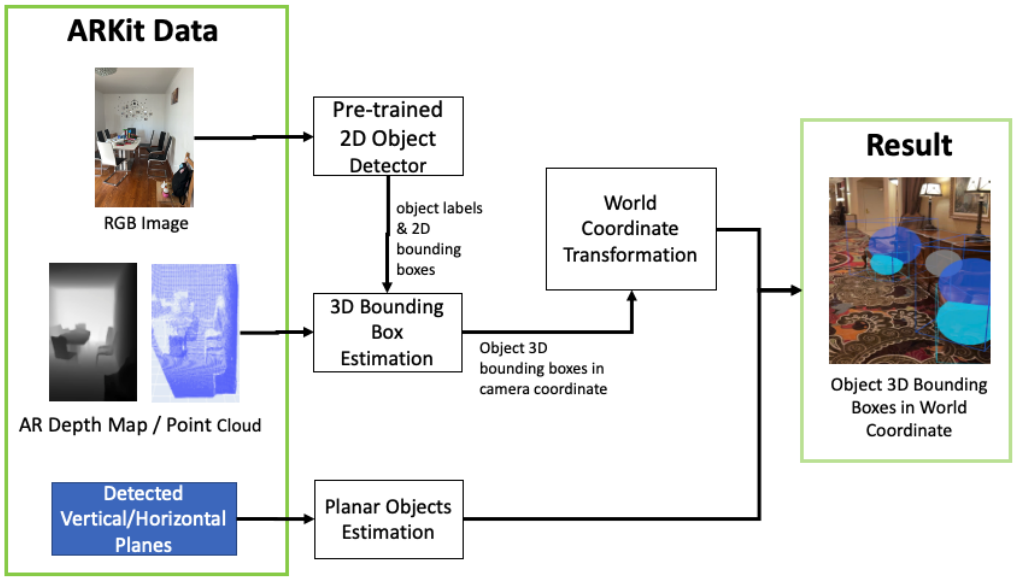

FIGURE 3.2: The general workflow of the 3D object detection and 3D bounding box estimation.

In a typical situation of assistive navigation, a mobile phone will need to run a mobile application in real time for the basic localization service. Processing 3D object detection within the same mobile device limits the flexibility to customize object detection methods based on different situations or request tasks and causes battery draining owing to the considerable computational power needed to support multiple functionalities simultaneously. Thus a cloud server was created to avoid these issues by processing the object

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

detection in the cloud. As communication with the cloud server would require a stable network connection to ensure real-time performance, the object detection process is also available in the mobile application as a backup for areas with poor network connections.

3.2.1 Planar Object Detection

ARKit can detect vertical and horizontal planes through 3D feature points collected over time and can further classify these planes into different classes, such as floor, wall, ceiling, and surface of rectangular objects such as a table. Our system considers each detected horizontal plane higher than the floor plane as an obstacle, as well as the vertical planes, which could be the wall or door. The 3D bounding boxes cannot be determined for the detected vertical planes as we usually miss the length or width information; thus, the system keeps track of the direction and distance between the user's camera location and the vertical plane.

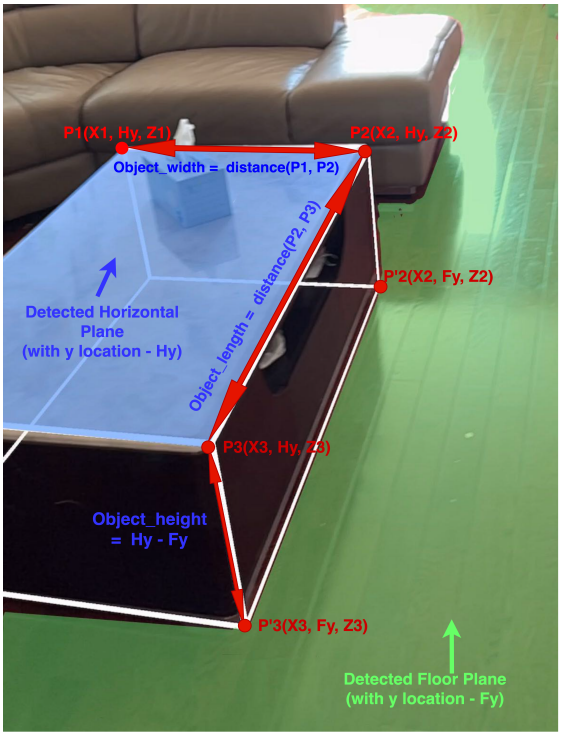

FIGURE 3.3: Estimated 3D bounding box (white rectangular box) of an object with the detected horizontal plane (blue) using the floor plane's (green) y-location (Fy).

The 3D bounding box for the object with a horizontal plane (e.g., the table with the detected blue horizontal plane in Figure 3.3) is determined by the four endpoints (P1, P2, P3, and P4) of its plane and the y location (Fy) (in the direction of gravity) of the detected floor plane (the green plane in Figure 3.3). The width and length of the object are the distances between P1 and P2, and P2 and P3, respectively, ignoring the y values. The height of the object is the difference between the horizontal plane's y location (Hy) and the floor plane's y location (Fy). The width and length of the object are updated along with the updated horizontal plane size over time (i.e., updated P1, P2, P3, and P4).

3.2.2 2D Object Detection

Plane detection can only detect objects with large and flat surfaces. Therefore, a pre-trained image object detection model was applied to detect other objects that would be further classified as obstacles, humans or targets, and then use our proposed method to determine the 3D bounding boxes of these objects. This thesis tested several object detection models, including YOLOv3 [20] and YOLOv5 [8]. However, the system can adapt to any pre-trained 2D object detector based on demand.

3.2.3 3D Bounding Box Estimation

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 detect the 3D bounding box of objects in the corresponding image frame.

3.2.3.1 Point Cloud based 3D Bounding Box Estimation

For mobile devices that do not contain LiDAR sensors, the AR point cloud of the corresponding frame is used to estimate the 3D bounding boxes of the objects. The typical range of the number of 3D feature points detected by each camera frame is between 0 and 200. Due to the limited number of feature points, we only processed the frame if it contained the number of feature points above a threshold (i.e., 30) after removing the points belonging to the already detected planar objects.

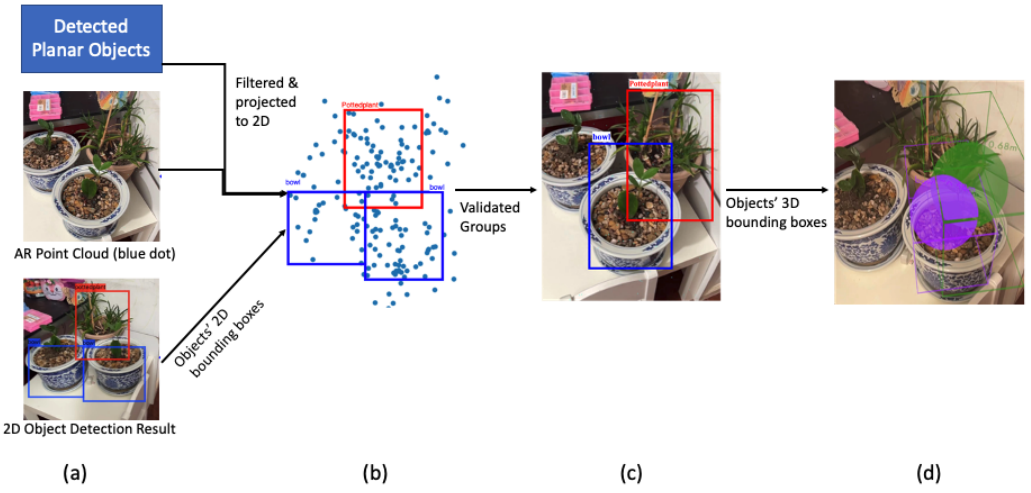

FIGURE 3.4: 3D bounding box estimation using the AR point cloud and 2D bounding boxes from 2D object detectors.

To convert the 2D bounding boxes of the detected objects into 3D bounding boxes, we first project the 3D raw feature points into the 2D image coordinate system (Figure 3.4). Second, we grouped the projected feature points based on the 2D bounding boxes of the detected objects (Figure 3.4 (a) to (b)). If the object group with the number of feature points is less than a certain threshold, its 3D bounding box is not computed for the current frame. For example, one of the bowls is ignored from the step in Figure 3.4 (b) to (c) because of the insufficient number of feature points. The 3D bounding box of the object is calculated using its feature point group's mean and standard deviation of the x, y, and z values (Figure 3.4 (c) to (d)). The resulting object's 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, which will be detailed in Section 3.3.1.

3.2.3.2 Depth Map based 3D Bounding Box Estimation

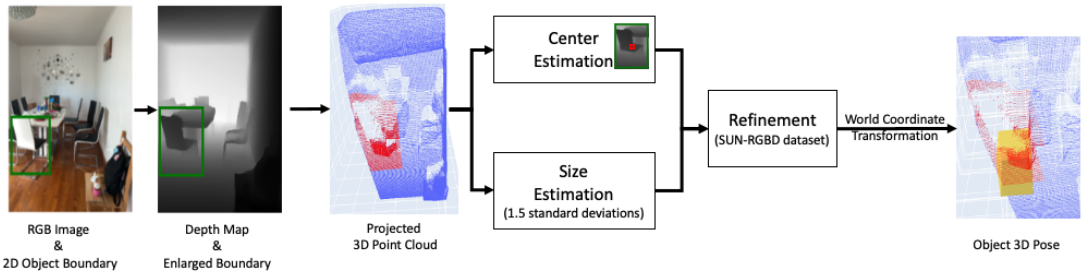

FIGURE 3.5: 3D bounding box estimation using the AR depth map and 2D bounding boxes from 2D object detectors.

The general pipeline for object detection using a depth map is shown in Figure 3.5. It first starts with the 2D object bounding box detected by the 2D object detector from the RGB image, and then converts 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, the depth map is projected onto the 3D point cloud using the camera's intrinsic parameters, where the points of the detected object are denoted in red (Figure 3.5). Subsequently, we estimate the object's 3D size and center location based on the object's point cloud. A 5x5 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 was 1.5 standard deviations of the x, y, and z value of its 3D point cloud. The 3D size of the resulting object was refined with the average object's 3D size computed from the SUN RGB-D dataset [24] (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 the object side is updated if it is within a certain deviation

of the corresponding average length. Finally, we converted the object's pose from the camera coordinates into world coordinates using the camera's extrinsic parameters.

3.3 3D Object Tracking

Tracking an object's 2D bounding box across frames 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 track in the new frame if the confidence score was below a threshold (i.e., 70%). If the confidence score is above the threshold, the new 3D bounding box of the object is determined using the 3D bounding box estimation method, as discussed in Section 3.2.3, with updated 2D bounding boxes, which can be either run on the mobile device or through the API call. The detection mode is 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 by more than 45 degrees from the latest detection location.

3.3.1 Point Cloud based Tracking

The ARKit point cloud is sparse and contains considerable noise owing to various circumstances in the context, such as colors, shadows, and lighting conditions. The sparsity

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

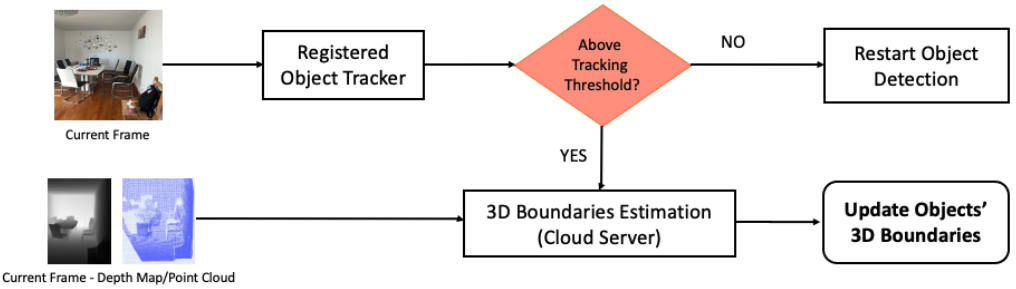

FIGURE 3.6: General workflow of the tracking process.

within the ARKit point cloud increases the error in estimating the 3D bounding box of the object from a single camera frame. Hence, we used the non-maximum suppression (NMS) method [22] for objects that contain five or more estimated 3D bounding boxes across multiple frames to obtain a more stable and accurate 3D bounding box (i.e., the validated bounding box). The object with the validated bounding box is passed through the information filtering process to consider for the voice notification and AR visualization.

3.3.1.1 3D Matching

When the app switches from tracking mode to detection mode with the point cloud data, previously detected/tracked objects might be detected again; 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 overlap for the bounding boxes of the same object. More importantly, the object may move.

Therefore, 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 its 3D distance will be updated. Otherwise, it is considered a new object and created a new object instance.

3.3.2 Depth Map based Tracking

Unlike the 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, only the dynamic moving objects were tracked. In this case, we assume that only people or objects with their 3D bounding box overlapping a person's bounding box are dynamic objects. 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.

3.4 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 computer 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 30fps. Note that the maximum speed of updating objects' 3D bounding boxes is 10fps in our current implementation, depending on the object detection and tracking run time.

3.4.1 AR Visualization

The AR mobile application uses ARKit to place AR assets based on the 3D bounding boxes of the detected object to provide a visual aid for people with low vision or ASD and as digital signage for different situations, such as task guidance and emergency evacuation.

FIGURE 3.7: 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.

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 3.7 shows some examples of AR visualization. The detected person is represented in red, as we feel a person is essential to show, especially for users with the challenge of social interaction.

AR visualization can be further improved, where the application also assists users in completing the assigned task. In combination with the object search feature, the application can display task instructions in the correct location based on the 3D pose of the target object. As shown in Figure 3.8, the AR instruction can be placed in the correct location with the detected 3D location of the tripod. With object detection available,

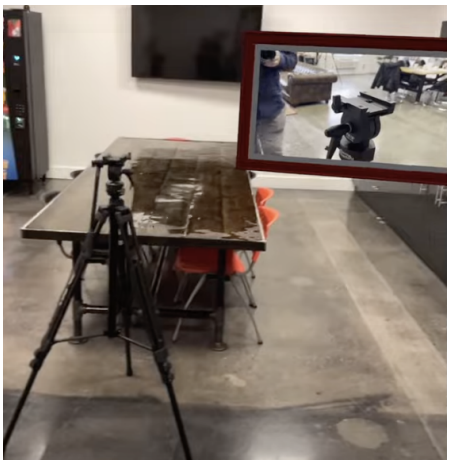

FIGURE 3.8: AR instruction example. The AR frame displays a video showing the user how to attach the camera to a tripod.

there is no need to pre-define the object's location or worry about the object changing its position over time. Additional use cases can be derived based on the capability of the AR visualization, which can be further discovered based on user studies.

3.4.2 Voice Interaction

Unlike the GUI interface provided for sighted users, touch-based interaction is challenging for users who are blind or have low vision; therefore, voice interaction is essential for making the application user-friendly to individuals with BLV. Voice interaction includes two major components: one receives the action command from the user, and the other presents the necessary information based on the action command.

Continuous listening to the user's voice command not only consumes extra computational power of the mobile device, but this might also not be correctly reflected in the user command as environmental noise can also be recorded. Therefore, instead of creating a new voice interaction system, we utilized the well-developed SiriKit [4] library to allow the user to open and give the in-app command for our application through Siri. It can also

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

reduce the requirement for the user to learn a new way of dealing with the app and save power to create another instance of continuing to listen to the user's voice command.

In addition to using SiriKit for initializing app activities, we also created our voice interaction within the app to handle specific commands to minimize the requirement for users to say 'Hi Siri' whenever they want to communicate with our app. It provides us the flexibility to create in-app actions. Our voice interaction can confirm the user action, determine the target object users are looking for, and provide users with information about detected objects. A video demonstration of the voice interaction is provided in Section 4.4.

3.4.2.1 Information Filtering

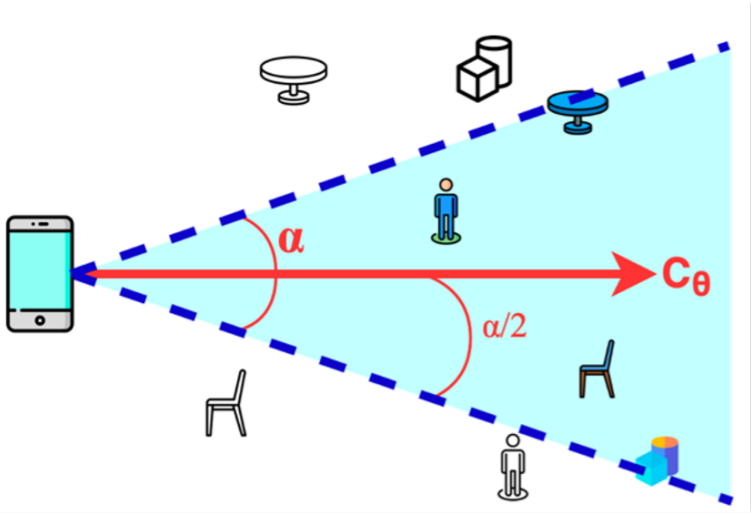

FIGURE 3.9: Information filtering process. Filter the objects within the angular range(α) with respect to the camera orientation (C_θ).

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 the direction of the user device for all validated objects detected. Second, the remaining objects are sorted by alert priority using Equation (3.1), which is calculated based on

the distance between the user's camera (C) and an object (O), as well as the object size. The distance has more weight than the object size ($\alpha \gg \beta$). Third, the voice notification is provided if the alert priority value exceeds a threshold.

$$AlertPrority(C, O) = \frac{\alpha}{distance(C, O)} + \beta * volume_0 \tag{3.1}$$

$$Distance(C, O) = \sqrt{(C_X - (O_{center_x} - \frac{O_{width}}{2}))^2 + (C_Z - (O_{center_Z} - \frac{O_{length}}{2}))^2} \tag{3.2}$$

The distance and direction of the alert object were provided in the voice notification. The distance between the object and camera is computed using Equation (3.2), where the differences in the y-axis are not considered when calculating the distance.

The orientation of the objects was calculated based on the difference between the y-direction of the current camera and the device to the object. The object direction (i.e., front, left, or right) was determined based on the orientation angle difference. Information filtering is performed whenever a new voice alert whenever a new object is detected or any object is within the dangerous range of the user's location.

Chapter 4

Experiments

We conducted several experiments to evaluate the proposed system's object detection accuracy and real-time performance. The experiments were conducted in open public areas and private residential areas, with consent from subjects who will be recorded in our video. We have also planned a series of usability experiments, but due to the limited time and other constraints such as pandemics, it will be conducted in the future.

4.1 Object Detection Evaluation

To evaluate our object detection accuracy, we explored several datasets of indoor scenes containing 3D object annotation, particularly SUN RGB-D [24] and NYUDv2 [23]. The typical depth map resolution collected by these datasets is 640x480, which is approximately six times the size of the depth map collected by the mobile device we used (256x192) and much larger than the AR point cloud data (approximately 200 points per frame). Due to this significant difference, the existing datasets cannot be used to evaluate the performance of our object detection method. Therefore, we developed a

data collection and annotation toolkit to construct our dataset using the mobile device sensors.

4.1.1 Data Collection and Annotation Toolkit

The data collection and annotation toolkit includes an iOS app for data collection and a data annotation interface for annotating the 3D bounding boxes of objects. The iOS app collects data, including the RGB image, AR point cloud, depth map, and intrinsic and extrinsic parameters of the camera for each dedicated frame.

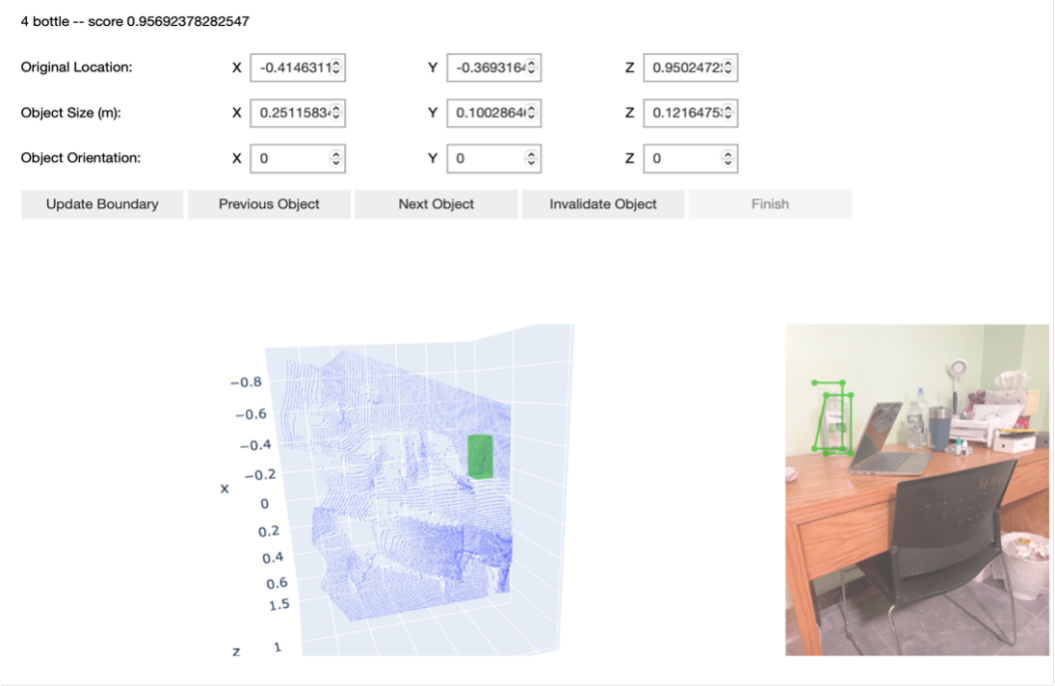

FIGURE 4.1: The data annotation interface. The 3D location, size, and orientation of the current annotated object are shown at the top. The bottom shows the corresponding image frame's point cloud and RGB image, where the green line segments and rectangular box represent the 2D and 3D positions of the annotated object, respectively.

The annotation interface was created using Python. It preprocesses the data to estimate a set of 3D object bounding boxes and allows the annotator to make corrections. The system first processes the RGB image with the Mask R-CNN model to obtain the object segmentation mask and then aligns it with the depth map to obtain the object's 3D point

	< 3m	3m $\sim$ 5m	5m $\sim$ 7m	> 7m
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

TABLE 4.1: Number of objects collected based on the distance range. The distance is calculated based on the camera location and the object center.

cloud. Next, we use our proposed method to pre-label the 3D bounding boxes of objects based on object point cloud data. Then, the interface 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. At this point, the annotator looked at the object one after another to correct the bounding boxes and invalidate the object if needed. We used this hybrid approach to reduce the burden of the data labeling process.

This annotation tool allows us to construct an object dataset based on mobile devices that can be used to train a deep learning model for more accurate predictions based on different purposes in the future.

4.1.2 Object Detection Accuracy Analysis

For the experiments, we collected data from a residential house, a number of public stores, and a street, using our data collection application. We obtained the ground truth 3D bounding boxes of the objects using our annotation interface. The objects we collected are shown in Table 4.1.

The accuracy of the depth map is closely related to the performance of 3D object detection because the depth values are critical in estimating the 3D pose of an object from its 2D detection result. Inaccurate depth values can result in incorrect 3D bounding boxes for the objects. Therefore, we evaluated the confidence level of the accuracy of depth data based on the collected experimental data.

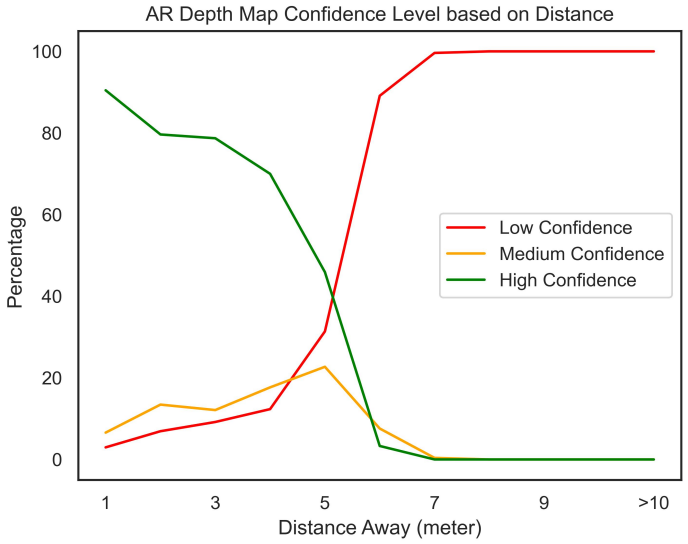

FIGURE 4.2: Distributions of the three confidence levels for accuracy of the AR depth values versus distances. At each distance, the three percentage numbers add up to 100%.

The depth map generated by the built-in LiDAR scanner of the selected iOS devices is usually accurate within 5 meters and can detect up to 15 meters, even though the depth accuracy decreases with increasing distance. The ARKit framework uses three confidence levels to indicate the accuracy of the collected depth data: low, medium, and high. We separated the distances into four major ranges: less than 3 meters, between 3 and 5 meters, between 5 to 7 meters, and greater than 7 meters. Objects with 3 meters are more important for people with BLV to know as they might bump into these objects if they are not altered; it should also be noted that the Microsoft Kinect sensors used in [18] have the limitation of inaccurate distance information beyond 3 meters. Based on the primary search, the built-in LiDAR sensor of iOS devices is usually accurate

	Distance Range			
Confidence Level	< 3m	3m $\sim$ 5m	3m $\sim$ 5m	> 7m
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%

TABLE 4.2: Distributions of the three confidence levels for accuracy of the AR depth value based on distance ranges

Distance Range	Number of Objects	Our Method (mAP)	Transferable3D (mAP)
< 3m	94	63.88%	38.13%
3m $\sim$ 5m	13	50.00%	33.33%
5m $\sim$ 7m	0	-	-
> 7m	8	37.50%	0.00%

TABLE 4.3: Object detection performance based on the distance range.

within 5 meters; thus, we set 5 meters as another break for our distance range. From the experiments, the farthest distance can be detected in the experimental residential house is within 7 meters; thus, 7 meters is the break for object detection between the indoor and outdoor environments. Figure 4.2 shows the distributions (in percentages) of the three confidence levels of the depth value accuracy versus distances based on our experiment data. Table 4.1 shows these breakdowns. Table 4.2 lists the percentages of numbers of points in the three confidence levels for accuracy of the AR depth value based on distance ranges. At each distance, the three percentage values add up to 100%.

We calculated the interest over union (IoU) of the predicted 3D object boundaries and the ground truth to evaluate our object detection method. The prediction is considered correct if the IoU is greater than 25%. We compared our results with one of the state-of-the-art methods, the transferable semi-supervised 3D object detection model (hereafter called Transferable3D) [27], which can predict other untrained objects. Because our annotated data were relatively small and unable to train Transferable3D, we used the SUN RGB-D dataset[24] to train Transferable3D.

	Object Perspective					
	Left		Center		Right	
Object Class	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 4.4: Object detection performance for objects capture from different perspectives.

The data in Table 4.1 were captured from random scenes in the experimental area since we wanted to test how our system can perform in the real-world environment. Therefore, the distribution of each object class is uneven in the annotated data, as shown in Table 4.1. This causes some objects to have a more significant influence than others, such that a bad prediction for a vase can dramatically decrease the mAP value. In addition, our data are much sparser than the SUN RGB-D dataset, so it is reasonable that Transferable3D does not perform as well as the SUN RGB-D dataset (41.8% mAP). Furthermore, the SUN RGB-D dataset contains only indoor objects and does not have any data for cars; thus, Transferable3D performed poorly with car prediction. However, with more data, we can refine Transferable3D to achieve better performance. Nevertheless, our method can achieve an mAP of 63.88% for objects within 3 meters, 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 performance of objects captured from different perspectives. In particular, we tested the laptop and chair, because both objects have different shapes when viewed from different angles.

From Table 4.4, there is not much difference in the average IoU for laptops or chairs when looking from different perspectives. The difference in average precision (AP) is

also small because it is a relatively small data sample, where one sample can greatly influence the AP value. However, the width and length of the laptop or chair are very similar, so the resulting 3D bounding boxes could still cover the objects at a different angle when they were placed in the nearby center.

4.2 Real Time System Performance Analysis

To evaluate the real-time performance of our system, we created an iOS app that can process object detection either on a mobile device or through a cloud server. We tested three different 2D object detectors (YOLOv3, YOLOv5 Nano, and YOLOv5 Medium) under two image resolutions (640x480 and 256x192). All experiments were performed using an iPhone 13 Pro Max. The object detection server ran at AWS Elastic Beanstalk using a t3 large instance containing two virtual central processing units (vCPU) and 8GB of memory.

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

Table 4.5 shows the run time based on a different combination of object detectors and image resolutions using the depth map data, where the point cloud data has a similar process time. The cloud server run time consists of an additional time for data transfer computed based on the time when the API call was initiated in the mobile app and the time when the cloud server received the API request and loaded the data. The "Mobile Device Run Time" in Table 4.5 is the total process time of the object detection module in the iPhone 13 Pro Max. The iPhone 13 Pro Max uses the A15 Bionic processor with a 6-core CPU and 5-core GPU, which in fact, has more computational power than

		YOLOv3 (248.4MB)		YOLOv5 Nano (7.9MB)		YOLOv5 Medium (85.1MB)	
Image Resolution		640x480	256x192	640x480	256x192	640x480	256x192
Num of Object Detected		7.4±3.8	5.3±2.1	3.6±1.9	3.1±1.9	5.8±3.4	4.4±2.4
Mobile Device Run Time (ms)		85±4	80±4	54±15	50±7	68±13	62±4
Cloud Server Run Time(ms)	Data Transfer	399±130	257±80	390±88	262±128	394±96	249±91
	Computation	2078±48	2035±48	188±26	171±22	791±26	773±23
	Total	2478±138	2291±97	578±96	433±131	1185±101	1022±92

TABLE 4.5: Comparison of system run time based on the model type and image resolution.

the cloud server host in the AWS, which has only two vCPU and 8GB of memory. In real-world applications, the cloud server can have much more powerful hardware. However, data transfer might be a bottleneck unless the mobile device lacks the power and computational budget to process object detection/tracking while performing basic localization services. Additionally, for the 2D object detector to run on the mobile device, it needs 2 to 3 seconds to initialize the model.

The 2D object detectors greatly influenced the run time, whereas the large model (i.e., YOLOv3) took nearly double the time to process in the mobile device and more than 10x time to process in the cloud server as the small model (i.e., YOLOv5 Nano). However, the larger model detected more objects than the smaller one in the same scenario.

The image resolution also affects the computation run time but is minor compared to the model type. However, image resolution becomes an important factor when aiming for real-time performance when using the object detection server. The 256x192 image resolution is usually around 50KB, whereas the 640x480 image resolution is around 300KB. With around 250KB difference in data size, it took more than 100ms to transfer the data to the cloud server. The difference in the data transfer time for the same image

resolution in Table 4.5 is due to the fluctuation of the internet speed. This data transfer time can be reduced with faster internet and edge computing techniques. Furthermore, a lower image resolution can also affect the performance of 2D object detectors, as shown in Table 4.5, 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 satisfy different user requirements in terms of run time and details for the 3D detection result. In addition, multiple cloud servers can be deployed to satisfy different user requirements.

4.3 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 goal of the experiment was to collect user feedback for voice interaction and the user interface with AR visualization. The baseline of the experiment was to let participants to explore an unfamiliar environment with their current tools, such as a white cane or guide dog for a blind user. Next, we let participants use our application to explore another similar environment. A user survey will be conducted at the end of each experiment to ask participants about their understanding of the environment, their challenges during exploration, and their suggestions for our application.

We also want to experiment with 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 the participants in a similar environment. We will collect various

data for each combination of the metrics, including the self-evaluated anxiety level, the time needed to explore the scene, the number of times hitting an obstacle, and times 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.

4.4 Demo

A video demo can provide better visualization of our system performance in the mobile applications; it can be viewed by clicking on this link. The demo is taken place in a residential house that shows two primary use cases:

- **Object Search.** The user opens and activates the object search using the 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 a laptop after confirming it with the user. The application provides the 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 while the user moves.

Chapter 5

Conclusions

In this thesis, we proposed a 3D object detection, recognition, and presentation system that operates on a mobile device. This system does not require specialized high-end sensors and works efficiently in real time while containing voice and AR visualization to assist the delivery of information to individuals with BLV, ASD, or other challenges.

A 3D object detection server was introduced to increase the flexibility to adapt to different data inputs and object detectors based on different requirements of run time and details for the 3D detection result, as well as to release the computational requirements for the mobile device and enable the system to be integrated with other applications, such as navigation applications. Moreover, the object detection module is available in the mobile app to handle areas with poor network connections.

Our data collection and annotation toolkit was created to construct an object dataset based on mobile device sensors. This dataset can be used to train a deep learning model to improve object detection accuracy. Additionally, the data collection application can be integrated with the modeling application [32] that we developed previously for indoor navigation to create a 3D reconstruction of the scanning environment by including the

static object's location and size on top of the 2D floorplan and AR model. The modeling app will also collect data for object detection along with environment scanning, and our object detection module processes the collected data to obtain the object's 3D poses and size for each frame. Subsequently, a merge module is created to match and refine the 3D bounding boxes of the same objects, using the detected object's location in the AR coordinate system. Finally, the 3D locations and sizes of the objects can be projected onto the 2D floorplan using the transformation matrices obtained in [32].

5.1 Future work

There is much room for future improvement in the current system. The first and most crucial step was to conduct the user experiments discussed in Section 4.3 to improve the usability of the mobile application. This is a critical step in validating whether our system can satisfy user needs and identifying the weaknesses of our system as a real-world application.

Next, and is the initial motivation of this thesis, integrating this system with our indoor navigation system, Nearabl, to improve the usability for users with BLV, ASD, or other challenges in indoor area exploration and navigation. In addition, integrating the data collection application with the modeling application would be ideal for constructing the 3D mapping of the scanned environment and expanding our dataset. This can eventually be used to develop a deep learning model to improve object detection accuracy.

Finally, we would like to explore finger tracking to provide an alternative method for user interaction, as it is sometimes challenging to deliver the spatial direction to a blind user with audio.

Bibliography

[1] Apple Inc. *Arkit - augmented reality.* [Online]. Available from: https://developer.apple.com/augmented-reality/arkit/.

[2] R. Bourne, J. D. Steinmetz, S. Flaxman, P. S. Briant, H. R. Taylor, S. Resnikoff, R. J. Casson, A. Abdoli, E. Abu-Gharbieh, A. Afshin, et al. Trends in prevalence of blindness and distance and near vision impairment over 30 years: an analysis for the global burden of disease study. *The Lancet global health*, 9(2):e130–e143, 2021.

[3] Y. Chen, S. Liu, X. Shen, and J. Jia. Dsgn: Deep stereo geometry network for 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12536–12545, 2020.

[4] R. Cheng, K. Wang, K. Yang, and X. Zhao. A ground and obstacle detection algorithm for the visually impaired. In *2015 IET International Conference on Biomedical Image and Signal Processing (ICBISP 2015)*, pages 1–6. IET, 2015.

[5] J. F. Domenech, F. Escalona, F. Gomez-Donoso, and M. Cazorla. A voxelized fractal descriptor for 3d object recognition. *IEEE Access*, 8:161958–161968, 2020.

[6] 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.

[7] N. Garnett, S. Silberstein, S. Oron, E. Fetaya, U. Verner, A. Ayash, V. Goldner, R. Cohen, K. Horn, and D. Levi. Real-time category-based and general obstacle detection for autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 198–205, 2017.

[8] Glenn Jocher, Ayush Chaurasia, Alex Stoken, Jirka Borovec, NanoCode012, Yonghye Kwon, TaoXie, Jiacong Fang, imyhxy, Kalen Michael; Lorna, Abhiram V, Diego Montes, Jebastin Nadar, Laughing, tkianai, yxNONG, Piotr Skalski, Zhiqiang Wang, Adam Hogan, Cristi Fati, Lorenzo Mammana, AlexWang1900, Deep Patel, Ding Yiwei, Felix You, Jan Hajek, Laurentiu Diaconu, Mai Thanh Minh. *ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference*. [Online]. Available from: https://doi.org/10.5281/zenodo.3908559/.

[9] C. He, J. Gong, Y. Yang, D. Bi, J. Lan, and L. Qie. Real-time track obstacle detection from 3d lidar point cloud. In *Journal of Physics: Conference Series*, volume 1910(1), page 012002. IOP Publishing, 2021.

[10] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[11] H.-C. Huang, C.-T. Hsieh, and C.-H. Yeh. An indoor obstacle detection system using depth information and region growth. *Sensors*, 15(10):27116–27141, 2015.

[12] K. Koldewyn, S. Weigelt, N. Kanwisher, and Y. Jiang. Multiple object tracking in autism spectrum disorders. *Journal of autism and developmental disorders*, 43(6):1394–1405, 2013.

[13] Laser Eye Surgery Hub. *Visual impairment & blindness global data & statistics*. [Online]. Available from: https://www.lasereyesurgeryhub.co.uk/data/visual-impairment-blindness-data-statistics/.

[14] D. Levi, N. Garnett, E. Fetaya, and I. Herzlyia. Stixelnet: A deep convolutional network for obstacle detection and road segmentation. In *BMVC*, volume 1(2), page 4, 2015.

[15] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[17] R. Manduchi and S. Kurniawan. Watch your head, mind your step: mobility-related accidents experienced by people with visual impairment. *Dept. Comp. Eng., Univ. California,*

Santa Cruz, Tech. Rep, 2010.

[18] H.-H. Pham, L. Thi-Lan, and N. Vuillerme. Real-time obstacle detection system in indoor environment for the visually impaired using microsoft kinect sensor. *Journal of Sensors*, 2016, 2016.

[19] E. Quintana, C. Ibarra, L. Escobedo, M. Tentori, and J. Favela. Object and gesture recognition to assist children with autism during the discrimination training. In *Iberoamerican congress on pattern recognition*, pages 877–884. Springer, 2012.

[20] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[21] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[22] R. Rothe, M. Guillaumin, and L. V. Gool. Non-maximum suppression for object detection by passing messages between windows. In *Asian conference on computer vision*, pages 290–306. Springer, 2014.

[23] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *European conference on computer vision*, pages 746–760. Springer, 2012.

[24] 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*, pages 567–576, 2015.

[25] N. Soquet, D. Aubert, and N. Hautiere. Road segmentation supervised by an extended v-disparity algorithm for autonomous navigation. In *2007 IEEE Intelligent Vehicles Symposium*, pages 160–165. IEEE, 2007.

[26] L. Sun, K. Yang, X. Hu, W. Hu, and K. Wang. Real-time fusion network for rgb-d semantic segmentation incorporating unexpected obstacle detection for road-driving images. *IEEE Robotics and Automation Letters*, 5(4):5558–5565, 2020.

[27] 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*, pages 1931–1940, 2019.

[28] J. N. van der Geest, C. Kemner, G. Camfferman, M. N. Verbaten, and H. van Engeland. Eye movements, visual attention, and autism: a saccadic reaction time study using the gap and overlap paradigm. *Biological Psychiatry*, 50(8):614–619, 2001.

[29] Vision Atlas. *The International Agency for the Prevention of Blindness*. [Online]. Available from: https://www.iapb.org/learn/vision-atlas/.

[30] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*, 2022.

[31] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[32] Z. Zhu, J. Chen, L. Zhang, Y. Chang, T. Franklin, H. Tang, and A. Ruci. iassist: An iphone-based multimedia information system for indoor assistive navigation. *International Journal of Multimedia Data Engineering and Management (IJMDEM)*, 11(4):38–59, 2020.