



Real-Time 3D Object Detection and Recognition using a Smartphone

Jin Chen¹^a and Zhigang Zhu^{1,2}^b

¹Visual Computing Laboratory/Data Science and Engineering Program, Computer Science Department,
The City College of New York - CUNY, New York, NY 10031 U.S.A.

²PhD Program in Computer Science, The Graduate Center - CUNY, New York, NY 10016, U.S.A.

Keywords: 3D Object Detection, 3D Object Tracking, Obstacle Detection, Assistive Computer Vision.

Abstract: Real-time detection of 3D obstacles and recognition of humans and other objects is essential for blind or low-vision people to travel not only safely and independently but also confidently and interactively, especially in a cluttered indoor environment. Most existing 3D obstacle detection techniques that are widely applied in robotic applications and outdoor environments often require high-end devices to ensure real-time performance. There is a strong need to develop a low-cost and highly efficient technique for 3D obstacle detection and object recognition in indoor environments. This paper proposes an integrated 3D obstacle detection system implemented on a smartphone, by utilizing deep-learning-based pre-trained 2D object detectors and ARKit-based point cloud data acquisition to predict and track the 3D positions of multiple objects (obstacles, humans, and other objects), and then provide alerts to users in real time. The system consists of four modules: 3D obstacle detection, 3D object tracking, 3D object matching, and information filtering. Preliminary tests in a small house setting indicated that this application could reliably detect large obstacles and their 3D positions and sizes in the real world and small obstacles' positions, without any expensive devices besides an iPhone.

1 INTRODUCTION


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 or low-vision (BLV) population has continuously grown over the past three decades and is expected to increase significantly as the population ages (Bourne *et al.*, 2020). There are an estimated 49.1 million people who are blind in 2020 globally, and 255 million people with moderate or severe visual impairment (MSVI), with visual acuity worse than 6/12 to 6/18 (Bourne *et al.*, 2020).


Being able to travel safely in an indoor environment is essential for BLVs to be successful in their jobs and assist their daily activities for a better quality of life. An indoor environment can be cluttered and changed more often than an outdoor environment. BLVs can easily bump into obstacles with limited space for movement in an indoor area. White canes (long canes) and guide dogs are the most common tools used to detect obstacles and navigate.

However, these tools cannot provide comprehensive information about the surroundings to help them avoid dangerous areas. Based on a survey of 300 legally blind or blind individuals (Manduchi and Kurniawan, 2010), over 40% of respondents experienced head-level accidents at least once a month, even with a white cane or a guide dog.

Furthermore, indoor environments are subject to more changes than outdoor environments, where objects and people move around. Many navigation applications cannot immediately reflect the environmental changes when planning paths. This may result in obstacles presenting along the intended routes, which increases the risk of BLV users bumping into obstacles when following guidance. Therefore, a real-time obstacle detection application is essential for improving safety while travelling.

Even though safety is the number one need, recognition of humans and other objects would also enhance BLVs' understanding of their surroundings, thus significantly improving their quality of life. Object detection is a fundamental topic in computer vision that has been extensively studied. Numerous

^a <https://orcid.org/0000-0003-1810-3828>

^b <https://orcid.org/0000-0002-9990-1137>

techniques have been developed and applied for multiple applications, particularly in autonomous vehicles.

Real-time 2D object detection techniques are well developed; however, 3D object detection helps BLVs better understand their surroundings, select a safe path, enjoy the environment, and interact with other people. However, multiple challenges exist in converting the 2D bounding boxes detected by 2D object detectors into 3D bounding boxes. Objects in the same category (e.g., chairs or tables) can have various appearances, shapes, and sizes. Real-time 3D object detection and recognition often involves high-end devices that are too expensive for BLVs to afford and/or too heavy to carry. Data show that 89% of BLV people live in low- or middle-income countries (Ackland *et al.*, 2017). 3D object detection often uses deep learning models, which require enormous computational power to ensure real-time performance. Moreover, most state-of-the-art 3D object recognition methods focus on the outdoor environment, owing to the high demand for autonomous driving. There is a lack of effective real-time techniques for detecting and tracking 3D objects in indoor environments.

Information filtering is another problem for BLV users with multiple detected objects, especially in real-time detection, where the detection information changes rapidly over time. BLV users may become confused and frustrated with the massive amount of information provided and may easily miss the information they need; thus, only reporting the essential information to BLV users is critical.

This paper proposes a low-cost and efficient real-time 3D obstacle detection iOS application for BLV users to help them overcome these problems and increase their accessibility in indoor areas. The main contributions of this study are as follows:

1. A real-time 3D obstacle detection and object recognition system integrating ARKit and a pre-trained 2D object detection model working in the iOS device.
2. 3D object tracking method utilizing the 2D image tracker and AR point cloud to improve the time performance and object identification.
3. An information filtering method extracting and reporting only essential information for BLV users from a set of massive objects detected over time.

The remainder of this paper is organized as follows. Section 2 provides a brief overview of the state-of-the-art 2D and 3D object detection methods. Section 3 describes the proposed 3D obstacle

detection and object recognition system. The results of our method and discussion are provided in Section 4. Finally, a summary of the proposed system and a discussion of future work are presented in Section 5.

2 RELATED WORK

2.1 2D Object Detection

2D object detection predicts the class labels and bounding boxes of objects from an image input. There are two main types of object detectors: two-stage and one-stage. R-CNN families (e.g., Fast R-CNN (Girshick, 2015) and Mask R-CNN (He *et al.*, 2017)) are two-stage object detectors where detection begins by extracting the region of interest (ROI) using a regional proposal network (RPN). In the second stage, the bounding boxes of the objects are refined using regression and classified into different classes. One-stage object detectors (e.g., YOLO (Redmon and Farhadi, 2018), SSD (Liu *et al.*, 2016), RetinaNet (Lin *et al.*, 2017)) do not have an ROI extraction stage and directly detect the bounding boxes of the objects from the image with a dense sampling of areas. Two-stage detectors usually require a longer process time than one-stage detectors but have higher accuracy (Soviany and Ionescu, 2018).

2D object detection techniques are sufficiently mature to support real-time performance. However, 2D bounding boxes with object labels do not provide sufficient information for BLVs to obtain a more comprehensive understanding of the indoor environment they are in to help them avoid obstacles along their path and 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 bounding boxes for objects in a scene. Compared with 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.

Denosing and ground detection are essential steps in 3D detection approaches for all the data types. (Huang *et al.*, 2015) removed depth map noise with morphological closing (dilation and then erosion), and then used the standard least-squares method and V-disparity method (Soquet *et al.*, 2007) to estimate ground curves and a height threshold for obstacles.

The region growth method was subsequently applied to distinguish different objects. (Cheng *et al.*, 2015) applied a seeded region growth method with Sobel edges to detect the ground and obstacles using depth maps generated from an RGB-D camera for obstacle detection. Their method had a refreshing frequency of 10 frames per second (fps). (Sun *et al.*, 2020) applied a semantic segmentation method with a deep learning model to RGB-D images that could support real-time performance at 22Hz using an Nvidia GTX2080Ti GPU for obstacle detection.

The deep stereo geometry network (DSGN) uses the 2D image features extracted from stereo images at both the pixel and semantic levels to construct the plane-sweep volume (PSV) with depth estimation (Chen *et al.*, 2020). A 3D geometric volume was constructed from the PSV and used to detect objects using a 3D neural network. The method has an average 0.682s process time per frame and an average depth error of 0.55m with NVIDIA Tesla V100.

Different lighting conditions can affect the estimated depth values and decrease the accuracy of the object positions in these methods. Furthermore, the object class information is missing, and smaller objects are not detected well in these methods. Therefore, they are not suitable for real-time systems combined with navigation purposes, as they consume enormous computational power in complex scenes.

2.2.1 3D Point Cloud Data

Point cloud obstacle detection is more popular for indoor obstacle detection as it contains richer information than depth maps. (Pham *et al.*, 2016) used RGB-D images with accelerometer data to reconstruct the point clouds of a scene. Voxelization and pass-through filters were applied to remove noise and the random sample consensus (RANSAC) algorithm (Fischler and Bolles, 1981) was used to segment planes and detect the ground plane of the scene. With the ground plane removed, various algorithms were applied to detect doors, stairs, and other loose obstacles.

In addition to depth images, LiDAR has been used more frequently in recent years as it constructs a more accurate point cloud and contains more information, such as intensity and angle. (He *et al.*, 2021) used the American Velodyne-16 line LiDAR, which can generate 300K points per second. A pass-through filter and voxel mesh method were used to filter the noise data as the sparsity of point cloud increased with distance. RANSAC and K-D trees were applied for plane segmentation and object clustering with thresholding. (Garnett *et al.*, 2017) proposed a unified

deep convolutional network with a LiDAR point cloud to achieve real-time performance (30fps) for both categorical-based and general obstacle detection in outdoor environments. It uses a column-based approach for general obstacle detection and StixelNet (Levi *et al.*, 2015) as the base. Nevertheless, these solutions are primarily workable for autonomous driving but are not readily usable for BLV users in indoor environments.

2.2.2 AR Point Cloud

Apple's augmented reality (AR) platform, ARKit (Apple Inc.), contains an AR point cloud with all detected feature point (i.e., distinctive markers) positions in the 3D camera coordinates. It generates a sparse point cloud for every camera frame that contains only features captured in a single frame. Despite its sparsity, it can detect horizontal and vertical planes in the scene, which is useful for detecting large obstacles. The user's movement and the captured 2D video frames can detect and track visual feature points to estimate their position in the real-world coordinate system. With the capabilities of ARKit, our proposed 3D object detection system will be built on top of ARKit.

2.3 3D Object Tracking

Unlike 2D trackers (Karunasekera *et al.*, 2019; Marques, 2020) which track objects across multiple frames, 3D trackers use point cloud data that often apply 3D Kalman filters to estimate object positions.

The IMM-UKF-JPDAF-based tracker (Sualeh and Kim, 2020) combines the unscented Kalman filter and joint probabilistic data association filter (Rezatofighi *et al.*, 2015) for the state estimation of multiple objects with a Gaussian assumption distribution. A clustering technique is applied to mitigate potential combinatorial explosions. (Wang and Wu, 2021) proposed a switching reference point method with both centroid and corner points for a LiDAR-based tracking system to handle abrupt changes in the position of an object. In our work, we modified the 2D tracking technique (Marques, 2020) for 3D tracking for the sake of time efficiency.

3 METHODS

The 3D obstacle detection and object recognition system consisted of four modules (Fig. 1): (1) 3D obstacle detection, (2) 3D object tracking, (3) 3D object matching, and (4) information filtering.

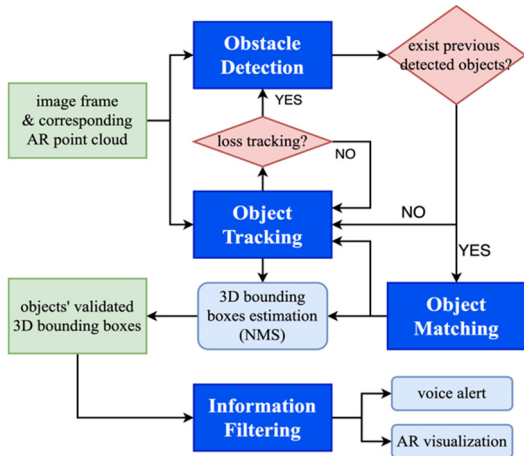


Figure 1: Workflow of the 3D obstacle detection and object recognition system.

The 3D obstacle detection module integrates the 2D bounding boxes and labels from the 2D object detector (YOLOv3) and point cloud and detected plane from ARKit to generate the 3D bounding boxes of the objects for each image frame. The 3D object tracking module tracks each object’s 2D bounding box from the current frame to the next frame and augments the AR point cloud data to generate the tracked 3D bounding box of each object. Next, in the object matching module, newly detected objects in the current frame are matched with the detected objects in the previous frames to reduce duplications of multiple instances of the same objects. Finally, the information filtering module determines the essential information to guide BLV users to avoid obstacles from the set of objects detected over time.

3.1 3D Obstacle Detection

The proposed 3D obstacle detection module utilizes both the point cloud generated by ARKit and 2D bounding boxes estimated by a pre-trained YOLOv3 object detector³ to obtain a rough estimation of the 3D bounding boxes of the detected objects. This module comprises three steps: plane segmentation, 2D object detection, and 3D bounding box determination.

An ARKit 3D model constructs a point cloud from the feature points collected over the camera frames to the current frame, and can estimate vertical and horizontal planes with sufficient 3D points collected. The floor plane determined based on the sizes and positions of all detected planes. The other detected planes were considered to be obstacles. The system cannot estimate the 3D bounding boxes for the

detected vertical planes because of the missing length or width information in the captured scene. The system aims to guide BLV users to avoid obstacles, so it tracks the distance between the device position and vertical planes. The 3D bounding box of an object with a horizontal plane (Fig. 2) is determined by the four endpoints (P1, P2, P3, and P4) of the horizontal plane and floor plane y location (F_y). The width and length of the object are the distances between P1 to P2 and P2 to P3, respectively, ignoring the y values:

$$Distance(P_i, P_j) = \sqrt{(X_i - X_j)^2 + (Z_i - Z_j)^2} \quad (1)$$

The height of the object is the difference between the horizontal plane y-location (H_y) and floor plane y-location (F_y). The plane segmentation method can only detect obstacles with large and flat surfaces. Therefore, a pre-trained image object detector was applied to handle other objects (e.g., small objects and humans). The goal is to determine the 3D positions and sizes of obstacles and to provide information to users in real time to assist with other applications (e.g., navigation). Therefore, one-stage object detectors were chosen to minimize the computation time. The YOLOv3 model (Redmon and Farhadi, 2018) was used in this study; however, it can be easily altered with other 2D object detectors.

To convert the 2D bounding boxes detected by YOLOv3 into 3D bounding boxes, we use the AR point cloud of the corresponding camera frame (Fig. 3). The app’s frame rate was set at 30 fps because it is difficult for the human eye to see differences above 30fps. We only processed the point cloud with number of feature points above a threshold (i.e., 30), and the typical range of the feature points detected in each frame was between 0 and 200. The 3D bounding

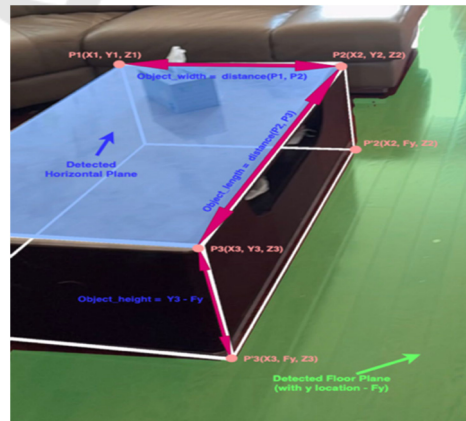


Figure 2: The 3D bounding box of an object with a horizontal plane.

³ <https://developer.apple.com/machine-learning/models/>

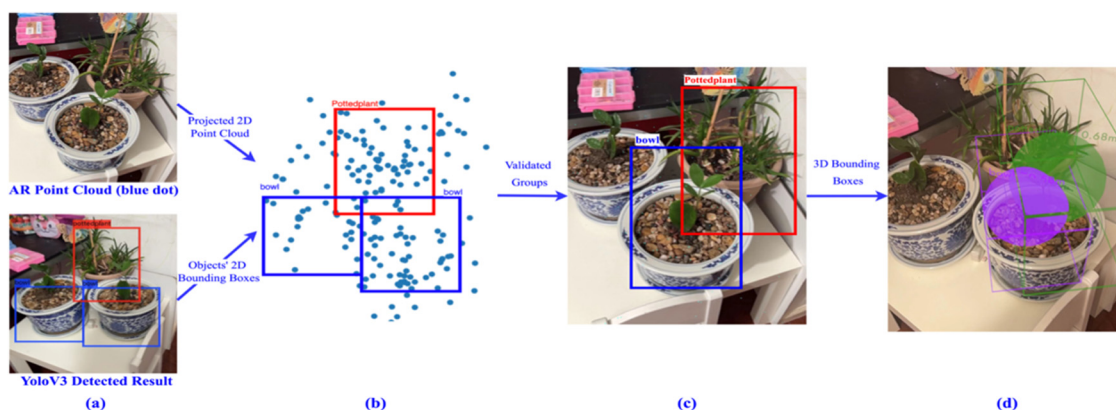


Figure 3: From 2D bounding boxes to 3D bounding boxes using YOLOv3 and AR point cloud.

box estimation method first removes all points belonging to the detected planes, and then projects the remaining points into the 2D image coordinate system and grouped feature points based on the 2D bounding boxes of each detected object (Fig. 3 (a) to (b)). If the object group contained fewer feature points than threshold, its 3D bounding box was not computed. For example, one of the flowerpots is ignored from the step in Fig. 3 (b) to (c) because of insufficient feature points within its 2D bounding box. The 3D bounding box of each object was calculated using the minimum and maximum values of the feature points group x , y , and z (Fig. 3 (c) to (d)).

After estimating the 3D bounding boxes for all the objects detected in the frame, the object matching module is activated if previously detected objects exist to either update the pre-detected objects' 3D bounding boxes or create new object instances. Each detected object is assigned a unique tracking ID (*tid*). Subsequently, the object tracking module tracks the 2D bounding boxes of objects in the next frame and updates the 3D bounding boxes with the AR point cloud of the new frame.

3.2 3D Object Tracking

AR point cloud is sparse and contains considerable noise owing to various circumstances of the context, such as low texture or colors, presence of shadows, and poor lighting conditions. The sparsity within the AR point cloud increases the errors in estimating the 3D bounding boxes of the objects. Hence, we used the non-maximum suppression (NMS) method (Rothe *et al.*, 2014) across multiple estimated 3D bounding boxes of the same object to obtain a more stable and accurate 3D bounding box of the object.

Tracking 2D bounding boxes of objects across frames requires less computational power and is more effective than performing object detection for each frame. Moreover, due to the sparsity of AR point cloud, 3D trackers are not suitable. By utilizing vision algorithms⁴, we can track multiple detected 2D objects across frames in real time.

Each tracking result provides an updated 2D bounding box and confidence score of the tracked object. If the confidence score of the track object is above the threshold (i.e., 70%), a new 3D bounding box is estimated with the corresponding frame's AR point cloud using the method described in Section 3.1 and recorded. When an object has five or more estimated 3D bounding boxes across multiple frames (Fig. 4 (a)), the NMS method is applied to obtain the final 3D bounding box (i.e., the validated 3D bounding box), as shown in Fig. 4 (b). An object track is considered to be lost in the new frame if its confidence score is below a threshold. If one or more objects lose tracks during the tracking mode, the system restarts the object detection process (Section 3.1) to detect new objects.

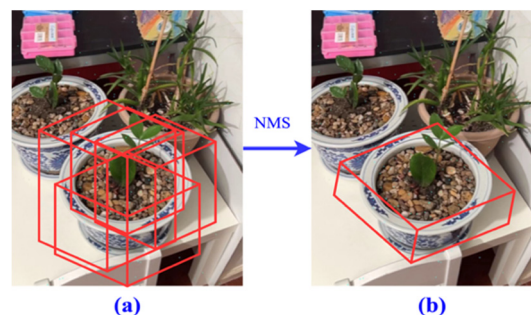


Figure 4: (a) Five estimated 3D bounding boxes of a flowerpot. (b) Validated 3D bounding box of the flowerpot after NMS.

⁴ https://developer.apple.com/documentation/vision/tracking_multiple_objects_or_rectangles_in_video

3.3 3D Object Matching

Previously detected/tracked objects may be detected again when the system switches between the tracking and detection modes. The object matching module is applied to avoid creating duplicate object instances for the same object, which consists of three major steps. First, each object in tracking mode updates its 3D bounding box with its tracking id (*tid*). Second, the estimated 3D bounding boxes of the newly detected objects were compared with 3D bounding boxes for previously detected objects with the same label. If the 3D bounding boxes overlap, it will not be considered a new object; instead, it will update the 3D bounding box of the overlapped object.

As mentioned previously, the estimated 3D bounding box of an object from a single frame has low accuracy; therefore, there might not always be an overlap area for the same object. More importantly, objects could be in motion between frames, therefore, we also need to compare closely located objects with the same label. The distance between the two 3D bounding boxes is calculated using the sum of the distance between two nearest endpoints in each of the three axes where bounding boxes do not overlap. If the distance is less than the threshold, it is considered to be the same object; otherwise, it is treated as a new object. After handling all objects, the system switches to the tracking mode for the next frame.

3.4 Information Filtering

With massive objects detected over time, information filtering is required to extract essential information to guide BLV users to avoid obstacles. It filters out all detected obstacles that are not within a certain angular range ($\alpha = 60^\circ$ in our current experiments) of camera orientation. As shown in Fig. 5, the obstacles outside the blue region were ignored. Afterward, the remaining objects are sorted by alert priority using Equation (2), which is calculated based on the distance between the user's camera position (C_p), object's center position (O_p), and object size (O_s). The distance between the object and camera is calculated using the closest point in object's 3D bounding box without considering the differences in the y-axis. The distance weighs more than the object size ($a \gg b$).

$$Priority(C_p, O_p, O_s) = \frac{a}{distance(C_p, O_p, O_s)} + b * O_s \quad (2)$$

If the object alert priority value exceeds a threshold, the distance and direction of an object are provided as voice notifications. AR visualizations of detected obstacle bounding boxes were added to show

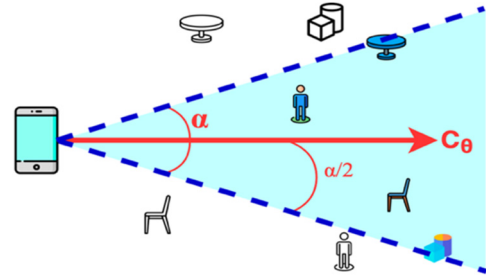


Figure 5: Filtering objects within the angular range (α) with respect to the camera orientation (C_θ).

and verify the detection results. These visualizations may provide additional visual aids for people with low vision or other disabilities (e.g., autism spectrum disorder) during an emergency evacuation. The colors and transparency levels of displayed AR assets were determined by the object labels and confidence scores from the detection or tracking results, respectively.

4 EXPERIMENTAL RESULTS

An iOS app was created to test real-time 3D obstacle detection results. Hosting image detection on cloud consumes more time for image uploading; therefore, 2D object detection is performed on an iPhone.

Table 1: Computation time (mean \pm std) of four module using iPhone 13 Pro Max.

	Computation Time (ms)	
	YOLOv3	YOLOv3 Tiny
Obstacle Detection	44.69 \pm 3.86	26.09 \pm 9.62
Object Tracking	12.13 \pm 11.79	12.64 \pm 8.37
Object Matching	0.65 \pm 0.77	0.29 \pm 0.42
Information Filtering	0.04 \pm 0.06	0.02 \pm 0.007

Table 1 shows the breakdown of the times spent on the four modules in milliseconds (ms) with YOLOv3 and YOLOv3 Tiny models using iPhone 13 Pro Max in the same environment. The YOLOv3 model (248.4MB) took approximately 3 to 5 seconds to initially load to the app. For each frame, it took an average of 44.69ms to detect objects and 12.13ms to track objects; both included the time for estimating 3D bounding boxes. In comparison, the YOLOv3 Tiny model (35.4MB) contains fewer convolution layers, which reduces the computation time. It took less than one second to load, and less time for object detection. However, YOLOv3 performs better than YOLOv3 Tiny for smaller object detection. Furthermore, YOLOv3 is more stable with a smaller standard deviation in the object detection time.

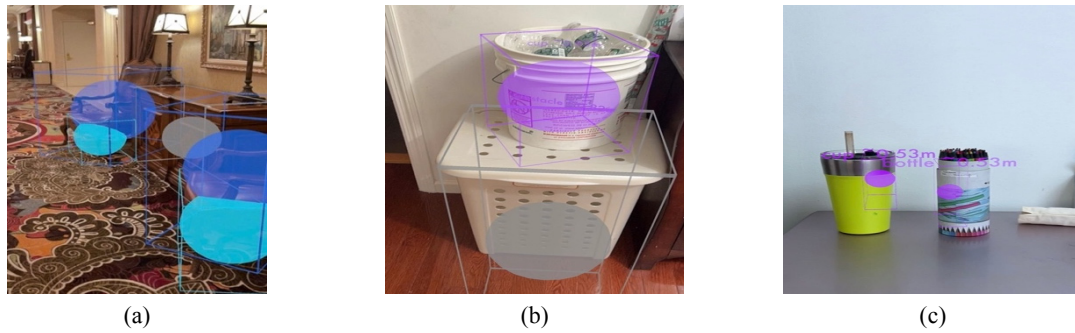


Figure 6: 3D obstacle detection results; (a) table and chair, (b) large obstacles, (c) small objects.

The computation time varied based on the number of objects detected in the image frames and the number of feature points in the AR point cloud. This uncertainty contributed to a large standard deviation in the computation time of the object tracking and object matching modules. From Table 1, most of the computation time was spent on the object detection module for both models. Alternatively, the system can use any 2D object detector based on the purpose of the application. On the average, the system can achieve a time performance of 10-15 fps using YOLOv3 on an iPhone 13 Pro Max.

Fig. 6 shows three detection results: a table and two chairs under a complex background, large obstacles under a clean background, and small bottles on table. The estimated 3D bounding boxes for large obstacles are more accurate than those for small obstacles and are determined faster and farther, as they have larger 2D image bounding boxes. Due to thresholding, smaller objects took more image frames to obtain a validated 3D bounding box, as they often do not contain sufficient context in the AR point cloud. Smaller objects also have low-accuracy 3D bounding boxes (Fig. 6 (c)). However, the app can solidly obtain the 3D positions of small objects. Our work aims to avoid BLV people colliding with obstacles. In most cases, small objects are placed on top of large objects, such as a table, so helping BLV users avoid large obstacles would help them avoid these small obstacles. Although the detected 3D bounding boxes do not always align with the actual object sizes in the real world, they are sufficient to warn BLVs to move away from it. The proposed algorithm can also work in real time, as shown in the demo video: https://youtu.be/L4zloslQ_8c.

5 CONCLUSION AND DISCUSSION

In the current work, the proposed 3D obstacle detection and object recognition system works well for detecting large obstacles in the real world with 3D positions and 3D bounding boxes. The system does not require multiple sensors or trains a new deep learning model besides a pre-trained 2D object detection model and works efficiently in real time. The system can easily adapt to any pre-trained object detection model for better performance. With the information filtering module, it is feasible to increase the safety of BLV users to travel indoors.

Few areas can be improved in current system. Due to the sparsity of the AR point cloud, the matching module also considers nearby objects with 3D bounding boxes within the distance threshold as the same object. It is possible to mistakenly consider two distinct objects of the same class to be the same object. Several methods have been planned to mitigate this issue. The first is to replace the AR point cloud with the LiDAR point cloud to obtain a denser point cloud and consider only overlapped 3D bounding boxes. Object motion can also increase the difficulty of the matching process such as people walking. To adapt to objects' motion change, the changes in camera positions and orientations, and changes in the objects' bounding boxes across camera frames are used to enhance the matching and tracking modules and update the object's position.

One limitation of the current system is the adaptation to object size changes. Usually, a non-human object has a static shape. However, we would also like to develop a method to detect people for interactions. The positions and postures of people can change over time, such as from sitting to standing, leading to a change in the 3D bounding box. Because our method uses the non-maximum suppression method across multiple estimated 3D bounding boxes

to determine the final 3D bounding box of an object, it is difficult to immediately update the 3D bounding box corresponding to the object's size change. Another limitation is that the system does not detect stairs; therefore, it cannot work for floor transition.

In the future, we will be working on the discussed area of improvement and solve the app's limitations. Also, we would like to integrate this system with our previous work (Zhu *et al.*, 2020) in indoor navigation apps and test with BLV users.

ACKNOWLEDGEMENTS

The research is supported by NSF (Awards #2131186, #2118006, #1827505, and #1737533), AFOSR (#FA9550-21-1-0082) and Intelligence Community Center for Academic Excellence (IC CAE) at Rutgers (#HHM402-19-1-0003 and #HHM402-18-1-0007).

REFERENCES

- Bourne, R. R., Adelson, J., Flaxman, S., Briant, P., Bottone, M., Vos, T., ... & Taylor, H. R. (2020). Global Prevalence of Blindness and Distance and Near Vision Impairment in 2020: progress towards the Vision 2020 targets and what the future holds. *Investigative Ophthalmology & Vis. Sci.*, 61(7), 2317-2317.
- Ackland, P., Resnikoff, S., & Bourne, R. (2017). World blindness and visual impairment: despite many successes, the problem is growing. *Community eye health*, 30(100), 71.
- Manduchi, R., & Kurniawan, S. (2010). Watch your head, mind your step: mobility-related accidents experienced by people with visual impairment. *Dept. Comp. Eng., Univ. California, Santa Cruz, Tech. Rep.*
- Huang, H. C., Hsieh, C. T., & Yeh, C. H. (2015). An indoor obstacle detection system using depth information and region growth. *Sensors*, 15(10), 27116-27141.
- Soquet, N., Aubert, D., & Hautiere, N. (2007). Road segmentation supervised by an extended v-disparity algorithm for autonomous navigation. In *2007 IEEE Intell. Veh. Symp.*, pp. 160-165. IEEE.
- Cheng, R., Wang, K., Yang, K., & Zhao, X. (2015). A ground and obstacle detection algorithm for the visually impaired. In *2015 IET ICBISP*, pp. 1-6.
- Sun, L., Yang, K., Hu, X., Hu, W., & Wang, K. (2020). Real-time fusion network for rgb-d semantic segmentation incorporating unexpected obstacle detection for road-driving images. *IEEE Robot. and Autom. Lett.*, 5(4), 5558-5565.
- Chen, Y., Liu, S., Shen, X., & Jia, J. (2020). Dsgn: Deep stereo geometry network for 3d object detection. In *Proc. IEEE CVPR*, pp. 12536-12545.
- Pham, H. H., Le, T. L., & Vuillerme, N. (2016). Real-time obstacle detection system in indoor environment for the visually impaired using microsoft kinect sensor. *J. of Sensors*, 2016.
- Apple Inc. Arkit - augmented reality. Apple Developer. <https://developer.apple.com/augmented-reality/arkit/>
- Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.
- He, C., Gong, J., Yang, Y., Bi, D., Lan, J., & Qie, L. (2021, May). Real-time Track Obstacle Detection from 3D LIDAR Point Cloud. In *J. of Phys.: Conf. Ser.*, 1910(1), p. 012002. IOP Publishing.
- Garnett, N., Silberstein, S., Oron, S., Fetaya, E., Verner, U., Ayash, A., Goldner, V., Cohen, R., Horn, K., & Levi, D. (2017). Real-time category-based and general obstacle detection for autonomous driving. In *Proc. IEEE ICCVW*, pp. 198-205.
- Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. of the ACM*, 24(6), 381-395.
- Levi, D., Garnett, N., Fetaya, E., & Herzlyia, I. (2015, September). StixelNet: A Deep Convolutional Network for Obstacle Detection and Road Segmentation. In *BMVC*, 1(2), p. 4.
- Rothe, R., Guillaumin, M., & Van Gool, L. (2014, November). Non-maximum suppression for object detection by passing messages between windows. In *ACCV*, pp. 290-306. Springer, Cham.
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. In *ECCV*, pp. 21-37. Springer, Cham.
- Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In *Proc. IEEE ICCV*, pp. 2980-2988.
- Girshick, R. (2015). Fast r-cnn. In *Proc. IEEE ICCV*, pp. 1440-1448.
- He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *Proc. IEEE ICCV*, pp. 2961-2969.
- Soviany, P., & Ionescu, R. T. (2018, September). Optimizing the trade-off between single-stage and two-stage deep object detectors using image difficulty prediction. In *2018 SYNASC*, pp. 209-214. IEEE.
- Karunasekera, H., Wang, H., & Zhang, H. (2019). Multiple object tracking with attention to appearance, structure, motion and size. *IEEE Access*, 7, 104423-104434.
- Sualeh, M., & Kim, G. W. (2020). Visual-LiDAR Based 3D Object Detection and Tracking for Embedded Systems. *IEEE Access*, 8, 156285-156298.
- Rezatofghi, S. H., Milan, A., Zhang, Z., Shi, Q., Dick, A., & Reid, I. (2015). Joint probabilistic data association revisited. In *Proc. IEEE ICCV*, pp. 3047-3055.
- Wang, M., & Wu, X. (2021). Multi-Object Tracking Strategy of Autonomous Vehicle Using Modified Unscented Kalman Filter and Reference Point Switching. *J. of Shanghai Jiaotong Univ. (Science)*, 26(5), 607-614.
- Zhu, Z., Chen, J., Zhang, L., Chang, Y., Franklin, T., Tang, H., & Ruci, A. (2020). iASSIST: An iPhone-Based Multimedia Information System for Indoor Assistive Navigation. In *IJMDEM*, 11(4), 38-59.
- Marques, O. (2020). Computer Vision and Image Analysis with the Vision Framework. In *Image Process. and Comp. Vision in iOS*, pp. 41-50. Springer, Cham.