Hybrid 3D Object Detection in Logistic Environments Using a Histogram Depth Filter

Daniel Vidal[1,*], Johannes Fottner[1], and Boyan Liu[2]

[1]Chair of Materials Handling, Material Flow, Logistics (fml), Technical University of Munich, Germany
[2]School of Engineering and Design, Technical University of Munich, Germany
Email: daniel.vidal@tum.de (D.V.); j.fottner@tum.de (J.F.); boyan.liu@tum.de (B.L.)
[*]Corresponding author
Manuscript received February 13, 2025; revised August 25, 2025; accepted November 21, 2025; published February 25, 2026

***Abstract*—In intralogistics, accurate 3D object detection is essential for enhancing robotic perception and enabling autonomous navigation. This paper presents a novel hybrid method that combines 2D object detection with depth-based point cloud segmentation for efficient and real-time 3D object pose estimation. The key contribution of this paper is the Cumulative Histogram Depth Filter (HDF), a lightweight algorithm that segments dominant depth regions corresponding to detected objects. This approach uniquely enables the reuse of existing 2D-labeled datasets, eliminating the need for extensive 3D annotations. The proposed method was evaluated using both simulated and real-world data, obtaining high detection accuracy. In the simulated environment, it achieved a mean distance error of 0.14 m, an Intersection over Union (IoU) of 0.90, and a mean Average Precision at 50 (mAP@50) of 0.95. Real-world experiments using Azure Kinect and ZED2 cameras yielded an average distance error of 0.13 m, IoU of 0.78, and mAP@50 of 0.65. Additionally, the system runs at 105 FPS, significantly outperforming more complex hybrid architectures in terms of computational efficiency, making it particularly suitable for real-time robotic applications.**

***Keywords*—hybrid 3D object detection, autonomous mobile robots, intralogistics, histogram depth filter, YOLO11, synthetic data**

I. Introduction

Intralogistics, which refers to the internal flow of materials and information within production environments, warehouses, or distribution centers, is a vital component of the modern supply chain [1]. As Autonomous Mobile Robots (AMR) become increasingly prevalent, the need for precise and adaptable object detection has grown. Reliable 3D perception is crucial for enabling robots to operate efficiently in dynamic environments, where they must detect, identify, and interact with diverse objects to perform tasks such as picking and retrieval [2, 3].

While traditional 2D object detection methods are effective for classification in image space, they lack the spatial information needed for depth-aware robotic tasks such as object localization, collision avoidance, and trajectory planning. Therefore, accurate 3D object detection is essential for underpinning practical robotic applications like warehouse automation, manipulation [3], navigation [4, 5], and smart logistics systems [6].

On the other hand, 3D detection methods often rely on fully annotated 3D datasets and powerful computation, which can limit scalability and generalization in real-world deployments. To address these challenges, this research introduces a hybrid approach for 3D object detection in logistics environments. It combines 2D object detection with a lightweight point cloud segmentation method called the Histogram Depth Filter (HDF). The HDF segments depth images by identifying dominant depth regions and enabling spatial localization of objects of interest.

This solution extends the utility of existing 2D detection frameworks by enabling their adaptation to 3D perception tasks without the need for large-scale 3D annotation. Experiments conducted in real and simulated data demonstrate the feasibility of this method for performing real-time 3D object detection in logistics scenarios.

II. Related Work

A. 3D Object Detection

3D object detection is a computer vision task that involves the recognition and classification of objects of interest in the three-dimensional space and the estimation of their position and orientation (x, y, z, θ) in the world reference system. Recent advancements in 3D object detection leverage deep learning models to process sensor data and produce pose estimation of objects. According to the type of data used, they can be classified as image-based, Point-cloud-based, or hybrid methods.

Some image-based methods rely on pre-existing object templates to detect and match keypoints within the image to known models [7, 8]. Others perform multi-view reconstruction using multiple cameras to recreate the 3D environment from different perspectives [9]. Additionally, approaches such as Cube R-CNN employ three-dimensional neural networks to predict 3D bounding boxes from Red-Green-Blue (RGB) images directly [10]. However, estimating object poses using only 2D data can lead to inaccuracies due to the lack of direct depth information.

An alternative approach involves using depth-measuring sensors, such as 3D LiDAR or depth cameras, to capture accurate spatial data. Methods like PointNet [11], VoxelNet [12], PointRCNN [13], and others [14] focus on processing point clouds to detect and localize objects in three dimensions. While these approaches provide precise distance measurements, they often lack the rich contextual information—such as color and texture—offered by RGB data. Visual features captured by cameras can be especially helpful for distinguishing adjacent or overlapping objects, such as identifying individual pallets within a stack.

Hybrid 3D detection methods seek to integrate the strengths of RGB and depth data by combining image-based object detection with point-cloud segmentation. For example, some techniques project 2D bounding boxes from images onto point clouds to determine object poses within a defined region of interest, known as the frustum [15–20]. Such

methods improve detection accuracy by using RGB data for initial object identification and depth information for spatial positioning, providing a more comprehensive understanding of the environment.

However, this pipeline requires sequential steps (i.e., image detection, point cloud segmentation, and pose estimation), which can introduce delays and affect real-time performance. To address this, some researchers propose single-shot methods that fusions RGB and depth data in one network to achieve real-time 3D object detection [21, 22]. However, propagating raw-depth data through the Deep Network structure might modify real data, resulting in inaccuracies, which remains a challenge in applications requiring high precision.

B. 3D Object Detection in Logistics

Despite the growing use of robotics in logistics, 3D object detection in this domain has received comparatively less attention than in areas like autonomous driving or household robotics. Hu and Shen [3] explored 3D object pose estimation methods tailored for intelligent logistics, emphasizing their relevance in robotic manipulation and automation. More recently, Achirei *et al.* [4] integrated 3D object detection using Convolutional Neural Networks (CNN) with predictive model control to improve navigation and decision making for omnidirectional robots in logistics environments.

In addition, Molter and Fottner [6] proposed a depth camera-based system to detect European pallets, which demonstrated good performance but was restricted to a single-object category. Although these approaches are promising, they often lack the generalization and flexibility needed to handle diverse object types and complex arrangements that are typically found in real-world warehouse scenarios.

Given the diverse requirements of logistics environments, there is a growing need for robust and adaptable 3D detection solutions that can handle multiple object types and support dynamic workflows. For example, the Logistic Objects in context dataset (LOCO [23]) does not contain 3D data annotation. Therefore, hybrid approaches that integrate RGB-based detection, like in LOCO, with depth-based localization hold promise for addressing these challenges. Using both data types, such methods could enhance object detection, interaction, and navigation, providing a more reliable and scalable solution tailored to logistics applications.

III. Framework of the Proposed Method

The proposed hybrid approach (Fig. 1) leverages a depth camera to combine the rich visual detail of RGB data with the precise spatial measurements of depth sensing. In this framework, the RGB image is processed by the state-of-the-art 2D object detection model YOLOv11 [24], which generates Regions of Interest (REoIs) for each class in the LOCO dataset. This initial detection step ensures that only relevant objects are considered for further processing, thereby reducing unnecessary computation on background or irrelevant regions. Each REoI is then used to crop the corresponding section of the depth image captured by the camera. The resulting cropped depth image is filtered using the proposed Histogram Depth Filter (HDF). To the best of our knowledge, this HDF represents a novel solution for estimating object poses in three-dimensional space.

A. 2D Object Detection

The first goal of this research is to ensure the performance of our method in terms of accuracy and real-time inference. In this paper, real-time refers to the ability of a computer system to process data at an equal or superior rate to the sensor readings [25]. This is, in our case, the ability to meet the Azure Kinect depth camera's max processing rate of 30 Frames Per Second (FPS) [26]. Achieving this frame rate is critical for seamless robot perception and responsive decision-making during navigation or manipulation tasks. Similar papers also use 30 FPS to define real-time 3D object detection [18].

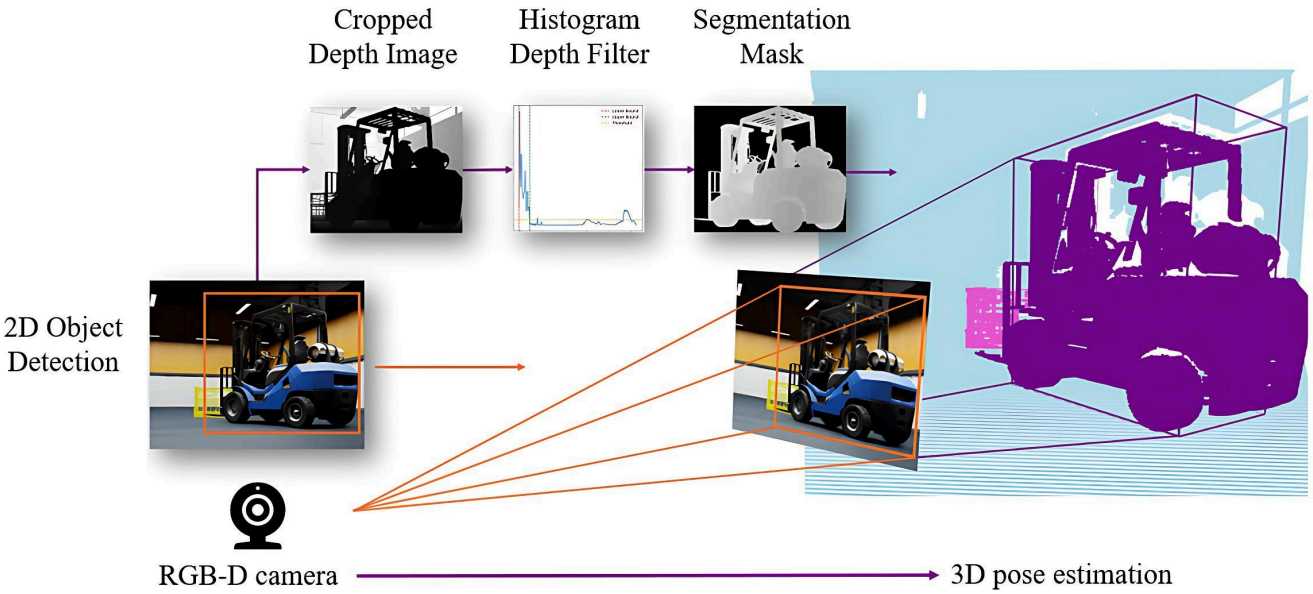

Fig. 1. Overview of the proposed approach. First, a Region of Interest (REoI) is generated using 2D object detection. The cropped depth image of this region is processed using a histogram depth filter to isolate only the points of the image that belong to the object of interest. In the last step, the pose of the object is generated using the filtered mask.

Therefore, the first step of our work was to test the state-of-the-art object detection algorithms to find the best solution that combines real-time performance and maximum mean average precision (mAP@50) using the LOCO dataset. As described in Table 1, the models to benchmark comprise the last releases of the You Only Look Once (YOLO) algorithms and one transformer-based object detector as RT-DETR-v2s. These models were selected for their versatility to produce accurate real-time detections. An NVIDIA RTX 3060 graphics card was used to train and test these models. The training parameters were as follows:

- Epochs = 25
- Batch size = 16
- Image size = 640 × 640 pixels

Table 1. Comparison of YOLO models on NVIDIA RTX 3060

Model	Parameters (Millions)	mAP@0.50	FPS (NVIDIA RTX 3060)
YOLO-NAS-S	15.2	56.29	36.06
YOLOv6-N-v3	4.7	43.30	**55.79**
YOLOv8-N	3.2	67.80	51.44
YOLOv9-T	2.0	67.40	43.31
YOLOv10-N	2.3	64.90	49.76
YOLOv11-N	2.6	**69.70**	50.83
RT-DETR-v2s	31	67.50	20.73

The results presented in Fig. 2 indicate that while YOLOv6 demonstrates faster inference speeds, YOLOv11-N achieves superior detection accuracy. YOLOv11-N is the latest model released by Ultralytics [24] and was trained using the library's default hyperparameters. This version introduces a more efficient backbone, designed to improve both speed and accuracy compared to its predecessor, YOLOv8. Considering the trade-off between inference time and mAP@50, YOLOv11-N was selected for this study. Furthermore, as the smallest model in the YOLOv11 family, it is particularly well-suited for deployment on resource-constrained edge computing devices.

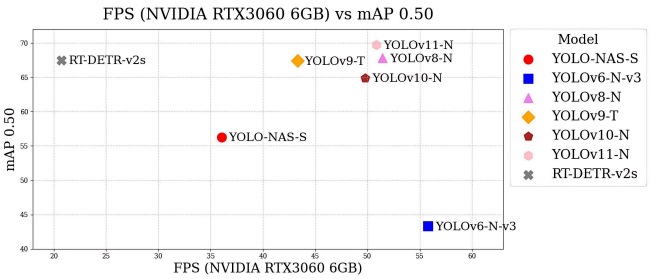

Fig. 2. Comparison of models on NVIDIA RTX3060 6 GB.

B. Histogram Depth Filter (HDF)

To the best of our knowledge, the use of Histogram Depth Filters (HDF) to isolate objects of interest in depth images is a novel solution for 3D object detection. The proposed method represents the depth data as a histogram to identify relevant regions in the image. In the histogram, the *x-axis* represents the distance between the camera and the different depth points of the point cloud, and the *y-axis* represents the accumulation of points at a certain distance.

As shown in Fig. 3, the different peaks in the histogram represent the different regions of interest. Consequently, the algorithm applies a cumulative approach, retaining the largest region within the depth map.

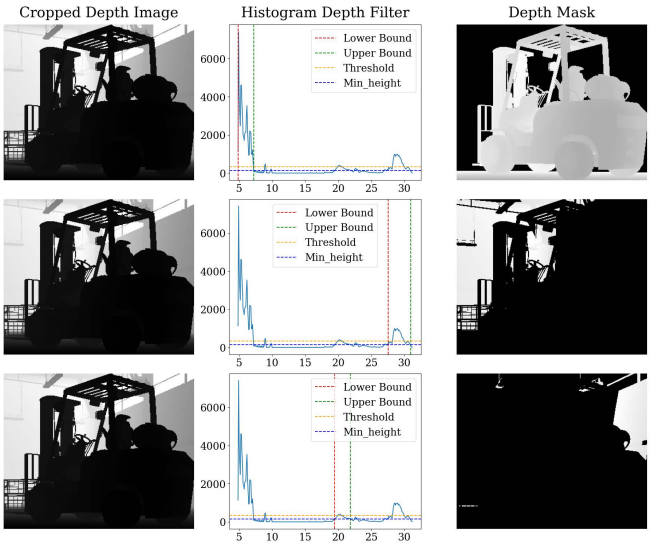

Fig. 3. The objects of interest in the depth image are characterized by peak regions in the histogram. The proposed method searches for the region with more relevance (*resolution* = 10, *max_height_percentage* = 5).

The following pseudocode (Algorithm 1) outlines the cumulative HDF process. It begins by pre-processing the depth image to remove invalid values. Next, it computes the histogram of depth values. The algorithm then identifies the dominant peak regions in the histogram by selecting peaks above the average height (orange dashed line in the histogram). For each significant peak, the algorithm determines the left and right boundaries and sums the heights of each peak within the region to calculate its cumulative value.

Algorithm 1: Pseudocode for Cumulative Hist Depth Filter

Input: depth_img → 2D array of depth values
ignore_bg → flag to ignore background
resolution → scaling factor for depth values
max_height_percentage → threshold to add new peaks
Output: filtered_image → filtered depth image

Initialization:
depth_values = flatten depth_image
bins = round(max(depth_values) × resolution)
(hist, bin_edges) = histogram(depth_values, bins)
max_peak_index = argmax(hist)
average_height = average(hist)
peaks = indices where (hist > average_height)
min_hght = hist[max_peak_index] * max_height_percentage
if peaks is not empty **then**
  results_dict = []
  for each peak_index in peaks **do**
    left_edge, rght_edge, count = calculate_region(peak_index, hist, min_hght)
    results_dict.add(l_edge, r_edge, count)
  end for
  sorted_results_dict = sort(results_dict by count).reverse
  if ignore_background is True **then**
    dom_peak = **next** sorted_results_dict[i] **when** sorted results_dict[i] **is not** hist[−1]

```
18:     else
19:         dom_peak = results_dict[0] → chose the largest region
20:     end if
21:     lower_bound = bin_edges[dom_peak ['l_edge']]
22:     upper_bound = bin_edges[dom_peak ['r_edge']]
23: end if
24: mask = depth_image[lower_bound : upper_bound]
25: filtered_image = depth_image && mask
26: return filtered_image
```

Next, the region with the largest cumulative sum, representing the object of interest, is selected as illustrated in the first row of Fig. 3. The logic behind this behavior is that, given a bounding box around an object, the depth values belonging to this object will likely form the largest region in the depth histogram.

Finally, the algorithm uses the points that form the largest cumulative region to generate a mask over the original depth image. This approach efficiently segments the most salient object in the region of interest, which is especially valuable for removing occluding objects and background data.

The parameters *resolution*, *max_height_percentage*, and *ignore_bg* define the behavior and performance of the function *cumulative_hist_depth_filter*. The *resolution* parameter defines the number of bins used in the histogram. For example, a value of 1 means the filter will use one bin per meter, as illustrated in the first row of Fig. 4. Conversely, a value of 100 results in 100 bins per meter, equivalent to accumulating depth points at a resolution of 1 cm (see last row of Fig. 4).

This parameter is useful in cases where two objects are too close to each other. If the distance between these objects is just a few centimeters, a small resolution will join these objects, as in the first row of Fig. 4. In this instance, a high resolution will help to separate the objects, but it will also separate the distant points from large objects, like with the further points of the forklift in the second row of Fig. 4.

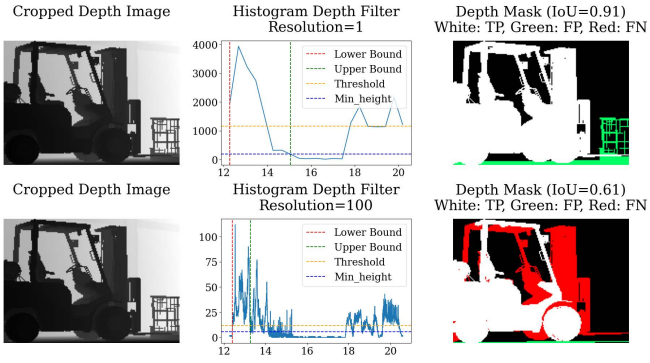

Fig. 4. The parameter resolution defines the number of bins in the histogram (*max_height_percentage* = 10). In this image, the white mask represents the True Positives (TP), the green mask the False Positives (FP), and the red mask represents the False Negatives (FN).

The parameter *max_height_percentage* controls how the function *calculate_peak_region* determines the left and right boundaries of the area of interest. Specifically, when a neighboring peak exceeds a given percentage of the maximum peak's height, it is included in the region. In other words, this parameter sets the minimum height a peak must reach (relative to the maximum peak) to be considered part of the object. This threshold is illustrated as a dashed blue line in the HDF plots.

This parameter is useful for filtering out outliers within the Region of Interest. For instance, in the first row of Fig. 5, a permissive *max_height_percentage* (5%) causes parts of the stillage box to be included in the mask. However, increasing the value to 10% (as shown in the second row) allows the method to filter out the stillage box more effectively.

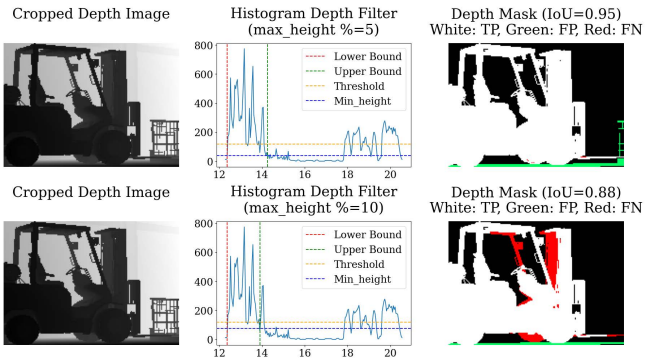

Fig. 5. The parameter *max_height_percentage* defines the horizontal limit to include an adjacent peak as part of the region of interest (*resolution* = 10).

The final parameter of interest is *ignore_bg*. This option is particularly important for classes like pallet trucks and forklifts, where the long forks often cause the background to occupy a large portion of the cropped depth image. As a result, the background may become the dominant peak in the histogram, as illustrated by the large peak at the end of the HDF in Fig. 6. Setting *ignore_bg* = True for these object classes helps to reduce false positives by ignoring such background-dominant peaks during segmentation.

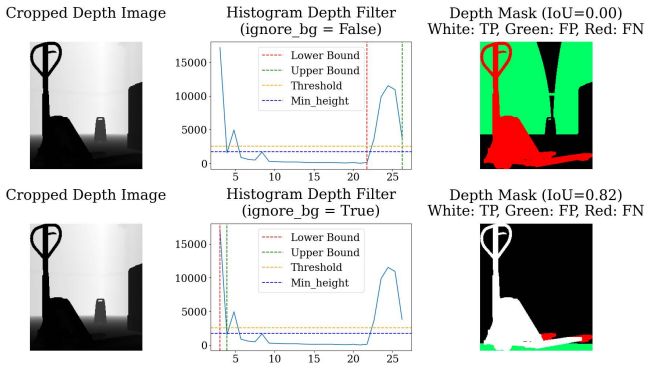

Fig. 6. The parameter *ignore_bg* is useful for classes where a large portion of the background is visible in the cropped depth image.

C. 3D Pose Estimation

Once the object depth mask is correctly isolated, the pose of its central point is calculated using the intrinsic parameters as in Eqs. (1)–(3). Here, (u, v) represents the pixel coordinate of the object's central point, (C_x, C_y) is the center of the image, and (f_x, f_y) is the focal length of the camera. X, Y, and Z are the real-world coordinates in the camera's reference frame.

In the same way, the closest and furthest points as well as the leftmost and rightmost points, are calculated. These edge values are used to generate the object's 3D bounding box, as in Fig. 7. Finally, its orientation is calculated using Eq. (4),

where (x_1, y_1) and (x_2, y_2) are the coordinates of Point I and Point II, respectively (Fig. 7).

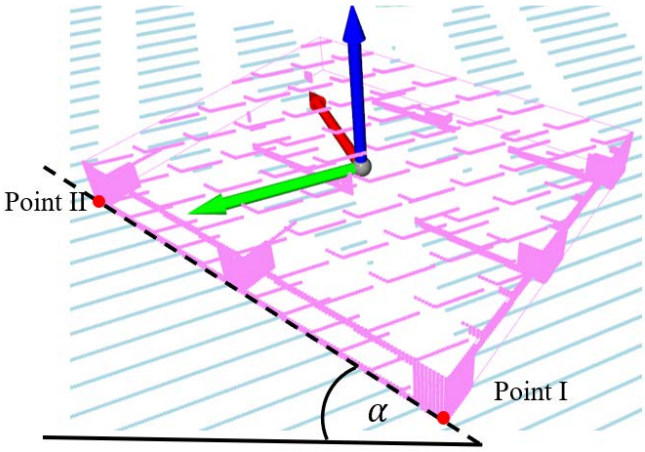

Fig. 7. Calculation of the object 3D pose.

$$X = \frac{(u - c_x)\cdot Z}{f_x} \quad (1)$$

$$Y = \frac{(v - c_y)\cdot Z}{f_y} \quad (2)$$

$$Z = \text{Depth}(u, v) \quad (3)$$

$$\alpha = \arctan(\left|\frac{y_2 - y_1}{x_2 - x_1}\right| \quad (4)$$

D. Test Dataset

As explained in the literature review, 3D object detection in logistic environments has not received as much attention as autonomous driving or household applications. Consequently, there are no public logistic datasets with 3D annotations, and no established benchmarks exist for comparison with state-of-the-art algorithms. To evaluate the developed method, a new synthetic dataset was generated using NVIDIA Isaac Sim. This dataset is based on the original LOCO dataset [24] and consists of 204 images, featuring the original five classes, as well as a robot transporter and distractor objects, in a warehouse environment.

To enhance generalization for multidomain detection, the images were created with variable lighting, randomized object and camera poses, and diverse occlusion and complexity conditions (Fig. 8). The randomization process followed uniform probability distributions within predefined spatial and illumination bounds, ensuring stochastic variability across samples while maintaining physical realism. The number of instances for each class is as follows: forklift (122), pallet truck (104), pallets (69), small load carrier (KLT, 101), and stillage (94). The full dataset and generation code are available on the project's GitHub [27].

Additionally, 33 images were captured in a real logistics environment using two different depth cameras: a Kinect Azure (14 images) and a ZED2 (19 images). This small dataset allows us to assess the suitability of the developed method with real data and sensors, which often differ from the readings obtained in simulation environments. The captured data was manually annotated (Fig. 9), and the class distribution is as follows: forklift (14), pallets (137), pallet trucks (18), small load carrier (KLT, 29), and stillage box (12).

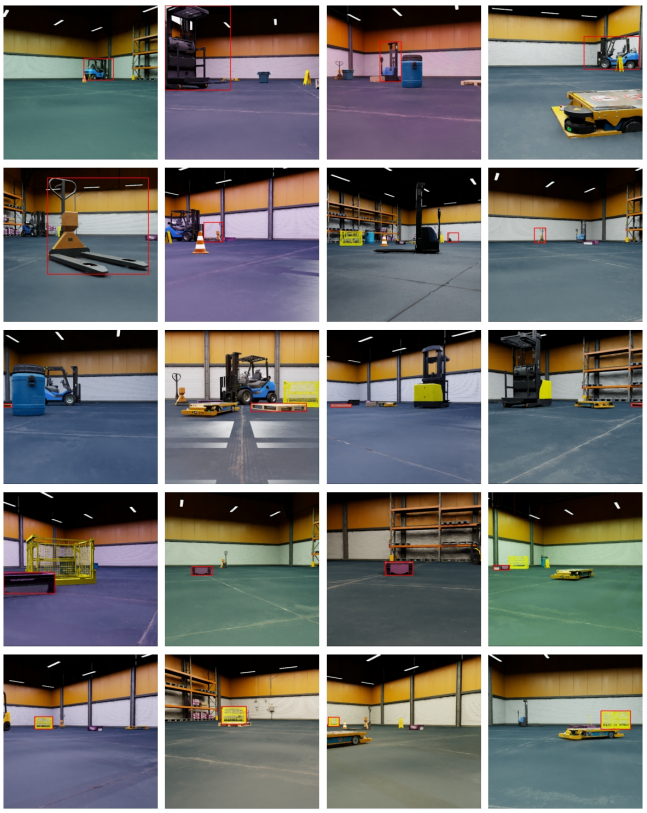

Fig. 8. Test dataset generated using the Nvidia Isaac Sim environment.

Fig. 9. Test dataset captured in a real logistics environment.

IV. Experimental Results

The experiment involved using the annotated bounding boxes to generate the REoIs that serve as input for the described HDF. This approach is important to analyze only the performance of the HDF, isolating the results from potential inaccuracies introduced by the 2D object detector.

A. Parameter Sensitivity Analysis on Simulated Data

Before conducting the performance tests, a sensitivity analysis was performed to determine how the HDF parameters influence the final results. The parameters *resolution*, *max_height_percentage*, and *ignore_bg* were varied independently while keeping the others fixed, as shown in Table 2. The resolution parameter was tested at values of 1, 10, and 100, corresponding to clustering the depth points every 1 m, 1 dm, and 1 cm, respectively.

The parameter *max_height_percentage* was set to either 5% or 10%. This means that, when the dominant peak within a region is chosen, all adjacent points with a height exceeding 5% or 10% of the dominant peak's height are added to the

region. This process continues until the criterion is no longer met, thereby defining the left and right bounds of the region. For the *ignore_bg* parameter, both possible values (True and False) were considered.

As this method aims to isolate the depth mask of objects of interest, the evaluation focused primarily on IoU and mAP between the ground truth and the generated mask. The number of false positives (IoU < 0.5) was also considered, as incorrect segmentation can negatively affect 3D pose estimation. The results, presented in Table 2, demonstrate stable performance across parameter combinations, with optimal settings: *resolution* = 1, *max_height_percentage* = 5, and *ignore_bg* = True for our synthetic dataset.

Using a 1 m resolution helps smooth the histogram representation, which is sufficient for separating isolated objects in warehouse environments. Fewer histogram bins also improve the speed of the filter. In contrast, higher resolutions such as 10 or 100 can help to separate closely spaced objects, but may also cause the filter to break different parts of the same large object (see Fig. 4).

Regarding the *max_height_percentage*, a relatively large value (10%) can help prevent merging of adjacent objects. However, it may also break up regions within the object of interest, such as the forks of a pallet truck or a forklift. Conversely, if this parameter is set too low, it may fail to separate adjacent objects. Nonetheless, a value of 5% yielded the best results on average through the generated test dataset.

Finally, the *ignore_background* option is particularly relevant for L-shaped objects like pallet trucks, which often include large background regions within their 2D bounding boxes. Enabling this option significantly improves segmentation accuracy and reduces false positives.

Table 2. Sensitivity analysis of HDF parameters: simulated data

Resolution	Height %	Ignore BG	Mean IoU	mAP@50	FP (IoU < 0.5)
1	5	False	0.71	0.71	90
1	**5**	**True**	**0.90**	**0.95**	**10**
1	10	False	0.71	0.71	89
1	10	True	0.88	0.94	12
10	5	False	0.61	0.59	123
10	5	True	0.72	0.72	71
10	10	False	0.58	0.56	126
10	10	True	0.65	0.66	88
100	5	False	0.39	0.27	222
100	5	True	0.40	0.27	218
100	10	False	0.36	0.21	245
100	10	True	0.36	0.21	245

B. Performance Test on Simulated Data

After determining the optimal parameter combination for the Cumulative Histogram Depth Filtering function across the dataset, a final performance test was conducted. The results, summarized in Table 3, report class-wise metrics including the mean Intersection over Union (IoU), distance error, mAP@50, and the number of false positives. The experiment yielded highly satisfactory outcomes, achieving an overall mean IoU of 0.90, an average distance error of 0.14 m, and an mAP@50 of 0.95. These results demonstrate the robustness of the proposed method across various logistic object classes.

Table 3. Evaluation results per class for the synthetic dataset

Class	Entries	Dist. Error (m)	Mean IoU	mAP@50	FP (IoU < 0.5)
forklift	120	0.13	0.95	0.96	3
Pallet_truck	72	0.16	0.87	0.90	4
pallet	64	0.20	0.84	0.95	1
klt	94	0.09	0.93	0.99	0
stillage	90	0.12	0.94	0.98	2
Overall	**440**	**0.14**	**0.90**	**0.95**	**10**

Qualitative results, such as those presented in Fig. 10, further confirm the method's ability to segment objects accurately under diverse conditions. Among the five classes, the pallet truck posed the greatest challenge, primarily due to its L-shaped geometry. In 32 cases, the object occupied less than 20% of the pixels within the Region of Interest, making it difficult for the HDF to retain the relevant depth points. This highlights a key limitation when dealing with complex or asymmetric geometries in logistics environments, which are less frequently encountered in conventional object detection benchmarks.

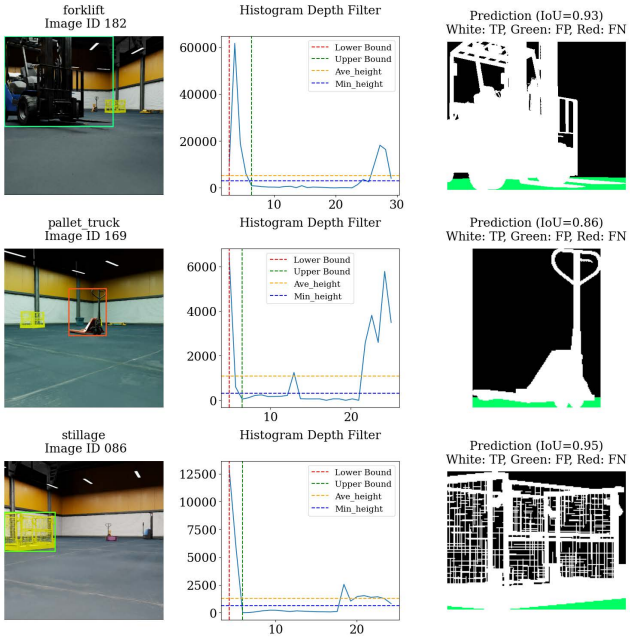

Fig. 10. Results obtained during the experiment with the Synthetic dataset.

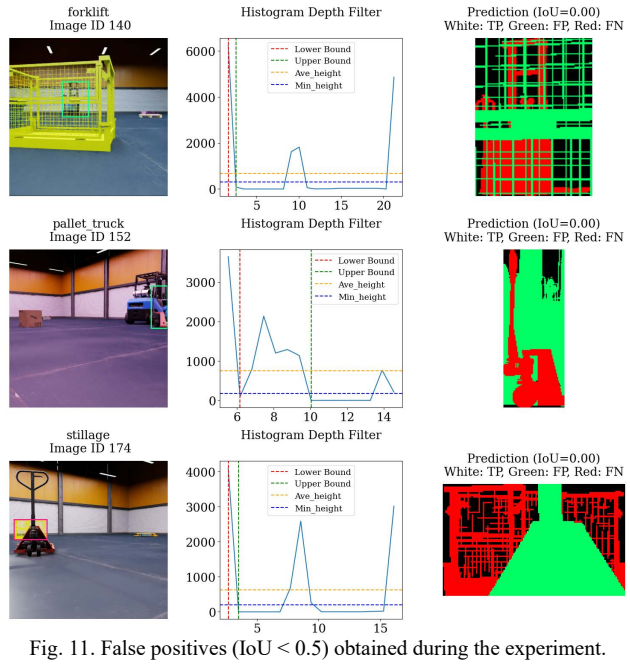

Fig. 11. False positives (IoU < 0.5) obtained during the experiment.

Regardless of these challenges, the method demonstrated strong generalization, with only ten false positives recorded across 440 evaluated instances (for the selected parameters). Most of these errors occurred in scenes with heavy occlusion, particularly involving forklifts and stillage boxes, where the object of interest was obstructed by closer elements in the scene (Fig. 11).

C. Performance Test on Real Data

Despite the successful result shown in the previous section, the domain gap between real and artificial data remains a significant challenge. Therefore, the sensitivity and performance tests were repeated with the previously described manually annotated real dataset.

Due to the complexity of manually labeling hundreds of objects in point clouds, this dataset is not yet large enough to serve as a benchmark. For its acquisition, we employed two widely used 3D cameras: the ZED2 (19 images) and the Azure Kinect (14 images). This process revealed another key difference between real and simulated data: real cameras produce noisier point clouds compared to the ideal outputs from simulation.

Moreover, differences in sensing technology affect the data quality. For example, structured-light cameras such as the Azure Kinect offer better point discretization (neighboring points are more distinctly separated) but struggle with black or distant objects—essentially, any surface that poorly reflects the projected pattern. In contrast, stereo cameras such as the ZED2 are less affected by object color but tend to distort distant objects and blur their edges (Fig. 12).

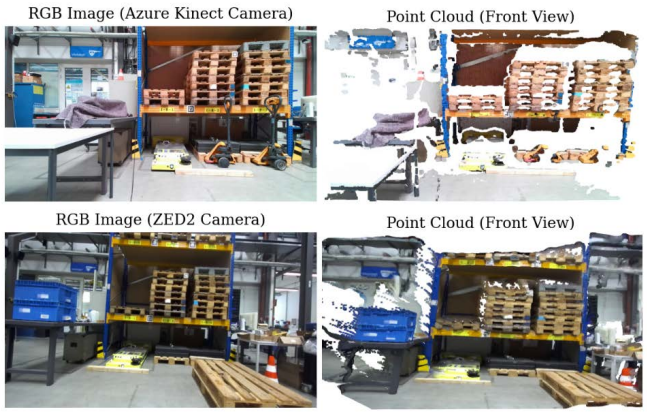

Fig. 12. Comparison of point clouds captured with an Azure Kinect and a ZED2 camera. The Azure Kinect (top row) exhibits finer point discretization but struggles with black and distant objects (e.g., pallet-truck handle and background wall). The ZED2 (bottom row) preserves the color consistency but shows edge distortions for distant objects, such as the blue KLT on the table.

Table 4 presents the sensitivity analysis results for the ZED2 camera. In this case, the *resolution* and *max_height_percentage* parameters were varied across the values 1, 10, and 100, and 5 and 10, respectively. The *ig_background* parameter was kept as True since it showed no impact on the results. This can be attributed to the characteristics of depth cameras, which have a limited sensing range (typically 5–8 m) and therefore cannot effectively capture distant background points.

Table 4. Sensitivity analysis of HDF parameters: ZED2 camera

Resolution	Height %	Ignore BG	Mean IoU	mAP@50	FP (IoU < 0.5)
1	5	True	0.75	0.61	38
1	10	True	0.75	0.61	38
10	**5**	**True**	**0.76**	**0.67**	**35**
10	**10**	**True**	**0.76**	**0.67**	**35**
100	5	True	0.73	0.65	51
100	10	True	0.73	0.65	51

For this experiment, the best configuration was obtained with a *resolution* of 10. The *max_height_percentage* had no significant impact when used with the same *resolution*. In contrast, setting the *resolution* to 100 led to the worst performance across all evaluated metrics, particularly due to the 51 misclassified elements resulting from this parametrization.

Table 5 presents the per-class results for this set of parameters. As shown in the table, the highest distance errors occur in the forklift and pallet truck classes. This is caused by the inclusion of the floor within the cropped region of interest (see Fig. 13). Since the floor is located close to the objects in depth space, it often appears as a continuous surface in the histogram. Consequently, the HDF fails to separate the object from the floor, reducing the accuracy of the pose estimation.

Table 5. Evaluation results per class for the ZED2 camera

Class	Entries	Dist. Error (m)	Mean IoU	mAP@50	FP (IoU < 0.5)
forklift	10	0.26	0.73	0.60	3
pallet	87	0.04	0.72	0.70	25
pallet_truck	10	0.55	0.84	0.30	6
small_load_carrier	18	0.02	0.85	0.89	1
stillage	6	0.02	0.90	0.83	0
Overall	**131**	**0.09**	**0.76**	**0.67**	**35**

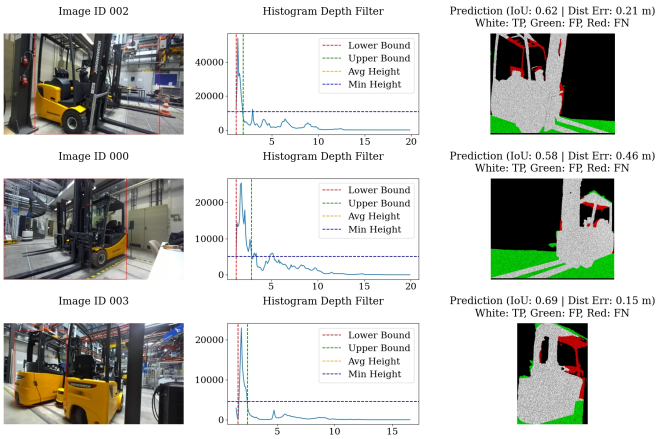

Fig. 13. Results obtained during the experiment with the ZED2 camera.

Another notable observation is the high number of false positives in the pallet class. In real warehouse settings, pallets are frequently stacked vertically, causing multiple instances to appear within a single bounding box (see Fig. 14). This overlap makes it difficult for the HDF to distinguish between individual pallets, leading to suboptimal mask-level segmentation. However, the distance error remains low because the stacked pallets are located at similar depths.

Similarly, the sensitivity analysis conducted with the Azure Kinect camera (Table 6) indicates that varying the HDF parameters does not lead to substantial differences in performance. This demonstrates the difference between real and simulated data.

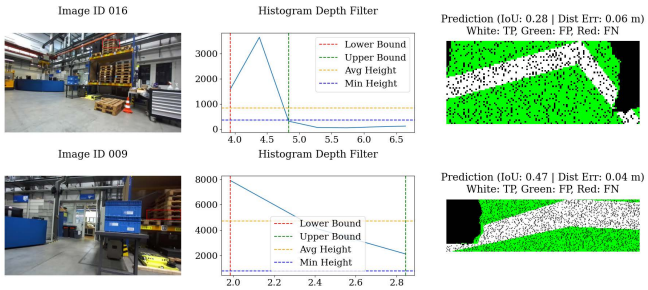

Fig. 14. False positives in the pallet class caused by stacked pallets in warehouse environments.

Table 6. Sensitivity analysis of HDF parameters: Azure Kinect camera

Resolution	Height %	Ignore BG	Mean IoU	mAP@50	FP (IoU < 0.5)
1	5	True	0.78	0.56	26
1	10	True	0.78	0.56	26
10	**5**	**True**	**0.80**	**0.63**	**20**
10	**10**	**True**	**0.80**	**0.63**	**20**
100	5	True	0.78	0.50	31
100	10	True	0.78	0.50	31

Among the tested configurations, the best results were obtained with a *resolution* of 10, achieving the highest mean IoU (0.80), mAP@50 (0.63), and the lowest number of false positives (20). In contrast, the *max_height_percentage* and *ig_background* parameters had no noticeable impact on the results. This suggests that the performance using this sensor is primarily influenced by the chosen *resolution*. The superior results achieved with a *resolution* of 10—compared to 1 in the case of simulated data—highlight how a centimeter-level scale adjusts better to dense, real-world logistic environments.

The per-class evaluation in Table 7 shows that, once again, the forklift and pallet truck classes exhibit the highest distance errors. These are also the largest objects, and incomplete capture of their geometry—such as missing forks or the inclusion of surrounding floor areas—can negatively affect pose estimation.

Table 7. Evaluation results per class for the Azure Kinect camera

Class	Entries	Dist. Error (m)	Mean IoU	mAP@50	FP (IoU < 0.5)
forklift	4	0.37	0.87	0.75	0
pallet	50	0.11	0.76	0.54	17
pallet_truck	8	0.48	0.76	0.13	3
small_load_carrier	11	0.06	0.87	0.91	0
stillage	6	0.24	0.86	0.83	0
Overall	**79**	**0.16**	**0.8**	**0.63**	**20**

The pallet class also shows suboptimal performance in terms of false positives. This is mainly due to the Azure Kinect's limitations in detecting distant objects. Additionally, black pallets present a specific challenge, as the sensor struggles to acquire reliable depth measurements from low-reflectivity surfaces.

Finally, the pallet truck class records the lowest mAP@50. This is attributed to its L-shaped geometry and, in the case of the Azure Kinect, the sensor's inability to detect black components such as the pallet truck's handle (see Fig. 15). These two factors—limited sensing range and poor performance on dark surfaces—significantly hinder the Azure Kinect's suitability for the intended application.

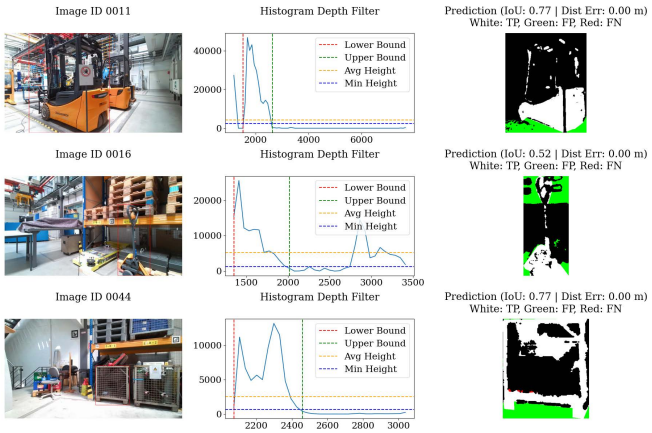

Fig. 15. Results obtained during the experiment with the Kinect Azure camera.

D. Computational Efficiency Analysis

In Table 8, it can be observed that the proposed hybrid 3D object detection method achieves a mean inference time of 9.46 ms, corresponding to a real-time processing speed of approximately 105 FPS. Notably, the majority of this computation time is consumed by the 2D object detector (YOLOv11-N), whose inference alone accounts for 7.16 ms.

On the other hand, our architecture is modular, and the detection head can be easily replaced with faster or more advanced 2D object detection networks, offering flexibility for future improvements. Consequently, the proposed HDF method introduces only minimal computational overhead (2.3 ms on average). This means it is an ideal lightweight resource for 3D point cloud segmentation.

Table 9 compares our method against other SOTA RGB-D based 3D object detection frameworks. Unlike many hybrid approaches that combine a 2D detector with a secondary deep learning module for 3D pose estimation, our method avoids the latency introduced by these additional networks. As a result, the proposed pipeline outperforms more complex architectures in terms of speed while maintaining competitive accuracy.

Another advantage is that the HDF process is analytical and interpretable. Conversely, deep neural networks often function as black boxes with behavior that is difficult to interpret or verify. This makes the proposed method particularly suitable for real-time applications in cognitive robotics, where computational efficiency and explainability are critical.

Table 8. Runtime analysis of the proposed pipeline tested on NVIDIA RTX3060

Module	Stage	Mean (ms)	Max (ms)	Min (ms)	Std (ms)
YOLO11n Object Detection	Preprocessing	1.57	2.30	1.20	0.28
	Inference	4.46	10.90	3.60	0.88
	Postprocessing	1.13	9.00	0.30	1.60
Depth Segmentation Pipeline	Preprocessing	1.80	3.90	1.40	0.30
	Inference Callback	0.50	1.20	0.00	0.50
Total	**-**	**9.46**	**27.3**	**6.50**	**3.56**

Table 9. Comparison of RGB-d-based 3D object detection methods

Method	Year	2D REoI	3D Pose Estimation	Inference Time/FPS
2D-Driven 3D Object Detection [16]	2017	Faster R-CNN	Geometric alignment	0.24 FPS
Frustum PointNets [15]	2018	FPN	PointNet segmentation + regression	5 FPS
VoxelNet [12]	2018	None	Voxel feature encoder + RPN	4.4 FPS
Expandable YOLO [21]	2020	YOLOv3	Anchor-based 3D regression	44 FPS
Hybrid 3D Object Detection (Previous) [17]	2022	YOLOv3-tiny	Point-cloud depth analysis	75 FPS
OMNI3D [10]	2023	Faster R-CNN	3D anchor-free regression	5–6.7 FPS
FusionVision [20]	2024	YOLOv8	Fast SAM	27.3 FPS
Hybrid 3D Object Detection using HDF (**Ours**)	2025	YOLOv11-N	Histogram-based depth filtering	**105 FPS**

V. Conclusion and Future Work

This paper presented a hybrid 3D object detection method that enables 3D pose estimation using only 2D-labeled data, significantly reducing the annotation effort typically required for training 3D perception systems. By integrating the real-time YOLOv11-N object detector with a lightweight histogram-based depth filtering (HDF) module, the proposed approach achieves real-time inference at up to 105 FPS, outperforming existing RGB-D-based methods in computational efficiency while maintaining competitive accuracy.

The architecture is modular, interpretable, and avoids the use of a second deep learning network for 3D pose estimation, making it particularly suitable for resource-constrained robotics applications. Extensive tests conducted with both synthetic and real data demonstrated promising results in terms of mAP@50, IoU, and distance error. However, the absence of a standardized benchmark dataset tailored to logistics environments limits direct comparisons with state-of-the-art methods.

Additional limitations were identified in the form of false-positive detections. Due to its nature, the HDF algorithm segments dominant depth regions, which can lead to suboptimal results. In particular, it struggles to distinguish individual elements in stacked configurations (e.g., vertically stacked pallets) and to separate the floor from large objects like forklifts. For objects with complex geometries (e.g., L-shaped pallet trucks, the method may mistakenly segment background structures, degrading 3D pose estimation performance.

Future work will focus on redesigning the pipeline by replacing the object detector with an instance segmentation algorithm. Unlike bounding-box-based detectors, instance segmentation methods produce object-specific masks that exclude surrounding clutter, floor regions, and occlusions, leading to more precise REoIs and improved segmentation quality. This change is expected to enhance the robustness of the 3D detection pipeline, particularly in cluttered or complex logistics scenarios. Further contributions will also focus on creating a large, real-data, annotated RGB-D dataset for logistics, which can serve as a benchmark for evaluating and comparing 3D object detection methods in this domain.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Daniel Vidal (DV) conducted the research, formulated the HDF algorithm, and performed the experimental tests. Boyan Liu (BL) produced the 2D object detection comparison and carried out the computational speed analysis. Johannes Fottner (JF) supervised the research and contributed to proofreading and reviewing the manuscript. All authors contributed to the discussion of results and approved the final version of the manuscript.

References

[1] J. Fottner, D. Clauer, F. Hormes *et al.*, "Autonomous systems in intralogistics: State of the art and future research challenges," *Logistics Research*, vol. 14, no. 1, pp. 1–41, 2021.

[2] C. Mayershofer and J. Fottner, "Towards an artificial perception framework for autonomous robots in logistics," in *Proc. Advances in Automotive Production Technology—Theory and Application: Stuttgart Conference on Automotive Production*, 2021, pp. 407–415.

[3] J. Hu and D. Shen, "Research on 3D object pose estimation method in intelligent logistics," in *Proc. 2019 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*, 2019, pp. 94–98.

[4] S. D. Achirei, R. Mocanu, A. T. Popovici *et al.*, "Model-predictive control for omnidirectional mobile robots in logistic environments based on object detection using CNNs," *Sensors*, vol. 23, no. 11, 4992, 2023.

[5] Z. Xu, X. Zhan, Y. Xiu *et al.*, "Onboard dynamic-object detection and tracking for autonomous robot navigation with RGB-D camera," *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 651–658, 2023.

[6] B. Molter and J. Fottner, "Real-time pallet localization with 3D camera technology for forklifts in logistic environments," in *Proc. 2018 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*, 2018, pp. 297–302.

[7] J. Tremblay, T. To, B. Sundaralingam *et al.*, "Deep object pose estimation for semantic robotic grasping of household objects," arXiv Preprint, arXiv:1809.10790, 2018.

[8] Y. Labbé, J. Carpentier, M. Aubry, and J. Sivic, "Cosypose: Consistent multi-view multi-object 6D pose estimation," in *Proc. 16th European Conference of Computer Vision–ECCV*, 2020, pp. 574–591.

[9] Y. Lin, J. Tremblay, S. Tyree, P. A. Vela, and S. Birchfield, "Multi-view fusion for multi-level robotic scene understanding," in *Proc. 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 6817–6824.

[10] G. Brazil, A. Kumar, J. Straub *et al.*, "Omni3D: A large benchmark and model for 3D object detection in the wild," in *Proc. the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 13154–13164.

[11] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3D classification and segmentation," in *Proc. the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 652–660.

[12] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3D object detection," in *Proc. the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4490–4499.

[13] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3D object proposal generation and detection from point cloud," in *Proc. CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 770–779.

[14] Z. Meng, X. Xia, R. Xu, W. Liu, and J. Ma, "Hydro-3D: Hybrid object detection and tracking for cooperative perception using 3D lidar," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 8, pp. 4069–4080, 2023.

[15] C. R. Qi, W. Liu, C. Wu *et al.*, "Frustum pointnets for 3D object detection from RGB-D data," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 918–927.

[16] J. Lahoud and B. Ghanem, "2D-driven 3D object detection in RGB-D images," in *Proc. IEEE International Conference on Computer Vision*, 2017, pp. 4622–4630.

[17] D. Vidal-Soroa, P. Furelos, F. Bellas *et al.*, "An approach to 3D object detection in real-time for cognitive robotics experiments," in *Proc. Iberian Robotics Conference*, 2022, pp. 283–294.
[18] M. Simony, S. Milzy, K. Amendey, and H.-M. Gross, "Complex-YOLO: An Euler-region proposal for real-time 3D object detection on point clouds," in *Proc. European Conference on Computer Vision (ECCV) workshops*, 2018.
[19] L. Zuo, Y. Li, M. Han, Q. Li, and Y. Liu, "Frustum fusionnet: Amodal 3D object detection with multi-modal feature fusion," in *Proc. 2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 2746–2751.
[20] S. El Ghazouali, Y. Mhirit, A. Oukhrid *et al.*, "Fusionvision: A comprehensive approach of 3D object reconstruction and segmentation from RGB-D cameras using YOLO and fast segment anything," *Sensors*, vol. 24, no. 9, 2889, 2024.
[21] M. Takahashi, Y. Ji, K. Umeda, and A. Moro, "Expandable YOLO: 3D object detection from RGB-D images," in *Proc. 21st International Conference on Research and Education in Mechatronics (REM)*, 2020, pp. 1–5.
[22] J. Piekenbrinck, A. Hermans, N. Vaskevicius, T. Linder, and B. Leibe, "RGB-D cube R-CNN: 3D object detection with selective modality dropout," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 1997–2006.
[23] C. Mayershofer, D. M. Holm, B. Molter, and J. Fottner, "LOCO: Logistics objects in context," in *Proc. 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2020, pp. 612–617.
[24] Electronic Article. (2024). [Online]. Available: https://github.com/ultralytics/ultralytics
[25] Electronic Article. (2025). Definition of real-time. *Cambridge Dictionary* [Online]. Available: https://dictionary.cambridge.org/dictionary/english/real-time
[26] Microsoft Documentation. (2021) Azure kinect dk hardware specifications. [Online]. Available: https://learn.microsoft.com/en-us/previous-versions/azure/kinect-dk/hardware-specification
[27] D. Vidal. (2025). 3D object detection GitHub repository. *GitHub*. [Online]. Available: https://github.com/vidaldani/3D_object_detection