electronics

Article

Simultaneous Object Detection and Distance Estimation for Indoor Autonomous Vehicles

Iker Azurmendi [1,2], Ekaitz Zulueta [1], Jose Manuel Lopez-Guede [1,*] and Manuel González [2]

[1] Department of Systems and Automatic Control, Faculty of Engineering of Vitoria-Gasteiz, University of the Basque Country (UPV/EHU), Nieves Cano, 01006 Vitoria-Gasteiz, Spain; iazurmendi@centrostirling.com (I.A.); ekaitz.zulueta@ehu.eus (E.Z.)
[2] CS Centro Stirling S. Coop., Avda. Álava 3, 20550 Aretxabaleta, Spain; mgonzalez@centrostirling.com
* Correspondence: jm.lopez@ehu.eus

Abstract: Object detection is an essential and impactful technology in various fields due to its ability to automatically locate and identify objects in images or videos. In addition, object-distance estimation is a fundamental problem in 3D vision and scene perception. In this paper, we propose a simultaneous object-detection and distance-estimation algorithm based on YOLOv5 for obstacle detection in indoor autonomous vehicles. This method estimates the distances to the desired obstacles using a single monocular camera that does not require calibration. On the one hand, we train the algorithm with the KITTI dataset, which is an autonomous driving vision dataset that provides labels for object detection and distance prediction. On the other hand, we collect and label 100 images from a custom environment. Then, we apply data augmentation and transfer learning to generate a fast, accurate, and cost-effective model for the custom environment. The results show a performance of mAP0.5:0.95 of more than 75% for object detection and 0.71 m of mean absolute error in distance prediction, which are easily scalable with the labeling of a larger amount of data. Finally, we compare our method with other similar state-of-the-art approaches.

Keywords: object detection; YOLO; distance estimation; autonomous vehicles; indoor navigation; AGV

check for updates

Citation: Azurmendi, I.; Zulueta, E.; Lopez-Guede, J.M.; González, M. Simultaneous Object Detection and Distance Estimation for Indoor Autonomous Vehicles. *Electronics* **2023**, *12*, 4719. https://doi.org/10.3390/electronics12234719

Academic Editors: Yi Xie, Dan Dan and Jiahao Liu

Received: 18 October 2023
Revised: 11 November 2023
Accepted: 17 November 2023
Published: 21 November 2023

1. Introduction

Autonomous navigation has become a hot topic in recent years [1], especially for autonomous road vehicles. Various development groups, from automobile manufacturers to technology and IT companies, are contributing to the autonomous transport of cargo and people. At the same time, robotics and automation are increasingly being developed to replace manual work and improve the performance of tasks in almost all fields. Unfortunately, the autonomous navigation of indoor mobile robots has not received as much attention as that of road vehicles, especially in the case of vision-guided robots [2].

Mobile robots are mechatronic devices that can move in a physical environment and are used to assist humans in various activities, whether hazardous or non-hazardous, repetitive, and disruptive. This category of robots interacts with the environment through sensors and actuators to move and perform their functions autonomously. They are typically used in offices, hospitals, production lines, or industry. In the latter case, automated guided vehicles (AGVs) stand out [3].

One of the main problems to be solved to make autonomous driving feasible is the reliable and accurate detection of obstacles [4]. Especially in recent years, automated driving technology has gained attention, and obstacle-avoidance algorithms have become an important and indispensable technology. As mentioned by Chang et al. [5], obstacle avoidance has always been a very important research topic in the field of robotics. Similarly, Hanumante et al. [6] describe real-time obstacle avoidance as one of the key issues for successful applications of mobile robot systems.

All mobile robots have some form of collision avoidance system, ranging from algorithms that detect an obstacle and stop the robot before it reaches it, to sophisticated algorithms that allow for the robot to avoid obstacles. The latter algorithms are much more complex, as they involve not only the detection of an obstacle but also some kind of quantitative measurement of the dimensions of the obstacle. Once the obstacle has been detected, the obstacle avoidance algorithm must steer the robot around the obstacle and resume movement towards the original path [7].

Automatic obstacle avoidance is critical for many autonomous systems, such as Unmanned Aerial Vehicles (UAVs) or Autonomous Mobile Robots (ARMs). In recent years, the development of intelligent and autonomous mobile robots has been at the heart of autonomous vehicle research, as they provide safety and intuitive control when working in different scenarios, such as surveillance, mapping, construction, delivery, and traffic monitoring, among others.

According to Yuri D. V. Yasuda et al. [2] the following terms are defined:

- **Obstacle.** An obstacle is a part of the environment, an agent, or any other object that the robot must avoid colliding with.
- **Obstacle detection.** Obstacle detection is the process of finding an obstacle and determining its position. This can be performed using distance measurements, images, and sounds. It is important to avoid collisions with the robot, which could result in injury or damage. As discussed above, obstacle detection is a sub-task of the locomotion problem.

Over the past decades, significant research has been devoted to the problems of obstacle detection [8,9] and environmental perception, including object detection and distance estimation. Even though much attention has been paid to the object detection task, distance estimation still receives very little interest in the computer vision community [10]. Radars and LiDaRs (Light Detection and Ranging) can be used to provide distance information, but they are expensive or provide poor information about objects compared to image sensors. Stereo vision and monocular vision can also be used for distance estimation. Stereo vision can calculate the distance to objects more intuitively and accurately. However, as mentioned by Huang et al. [11], stereo vision systems require a long execution time due to calibration and matching between both cameras and exhibit low efficiency and considerable computational complexity. On the other hand, although distance estimation using a monocular camera is currently being studied in the literature and is one of the most classical tasks in computer vision, current monocular distance estimation methods require a lot of data acquisition to obtain accurate results [12].

This study proposes an accurate and lightweight deep learning model that can predict object-specific distances from monocular images. The idea has been to modify an existing object-detection algorithm to simultaneously detect objects and estimate distances between them and the camera in an image.

An object distance estimator can be useful in many areas. In autonomous driving, it could allow an integrated vehicle system to inform and alert the driver, either to prompt the driver to take preventive action or to prepare the vehicle's safety systems for an imminent collision [13,14]. As discussed by Davydov et al. [15], accurate distance estimation is a prerequisite for Advanced Driver Assistance Systems (ADAS) to provide the driver with safety-related functions such as adaptive cruise control and collision avoidance. In robot motion, it can provide distance information for collision avoidance. In video surveillance perception, it can provide 3D information for object tracking [16].

Our idea is to develop a low-cost camera-based algorithm that detects objects and estimates distances, which could be used as the first part of a fully automated obstacle avoidance algorithm for an AGV that would be used for the autonomous transport of cargo in industry. The following is an overview of the research contributions:

- To the best of our knowledge, this is the first time that simultaneous object detection and distance prediction has been performed in an autonomous indoor vehicle using only a monocular camera;

- The results show a precise and lightweight object detection and distance-estimation algorithm that can be used for obstacle avoidance in autonomous indoor vehicles;
- Different sized object detection and distance prediction models have been trained on a custom dataset and their comparative has been presented;
- The article demonstrates how an accurate deep learning algorithm can be obtained with few images by using transfer learning;
- A comparison with other state-of-the-art obstacle detection methods for autonomous indoor vehicles is presented.

The rest of the paper is organized as follows. In Section 2, a state-of-the-art overview of obstacle detection work for indoor autonomous vehicles is reviewed. Section 3 presents the YOLO object-detection algorithm and the modified architecture for simultaneous object detection and distance estimation. Additionally, the used datasets and the applied image augmentation techniques are described. The training results, comparing different object detection models, are discussed in Section 4. Moreover, some examples of the algorithm performance are also shown. The results are discussed in Section 5. The paper is concluded in Section 6.

2. Related Work

As noted by Nai-Hsiang Chang et al. [5], obstacle avoidance has always been a very important research topic in the field of robotics. The most important thing is to find out the obstacle location in relation to the robot as accurately as possible [17]. For obstacle avoidance in autonomous vehicles, some basic requirements for image processing include the following features. On the one hand, the prediction must achieve almost 100% accuracy to obtain a reliable obstacle-detection system. On the other hand, the prediction must guarantee real-time processing and fast inference speed to reduce the latency of the autonomous vehicle control loop.

To find a solution to the obstacle avoidance problem, several methods have been proposed in the literature in recent years [18–22]. There are a variety of sensors that can be used for obstacle detection. Some of the most popular sensors [23] are infrared sensors [24,25], ultrasonic sensors [26,27], cameras [28,29], radars [30,31], or LiDaRs [32,33].

Additionally, the rapid development of Machine Learning (ML), especially Deep Learning (DL) [34], has promoted self-learning as a new area of research for robot obstacle avoidance [35]. In this category of techniques, different neural network architectures have been used to detect obstacles using computer vision. Nowadays, due to the development of large-scale computation, Convolutional Neural Networks (CNNs) are one of the most powerful vision-based algorithms. Therefore, their use in mobile robots for obstacle detection has increased.

For example, Liu et al. [36] present an end-to-end CNN-based model for detecting obstacles in images and consequently, generating steering commands for controlling the robot. Similarly, Rezaei and Darabi [29] use a deep neural network to inform the robot about the suitable direction of movement. Moreover, Christiansen et al. [37] introduce DeepAnomaly, an algorithm combining DL and anomaly detection to exploit the homogenous characteristics of a field to perform accurate and reliable anomaly detection. With the combination of background subtraction and DL, they develop a fast state-of-the-art detector for obstacles that are distant, heavily occluded, and unknown.

Furthermore, different object-recognition techniques such as CNN-based architectures are used to find the object in the image and estimate the position of the obstacle in relation to the vehicle. RCNNs (Region-based CNNs) and the YOLO (You Only Look Once) family are the most commonly used CNN-based models for this purpose [38]. For instance, Singh et al. [39] use YOLO object-detection algorithm results to aid the mobile robot in detecting obstacles and navigating in an indoor environment. Another example is provided by Su et al. [40], where an improved YOLOv5s object detection method is used for a semi-structured apple environment.

The use of semantic segmentation in obstacle detection in mobile robotics has also been used in the literature. For example, Teso-Fz-Betoño et al. [41] used this technique to distinguish between the floor, which is the navigation-free space, and the walls, which are the obstacles. Something similar was performed by Dang and Bui [28], who provided a real-time solution to the problem of obtaining hallway scenes from a single image.

On the other hand, as mentioned in the introduction, another objective of this work is to predict the distance to the detected obstacles. To detect distance to objects using computer vision several methods have been proposed in the literature. One solution is to use stereo vision and combine it with an obstacle detection algorithm to determine the distance to objects. The methods proposed by Macias-Garcia et al. [42] and Luo et al. [43] are two examples where deep learning and stereo vision are combined for obstacle detection in mobile robotics.

Another solution is proposed by Scokzeń et al. [44]. They used an RGB-D camera to detect obstacles in an agricultural mobile robot using a four-stage method: collection of RGB and depth images, generation of semantic segmentation based on RGB image, reconstruction of the point cloud with segmentation results and depth image, and projection of point in 2D occupancy grid for detecting obstacle position on the map.

Nevertheless, this study aimed to generate simultaneous object detection and distance estimation using a low-cost monocular camera. Therefore, if only a single camera should be used, some of the most widely used approaches in image-based obstacle detection are the depth-based methods [45]. In these methods, a DL network was trained with appropriate data to determine the depth of each pixel (depth map) in a single image. An obstacle was detected if the predicted pixels in an image had a value lower than a threshold. Monodepth2 [46], Fastdepth [47], and DNet [48] are some examples of monocular depth estimation techniques.

Chen et al. [49] used YOLOv3 and Monodepth [50], a combination of monocular object detection and depth estimation models, to generate a disparity map in a single camera during inference. The distance was estimated from the Monodepth output and used in the predicted YOLO boxes to get the position of the obstacles. Also, Zhang et al. [16] followed the method of [10] to develop deep a neural network that can output object distance directly using an R-CNN-based deep regression network. Their pipeline only requires 2D image datasets with annotations of bounding boxes and distances, which are less expensive to collect.

Another approach used in the literature was the prediction of the distance after detecting the objects. DisNet [4] is an example of a distance estimation network, where the authors used a YOLO object-detection algorithm for bounding box prediction and a neural network for predicting the distances of different objects in an image from the data generated by the regressor and other object-related default parameters. Like DisNet, Natanael et al. [14] used YOLOv3 to detect bounding boxes along with coordinates, and from these coordinates, they calculated distances analytically.

Going a step further, Marek et al. [51] present a modified YOLOv3 architecture to perform object detection and distance prediction simultaneously. They also compare two alternatives: adding a single output to the prediction vector to predict the distance to the object and adding a distance output for each of the classes that can be detected. As they comment, the addition of a single output gives better results.

Additionally, Zhu et al. [10] first proposed the use of distance-valued labels in the training process, which can automatically predict the distance of a given object in RGB images without the camera parameters. The framework of the proposed model consists of a feature extractor, a distance regressor, and a multiclass classifier.

To conclude this section, a comparison of our method with other obstacle detection/avoidance systems for autonomous indoor vehicles is presented in Table 1.

Table 1. State of the art comparison of obstacle detection methods.

Ref	Obstacle Detection				Obstacle Avoidance	Pros [🟢] and Cons [🔴]
		Sensor	Method	Distance Estimation		
[26]		Ultrasonic sensor	Processing of the data collected from the sensor	✓	✓	🟢 Compact size, low cost, and easy implementation. 🟢 Sensing capability with all matering types. 🔴 Short measure distance for low cost sensors (10 m). 🔴 Influenced by air temperature and humidity. 🔴 Not customisable for custom types of obstacles.
[27]				✓	✓	
[52]				✓	✓	
[53]				✓	✓	
[24]		Infrared sensor	Combination of three infrared sensors around the chassis	✓	✓	🟢 Small size. 🟢 Low cost and fast. 🔴 Cannot detect transparent and black objects. 🔴 Several sensors are needed for good performance.
[25]			Combination of data from infrared sensors and a camera	✓	✗	
[54]	LiDaR	2-D RPLiDAR	Filtering, processing, and clustering lidar raw data	✓	✓	🟢 Very-high accuracy measurements. 🟢 High resolution at range. 🟢 Unaffected by darkness or bright light conditions. 🔴 Slower and more expensive than other methods. 🔴 Complex data interpretation. 🔴 Sensitive to dirt.
[55]		LiDaR	Lidar raw data processing	✓	✓	
[17]		2D LiDaR		✓	✗	
[56]	Vision	Gray Scale Camera	Inverse perspective mapping + image abstraction and geodesic distance computation	✗	✗	🟢 Fast and accurate. 🟢 Low cost. 🔴 No distance to obstacle information. 🔴 Manual labelling for quantitative evaluation.
[57]		Omnidirectional vision	Improved dynamic window approach and artificial potential field	✗	✓	🟢 360° vision. 🟢 Robust and effective method (won the 2017 FIRA avoidance challenge). 🔴 No distance to obstacle information.
[58]		Stereo Camera	Depth-map mapping with world coordinates	✓	✓	🟢 High precision compared to monocular vision. 🔴 Large computational complexity. 🔴 High hardware cost.
[44]		RGB-D Camera	Semantic segmentation	✓	✓	🟢 Information for each pixel. 🔴 Laborious image labelling work. 🔴 Powerful hardware needed for fast training and inference.
[28]		RGB Camera		✗	✓	
[40]			Object detection	✗	✗	🟢 Flexible customisation for obstacle detection. 🟢 Accurate results for different seasons. 🔴 No direct distance information.
[29]			Obstacle classification with CNNs	✗	✓	🟢 Easy to train and label. 🟢 Accurate results for trained objects. 🔴 No distance to obstacle information. 🔴 No multi-obstacle detection.
[36]				✗	✓	
[59]			Obstacle edge detection	✓	✓	🟢 Fast, accurate, and easy to implement. 🔴 Only useful for reduced type of obstacles.
[60]			Image processing	✗	✓	🟢 Simple and efficient. 🔴 No distance to obstacle information.
Ours			Object-detection algorithm modification	✓	✗	🟢 Flexible customisation for obstacle detection. 🟢 Fast and accurate. 🟢 Low cost. 🟢 Easily scalable. 🔴 Light and visibility dependent.

The results in Table 1 show that each of the studied methods has its positive and negative aspects. For example, ultrasonic sensors are a low-cost and easy-to-apply solution but have a short detection range. If more sensing distance is required, the price increases. Additionally, LiDaR sensors are more expensive, and the processing of the information is more complex and time-consuming, but they offer very accurate detections. Therefore, the main limitations of the algorithms in the literature so far were that they did not combine obstacle detection and position estimation with a single sensor (you needed to use a sensor combination) and were not customizable to the working environment. On the other hand, our method is fast, accurate, low-cost (only one camera is needed), and it can be customized for each working environment labeling new images of the potential obstacles that the vehicle will meet. Finally, it is easily scalable by tagging as many images as needed.

3. Simultaneous Object Detection and Localization

The idea behind the simultaneous object detection and distance estimation model of this work is the modification of an existing object-detection algorithm. The modification consists of adding an output to the prediction vector of the existing model, which involves modifying the network architecture, the loss function, the way labels are read, and the calculation of model validation metrics. This section details the original object-detection model used as a baseline, the datasets used to train and test the model, and the pseudocode of the overall process followed to achieve the objective of the work.

3.1. YOLO (You Only Look Once)

The base model used to achieve simultaneous object detection and distance estimation was YOLOv5. The YOLOv5 algorithm, an evolution of the YOLOv1, YOLOv2, YOLOv3, and YOLOv4 models, is a family of pre-trained object detection architectures and models designed for the COCO dataset [61].

YOLO is a popular and influential object-detection algorithm and model architecture in the field of computer vision and deep learning. The key innovation of YOLO is that it performs object detection in real time by dividing the image into a grid and making predictions for each grid cell. Unlike traditional object detection methods that involve multiple passes over an image, YOLO takes a one-stage approach, where it simultaneously predicts the bounding boxes and class probabilities of objects in a single forward pass through the neural network. Although YOLO is not the only one-step detection model, it is generally more efficient than the other state-of-the-art algorithms in terms of speed and accuracy [62].

The YOLO algorithm has gone through several versions, with each version (YOLOv1, YOLOv2, YOLOv3, YOLOv4, YOLOv5, YOLOv6, YOLOv7, YOLOv8, YOLOX, YOLOR, PP-YOLO, DAMO-YOLO, or YOLO-NAS) introducing improvements in accuracy and efficiency. Nowadays, the good results of the architecture have allowed for object detection using YOLO algorithms to be used in a wide range of applications such as autonomous driving [63,64], defect detection [65,66], or healthcare [67,68], among others [62,69].

3.1.1. Updating the Prediction Vector

As mentioned above, this work proposes the modification of a YOLOv5 algorithm that, besides locating and labeling the different objects in an image, predicts their distance from the camera. For this purpose, we start with a YOLOv5 algorithm implemented in Keras (https://github.com/yyccR/yolov5_in_tf2_keras, accessed on 11 November 2023). This algorithm, like the original YOLOv5, was designed with a specific number of outputs (see Figure 1), which depend on the number of classes to be identified.

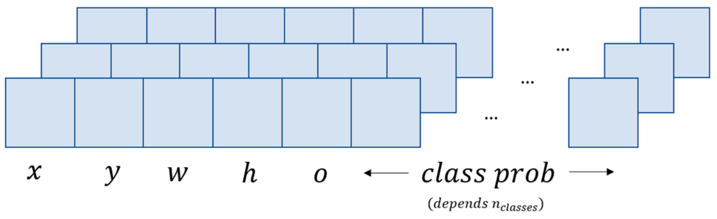

Figure 1. YOLOv5 model prediction vector.

A prediction vector is given as $p = (b, o, c) = (x, y, w, h, o, c_1, c_2, ..., c_n)$, where (x, y, w, h) are coordinates produced by the bounding box regressor, (o) denotes the objectness, the confidence that the prediction vector p captures a real object, and $(c_1, c_2, ..., c_n)$ represents the confidence that the detected object is each of the n classes. The n classes represent the different n objects that can be detected by the model. In total, each prediction vector consists of $5(x,\ y,\ w,\ h,\ o)$, which are constant for each object detector, plus n values, which depends on the number of classes to detect, i.e., $5 + n$.

On the other hand, besides the prediction vector, the output of the YOLOv5 algorithm is a set of tensors representing the detections of objects at different levels of image resolution, corresponding to different anchor scales (anchors). Each tensor has a form (batch size, num anchors, grid size, grid size, num attributes), where

- $batchsize$: the number of images in the input batch;
- $numanchors$: the number of anchors used for each grid cell;
- $gridsize$: the size of the grid that divides the image into cells;
- $numattributes$: the number of attributes by detection, including bounding box coordinates, object confidence scores, class scores, and other related values. This is the explained prediction vector $p = (b, o, c)$.

The objective is to modify the output of the model so that for each prediction it makes, it provides the distance of the object from the camera, as shown in Figure 2.

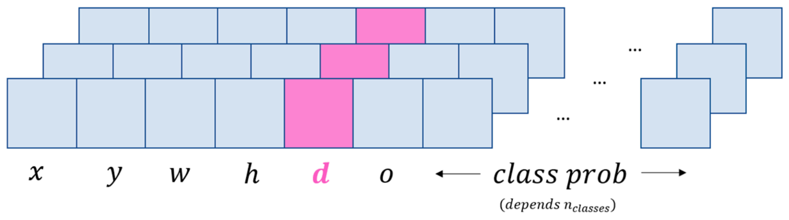

Figure 2. Modified prediction vector of the YOLOv5 model for simultaneous object detection and distance estimation.

For the object-detection algorithm, adding distance to the prediction vector means increasing the number of outputs and modifying the architecture so that the model learns a new prediction output, using the characteristics of a regression output and applying the ability to train itself to minimize the loss of distance estimation as well. Then, for distance detection, the prediction vector is extended in the form $p = (b, d, o, c) = (x, y, w, h, d, o, c_1, c_2, \ldots, c_n)$, where d is the distance to the object and is class independent. In total, each new prediction vector will consist of $6 + n$ values, where n is the number of different objects that can be detected by the model.

The algorithm also uses the following four approaches to improve accuracy and efficiency: Residual Blocks, Bounding Box Regression, Intersection over Unions (IoU), and Non-Maximun Suppression (NMS). First, Residual Blocks are often used as building blocks within the neural network architecture to extract features from input images efficiently. In bounding box regression, the coordinates of the predicted bounding box are adjusted to closely match the true location of the object. IoU measures the overlap between the predicted bounding box and the ground truth bounding box of an object. Finally, NMS is a post-processing technique used to filter out duplicate or redundant object detections.

3.1.2. New YOLO Loss Function

As mentioned above, the original YOLOv5 prediction vector returns three outputs: the probability of each class for the detected objects, the objectness score, and the bounding boxes. Thus, the YOLOv5 loss consists of three parts: class loss (Binary Cross Entropy loss), objectness loss (Binary Cross Entropy loss), and bounding box loss (Complete Intersection over Union loss) (see Equation (1)). On the one hand, the classification loss $(\mathrm{L}_{\mathrm{cls}})$ measures the accuracy of the predictions of the classes of objects detected in each grid cell. On the other hand, the objectness loss $(\mathrm{L}_{\mathrm{obj}})$ measures how well the model predicts whether an object is present in each grid cell. Finally, the bounding box regression loss $(\mathrm{L}_{\mathrm{bbox}})$ assesses how accurate the predictions of the bounding box coordinates around the object are. $\lambda_1,\ \lambda_2,$

and λ_3 are the weighting coefficients (hyperparameters) that control the relative importance of each term in the overall loss function.

$$\text{Loss} = \lambda_1 \cdot L_{cls} + \lambda_2 \cdot L_{obj} + \lambda_3 \cdot L_{bbox} \tag{1}$$

In addition, the balance loss is used to adjust the confidence and regression loss weights in the different detection layers, depending on the size of the objects and the scale of the prediction layer. The balance weights are [4.0, 1.0, 0.4], respectively, and they ensure that the predictions at different scales contribute appropriately to the total loss (see Equation (2)).

$$L_{obj} = 4.0 \cdot L_{obj}^{small} + 1.0 \cdot L_{obj}^{medium} + 0.4 \cdot L_{bbox}^{small} \tag{2}$$

This strategy aims to consider objects of different sizes more equally in terms of loss, which can help the model focus on accurate detection regardless of object size. This technique of differentially weighting losses according to object size and scale is a way of adjusting the relative importance of different aspects of detection in different prediction layers. The new YOLOv5 loss used in this work consists of four parts: class loss (Categorical Cross Entropy loss), objectness loss (Binary Cross Entropy loss), bounding box loss (Complete Intersection over Union loss), and distance loss (MAE) (see Equation (3)).

$$\text{Loss} = \lambda_1 \cdot L_{cls} + \lambda_2 \cdot L_{obj} + \lambda_3 \cdot L_{bbox} + \lambda_4 \cdot L_{dis} \tag{3}$$

Correctly balancing the weights in each term of the loss function is critical to achieve effective training and optimal object detection performance. Each term has a specific purpose, and adjusting the weights allows you to assign the appropriate importance to each feature of the problem improving the accuracy and generalization of the model.

3.2. Datasets

In deep learning, as in machine learning, one of the most important aspects to consider is the type of data you feed the model. The more data you have, the more likely a ML algorithm is to understand them and make accurate predictions on new data. As mentioned by Li Liu et al. [70], datasets have played a crucial role throughout the history of DL research, not only as a common ground for measuring and comparing the performance of competing algorithms but also for pushing the field towards increasingly complex and challenging problems [62].

3.2.1. KITTI Dataset

KITTI is an autonomous driving vision dataset developed by the Karlsruhe Institute of Technology and the Toyota Technological Institute in Chicago for various tasks such as stereo, optical flow, visual odometry, 3D object detection, and 3D tracking [8]. This work uses the KITTI 3D object detection dataset which consists of 7481 training images and 7518 test images and their corresponding point clouds, for a total of 80.256 labeled objects. The description of the dataset and an example are given in Table 2.

Table 2. Description of the KITTI 3D Object Detection dataset.

Name	Type	Truncated	Occluded	Alpha	BBox	Dimensions	Location	Rotation ry
N° of values	1	1	1	1	4	3	3	1
Example	Car	0.0	0	−1.57	596.71 174.68 624.59 201.52	1.66 1.73 3.05	0.01 1.8 46.71	−1.57

Where

- Type describes the type of object: 'Car', 'Van', 'Truck', 'Pedestrian', 'Person_sitting', 'Cyclist', 'Tram', 'Misc' or 'DontCare';
- Truncated is a float from 0 (non-truncated) to 1 (truncated), where truncated refers to the object leaving image boundaries;

- Occluded is and integer (0, 1, 2, 3) indicating occlusion state: 0 = fully visible, 1 = partly occluded, 2 = largely occluded, 3 = unknown;
- Alpha is the observation angle of the object, ranging [-pi...pi];
- Bbox is the 2D bounding box of the object in the image (0-based index): contains left top and right bottom pixel coordinates;
- 3D object dimensions: height, width, length (in meters);
- 3D object location (x,y,z) in camera coordinates (in meters);
- Rotation ry is the rotation around the Y-axis in camera coordinates [−pi...pi].

For this work, we will be calculating the absolute distance from the object to the camera, not just the depth. For this reason, it is necessary to process the location information provided by the KITTI image set. So, from the KITTI dataset, we only need the type, bbox, and location data for our application. In addition, the data can be filtered with the occlusion variable to detect just fully visible objects. Some examples of the KITTI image set are shown in Figure 3.

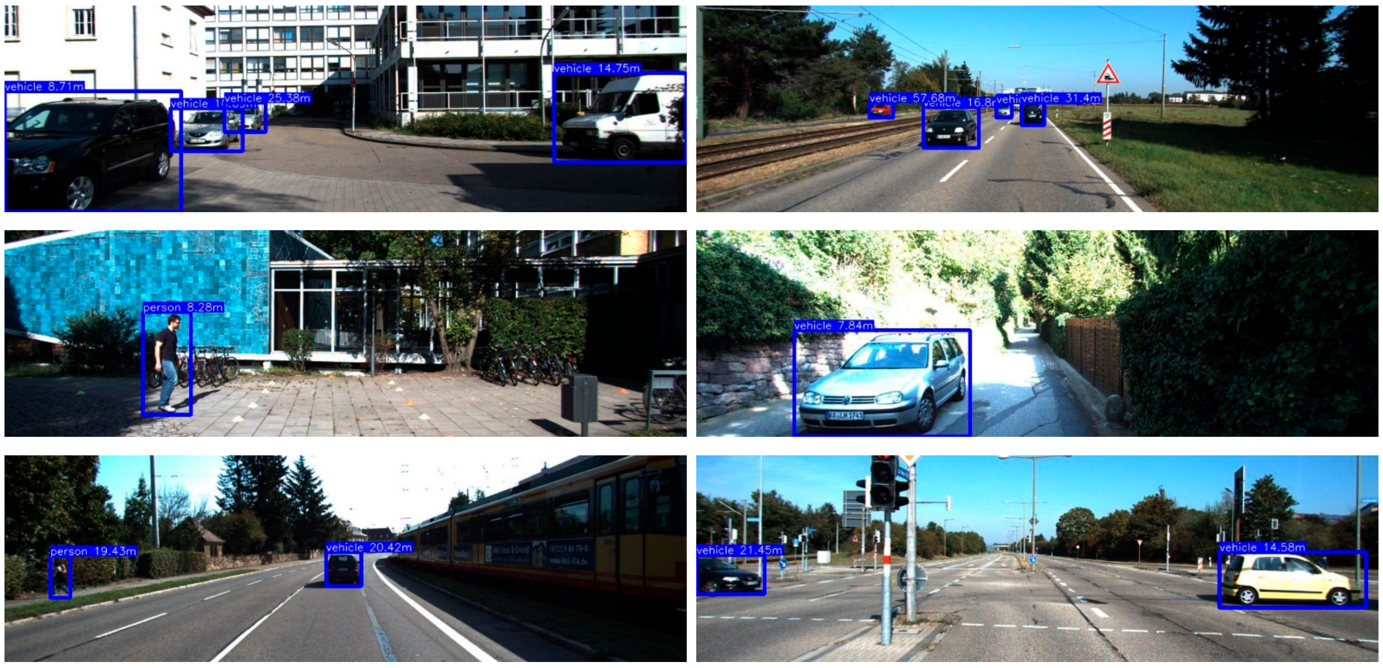

Figure 3. Some labelled KITTI 3D Object Detection dataset images.

3.2.2. Custom Dataset

As mentioned in the introduction, the goal is to detect obstacles in a specific environment. Training the designed object detection and distance-estimation algorithm using only the KITTI dataset would not produce good results for the application, as it would not correctly represent the vehicle's working environment. For this reason, 104 images of the specific environment in which the robot moves were collected and labeled. The images were captured at a resolution of 640 × 480 at the Vitoria-Gasteiz University of Engineering with a Logitech C920s webcam, and they were manually labeled using a laser meter to measure the distance to the obstacles (in this case people). The reason for detecting people as obstacles is that the vehicles we are working with (AGVs) are used for the transport of cargo in an industrial environment, so a possible application of this type of algorithm is to make the vehicle act autonomously when it encounters a worker. Another possible application could be the interaction of a service robot with people. Some images of the dataset are shown in Figure 4.

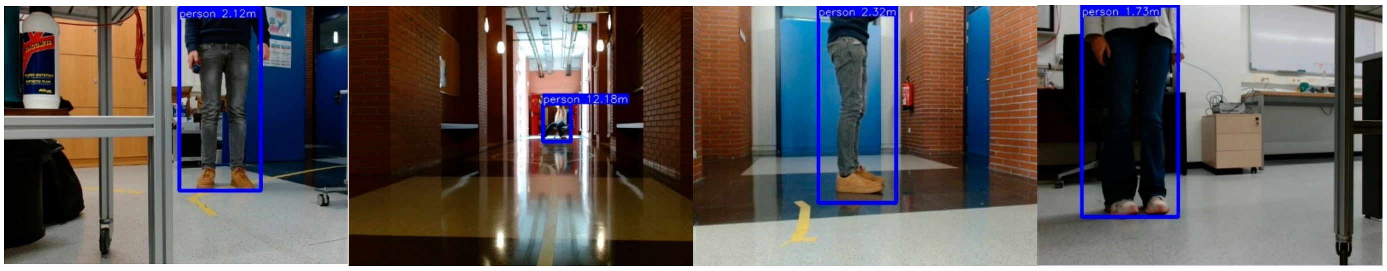

Figure 4. Images from custom object detection and distance estimation dataset.

3.3. Data Augmentation

As described in *Albumentations: Fast and Flexible Image Augmentations* [71], data augmentation is a commonly used technique to increase both the size and diversity of labeled training sets by using input transformations that preserve the corresponding output labels. Usually, modern computer vision-based deep learning models require large amounts of high-quality labelled images to perform successfully. The use of data augmentation allows for an increase in the volume, quality, and variety of training data, thus reducing manual-labelling times and improving the results of the developed models [72].

In this project, the original image set was increased to approximately 500 images ($\times$5 original images) using image-augmentation techniques such as horizontal flipping, random brightness, contrast modification, RGB shift, or the addition of random noise, as these are modifications that do not change the distance to the objects (see Figure 5).

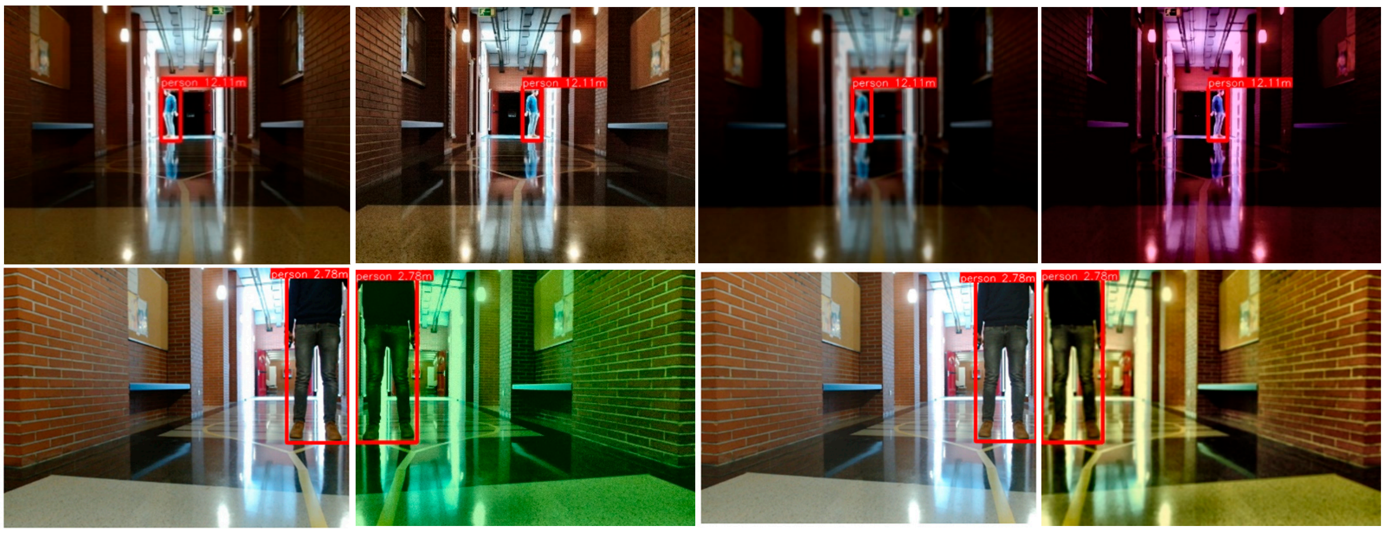

Figure 5. Examples of images with data augmentation (the left image is the original).

The idea was to first train the network on the KITTI dataset and then apply transfer learning to our images from the custom dataset. Transfer learning is the reuse of a pre-trained model on a new problem and allows for the knowledge gained from previous training to be used to improve the results of a future task without the need to use large amounts of labeled data. A list of step-by-step instructions for training the custom object detection and distance estimation model is shown below.

1. Download and preprocess the KITTI dataset. In this work, the KITTI 3D Object Detection (https://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d, accessed on 11 November 2023) dataset will be used in the first stage to train the algorithm.
2. Generate object detection and distance estimation custom dataset. To use the developed model in a custom environment, it is necessary to collect and label a dataset that describes the new environment.

3. Create or find an object-detection algorithm. There are in the literature several object-detection algorithms. However, you should look for or design one that allows you to modify the architecture easily.
4. Modify object detection model architecture to estimate distance to objects as well. Once the object-detection algorithm is working correctly, it will be necessary to modify the architecture so that it can also predict distances to detected objects.
5. Train the model with object detection and distance prediction dataset. The first training of the new model will be performed on a dataset with many labelled images, like KITTI or nuScenes. This will allow the network to optimise its weights for better training on customised images.
6. Transfer learning of the model weights with the custom dataset. After training the model with the large database, the model is re-trained with the images of the customised environment where the vehicle will move. In this way, the network can adapt correctly to the environment with a low amount of data.

Repeat steps five and six to find the best model performance. Finally, the models will be trained until the correct architecture and hyperparameters are found for which the network detects objects and estimates distances correctly.

4. Results

In this section, the results of the proposed simultaneous object detection and distance-estimation algorithm will be discussed. On the one hand, to evaluate the object detection performance of the models with the custom dataset, the precision (P, Equation (4)), recall (R, Equation (5)), and mean average precision (mAP, Equations (6) and (7)) were calculated and compared. Precision measures the model's ability to make accurate positive predictions, while recall measures the proportion of actual positive cases that the model correctly identifies. To calculate mAP, you first need to create a precision–recall curve for each class that the model is trying to detect. The precision–recall curve is generated by varying the confidence threshold for the detection model and calculating the precision and recall at each threshold. The total number of different confidence thresholds or detection scores is N. For each class, you calculate the area under the precision–recall curve. This area is called the Average Precision (AP, Equation (6)). To obtain the mAP (Equation (7)), the average of the AP values across all classes ($\mathrm{n_{classes}}$) is calculated, which summarizes the overall performance of the object detection model.

$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{4}$$

$$\mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{5}$$

$$\mathrm{AP} = \sum_{\mathrm{k}=1}^{\mathrm{N}} \mathrm{Precision}(\mathrm{k}) \cdot \Delta \mathrm{Recall}(\mathrm{k}) \tag{6}$$

$$\mathrm{mAP} = \frac{1}{\mathrm{n_{classes}}} \sum_{\mathrm{i}=1}^{\mathrm{n_{classes}}} \mathrm{AP_i} \tag{7}$$

where TP, FP, and FN refer to the true positives, false positives, and false negatives, respectively. TP are positive samples with correct classification, FP are negative samples with incorrect classification, and FN are positive samples with incorrect classification. On the other hand, to evaluate the distance estimation accuracy, the MAE (Mean Absolute Error, Equation (8)) and MAPE (Mean Absolute Percentage Error, Equation (9)) values are used, because it is more important to detect a close obstacle than a far obstacle, in order to avoid potential imminent collisions. The metrics were calculated with the help of the following equations.

$$\mathrm{MAE} = \frac{1}{N}\sum_{k=1}^{N}\left|y_{k,\mathrm{real}} - y_{k,\mathrm{pred}}\right| \tag{8}$$

$$\mathrm{MAPE} = \frac{1}{N}\cdot\sum_{k=1}^{N}\left|\frac{y_{k,\mathrm{real}} - y_{k,\mathrm{pred}}}{y_{k,\mathrm{real}}}\right| \tag{9}$$

The results of the different YOLOv5-based models can be seen in Table 3.

Table 3. Results.

Model		Object Detection				Distance Estimation		Speed
Type	**Params (M)**	**mAP 0.5**	**mAP 0.5:0.95**	**Precision**	**Recall**	**MAE (m)**	**MAPE (%)**	**Inf. Time (gpu | cpu) (ms)**
YOLOv5n	1.8	0.867	0.731	0.510	0.930	0.87	18.3	51 | 65
YOLOv5s	7.1	0.882	0.785	0.594	0.934	0.72	28.9	57 | 87
YOLOv5m	20.9	0.921	0.782	0.615	0.936	0.71	14	65 | 135
YOLOv5l	46.2	0.897	0.817	0.641	0.936	0.83	23.9	76 | 223

The results in Table 3 show that there is a progressive improvement in performance with the increase in the model parameters. On the one hand, the object detection results show good accuracy, with a mAP0.5:0.95 of more than 70% for all models. On the other hand, the distance estimation results show an average absolute error of about 0.7–0.8 m and a mean average precision error between 10 and 30%. The best distance prediction model YOLOv5m has a 0.71 m MAE and 14% MAPE. A MAE of 0.71 m and a MAPE of 14% means that, on average, the predictions made by a model are off by 0.71 m and 14% from the actual or real values.

For the visualization of the training results, Tensorboard was used, which is the TensorFlow visualization toolkit. The device used in this experiment was a laptop with an Intel Core i7-11800H CPU and a NVIDIA GeForce RTX 3070 GPU. The software environment was CUDA 11.6 and Python 3.7.

Figure 6 provides several images of the test set where the results of the trained models are compared. The results show that although the lighter models still fail in some estimations of distance, the heaviest model nearly predicts all the results perfectly. The labels in the images of Figure 6 have the (Detected class: Predicted distance by de net [m], Real distance [m]) form.

The images in Figure 6 show that the simultaneous object detection and distance estimation model works quite well for the four displayed models. On the one hand, all four models correctly perform the part of detecting people. On the other hand, distance estimation is a factor where a progressive improvement is seen with the increasing number of model parameters: the YOLOv5n model still estimates distances of 0m for some images, while the YOLOv5l, the heaviest one, practically hits all distances.

Additionally, in Table 4 our method is compared quantitatively with other object-detection algorithms available in the literature, including official object-detection models trained on big databases as well as different object-detection modifications made by researchers. The model comparison has been carried out for similar tasks in autonomous (mostly indoor) vehicles, which do not necessarily involve obstacle detection. The results show that the performance of the model depends on different factors such as the selected detection algorithm, the complexity of the task the vehicle will perform (such as the detection of objects with easily distinguishable patterns as well as the working environment), or the number of training images/training labels and their quality.

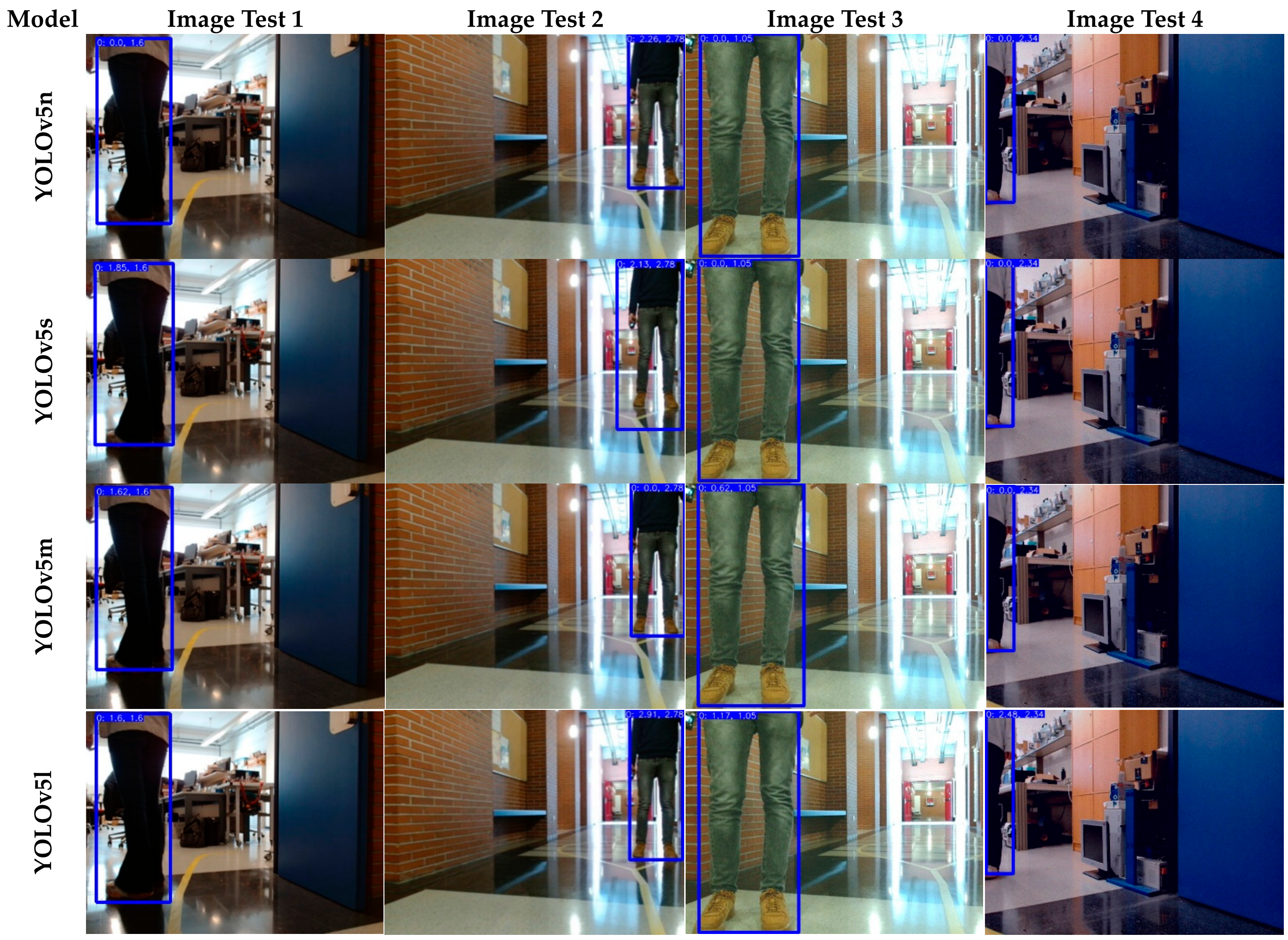

Figure 6. Image model comparison.

Table 4. Quantitative state of the art object detection model comparison.

Ref	Object Detection Model		Data		Work Environment
	Model	**mAP 0.5**	**Dataset**	**N Images**	
[73]	YOLOv5n	45.7	Mixed	-	Official YOLOv5 algorithm. General object detection.
	YOLOv5l	67.3			
[40]	Improved YOLOv5s	95.2	Custom	1800	Semi-structured apple orchard environment.
[74]	YOLOv3	49.4	BDD100K	+100,000	Autonomous vehicles in outdoor environment in clear (1) and rainy (2) conditions.
		52.6			
[75]	JET-Net	59.1	Mixed	+55,000	Football environment for autonomous robots.
[76]	Tiny-YOLO	67.6	Mixed	7700	General indoor environment for mobile robots.
[77]	Faster R-CNN	82.8	Custom	1625	Different conditions outdoor environment for mobile robots.
[78]	Improved YOLOv4	86.8	DJI ROCO	2065	Robomaster Competition environment for mobile robots.
Ours	YOLOv5n	86.7	Custom	104	Custom indoor environment for automated guided vehicles.
	YOLOv5l	89.7			

Finally, Table 5 compares the mean absolute error with other distance estimation models using a single monocular camera. It should be noted that the working environment of the other models (outdoor) is different from ours (indoor) and that the maximum detection distances are larger, so there is no direct comparison of the distance error. However, these values serve to determine the functionality and feasibility of our model.

Table 5. Quantitative state of the art monocular distance to object estimation model comparison.

Ref	MAE (m)	Distance Estimation Method	Task
[4]	2.0	Deep Neural Network	Distance estimation in railway environment.
[51]	2.57	YOLOv3 prediction vector modification	Distance to multiples classes (vehicles, pedestrians, trams, trucks, etc.) estimation for autonomous vehicles.
[10]	46.2	End-to-end learning-based model	Distance to multiples classes (vehicles, pedestrians, trams, trucks, etc.) estimation in autonomous vehicles.
[16]	1.83	R-CNN based structure	Distance estimation to cars, pedestrians, and cyclists for autonomous vehicles.
Ours	0.71	YOLOv5 prediction vector modification	Distance to obstacles prediction in indoor environment.

5. Discussion

This paper presents a set of algorithms for detecting objects and estimating distances to them simultaneously. The results demonstrate the feasibility of training a deep learning model using very few samples of data, saving the laborious time-consuming manual labeling of images. The use of data augmentation techniques that do not alter the distance to the objects allows for the generation of new synthetic images and obtaining satisfactory results for the task at hand. Furthermore, applying transfer learning from a trained model with a large dataset makes training in a customized environment much quicker and simpler.

The main limitations of the algorithm are that it has only been trained with a reduced set of images for a custom environment and just one type of obstacle. However, this can easily be improved by manually labeling more images, although this is time-consuming, and is the major challenge of deep learning algorithms. Moreover, not all models offer the same performance characteristics, so it is necessary to find the appropriate model for each application, considering, for example, aspects such as detection accuracy and inference time. Similarly, it is necessary to have hardware that can quickly run these algorithms. However, nowadays it is even possible to find microcontrollers that support deep learning libraries. Another negative aspect to consider is that currently no obstacle avoidance algorithm has been developed, although it can be used to avoid collisions in a simple way, such as the reduction of vehicle speed if the distance to the obstacle is more than 5 m and a full stop if it is less than 5 m.

To conclude this section, the contributions of this paper have been compared with other papers covering similar topics in Tables 1, 4 and 5. On the one hand, the literature review in Table 1 shows that there are several methods for obstacle detection, all of which have their advantages and disadvantages. For example, ultrasonic sensors are low cost and easy to implement but have a short range of action and cannot be customized for the detection of only some types of obstacles. Likewise, LiDaRs have good resolution and provide distance information but are more expensive than other sensors. Additionally, camera-based methods are low-cost compared to some specialized sensors and offer detailed information about the surroundings. However, most computer vision-based methods did not provide distance estimation, especially when using a single, uncalibrated camera.

On the other hand, taking this into account, our algorithm uses a single monocular uncalibrated camera, as the only source of information. The use of deep learning algorithms allows for the generation of a flexible model in terms of the obstacles to be avoided: it can be trained in a customized way for just the desired obstacles. In addition, the proposed model offers very high performance with a very small dataset, with more than 75% of mAP0.5:0.95.

For example, the official YOLOv5l algorithm has 49% mAP for general-purpose object detection. Furthermore, Table 4 shows that our algorithm provides accurate results when compared to other object-detection algorithms in similar mobile robot tasks. Some results of similar studies on environment recognition include Tiny-YOLO [76] with and improved YOLOv4 [78] with 67.6% and 86.8% of mAP, respectively. In comparison, our algorithm offers better accuracy, and the number of training images is more than 10 times lower. For distance estimation (see Table 5), our best model has an MAE of 0.71 m, slightly better than monocular camera distance estimation models in outdoor autonomous vehicles. Finally, our algorithm was easy to use and implement, and different versions of the algorithm are provided depending on the hardware available to deploy it.

6. Conclusions

The present work aimed to develop an algorithm that simultaneously performs object detection and distance estimation for an autonomous indoor vehicle. A modification of a YOLOv5-based Keras implementation is proposed for obstacle detection in a custom environment. The results show an incredible performance with a custom dataset of 100 images. The developed algorithm can detect obstacles in an image and predict its distance to the camera with high accuracy and low inference time.

In the results, different models for simultaneous object detection and distance estimation, with parameters from 1.8 M up to 46.2 M, are compared. A progressive improvement is shown as the volume of parameters to be trained increases, with the YOLOv5l model providing very accurate results for the task at hand. These results illustrate the usefulness of transfer learning and the possibility of detecting customized obstacles with a low-cost vision sensor that can be used for autonomous driving in both indoor and outdoor vehicles.

Future research directions can be divided into two main areas: on the one hand, to use this algorithm as a baseline for developing an obstacle avoidance algorithm using techniques such as reinforcement learning, and on the other hand, to deploy this type of algorithm on low-cost hardware, such as a microcontroller or a Coral Edge TPU.

Author Contributions: Conceptualization, I.A.; methodology, I.A. and M.G.; software, I.A.; validation, I.A., M.G., E.Z. and J.M.L.-G.; formal analysis, M.G., E.Z. and J.M.L.-G.; investigation, I.A. and M.G.; resources, I.A. and M.G.; data curation, I.A. and M.G.; writing—original draft preparation, I.A.; writing—review and editing, I.A., M.G., E.Z. and J.M.L.-G.; visualization, I.A., M.G., E.Z. and J.M.L.-G.; supervision, M.G., E.Z. and J.M.L.-G.; project administration, I.A., M.G. and E.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: The authors are grateful for the support provided by the UPV/EHU.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Taketomi, T.; Uchiyama, H.; Ikeda, S. Visual SLAM Algorithms: A Survey from 2010 to 2016. *IPSJ Trans. Comput. Vis. Appl.* **2017**, *9*, 16. [CrossRef]
2. Yasuda, Y.D.V.; Martins, L.E.G.; Cappabianco, F.A.M. Autonomous Visual Navigation for Mobile Robots: A Systematic Literature Review. *ACM Comput. Surv.* **2020**, *53*, 1–34. [CrossRef]
3. Mota, F.A.X.D.; Rocha, M.X.; Rodrigues, J.J.P.C.; Albuquerque, V.H.C.D.; Alexandria, A.R.D. Localization and Navigation for Autonomous Mobile Robots Using Petri Nets in Indoor Environments. *IEEE Access* **2018**, *6*, 31665–31676. [CrossRef]
4. Haseeb, M.A.; Guan, J.; Ristić-Durrant, D.; Gräser, A. DisNet: A Novel Method for Distance Estimation from Monocular Camera. In Proceedings of the 10th Planning, Perception and Navigation for Intelligent Vehicles (PPNIV18), IROS, Madrid, Spain, 1 October 2018.
5. Chang, N.-H.; Chien, Y.-H.; Chiang, H.-H.; Wang, W.-Y.; Hsu, C.-C. A Robot Obstacle Avoidance Method Using Merged CNN Framework. In Proceedings of the 2019 International Conference on Machine Learning and Cybernetics (ICMLC), Kobe, Japan, 7–10 July 2019; pp. 1–5.

6. Hanumante, V.; Roy, S.; Maity, S. Low Cost Obstacle Avoidance Robot. *Int. J. Soft Comput. Eng.* **2013**, *3*, 52–55.
7. Borenstein, J.; Koren, Y. Real-Time Obstacle Avoidance for Fast Mobile Robots. *IEEE Trans. Syst. Man Cybern.* **1989**, *19*, 1179–1187. [CrossRef]
8. Geiger, A.; Lenz, P.; Urtasun, R. Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; pp. 3354–3361.
9. Bernini, N.; Bertozzi, M.; Castangia, L.; Patander, M.; Sabbatelli, M. Real-Time Obstacle Detection Using Stereo Vision for Autonomous Ground Vehicles: A Survey. In Proceedings of the 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), Qingdao, China, 8–11 October 2014; pp. 873–878.
10. Zhu, J.; Fang, Y. Learning Object-Specific Distance from a Monocular Image. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019; pp. 3838–3847.
11. Huang, L.; Zhe, T.; Wu, J.; Wu, Q.; Pei, C.; Chen, D. Robust Inter-Vehicle Distance Estimation Method Based on Monocular Vision. *IEEE Access* **2019**, *7*, 46059–46070. [CrossRef]
12. Liang, H.; Ma, Z.; Zhang, Q. Self-Supervised Object Distance Estimation Using a Monocular Camera. *Sensors* **2022**, *22*, 2936. [CrossRef]
13. Leu, A.; Aiteanu, D.; Gräser, A. High Speed Stereo Vision Based Automotive Collision Warning System. In *Applied Computational Intelligence in Engineering and Information Technology*; Precup, R.-E., Kovács, S., Preitl, S., Petriu, E.M., Eds.; Topics in Intelligent Engineering and Informatics; Springer: Berlin, Heidelberg, 2012; Volume 1, pp. 187–199, ISBN 978-3-642-28304-8.
14. Natanael, G.; Zet, C.; Fosalau, C. Estimating the Distance to an Object Based on Image Processing. In Proceedings of the 2018 International Conference and Exposition on Electrical and Power Engineering (EPE), Iasi, Romania, 18–19 October 2018; pp. 211–216.
15. Davydov, Y.; Chen, W.-H.; Lin, Y.-C. Supervised Object-Specific Distance Estimation from Monocular Images for Autonomous Driving. *Sensors* **2022**, *22*, 8846. [CrossRef]
16. Zhang, Y.; Ding, L.; Li, Y.; Lin, W.; Zhao, M.; Yu, X.; Zhan, Y. A Regional Distance Regression Network for Monocular Object Distance Estimation. *J. Vis. Commun. Image Represent.* **2021**, *79*, 103224. [CrossRef]
17. Mochurad, L.; Hladun, Y.; Tkachenko, R. An Obstacle-Finding Approach for Autonomous Mobile Robots Using 2D LiDAR Data. *Big Data Cogn. Comput.* **2023**, *7*, 43. [CrossRef]
18. Horan, B.; Najdovski, Z.; Black, T.; Nahavandi, S.; Crothers, P. OzTug Mobile Robot for Manufacturing Transportation. In Proceedings of the 2011 IEEE International Conference on Systems, Man, and Cybernetics, Anchorage, AK, USA, 9–12 October 2011; pp. 3554–3560.
19. Yildiz, H.A.; Can, N.K.; Ozguney, O.C.; Yagiz, N. Sliding Mode Control of a Line Following Robot. *J. Braz. Soc. Mech. Sci. Eng.* **2020**, *42*, 561. [CrossRef]
20. Shitsukane, A.; Cheriuyot, W.; Otieno, C.; Mvurya, M. A Survey on Obstacles Avoidance Mobile Robot in Static Unknown Environment. *Int. J. Comput.* **2018**, *28*, 160–173.
21. Joshi, K.A.; Thakore, D.G. A Survey on Moving Object Detection and Tracking in Video Surveillance System. *Int. J. Soft Comput. Eng. (IJSCE)* **2012**, *2*, 2231–2307.
22. Lee, H.; Yoon, J.; Jeong, Y.; Yi, K. Moving Object Detection and Tracking Based on Interaction of Static Obstacle Map and Geometric Model-Free Approachfor Urban Autonomous Driving. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 3275–3284. [CrossRef]
23. Kinsky, P.; ZHou, Q. Obstacle Avoidance Robot. Available online: https://digital.wpi.edu/concern/student_works/mg74qn550?locale=en (accessed on 16 June 2023).
24. Al-Mallah, M.; Ali, M.; Al-Khawaldeh, M. Obstacles Avoidance for Mobile Robot Using Type-2 Fuzzy Logic Controller. *Robotics* **2022**, *11*, 130. [CrossRef]
25. Crnokic, B.; Peko, I.; Grubisic, M. Artificial Neural Networks-Based Simulation of Obstacle Detection with a Mobile Robot in a Virtual Environment. *Int. Robot. Autom. J.* **2023**, *9*, 62–67. [CrossRef]
26. Azeta, J.; Bolu, C.; Hinvi, D.; Abioye, A.A. Obstacle Detection Using Ultrasonic Sensor for a Mobile Robot. *IOP Conf. Ser. Mater. Sci. Eng.* **2019**, *707*, 012012. [CrossRef]
27. Derkach, M.; Matiuk, D.; Skarga-Bandurova, I. Obstacle Avoidance Algorithm for Small Autonomous Mobile Robot Equipped with Ultrasonic Sensors. In Proceedings of the 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Kyiv, Ukraine, 14–18 May 2020; pp. 236–241.
28. Dang, T.-V.; Bui, N.-T. Obstacle Avoidance Strategy for Mobile Robot Based on Monocular Camera. *Electronics* **2023**, *12*, 1932. [CrossRef]
29. Rezaei, N.; Darabi, S. Mobile Robot Monocular Vision-Based Obstacle Avoidance Algorithm Using a Deep Neural Network. *Evol. Intel.* **2023**, *16*, 1999–2014. [CrossRef]
30. Gao, M.; Tang, J.; Yang, Y.; He, Z.; Zeng, Y. An Obstacle Detection and Avoidance System for Mobile Robot with a Laser Radar. In Proceedings of the 2019 IEEE 16th International Conference on Networking, Sensing and Control (ICNSC), Banff, AB, Canada, 9–11 May 2019; pp. 63–68.
31. Guo, L.; Antoniou, M.; Baker, C.J. Cognitive Radar System for Obstacle Avoidance Using In-Motion Memory-Aided Mapping. In Proceedings of the 2020 IEEE Radar Conference (RadarConf20), Florence, Italy, 21–25 September 2020; pp. 1–6.
32. Gia Luan, P.; Thinh, N.T. Real-Time Hybrid Navigation System-Based Path Planning and Obstacle Avoidance for Mobile Robots. *Appl. Sci.* **2020**, *10*, 3355. [CrossRef]

33. Hutabarat, D.; Rivai, M.; Purwanto, D.; Hutomo, H. Lidar-Based Obstacle Avoidance for the Autonomous Mobile Robot. In Proceedings of the 2019 12th International Conference on Information & Communication Technology and System (ICTS), Surabaya, Indonesia, 18 July 2019; pp. 197–202.
34. Deng, L.; Yu, D. Deep Learning: Methods and Applications. *Found. Trends®Signal Process.* **2014**, *7*, 197–387. [CrossRef]
35. Jia, B.; Feng, W.; Zhu, M. Obstacle Detection in Single Images with Deep Neural Networks. *Signal Image Video Process.* **2016**, *10*, 1033–1040. [CrossRef]
36. Liu, C.; Zheng, B.; Wang, C.; Zhao, Y.; Fu, S.; Li, H. CNN-Based Vision Model for Obstacle Avoidance of Mobile Robot. *MATEC Web Conf.* **2017**, *139*, 00007. [CrossRef]
37. Christiansen, P.; Nielsen, L.; Steen, K.; Jørgensen, R.; Karstoft, H. DeepAnomaly: Combining Background Subtraction and Deep Learning for Detecting Obstacles and Anomalies in an Agricultural Field. *Sensors* **2016**, *16*, 1904. [CrossRef] [PubMed]
38. Lin, B.-S.; Lee, C.-C.; Chiang, P.-Y. Simple Smartphone-Based Guiding System for Visually Impaired People. *Sensors* **2017**, *17*, 1371. [CrossRef]
39. Jot Singh, K.; Singh Kapoor, D.; Thakur, K.; Sharma, A.; Gao, X.-Z. Computer-Vision Based Object Detection and Recognition for Service Robot in Indoor Environment. *Comput. Mater. Contin.* **2022**, *72*, 197–213. [CrossRef]
40. Su, F.; Zhao, Y.; Shi, Y.; Zhao, D.; Wang, G.; Yan, Y.; Zu, L.; Chang, S. Tree Trunk and Obstacle Detection in Apple Orchard Based on Improved YOLOv5s Model. *Agronomy* **2022**, *12*, 2427. [CrossRef]
41. Teso-Fz-Betoño, D.; Zulueta, E.; Sánchez-Chica, A.; Fernandez-Gamiz, U.; Saenz-Aguirre, A. Semantic Segmentation to Develop an Indoor Navigation System for an Autonomous Mobile Robot. *Mathematics* **2020**, *8*, 855. [CrossRef]
42. Macias-Garcia, E.; Galeana-Perez, D.; Bayro-Corrochano, E. CNN Based Perception System for Collision Avoidance in Mobile Robots Using Stereo Vision. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–7.
43. Luo, W.; Xiao, Z.; Ebel, H.; Eberhard, P. Stereo Vision-Based Autonomous Target Detection and Tracking on an Omnidirectional Mobile Robot. In Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics; SCITEPRESS—Science and Technology Publications, Prague, Czech Republic, 29–31 July 2019; pp. 268–275.
44. Skoczeń, M.; Ochman, M.; Spyra, K.; Nikodem, M.; Krata, D.; Panek, M.; Pawłowski, A. Obstacle Detection System for Agricultural Mobile Robot Application Using RGB-D Cameras. *Sensors* **2021**, *21*, 5292. [CrossRef]
45. Badrloo, S.; Varshosaz, M.; Pirasteh, S.; Li, J. Image-Based Obstacle Detection Methods for the Safe Navigation of Unmanned Vehicles: A Review. *Remote Sens.* **2022**, *14*, 3824. [CrossRef]
46. Godard, C.; Mac Aodha, O.; Firman, M.; Brostow, G. Digging Into Self-Supervised Monocular Depth Estimation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Soul, Republic of Korea, 27 October–2 November 2019.
47. Wofk, D.; Ma, F.; Yang, T.-J.; Karaman, S.; Sze, V. FastDepth: Fast Monocular Depth Estimation on Embedded Systems. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 6101–6108.
48. Xue, F.; Zhuo, G.; Huang, Z.; Fu, W.; Wu, Z.; Ang, M.H. Toward Hierarchical Self-Supervised Monocular Absolute Depth Estimation for Autonomous Driving Applications. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October 2020–24 January 2021; pp. 2330–2337.
49. Chen, Z.; Khemmar, R.; Decoux, B.; Atahouet, A.; Ertaud, J.-Y. Real Time Object Detection, Tracking, and Distance and Motion Estimation Based on Deep Learning: Application to Smart Mobility. In Proceedings of the 2019 Eighth International Conference on Emerging Security Technologies (EST), Colchester, UK, 22–24 July 2019; pp. 1–6.
50. Godard, C.; Mac Aodha, O.; Brostow, G.J. Unsupervised Monocular Depth Estimation with Left-Right Consistency. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017.
51. Vajgl, M.; Hurtik, P.; Nejezchleba, T. Dist-YOLO: Fast Object Detection with Distance Estimation. *Appl. Sci.* **2022**, *12*, 1354. [CrossRef]
52. Yanmida, D.Z.; Imam, A.S.; Alim, S.A. Obstacle Detection and Anti-Collision Robot Using Ultrasonic Sensor. *Elektrika* **2023**, *22*, 11–14. [CrossRef]
53. Anh, P.Q.; duc Chung, T.; Tuan, T.; Khan, M.k.a.A. Design and Development of an Obstacle Avoidance Mobile-Controlled Robot. In Proceedings of the 2019 IEEE Student Conference on Research and Development (SCOReD), Seri Iskandar, Malaysia, 15–17 October 2019; pp. 90–94.
54. Madhavan, T.R.; Adharsh, M. Obstacle Detection and Obstacle Avoidance Algorithm Based on 2-D RPLiDAR. In Proceedings of the 2019 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 23–25 January 2019; pp. 1–4.
55. Ravankar, A.; Ravankar, A.A.; Rawankar, A.; Hoshino, Y. Autonomous and Safe Navigation of Mobile Robots in Vineyard with Smooth Collision Avoidance. *Agriculture* **2021**, *11*, 954. [CrossRef]
56. Kaneko, N.; Yoshida, T.; Sumi, K. Fast Obstacle Detection for Monocular Autonomous Mobile Robots. *SICE J. Control. Meas. Syst. Integr.* **2017**, *10*, 370–377. [CrossRef]
57. Li, S.-A.; Chou, L.-H.; Chang, T.-H.; Yang, C.-H.; Chang, Y.-C. Obstacle Avoidance of Mobile Robot Based on HyperOmni Vision. *Sens. Mater.* **2019**, *31*, 1021. [CrossRef]
58. Mane, S.B.; Vhanale, S. Real Time Obstacle Detection for Mobile Robot Navigation Using Stereo Vision. In Proceedings of the 2016 International Conference on Computing, Analytics and Security Trends (CAST), Pune, India, 19–21 December 2016; pp. 637–642.

59. Widodo, N.S.; Pamungkas, A. Machine Vision-Based Obstacle Avoidance for Mobile Robot. *J. Ilm. Tek. Elektro Komput. Dan Inform.* **2020**, *5*, 77. [CrossRef]
60. Saidi, S.M.; Mellah, R.; Fekik, A.; Azar, A.T. Real-Time Fuzzy-PID for Mobile Robot Control and Vision-Based Obstacle Avoidance. *Int. J. Serv. Sci. Manag. Eng. Technol.* **2022**, *13*, 1–32. [CrossRef]
61. Ahmad, I.; Yang, Y.; Yue, Y.; Ye, C.; Hassan, M.; Cheng, X.; Wu, Y.; Zhang, Y. Deep Learning Based Detector YOLOv5 for Identifying Insect Pests. *Appl. Sci.* **2022**, *12*, 10167. [CrossRef]
62. Azurmendi, I.; Zulueta, E.; Lopez-Guede, J.M.; Azkarate, J.; González, M. Cooktop Sensing Based on a YOLO Object Detection Algorithm. *Sensors* **2023**, *23*, 2780. [CrossRef]
63. Jia, X.; Tong, Y.; Qiao, H.; Li, M.; Tong, J.; Liang, B. Fast and Accurate Object Detector for Autonomous Driving Based on Improved YOLOv5. *Sci. Rep.* **2023**, *13*, 9711. [CrossRef]
64. Mahaur, B.; Mishra, K.K. Small-Object Detection Based on YOLOv5 in Autonomous Driving Systems. *Pattern Recognit. Lett.* **2023**, *168*, 115–122. [CrossRef]
65. Guo, Y.; Kang, X.; Li, J.; Yang, Y. Automatic Fabric Defect Detection Method Using AC-YOLOv5. *Electronics* **2023**, *12*, 2950. [CrossRef]
66. Li, L.; Wang, Z.; Zhang, T. GBH-YOLOv5: Ghost Convolution with BottleneckCSP and Tiny Target Prediction Head Incorporating YOLOv5 for PV Panel Defect Detection. *Electronics* **2023**, *12*, 561. [CrossRef]
67. Yücel, Z.; Akal, F.; Oltulu, P. Mitotic Cell Detection in Histopathological Images of Neuroendocrine Tumors Using Improved YOLOv5 by Transformer Mechanism. *Signal Image Video Process.* **2023**, *17*, 4017–4114. [CrossRef]
68. Nguyen, H.-C.; Nguyen, T.-H.; Scherer, R.; Le, V.-H. Unified End-to-End YOLOv5-HR-TCM Framework for Automatic 2D/3D Human Pose Estimation for Real-Time Applications. *Sensors* **2022**, *22*, 5419. [CrossRef] [PubMed]
69. Fathy, C.; Saleh, S.N. Integrating Deep Learning-Based IoT and Fog Computing with Software-Defined Networking for Detecting Weapons in Video Surveillance Systems. *Sensors* **2022**, *22*, 5075. [CrossRef] [PubMed]
70. Liu, L.; Ouyang, W.; Wang, X.; Fieguth, P.; Chen, J.; Liu, X.; Pietikäinen, M. Deep Learning for Generic Object Detection: A Survey. *Int. J. Comput. Vis.* **2020**, *128*, 261–318. [CrossRef]
71. Buslaev, A.; Iglovikov, V.I.; Khvedchenya, E.; Parinov, A.; Druzhinin, M.; Kalinin, A.A. Albumentations: Fast and Flexible Image Augmentations. *Information* **2020**, *11*, 125. [CrossRef]
72. Mumuni, A.; Mumuni, F. Data Augmentation: A Comprehensive Survey of Modern Approaches. *Array* **2022**, *16*, 100258. [CrossRef]
73. Glenn Jocher YOLOv5. Available online: https://github.com/ultralytics/yolov5 (accessed on 8 November 2022).
74. Hnewa, M.; Radha, H. Object Detection Under Rainy Conditions for Autonomous Vehicles: A Review of State-of-the-Art and Emerging Techniques. *IEEE Signal Process. Mag.* **2021**, *38*, 53–67. [CrossRef]
75. Poppinga, B.; Laue, T. JET-Net: Real-Time Object Detection for Mobile Robots. In *Proceedings of the RoboCup 2019: Robot World Cup XXIII*; Chalup, S., Niemueller, T., Suthakorn, J., Williams, M.-A., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 227–240.
76. Jiang, L.; Nie, W.; Zhu, J.; Gao, X.; Lei, B. Lightweight Object Detection Network Model Suitable for Indoor Mobile Robots. *J. Mech. Sci. Technol.* **2022**, *36*, 907–920. [CrossRef]
77. Nilwong, S.; Hossain, D.; Kaneko, S.; Capi, G. Deep Learning-Based Landmark Detection for Mobile Robot Outdoor Localization. *Machines* **2019**, *7*, 25. [CrossRef]
78. Hu, Y.; Liu, G.; Chen, Z.; Guo, J. Object Detection Algorithm for Wheeled Mobile Robot Based on an Improved YOLOv4. *Appl. Sci.* **2022**, *12*, 4769. [CrossRef]