Title:

Real-time 3D point-cloud segmentation using YOLO-based object detection method for augmented reality applications.

Authors:
Rabab Ahniguira, rabab.ahniguira@ensam.eu, Arts et Métiers Institute of Technologies
Arnaud Polette, arnaud.polette@ensam.eu, Arts et Métiers Institute of Technologies
Jean-Philippe Pernot, jean-philippe.pernot@ensam.eu, Arts et Métiers Institute of Technologies

Keywords:
Point cloud segmentation, mobile applications, Lidar sensor, YOLO v7, augmented reality

DOI: 10.14733/cadconfP.2023.307-312

Introduction:

Point cloud segmentation is a key technology for recent Augmented Reality (AR) applications, as it enables the separation of objects in a 3D scene into meaningful groups or parts. In recent years, there has been growing interest in using point clouds for a range of applications, including AR, robotics, and virtual reality. However, there are still many challenges in the field of point cloud segmentation for mobile applications, particularly in terms of computational efficiency, accuracy, and scalability.

The current state of knowledge on point cloud segmentation for mobile applications has seen a number of advances in the development of deep learning-based methods, which have shown promising results in terms of accuracy such as PointNet [1] and PointNet++[2]. However, these methods are often computationally expensive and may not be well-suited for real-time applications on mobile devices. Additionally, there is a lack of publicly available datasets and benchmarks for evaluating the performance of point cloud segmentation algorithms on mobile platforms.

In this paper, the idea is to make use of the well-known YOLO (You Only Look Once) [3] 2D image-based detection method and integrate it in a real-time 3D point cloud segmentation framework, with the objective to apply it on some use cases that need to be implemented in wireless applications such as in an AR context. Unlike 3D object detection on point clouds, 2D object detection can be trained rapidly and with high accuracy. The main reason for this is the availability of large annotated image datasets, such as COCO [4] and PASCAL VOC [5], that allow deep learning models to learn the visual appearance and context of objects in images. Another advantage of 2D object detection, especially on custom data is that it is easier to perform data augmentation techniques on images, such as random cropping, flipping, and color jittering, to increase the size of the training dataset and reduce overfitting. These techniques allow models to learn from multiple variations of the same image, making them more robust to changes in the input data. In contrast, 3D object detection on point clouds is a more challenging problem. While there have been some recent advances in 3D object detection and segmentation using deep learning models, these models are typically slower to train and less accurate than 2D object detection models. This is because point clouds are much less dense than images and contain fewer features, making it harder for models to learn the visual appearance of objects. In addition, there is a limited amount of annotated point cloud data available, which makes it difficult to train large and accurate models.

The proposed approach has the potential to improve the speed and accuracy of 3D point cloud segmentation for AR applications, while performing a fast and efficient YOLO-based object detection in 2D before coming back to the 3D point cloud segmentation.

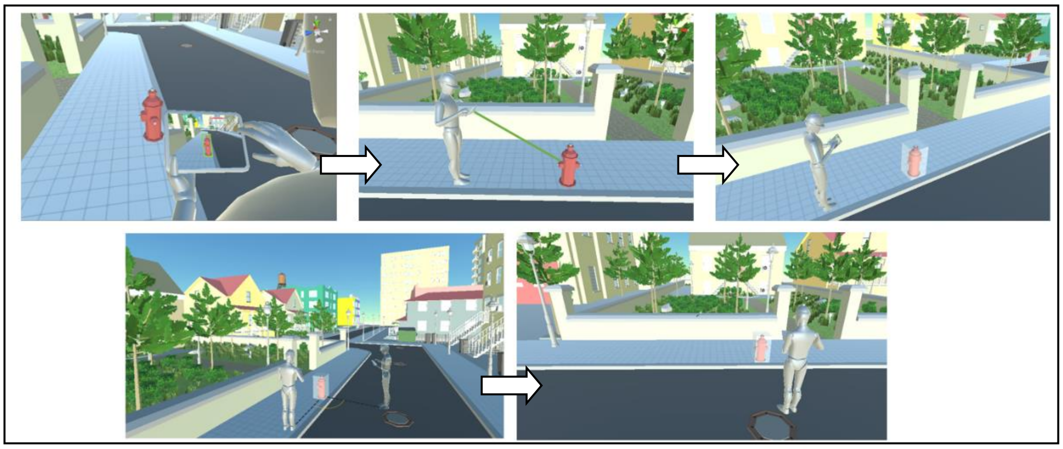

Fig. 1: Illustration of the proposed pipeline: (a) object detection in 2D images, (b) estimating their 3D location through ray-casting, (c) creating a 3D bounding box around the object, (d) consolidating the segmentation using multiple positions around the object, (e) finally segmenting the 3D point cloud of the object.

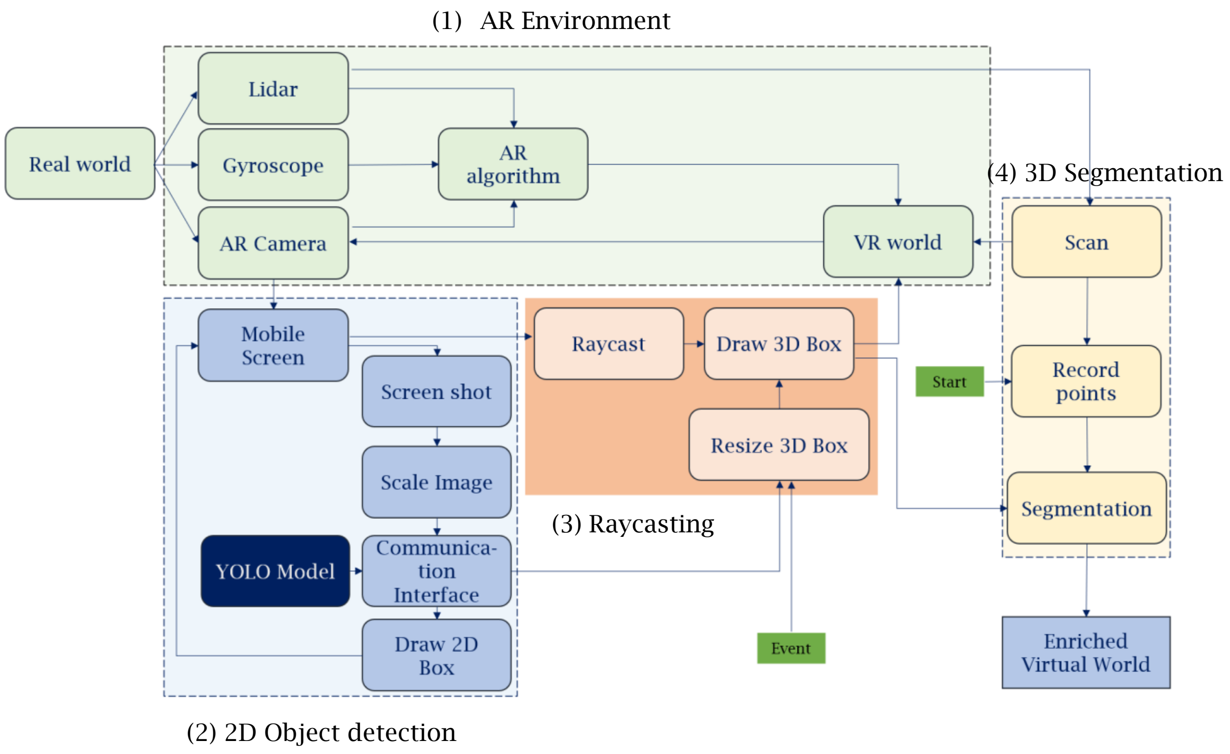

Fig. 2: Overview of the proposed method and its different modules.

Main Idea:

The main idea of the proposed approach is illustrated on figure 1. First, in step (a) an object is detected in AR using YOLO v7 [6]. Then, in step (b) using ray-casting the link with the 3D reference frame is obtained, and this allows finding the center of the 3D bounding box in step (c). The process then involves running a 3D scan, extracting the 3D coordinates of points, and isolating the points that are within the 3D coordinate system (d), and this is repeated several times to get information from different viewpoints. At the end, the 3D point cloud can finally be segmented in step (e).

Materials and Methods

The flowchart of this process is shown in Figure 2. In brief, the four main steps of this process are as follows:

(1) AR ENVIRONMENT SETTING

For this part, ARFoundation, a Unity package has been used. By leveraging these tree sensors, ARFoundation enables the creation of AR settings, where virtual objects can be seamlessly integrated into the real world. Lidar sensor is used to capture the environment in 3D, providing a high-resolution point cloud that can be used to detect and track objects in real-time. ARFoundation uses this information to create a detailed 3D map of the environment, which allows virtual objects to be placed in the correct position and orientation relative to the real world. Gyroscopes are used to track the movement and orientation of the device in real-time, allowing ARFoundation to update the position and orientation of virtual objects in response to changes in the physical environment. This creates a seamless and natural experience, where virtual objects appear to interact with the real world in real-time. By combining the data from lidar cameras and gyroscopes, ARFoundation creates a rich and immersive augmented reality setting, where virtual objects can be seamlessly integrated into the real world to interact with the real world in a natural and intuitive way.

(2) 2D OBJECT DETECTION

YOLO v7 [6] is a state-of-the-art object detection algorithm that has been optimized for improved accuracy and speed. The ability to use YOLO v7 on custom data allows the algorithm to be tailored to a specific application, increasing its effectiveness. In our case, YOLO v7 was trained using Roboflow's Annotation Tool on fire hydrant images taken in the city of Aix-en-Provence as well as some similar images from Internet (Figure 2).

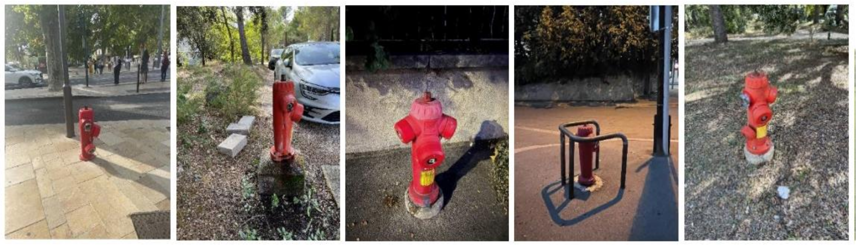

Fig. 3: Images from the training dataset.

To ensure optimal performance of the model in the application, a thorough benchmarking process has been performed to compare all possible communication interfaces for integrating YOLO v7 into the AR application taking into consideration factors such as accuracy, speed, and interoperability.

The Unity game engine and the Barracuda neural network inference engine are two common platforms for implementing YOLO v7 in AR applications. The Open Neural Network Exchange (ONNX) is an open format for representing deep learning models, particularly neural networks. It provides a standard for model representation by specifying standard operators, a computational graph model, and standard data types. The metadata included in ONNX provides semantic information about the input and output types of the model. ONNX enables interoperability between different deep learning frameworks and platforms, allowing models to be shared and reused easily. Frameworks like PyTorch have built-in support for exporting models in ONNX format, while models from other frameworks like TensorFlow and Keras can be converted to ONNX using framework-specific conversion tools. Barracuda is a library for running ONNX-formatted neural networks in Unity, providing support for trained models from PyTorch, TensorFlow, and Keras. While Barracuda does not support training, it allows the use of pre-trained models in an offline adaptation pipeline. The first unit tests have validated the effectiveness of this pipeline in an augmented reality application.

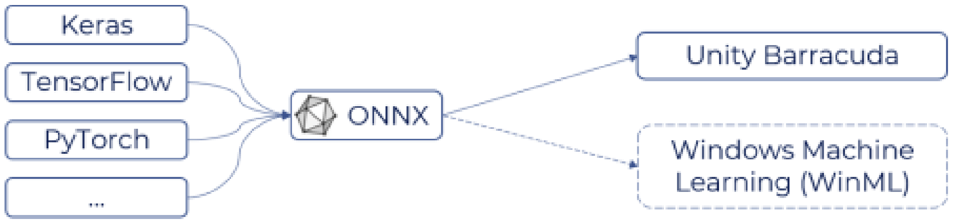

Fig. 3: Communication interface to adapt the model to the mobile support with ONNX.

(3) RAYCASTING AND CREATION OF A 3D BOUNDING BOX

In AR, a ray-cast is a method used to determine the intersection point of a virtual ray with a real-world object or surface. In 2D AR, the ray-cast is used to project a 2D point on the screen to a 3D point in the real world, allowing virtual objects to appear as if they are interacting with the real environment. This process involves using the device's sensors, such as the camera and Lidar, to gather information about the device's position and depth, and then using that information to calculate the intersection point of the virtual ray with the real-world environment. Our study case aims to use the coordinates of the center of bounding box of the detected object in 2D. This step calculates the location and orientation of object relative to the camera in 3D space by tracing rays along different directions through the scene and finding their intersections with objects in it.

The goal is to use information from 2D bounding boxes and ray casted points to identify objects more accurately by extracting further features such as size, shape, and surface details. The process starts with a default depth, which is estimated based on pre-existing information about the object's shape. By changing the angle at which the image is captured, a different 2D representation of the object is obtained, allowing for a recalculation of the depth. Indeed, as depth information can provide valuable information for object recognition. By combining information from multiple 2D views of an object, it's possible to obtain a more complete and accurate representation of the object's geometry, which can be used to identify it more effectively.

(4) 3D SCAN ACTIVATION AND RESULTING POINT CLOUD SEGMENTATION

This stage first considers developing a scan application where the user turns around full object by activating multiple scanning processes at different angles simultaneously, thus obtaining further information about its entire shape including any hollow or curved sections etc. This stage in the development of the 3D application focuses on capturing the full shape of objects, including all the object points, by using multiple scanning processes provided by Lidar and ARKIT libraries.

Having access to the 3D coordinates and color information of all points captured by the 3D scan application enables various mathematical operations to be performed on the data. This allows for the extraction of specific points and the application of operations such as union and if-then-else conditions. Thus, it is possible to extract all points that meet certain spatial criteria, or to perform operations that combine multiple point sets. By providing the ability to extract only points contained within a 3D bounding box, the final result is a segment of the point cloud corresponding to the detected object.

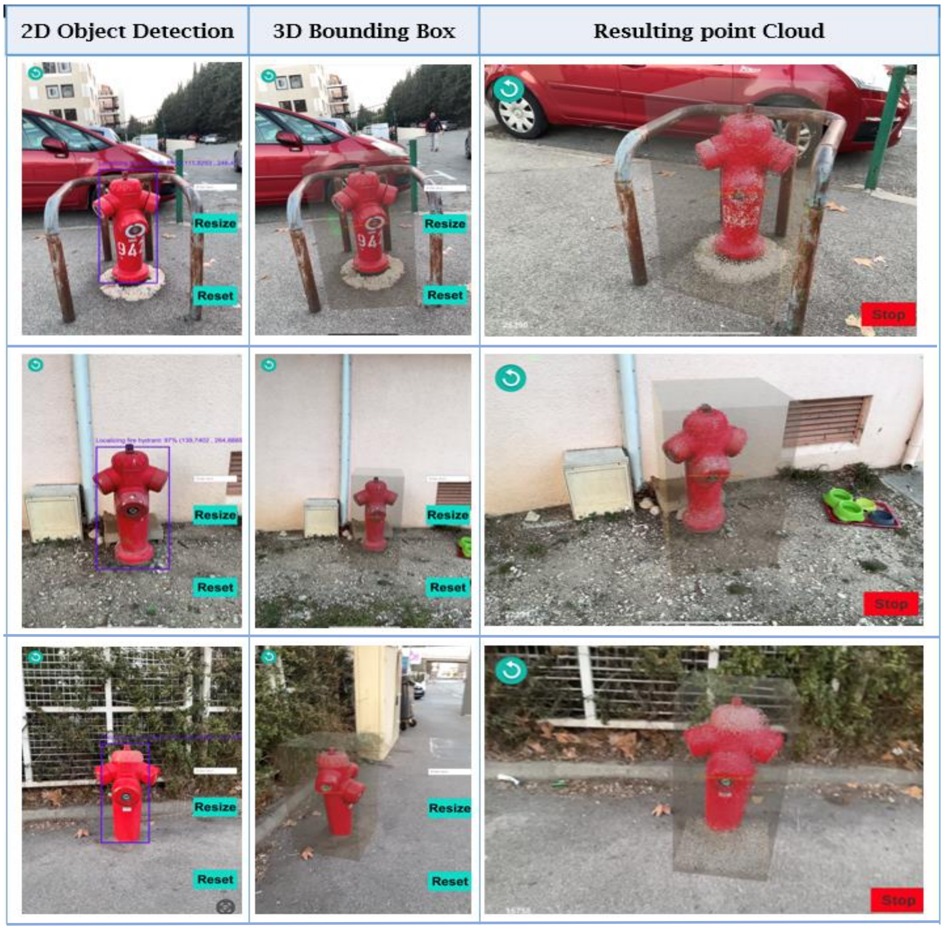

Fig. 4: Result of implementation of the approach on three different fire hydrants.

Implementation and results

The use case involves developing an AR application for mobile devices, which enables point cloud segmentation in the context of a BIM project.

The implementation of this application is done using Unity 3D. The application has been tested on different fire hydrants in the city using an iPad Pro equipped with a Lidar sensor, as shown in figure 4. The user interface includes a button to start the 2D detection, along with a button to resize the 3D box. The 3D box is initially created automatically, and the user turns around the object to capture information from multiple perspectives to resize the 3D box and achieve a better fit to ensure that all the relevant details are captured in the point cloud data. This helps in obtaining accurate segmentation results, which can be further used for the BIM model registration.

The augmented reality interface allows the user to activate the scanning process and to view the point cloud data in real-time, enabling them to interact with the object and make any necessary adjustments to ensure accurate segmentation results. After the user has resized the 3D box the system identifies and separates the points inside the 3D box as a segmented point cloud. The application also offers the ability to export the segmented point cloud data in multiple formats. This makes it easier for project stakeholders and for registration.

Overall, this AR application provides a fast and efficient way of segmenting the detected object. In this case, the specific objects being detected and segmented are fire hydrants, but the method can be extended to other customized elements as well. The approach presented here provides a useful starting point but requires further refinement to reach a higher level of accuracy and reliability.

Conclusion:

In summary, this paper has demonstrated that by combining the power of 2D object detectors with advanced 3D point cloud processing techniques, it is possible to develop an efficient framework for detecting objects in real-world scenes. It offers a promising solution that can find application in robotics and autonomous vehicle navigation, as well as many other areas where accurate 3D object detection is needed. Furthermore, this paper has proposed a novel approach for 3D object detection with limited data that can be easily adapted to other applications. With further development, it will provide a more reliable and accurate results.

Acknowledgements: This work was supported by PSPC Provence Alpes Cote d'Azur, with the participation of Perspective(s): Romain Senatore, Stéphane Kyles and Nicolas Cortell.

References:

[1] C. R. Qi, H. Su, K. Mo, et L. J. Guibas, « PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation ». arXiv, 10 avril 2017. Consulté le: 13 septembre 2022. [En ligne]. Disponible sur: http://arxiv.org/abs/1612.00593

[2] C. R. Qi, L. Yi, H. Su, et L. J. Guibas, « PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space ». arXiv, 7 juin 2017. Consulté le: 13 septembre 2022. [En ligne]. Disponible sur: http://arxiv.org/abs/1706.02413

[3] J. Redmon, S. Divvala, R. Girshick, et A. Farhadi, « You Only Look Once: Unified, Real-Time Object Detection ». arXiv, 9 mai 2016. Consulté le: 13 septembre 2022. [En ligne]. Disponible sur: http://arxiv.org/abs/1506.02640

[4] T.-Y. Lin *et al.*, « Microsoft COCO: Common Objects in Context ». arXiv, 20 février 2015. Consulté le: 13 septembre 2022. [En ligne]. Disponible sur: http://arxiv.org/abs/1405.0312

[5] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, et A. Zisserman, « The Pascal Visual Object Classes Challenge: A Retrospective », *Int. J. Comput. Vis.*, vol. 111, nº 1, p. 98-136, janv. 2015, https://doi.org/10.1007/s11263-014-0733-5.

[6] C.-Y. Wang, A. Bochkovskiy, et H.-Y. M. Liao, « YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors ». arXiv, 6 juillet 2022. Consulté le: 7 octobre 2022. [En ligne]. Disponible sur: http://arxiv.org/abs/2207.02696