electronics

MDPI

Article

Navigating Unstructured Space: Deep Action Learning-Based Obstacle Avoidance System for Indoor Automated Guided Vehicles

Aryanti Aryanti [1,2], Ming-Shyan Wang [1,*] and Muslikhin Muslikhin [3,*]

[1] Department of Electrical Engineering, Southern Taiwan University of Science and Technology, Tainan 710, Taiwan; da72b201@stust.edu.tw or aryanti@polsri.ac.id
[2] Department of Electrical Engineering, Politeknik Negeri Sriwijaya, Palembang 30139, Indonesia
[3] Department of Electronics Engineering Education, Universitas Negeri Yogyakarta, Yogyakarta 55281, Indonesia
* Correspondence: mswang@stust.edu.tw (M.-S.W.); muslikhin@uny.ac.id (M.M.)

Abstract: Automated guided vehicles (AGVs) have become prevalent over the last decade. However, numerous challenges remain, including path planning, security, and the capacity to operate safely in unstructured environments. This study proposes an obstacle avoidance system that leverages deep action learning (DAL) to address these challenges and meet the requirements of Industry 4.0 for AGVs, such as speed, accuracy, and robustness. In the proposed approach, the DAL is integrated into an AGV platform to enhance its visual navigation, object recognition, localization, and decision-making capabilities. Then DAL itself was introduced to combine the work of You Only Look Once (YOLOv4), speeded-up robust features (SURF), and k-nearest neighbor (kNN) and AGV control in indoor visual navigation. The DAL system triggers SURF to differentiate two navigation images, and kNN is used to verify visual distance in real time to avoid obstacles on the floor while searching for the home position. The testing findings show that the suggested system is reliable and fits the needs of advanced AGV operations.

Keywords: automated guided vehicles; deep action learning; k-nearest neighbor; speeded-up robust features; visual navigation; YOLOv4

check for updates

Citation: Aryanti, A.; Wang, M.-S.; Muslikhin, M. Navigating Unstructured Space: Deep Action Learning-Based Obstacle Avoidance System for Indoor Automated Guided Vehicles. *Electronics* **2024**, *13*, 420. https://doi.org/10.3390/electronics13020420

Academic Editor: Andrea Bonci

Received: 3 November 2023
Revised: 10 January 2024
Accepted: 15 January 2024
Published: 19 January 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Achieving the large-scale use of automatic guided vehicles (AGVs) called for by Industry 4.0 is a daunting undertaking. Most AGVs nowadays are used in massive industries, such as Amazon, Alibaba, Lotte, Carrefour, Walmart, and Pinduoduo [1–6]. AGVs have shown great benefits in the logistics field and have led to a significant reduction in handling and transportation costs. However, to meet the anticipated needs of Society 5.0, further work is required to improve the cost, speed, safety, and versatility of AGV systems [5,6]. Industry 4.0 requirements include interoperability (compatibility), information transparency, technical assistance, and independent decisions. The system built for the AGV fulfills the element of the independent decision making, where the AGV will decide autonomously to navigate and stop through a DAL mechanism. The proposal of DAL can only be optimal with the support of other existing systems, such as YOLO for the detection system, SURF for route confirmation, and kNN used for AGV verification against references, such as start, home, and obstacles.

Continuous advancements in wireless connectivity, sensor miniaturization, cloud computing, big data, and analytics have led to a new concept called the Internet of Things (IoT), in which devices collect and exchange information with one another with little or no human invention [5,7,8]. The integration of IoT and Artificial Intelligence (AI) technology has led to the development of the Artificial Intelligence of Things (AIoT), which is essentially

a more intelligent and capable ecosystem of connected devices and systems. In the context of AGVs, AIoT primarily aims to emulate the execution of human tasks within logistics and storage systems through the utilization of Internet networks and intelligent decision making [8–11]. Although Industry 4.0 already leverages such AGV technology to a certain extent, its use is generally limited to larger enterprises and structured environments [2].

One of the many components of Industry 4.0 is warehousing, which integrates technology and automation to optimize various tasks in the storage and distribution of goods. Within warehouse environments, AGVs are mainly used for order picking and material transport and are typically guided by line tracking, barcodes, laser sensors, and camera techniques [1]. Such methods work well in structured environments with clearly defined paths, controlled obstacles, predictable workflows, and minimal human interactions [12,13]. However, in environments with a dynamic layout, obstacle variability, and the need for complex decision making, existing AGV systems face significant challenges. Unfortunately, these problems are yet to be adequately resolved in the context of warehousing. Broadly, other studies link Industry 4.0 with several machine models and simulations through dynamic, intelligent, flexible, and open applications. Including photogrammetry models is integral to Industry 4.0 regarding flexibility, primarily assisted by GNSS and GCP navigation. It has also been developed to forge a wheeled AGV, using artificial neural networks to deal with high accuracy [14,15].

In the literature, AIoT approaches have been widely used for applications such as roadside recognition for AGV forklift equipment, unmanned vehicle obstacle detection, autonomous lane detection, crop and weed detection, and collision avoidance [2,16]. The authors in [17] used a recurrent neural network (RNN) to guide musculoskeletal arms and robotic arms to achieve the precise and efficient completion of goal-directed tasks. Where research [17] was developed from Krizhevsky et al. [18] who earlier developed a convolutional neural network (CNN) called AlexNet that is efficient and has dropout regularization to reduce overfitting [19]. Zhang et al. [5,11,12,20,21] proposed that features be generated from two consistent domains using Generative Adversarial Networks (GANs) But the training procedure is slow, so updates appear through unsupervised cross-domain object detection; this detection is known as CA-FRCNN (Cycle-Consistent Domain Adaptive Faster RCNN). Another method, You Only Look Once (YOLO), is a real-time object identification method that has been developed by Fang et al. Several variants of the original YOLO model have been proposed [22]. In general, the results have shown that YOLO has many advantages over other lightweight models, including real-time processing capabilities, multi-object detection, and customization for specific applications, making it a versatile and efficient choice for many computer vision tasks.

YOLO has found many applications across various industries, including robotics, agriculture, medicine, health, education, and the military. As the detection speed of YOLO has improved over the past few years, the scope of its applications has expanded [23], and there is now growing interest in applying YOLO to AGVs. However, YOLO has a high power consumption and relies on fast GPU cards and complex computational processes. Furthermore, while YOLO facilitates autonomous driving, navigation, and obstacle avoidance, particularly in unstructured environments, there remain many challenging concerns to be overcome [18–20,22–24].

The reliable navigation of modern AGVs generally depends on the successful recognition and detection of routing markers. Stereo camera systems, such as D435i, Kinect, or RealSense, provide an effective solution for the detection of fixed objects. However, they suffer from several severe limitations in practical situations, such as vulnerability to environmental conditions, the need to maintain accurate calibration and camera alignment over time, and a high computational complexity. Consequently, the feasibility of using mono camera systems for AGV navigation has gained increasing traction in recent years. In such approaches, a relative baseline is calculated as the AGV moves based on the pixel shift at a fixed point on an object, and the speeded-up robust features (SURF) law is then applied to the pixel shift for navigation purposes. Compared with stereoscopic vision systems,

mono camera systems have a lower power consumption, thus facilitating a longer AGV running time. Moreover, in automatic navigation scenarios, the ability to detect random obstacles, narrow positions, and changes in the object size, orientation, and type is typically more important than performing depth estimation [7,25,26].

The studies in [3,27,28] examined the roles of AI, robotics, and data mining in AGV navigation, and concluded that effective algorithms for navigating indoor spaces rely heavily on the extraction of appropriate local features for performing keyframe selection, localization, and relative posture calculation. Many features and feature processing methods have been proposed, including segments of invariant column [4], SIFT (Scale Invariant Feature Transform) [29,30], and FREAK (Fast Retina Keypoint) [29]. It was shown in [31] that the feature processing speed can be accelerated through a bag-of-words (BoW) technique, in which a histogram of visible words is used to represent the quantified image. The Term Frequency–Inverse Document Frequency (TF-IDF), as statistics-based methods, can also be applied to each histogram bin to quantify the relevance of a particular visual term to any image within the image set [27,32]. However, although local features are theoretically less sensitive to lighting variations and motion blur, indoor environments still pose a significant challenge owing to their extreme visual diversity, the presence of repeated patterns, and the potential for occlusion. Random Sample Consensus (RANSAC) [2,28,33] provides a means of overcoming these problems through more accurate and robust feature matching. However, in complex, unstructured environments, the resulting substantial mismatch ratio increases the computation time required by RANSAC to estimate the relative poses with precision.

In previous studies [34–36], the present group developed a R-CNN (region-based CNN) with a structure of eye-in-hand that utilized a single camera to perform an estimate of depth and object location. In a later study, this approach was extended to the task of object picking. An action learning (AL) method was also proposed to help the manipulator robot learn from its mistakes [32]. Although the system can learn from actions, the working area coverage is very narrow, and the object detection area is fixed. So, the AL method is less suitable for unstructured areas, distances, and dynamic object positions. In the present study, a robust and efficient navigation method is proposed for AGVs by merging AL with deep action learning (DAL) and utilizing SURF and the k-nearest neighbors (kNN) method as feedback guides for the navigation process.

This study's primary contributions can be summed up as follows:

- A DAL architecture is employed to perform robust and accurate detection of objects in an indoor environment.
- Object localization is performed using a single monochrome camera fixed to the AGV. An automated navigation capability is realized through the amalgamation of YOLOv4, SURF, and kNN in a seamless DAL architecture.
- The AGV's self-navigation performance is enhanced by representing obstacles as points or nodes in the AGV mapping system, thereby improving its ability to plan routes around them.
- The experimental outcomes demonstrate that the suggested system performs robustly and meets the requirements of advanced AGV operations.

The remainder of this paper is organized as follows. The general system design is presented in Section 2, including both the AGV robotic platform and the navigation system. The suggested visual navigation system, the localization, and detection techniques are described in depth in Section 3. Section 4 discusses the visual navigation, obstacle avoidance, safety, and moving obstacle detection issues for the AGV platform in an indoor warehouse environment. The results of the experiment are presented and analyzed in Section 5. Section 6 concludes by offering a brief conclusion and suggesting future research directions.

2. System Design

The proposed system comprises two main components: the physical AGV robotic platform and the navigation system used to control its motion. The navigation system is designed to allow the AGV to perform naturally inside an environment containing various objects, such as walls, aisles, shelves, and objects on the floor. For evaluation purposes, in this research, the AGV navigates by passing various obstacles or markers from the starting position to the home position. As the AGV moves, it performs continuous object detection and recognition using a visual navigation system implemented using a DAL architecture based on YOLOv4 for segmentation purposes and SURF to perform collision avoidance.

The fundamental elements of the suggested DAL architecture are shown in Figure 1. The brown boxes refer to the simulated localization environment, which contains various objects, including sports cones, gallon water containers, and cardboard boxes. A dataset consisting of images showing these objects and the associated environment was compiled to support the YOLO segmentation process. While the AGV is moving, the paired RGB images captured by the mono camera are used as input data to the SURF algorithm (shown in purple) to perform obstacle avoidance according to the object detection results obtained from the kNN-assisted YOLOv4 model. Finally, a set of commands was produced to instruct the AGV to move forward, backward, left, right, or stop, as required, to safely reach the designated home position (depicted in white).

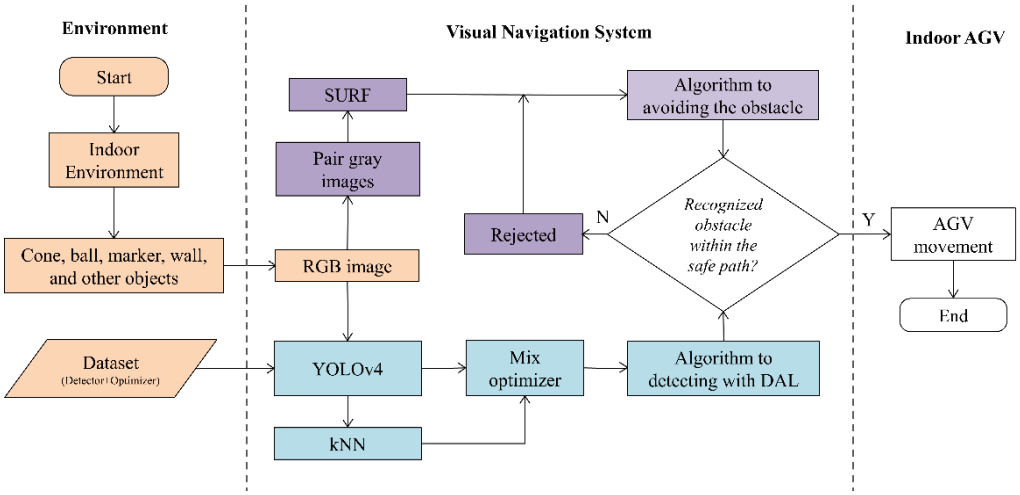


Figure 1. Overall structure of DAL architecture for indoor AGV navigation.

In general, DAL is divided into three main parts. So, it needs to be declared that the DAL concept adopts AL and reinforcement learning (RL) so that the first part is the environment. This environment is set in the form of cones, water containers, cardboard, and other items. On the other hand, a previously collected dataset is inserted into a passive environment. In the middle is the visual navigation system. The RGB image in the form of an indoor environment becomes input from the system for detection and recognition by YOLOv4 with consideration of three optimizers: SDGM, RMSProp, and ADAM. Then, the results of YOLOv4 recognition are also used by kNN to find connection points between the previous image and the current image. The results of this search for the shortest distance are used to consider the AGV's movements in finding a safe route. The RGB input from the AGV is taken continuously and compared; therefore, as soon as the AGV moves, the first image will be taken, and then it will move again for the following image. In this case, there is a shift in the baseline, both on the x-axis and/or z-axis, so it seems that the AGV is using a stereo camera with a pair of identical images.

SURF processes identical images to find link points between the first and second images. There is no limit to the number of matching points. Naturally, many matching points are more valid in this case, but it is also necessary to limit it to make performance efficient; ten nodes are enough. The navigation algorithm uses the results of these nodes to avoid obstacles, approach the target, turn, and stop. Another algorithm related to DAL tests the accuracy of target detection. The two algorithms are combined to find the target and determine navigation by the AGV in an indoor context.

3. AGV with Visual Navigation System

3.1. Visual Navigation

A visual navigation system must be able to find its way and position itself in an environment by understanding visual information. A possible approach is performing simultaneous mapping, visual odometry, semantic mapping, object detection, and tracking. Visual detection–odometry is used in this paper by examining changes in visual information in successive frames. By investigating and comparing the visual features of the obstacles specified in Table 1, the AGV will estimate the displacement and update its perception of the position of the obstacles in the environment.

Table 1. Characteristics of obstacles in present environment.

Type of Obstacle	Size (mm)	Weight (gr)	Package	Role
Carton box	310 × 215 × 235	255	cubical	stop
Sporting cone	475 × 175	59	cone	turn left/right
Water container	510 × 265	108	cylindrical	backward

The navigation system proposed in this study utilizes a combination of object recognition using YOLOv4 assisted by kNN and tracking via semi-visual odometry based on SURF. The entire segmentation, tracking, and obstacle avoidance algorithms are seamlessly implemented within a DAL architecture. Figure 2 illustrates the AGV with an independent four-wheel drive system and a camera positioned at a 15 mm center-point to floor distance. As shown, the AGV is positioned within an environment containing three obstacles, and its aim is to navigate through the environment without hitting any obstacles, using a visual odometry method based on the segmentation results obtained from sequential frames.

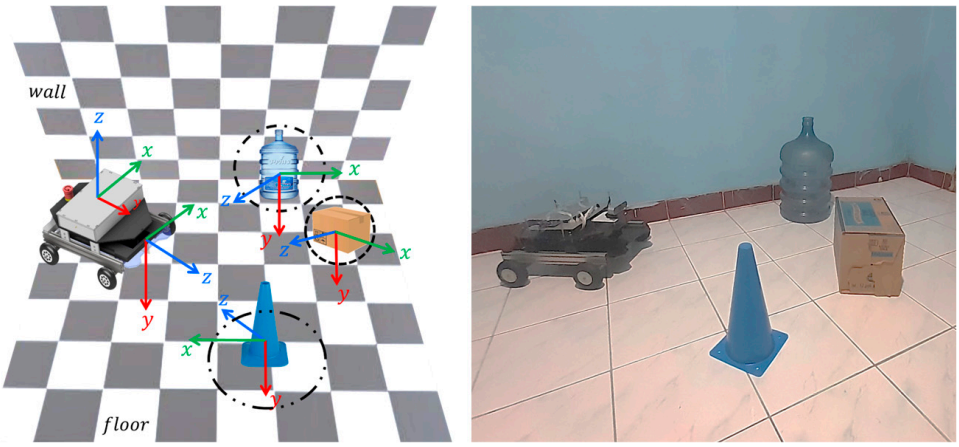


Figure 2. Cartesian coordinate frames of camera with obstacles (**left**) and real world (**right**).

In Figure 2, the Cartesian coordinates of the AGV are always perpendicular to the obstacle, while the z-axis of the camera is opposite the z-axis of the obstacle. The y-axis moves to the right, and the y-axis is opposite the z-axis of the AGV. In other words, the z-axis of the camera will try to approach the z-axis of the obstacle/target. In the real world,

the z-axis will get shorter when the AGV approaches the obstacle/target and vice versa. Environmental change is a problem, but so far, it has been resolved with variations in the dataset that has been trained.

3.2. Indoor AGV with DAL Data-Driven Algorithm

According to Chen et al. [36], indoor AGVs rely on appropriate and sophisticated data-driven approaches to operate precisely. In the present study, the AGV is guided by a DAL data-driven algorithm that makes use of data interpretation and analysis to make educated decisions, address issues, and formulate appropriate movement strategies. The DAL algorithm facilitates the continuous improvement of the AGV's navigation performance through a closed-loop mechanism based on feedback, new data, and ongoing analysis, which allows the navigation system to adapt and optimize its strategies and processes over time.

In the present study, the navigation process is based on the images acquired from a camera mounted at the front of the AGV, together with velocity information acquired from onboard inertial sensors. Monocular cameras are often used for indoor navigation purposes owing to their high performance and small size. However, they have no means of directly estimating the distance between objects and themselves, and hence they are unreliable in determining the AGV's speed. Although several methods have been proposed to overcome this issue, they are restricted to particular circumstances [1]. Fortunately, the most recent developments in tiny RGB-D cameras, such as the Intel RealSense RGB product series [12], now offer a promising solution. These cameras not only enable the tracking of pre-defined objects but also address the scaling issue, thereby allowing the AGV's velocity to be directly determined.

In the camera plane, objects are detected by calculating the distance d between the camera optical center and an object point $X = (X, Y, Z)^T$. A pinhole camera principle applies to real-world 3D coordinates of X obtained as

$$X = \frac{d}{\|K^{-1}x\|} K^{-1}x \tag{1}$$

where the K is a matrix of calibration comprising the camera intrinsic parameters and x represents a pixel coordinate for a uniform image $(x, y, 1)^T$ [19]. The X-point rigid conversion from the coordinates centered on the camera scheme to random world coordinate frame X_w is obtained by executing the translation T and camera rotation world frame R as follows:

$$\begin{bmatrix} X \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ 1 \end{bmatrix} \tag{2}$$

Investigating the transition between two identical images, the actual metric movement of the camera can be obtained by setting the starting point of the axis of the coordinate system to the camera focal point ($T_1 = (x, y, 1)^T$). Then, identify the axis coordinates set by the camera orientation ($R_1 = I$; I is a matrix of identities), and define $T_2 = -R_2C_2$, where C_2 is the world coordinate of the camera optical center so that at that time, it can be known where both images were taken. The results can then be converted into corresponding pose changes and speed data for navigation.

Previous research [37] showed that the relative distance between the positions of separate items detected in the first and second images remains consistent if modeled based on a rigid transformation in Euclidean space. According to this model, when the object points seen in the first image are specified as $X_{1i},\ i,\ k = 1, \ldots, n$, while those observed from the second image are defined as $X_{2i},\ i,\ k = 1, \ldots, n$, the constant relative distance restraint is given as:

$$\|X_{1i} - X_{1k}\| = \|X_{2i} - X_{2k}\| \tag{3}$$

In Equation (3) a subset of $\|X_{1i} - X_{1k}\|$ is constructed using the matched point to compute the object points; the most likely just contains inliers and Equation (1) and

comparing combinations of object point pairs using Equation (2). These points are then used to determine the translation parameter T.

3.3. Deep Action Learning for Obstacle Detection

The mono camera placed at the front of the AGV was used to examine the obstructions in the inside environment in 2D coordinates. To accomplish the navigation process, the camera frame (C) obstacle coordinates must be transformed into the AGV frame (E) coordinates. Figure 3 shows the geometric relationships between the two frames, where O denotes an obstruction and A denotes an AGV frame.

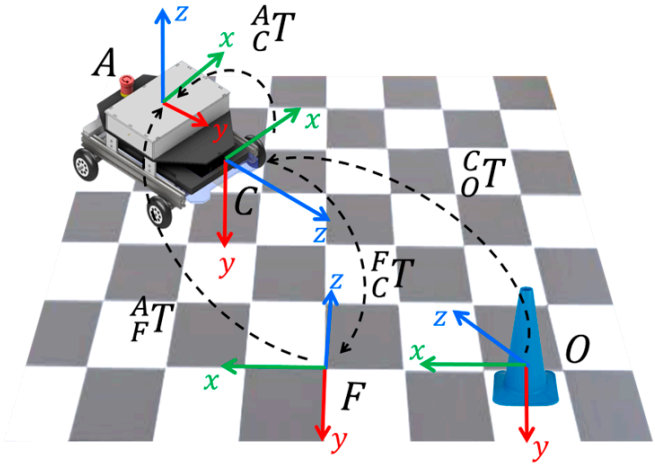


Figure 3. The frame transformation process commences from obstacle in floor coordinate frame F to camera coordinate frame C and finally AGV base in coordinate frame A.

The controller of the robot manages the transitions between different frames, optimizes perception downsizing, and facilitates the AGV to perform course corrections. Let ${}^{A}O$ be the location of obstacle O with respect to the A-frame and ${}^{C}O$ be the location of the obstacles in the C-frame. The coordinate transformation of the object from the C-frame to the A-frame can then be given as

$$ {}^{O}_{A}T = {}^{O}_{C}T {}^{C}_{F}T {}^{F}_{A}T \tag{4} $$

In each navigation trial, the AGV was positioned initially such that the camera was aligned with the sports cone (see Figure 2). The AGV was programmed to move from its starting position to home position $movh \leftarrow gpc$, see Algorithms 1 and 2, where the starting position is denoted as O_{cone}, and home position is denoted as O_{cbox}.

Algorithm 1 Search Obstacles in Indoor Environment

predefined : a scenario and the obstacles on the floor; a DAL-trained system (AI)
input : images pair
for *AGV sends image pairs P_(i = 1:n) for each movement.* **do**
  recognizing a dual obstacle O in a given environment $\forall D(x,y)$, $Z_{c[i]}D(x,y)_{[i]}$;
  if *the evaluation is matching* $D(x,y)_{[i]} \approx SURF$ **then**
    world monocular camera depth being saved as world robot depth $Z_{r[i]} \leftarrow Z_{c[i]}$; i++;
    running position point of SURF $MME(x,y) \longleftarrow d^2_{st}$; $MME \geq 9$
  else
    return to **input**;
  end
end

Algorithm 2 Verification of Obstacle for AGV Step Selection

predefined : obstacle coordinates O_i;
the depth of obstacle Z_i.
input : **Algorithm 1**.
for *every barrier is identified* $O_{i=1}$ **to** $O_{i=n}$ **do**
  if *numerous detections sorting* $O_{(n)} = \sum_{i=0}^{n-1} n - i$ **then**
    obtain the max of level confidence $O_{(n)[max]}$;
    call **function a.**
  else if *the levels of confidence overlap to the next obstacle* $O_{(n)[max]} \cup O_{i=n}$ **then**
    use the SURF disparity map to evaluate $O_{i=n} \rightarrow det\left(\mathcal{H}_{\text{approx}}\right)$ && $D(x, y)$;
    call **function a.**
  else *confidence levels combine with obstacle* $O\binom{n}{k}$ **then**
    obtain the maximum level of confidence $O_{(n)[max]}$ assess using the SURF disparity map $O_{(n)[\max SURF\ D]} \rightarrow det\left(\mathcal{H}_{\text{approx}}\right)$ && $D(x, y)$;
    reassess with the following inner bounding box $B_{(box\ [a_{[1+i,1]},\ldots,a_{[1+i,4]}])} O_{(n)} \rightarrow O_{(n)[\max SURF\ D]}$;
    call **function a.**
  end
  return to **predefined**.
end
function a toward the AGV to the homing obstacle $O_{(n)[max]} \leftarrow \exists\, p_{[x,y,z]}$;
homing position $movh \leftarrow gpc$.

3.3.1. Localization of Recognized Obstacles

Many targets and obstacles must be identified while the AGV is moving. These obstacles were then detected as the AGV traveled over the floor using the YOLOv4 detection model with the assistance of a kNN algorithm. The obstacle centroid locations and depths frame *O* in camera frame *C* were decisive, based on the bounding boxes detected by YOLO. As shown in Figure 4, the centroids were confirmed using SURF, with the assistance of a disparity map constructed after the double-check bounding box depths were obtained. In the subsequent stage, the depths were validated using the bounding box and centroids using the kNN algorithm. Finally, the recovered obstacle depth was translated from the target *O*-frame to the mono camera *C*-frame using the intrinsic camera settings.

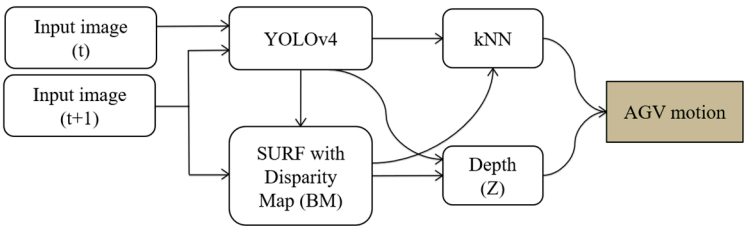


Figure 4. Block diagram for coordinated transformation.

Figure 4 illustrates the triangulation scheme used by the AGV controller to perform forward/reverse motion capture and focal point capture. The area of the workspace was fixed along a parallel line *x* with *O* as its optical center utilizing an attached monochrome camera. The camera constructed the baseline *b* along line *x* from *O* to O_{t+0} and it only had one camera because of the *f* focal length. Given a reference point $P(X_p, Y_p, Z_p)$, the projections at *X* and *Y* are $p_1(x_{t+0}, y_{t+0})$ and $p_1(x_{t+1}, y_{t+1})$, respectively, in image plane. Figure 5 shows the image coordinates *P* in a single plane with magnification, as shown in the n projection. This triangulation scheme to simplify the perspective projection calculation method. In a single picture plane, its image corresponds to a *P* coordinate.

As shown in Figure 5, the object was determined using just one camera. Following the camera shift, the *Y*-axis was perpendicular to the page, and both images were placed on the movable camera as a *b* line. The equation of (4) can be obtained using the principle of identical triangles, as shown in Equation (5). From basic geometric principles, line b

in Figure 5 can be determined using Equation (6), while Z (the depth of point P) can be calculated using Equation (7).

$$\frac{X}{x_1} = \frac{Z}{f}; \frac{X}{y_1} = \frac{Z}{f}; \frac{b-X}{x_2} = \frac{Z}{f} \tag{5}$$

$$b = \frac{Z}{f}x_1 + \frac{Z}{f}x_2 \tag{6}$$

$$Z = \frac{b \times f}{x_1 + x_2} \tag{7}$$

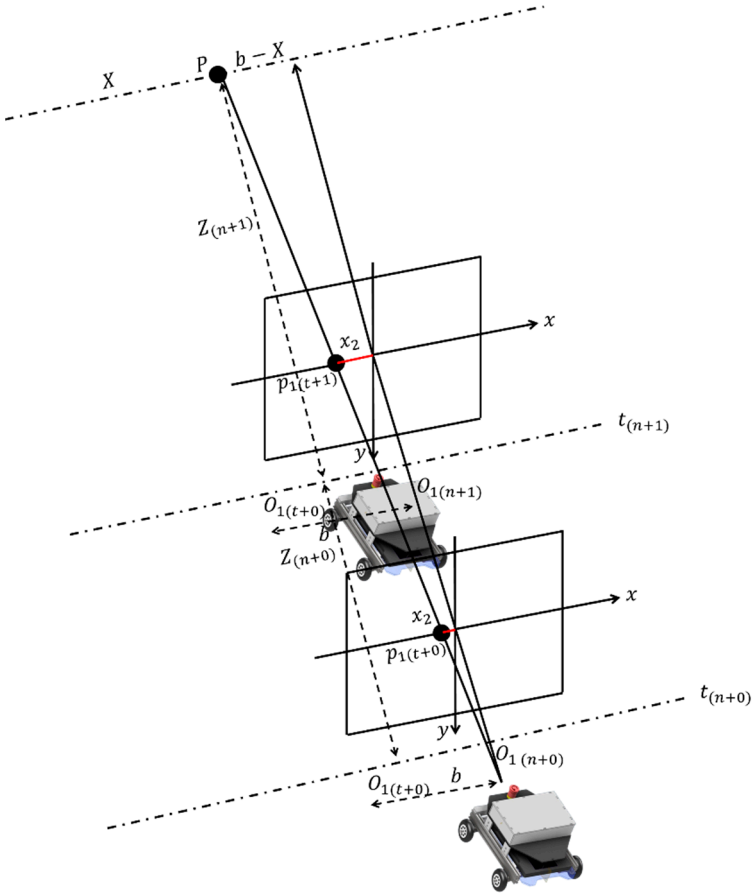


Figure 5. Triangulation scheme of stereo $O_{1(n+0)}$ comprises camera plane 1 ($t + 0$) in initial AGV position, followed by AGV motion from $O_{1(n+0)}$ to $t_{(n+1)}$, where picture taken at $O_{2(n+1)}$ is camera frame of plane 1 ($t + 1$).

Comparing images 1 and 2, the disparity d between them is obtained simply as the difference in their x coordinates, in Equation (8). After Z has been gained, X, Y, and also point coordinates P can be derived based on Equation (9).

$$d = x_1 - x_2 \tag{8}$$

$$X = \frac{Z \times x_1}{f}; Y = \frac{Z \times y_1}{f}; Z = \frac{b \times f}{d} \tag{9}$$

where x_1 and x_2 are the 2D image pixel coordinates, while X, Y, and Z are the 3D image real positions.

It was supposed that the barriers were immovable and permanent. Therefore, the centroid position of each detection result must be established. The segmented bounding

boxes are used as the basis for the simplest approach for determining centroids, as indicated in Equation (10).

$$B_{box} = \begin{bmatrix} o_{11} & o_{12} & o_{13} & o_{14} \\ o_{21} & o_{22} & o_{23} & o_{24} \\ o_{31} & o_{32} & o_{33} & o_{34} \\ \vdots & \vdots & \vdots & \vdots \\ o_{n1} & o_{n2} & o_{n3} & o_{n4} \end{bmatrix} \tag{10}$$

The bounding box B_{box} matrix comprises four columns $o_{[1\ldots,4]}$, corresponding to X, and the number of rows is determined by the number of barriers found $o_{[n,4]}$ on the camera angle. The centroid coordinates X_{cen} and Y_{cen} can then be obtained from Equations (11) and (12).

$$\begin{cases} X_c = B_{box}\,(:, o_{11}); \;\; Y_c = B_{box}\,(:, o_{12}) \\ o = B_{box}\,(:, o_{13}); \;\; b = B_{box}\,(:, o_{14}) \end{cases} \tag{11}$$

$$\begin{cases} X_{cen} = X_c + \frac{o}{2} \\ Y_{cen} = Y_c + \frac{b}{2} \end{cases} \tag{12}$$

The centroid calculated using Equation (12) is taken as the AGV reference point to plan its next movement. It should be noted that the centroid coordinates are given in the 2D plane, and the Z-value is no longer required because the AGV just signals movement and will not grab.

3.3.2. SURF Approach

In the traditional SURF method [29,38] the convolution of the 2D image is approximated using a box filter, with the second order of the Gaussian derivative used for simplifying and improving computational effectiveness [34]. In the present study, the SURF detector makes use of the Hessian matrix due to its speedier and more accurate processing. Presented a point $x = (x, y)$ in image I, the Hessian matrix $H(x, \sigma)$ in x at scale σ, can be described as

$$\mathcal{H}(x, \sigma) = \begin{bmatrix} L_{xx}(x, \sigma) & L_{xy}(x, \sigma) \\ L_{xy}(x, \sigma) & L_{yy}(x, \sigma) \end{bmatrix} \tag{13}$$

where $L_{xx}(x, \sigma)$ is the Gaussian second-order convolution derivative $\frac{\partial^2}{\partial x^2} g(\sigma)$ with image I at point x, and similarly for $L_{xy}(x, \sigma)$ and $L_{yy}(x, \sigma)$,

$$\det\left(\mathcal{H}_{approx}\right) = D_{xx}, D_{yy} - \left(0.9\, D_{xy}\right)^2 \tag{14}$$

where D_{xx}, D_{yy}, D_{xy} are approximations. The image (x, y) must first be detected and converted into a gray image and then filtered using four types of filters, as illustrated in Figure 6. To enhance the precision, the number of appropriate points was initially set to 20, with the expectation that the level of precision must be increased with a lower standard deviation. However, the SURF method, in the authors' experience, was found to be deficient in accuracy. Consequently, its performance was enhanced using a disparity map, as described in the following.

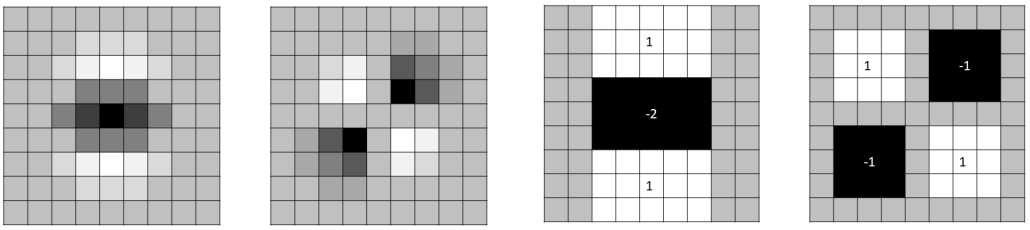


Figure 6. Gaussian second-order clipped and discretized (9 × 9 boxes) partial derivatives in (gray) in the y-direction are both similar to zero (black), while one represents in white.

In general, a disparity map, $D(x, y)$, deputizes the transfer of the appropriating pixels among the right and left pictures in a stereo vision system. However, in practice, comparable pixels in different images are difficult to discern. For instance, some elements, such as non-textured, homogenous, repeating textures and camera noise, may cause issues even with non-occluded pixels [38]. Block matching (BM) was used to calculate the disparity in the current study for each and every pixel. The following is how the validity of the discrepancy significance was assessed:

$$D_{L\to R}(x,y) = \underset{d\epsilon[0,D_{max}]}{\operatorname{argmin}} \varepsilon^{d}_{L\to R}(x,y) \tag{15}$$

$$\varepsilon^{d}_{R\to L}(x,y) = \frac{\sum_{(u,v)}\sum_{\in W}|\, f_r(x-u,\ y-v) - f_l(x-u+d, y-v)|}{\sum_{(u,v)}\sum_{\in W}|\, f_r(x-u,\ y-v) + f_l(x-u+d, y-v)|} \tag{16}$$

The disparities among the right and left pictures can be derived from Equations (12)–(16) and Equations (13)–(17), where $\varepsilon^{d}_{R\to L}(x,y)$ is the match error in a block that has been normalized and has a block matching windows W and a horizontal discrepancy d. In addition, D_{max} is the highest value disparity within the permissible limit. Note that the equations are equally valid for obtaining the disparity between the right and left image frames in a conventional stereo vision system or between the first and second images $f_{l,r,b,f}$ during AGV movement in the present mono camera setup.

$$D_{R\to L}(x,y) = \underset{d\epsilon[-D_{max},0]}{\operatorname{argmin}} \varepsilon^{d}_{R\to L}(x,y) \tag{17}$$

$$MME(x,y) = \varepsilon^{d}_{L\to R}(x,y)\Big|_{d=D_{R\to L}(x,y)} \tag{18}$$

where u and v are the number of pixels in the xy-camera image plane.

According to Equation (18), the minimum matching error (MME) represents how closely (or not) the pair image values in the left image (x, y) and the right image with disparity ($x + d$, y) are related to the same spots.

3.3.3. Verified by kNN

The images processed by the DAL architecture were obtained from a video sequence captured as the AGV moved through the environment. The detection results obtained from these images have a fundamental effect on the performance of the AGV in traversing, turning, stopping, and reversing. In preliminary experiments, it was found that the obstacles could be successfully detected at distances of up to 1–3 m. If so, the system requires additional techniques for determining when to halt, turn, or go backward. Referring to Figure 2, the AGV first traversed in the forward direction and then, at a certain distance from the cone, stopped and turned to avoid it. In the proposed navigation system, the direction of the turn (i.e., right or left) was determined with the assistance of a kNN classifier.

The kNN classifier was trained using a distance measure. In general, the better the metric indicates the label similarity, the better the categorization outcome becomes. In general, a kNN classifier operates by measuring the gaps between a query and all of the data instances. It then selects a certain number (k) of the examples closest to the query and either votes for the most popular label (in classification tasks) or averages the labels (in regression tasks). Using query points to categorize points in a training dataset based on their proximity is a fundamental technique, but an efficient way of classifying fresh points. In this study, the distance was computed using a variety of measures, as detailed below. The distance between a set of data and query points is computed using a kNN.

Given an m_x-by-n data matrix X, which can be expressed with m_x (1-by-n) row vectors x_1, x_2,. . ., x_{mx}, and an m_x-by-n data matrix Y, which can be expressed with m_x (1-by-n) row

vectors $y_1, y_2, \ldots, y_{my}$, the various distances between the vectors x_s and y_t can be evaluated as follows:

$$d_{(s,t)} = \sqrt{(x_s - y_s)^2 + (x_t - y_t)^2} \quad (19)$$

The location points or the SURF points can both be used to get the x_s and y_t values in Equation (19). Then, by comparing these three inputs, it is possible to decide when to turn.

3.3.4. DAL with YOLOv4

This section describes the obstacle detection process performed using the YOLOv4 architecture. The overall process involved collecting an obstacle dataset, labeling it, training the YOLOv4 model with DAL, and testing its efficiency. During the training process, data augmentation was performed to enhance the network performance. The trained model was able to detect up to three obstacles in each image and was built on CSPDarkNet-53, as shown in Figure 7.

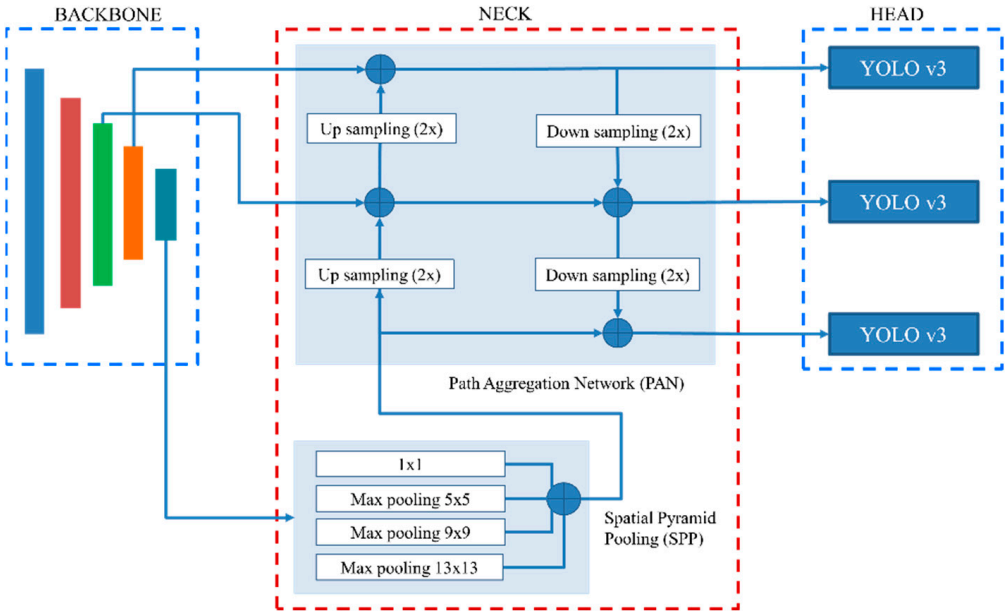


Figure 7. YOLOv4 within SqueezeNet feature extraction network with neck, head, and backbone.

A dataset consisting of 1460 images with one or two obstacles was manually compiled and carefully labeled. In general, small datasets are sufficient for analyzing the YOLOv4 training process; however, larger datasets are needed to train functional detectors and accordingly, to develop the YOLOv4 object detector in this research. Firstly, create a network based on the size of the input object for training. Second, perform data augmentation to improve training accuracy. Third, train the object detector to estimate using three combinations of Intersection over Union (IoU), anchor box offsets, and class probability. Finally, perform a detector performance test through evaluation on a large number of images.

Each DAL cycle comprised four stages: planning, observation, action, and reflection. A parameter referred to as the beta value β_p was introduced to the plan stage. In practical implementations, the number of cycles must be carefully determined to reduce the computational cost and optimize the AGV response time. Thus, the DAL operation was governed by two factors: the beta value and the cycle count, shown in Figure 8.

The validity of the proposed DAL-based AGV indoor visual navigation system was evaluated by performing an approach task with the target. Although the navigation procedure used DAL, the reinforcement learning approach served as its inspiration. The Bellman equation provides a discounted value from a goal point in reinforcement learning. Different routes are discovered in order to find the one that optimizes this value. In reinforcement learning, this approach is used to determine the value associated with each

state in advance. However, in the DAL approach, these values are updated continuously via real-time evaluation and are used as passing grades during the learning process. The passing grade value is denoted by β, and the assessment indicators are listed in Table 2.

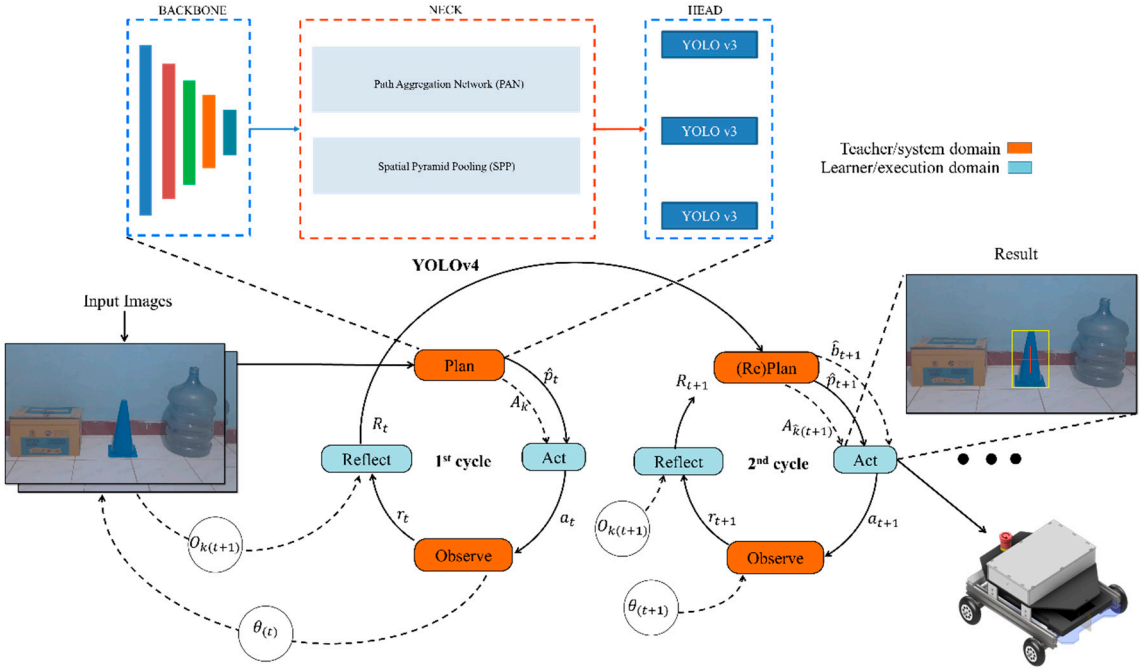


Figure 8. DAL flow with each cycle inside YOLOv4 into the plan stage.

For YOLOv4 to function optimally, the head (YOLOv3) detection output must meet certain criteria for δ and ε, which are indicators of the localization of the target and belong to the real number set δ, $\varepsilon \in R$. The seven evaluation indicators in Table 2 were used to obtain the following beta values:

$$\beta = \sum_{i=1}^{7} n_i \omega_i + n_{i+1}\omega_{i+1} + \cdots + n_{i+6}\omega_{i+6} \tag{20}$$

The first DAL cycle planning phase can be expressed using Equation (21), where the planning stage is denoted by p_t, the action stage by a_t, the observation stage by o_t, and the reflection stage by r_t.

$$p_t = \beta \wedge \delta \wedge \varepsilon \Rightarrow a_t \tag{21}$$

If p_t fulfills the circumstances specified by $\beta, \delta, \varepsilon$, the cycle continues to stage a_t, with the following settings.

$$a_t \neq 0 \Rightarrow o_t \tag{22}$$

Equation (22) can be rewritten as (23), where the reflection value outcome be determined by o_t with binary,

$$\begin{cases} o_t = \beta \wedge \delta \Rightarrow r_t;\ \beta,\ \delta,\ \varepsilon\ \neq 0 \\ r_t = o_t,\ o_t \in \{0,\ 1\} \end{cases} \tag{23}$$

If $r_t = 1$, the cycle terminates; in another way, if $r_t = 0$, it scrolls to evaluate the next cycle $\beta_{(t+1)}$ and returns its value. See in Equation (23), when preview o_t gathers inputs from the value $\beta \wedge \delta$, YOLOv4 is employed for a second time to verify whether the obstacle has been recognized. Depending on the outcome, it proceeds to the second or the next cycle.

To impose the action learning process described above on the AGV robot, the perception of the robot of the obstacles on the ground must be valid [6]. Table 2 details the seven assessment indicator items used to determine robot perceptions in each cycle. For each AGV movement, a single assessment result was obtained in the range of 1–100. The resulting β_p values were compared to the outcomes produced using the probability density method.

The normal distribution is a collection of two-parameter curves, commonly referred to as the Gaussian distribution. As the sample size approaches infinity, the central limit theorem states, any distribution with a finite mean and variance with a limited number of independent samples converges to a normal distribution. The basic properties of this normal distribution are frequently employed in the development of deep learning. As shown in Algorithm 1, the DAL process works to find obstacles. The AGV takes the first image and then moves to take the second image as input. Both images are recognized to look for obstacles and then given for environmental assessment. The environment is re-evaluated with a disparity map and SURF to find the distance between the obstacle and the AGV. This process will continue and repeat until $MME \geq 9$.

Table 2. Assessment indicators in present experiments.

Aspect of the Assessment (Indicators)	Scale/Probability				ω
	0	**1**	**2**	**3**	
Prior to Acting, Plan (β_1)					
Calculate the outcome of the YOLOv4 (detection) on one process δ (%)	-	≤80	81~95	≥96	5
Evaluate the present barrier in an indoors of β_p value	-	≤75	76~90	≥91	3
SURF-kNN is used to calculate the distance. ε (%)	-	≤10	11~60	≥61	7
Prior to Observe and Consider (β_2)					
Verify the prior act of the obstacle achievement	No	-	-	Yes	15
Make sure the obstacle is identified		Yes	-	No	9
Compare the current of β with the previous of β_0 (%)	-	≥21	6~20	≤5	3
The cumulative distribution function (CDF) value for the subsequent cycle	-	≤75	76~90	≥91	5

4. Visual Navigation Algorithm

The navigation system for an AGV robot must be aware of its surroundings to avoid collisions and fulfill the assigned movement task. To minimize any damage, the AGV system must also evaluate the rule design and accurately detect the obstruction [31].

4.1. Obstacles on the Floor

In an actual warehouse, pallets and shelves are often arranged statically, unless the number of saved items is dynamic. However, the present experiments considered various possible arrangements of the obstacles. Three things are depicted on the ground with each rule in Figure 9, as stated in Table 1 above. Even though specific rules are determined for each object, maintaining the reliability of the system is critical. Thus, the evaluation experiments considered not only the ability of the AGV to detect and recognize the obstacles but also the decision-making process that the AGV then adopts in navigating around the obstacle. For example, when the AGV encounters a sports cone after detection, at an unknown distance, it should turn; conversely, when it detects the cardboard box at close range, it should stop.

Figure 9 depicts snapshots of the complete sequence of detection results as the AGV moved from the initial to the home position. In the detection process, all the obstacles were lying on the floor, and the AGV did not consider the problem of measuring the distance between itself and the obstacles because the objective was simply to avoid hitting them during the navigation process. Therefore, throughout the SURF-kNN turning process, the AGV continued to calculate identical points to complete the matching process.

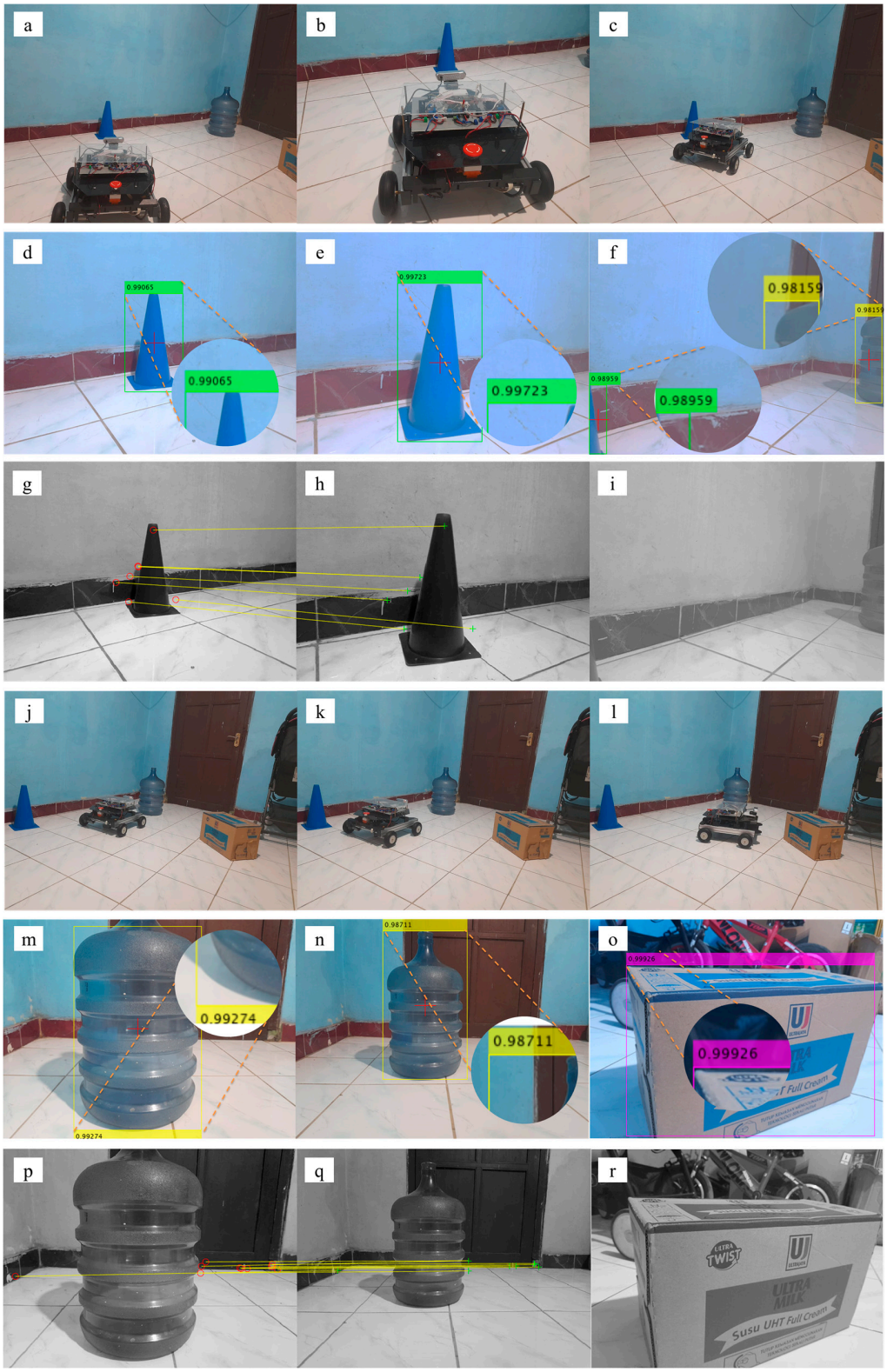


Figure 9. The visual navigation path taken by an AGV from its initial point to its home position. AGV begins in its initial location (**a**), advances toward the sports cone in step (**b**), then rotates to face the water container in steps (**c**–**f**,**j**), goes backward because it is too near in step (**k**), and finally recovers its original position in step (**l**). Outcomes of sports cone, water container, and carton box detection when traveling forward (**j**–**o**), SURF-kNN results are (**g**–**i**); when moving backward, there are (**o**–**r**) no SURF-kNN results, and there are (**p**,**q**) SURF-kNN results.

4.2. Avoiding Obstacles

4.2.1. Forward Detection

The AGV detects obstacles in the environment in real time and moves forward, backward, turns, or stops, as appropriate to fulfill its navigation mission. This ongoing process involves comparing the current obstacle detection data with previous instances, with parameter differences between them serving as the input for the AGV's decision-making process.

During motion, the AGV can execute three basic types of motions: go forward (or backward), turn (left or right), and stop. In the first case, after detecting the sports cone obstacle, the AGV interprets the object as a forward command. The AGV was assumed to start at position $T_{(n+0)}$. As it moved forward during period $T_{(n+1)}$, it took images of the cone. The detection results were compared to a 1:7 ratio, and the AGV continued to move in the forward direction until the robot's basic dimensions are turned and the final detection ratio is reached. On the other hand, the AGV approached the sports cone for six steps after spotting it, then decided whether to turn either way after the seventh step.

In Figure 9d,h, it is discernible that point *P*, as a sports cone, experiences a size increase proportional to the AGV motion as it approaches the target. The x_2 point on $t_{(n+0)}$ becomes wider, and in terms of optics, the object in the image (x, y) appears larger than in terms of optics; the object in the image (x, y) appears larger than in the foregoing step. When the $t_{(n+7)}$ point of inflection is achieved, the AGV formulates an appropriate turn decision using the method described in the subsection that follows.

4.2.2. Decision to Turn

The AGV is built with a separate drive on each wheel, and consequently, the turning maneuver involves a pivot motion. For such a pivot motion, detection utilizing a 1:7 ratio is no longer viable, and hence an alternative approach is required. In general, pivot motion forms the radius of the rotating radius. As a result, it is possible to estimate the camera field of vision (FoV) angle. This FoV curve is an extension of the picture plane x-axis. As a result, Figure 9a should be updated to reflect Figure 9b.

Determining the turn direction as left or right is absolutely random, when the pivot mechanism quickly recognizes the following obstacle arrival. Thus, the potential for time inefficiencies can be avoided. The DAL system records prior detections that have been made and continues to search for the upcoming detection outcomes using YOLOv4. In the turning process, there could be multiple results from the detection.

Figure 9c shows a typical right-turn maneuver scenario. Initially, the AGV detected the cone with a detection accuracy of 0.997 and initiated a 90° (degree) turn. During the subsequent 20–35-degree turn segment, the AGV detected the sports cone with 0.989 accuracy and the gallon container with 0.981 accuracy. As the AGV continued its rightward pivot, it maintained a vigilant strategy, and when it identified the gallon container, it continued to pivot until it was oriented at 90 degrees to the target. In the event that the AGV detects that it is too close to the gallon container, as indicated by the bounding box area exceeding 50% of the total number of image pixels, it initiates a reverse maneuver followed by a corrective turn.

4.2.3. Home Position

The AGV movement terminates when it reaches the home position (a cardboard box). The AGV assumes it has arrived at this position when the size of the bounding box for the cardboard box exceeds 50% of the image pixels. The image displays the AGV position after backing up, turning right, and then spotting the carton box. The bounding box detection results exhibited an impressive accuracy of 0.998. The implementation of a turn, reverse, or stop are the three possible responses presented in Algorithm 2.

The sequence of Algorithm 2 starts by detecting obstacles by looking for the highest accuracy value and changing the optimizer. If the detection results are known, the Hessian matrix calculates the same image and the distance between the two images. The distance between the two images actually represents the movement of the AGV. Then, combine

the confidence level with SURF in assessing the bounding box, and then control the AGV towards the home position.

5. Experimental Results

5.1. Experimental Settings

The present study considered the problem of building an AGV navigation system for SMEs. Thus, the aim was to reduce the overall cost of the devices used, while maintaining an acceptable system performance. The proposed system runs on a CPU with i7 1165G7 @2.80 GHz and memory of 16 GB RAM. In order to overcome the graphical requirement, NVIDIA®GeForce RTX™ 3060 GPU 6 GB GDDR6 was installed, along with all technical specifications onboard in a customized AGV with four-wheel drive. The evaluation of the proposed navigation system was limited to the case of detecting three objects and performing maneuvering as appropriate to reach the home position without obstacle contact. The feasibility of the system is evidenced by Figures 9 and 10.

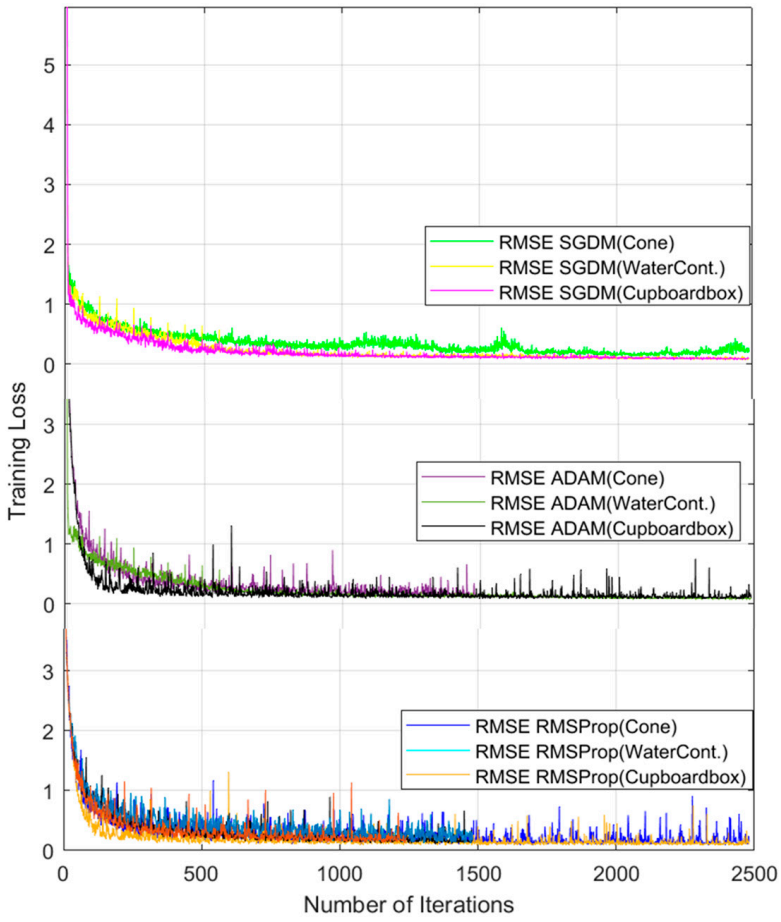


Figure 10. The YOLOv4 detector iteration RMSEs over training loss in each class for the SGDM (**top**), ADAM (**middle**), and RMSProp optimizer (**bottom**).

5.2. Detection Evaluation

It is necessary to evaluate the metrics considered in the present study: accuracy, precision, recall, F1 score, and average precision (AP); see Equation (24). This evaluation is obtained from calculations of the true-positive (TP), false-positive (FP), true-negative (TN), and false-negative (FN) statistics collected for around 560 images. The precision, memory, F1 score, and AP were calculated using a confidence value of 0.85. In the experiments, TP was taken as a detection outcome with an IoU value > 0.75 and a single bounding box, and FP was taken otherwise. The TN criterion was specified as the absence of a

double bounding box and an IoU value of <0.75. Finally, the FN output was specified as an incorrect detection output and an IoU value of <0.75 or the absence of a bounding box.

$$\begin{cases} accuracy = \frac{TP+TN}{TP+TN+FP+FN} \\ precision = \frac{TP}{TP+FP} \\ recall = \frac{TP}{TP+FN} \\ F_1 = \frac{2\times precision\times recall}{precission+recall} \\ AP = \int_0^1 p(r)\, dr \end{cases} \quad (24)$$

As shown in Table 3, the maximum confidence, accuracy, and precision values were obtained for the carton box, while the optimal recall and F1 performance were obtained for the water container, while the ideal AP and detection speed were obtained for the sports cone, with an average time of μ = 0.004 s with σ = 0.004. The training time of YOLOv4 was evaluated for three different optimizer schemes: Root Mean Square Propagation (RMSProp), Adaptive Moment Optimization (ADAM), and Stochastic Gradient Descent with Momentum (SGDM) as shown in Figure 10.

Table 3. Comparison of detection performance of YOLOv4 and DAL with that of other methods.

Obstacles	Method	Parameters						
		Confid.	**Accur.**	**Precis.**	**Recall**	**F1**	**AP**	**Time (s)**
Sports cone	YOLOv4	0.907	0.933	0.940	0.950	0.948	0.944	0.937
	DAL (YOLOv4)	0.938	0.963	0.943	0.950	0.967	0.933	0.949
Water container	YOLOv4	0.908	0.922	0.921	0.934	0.948	0.877	0.918
	DAL (YOLOv4)	0.916	0.875	0.950	0.950	0.950	0.928	0.939
Carton box	YOLOv4	0.932	0.940	0.950	0.938	0.943	0.921	0.937
	DAL (YOLOv4)	0.923	0.964	0.950	0.908	0.969	0.977	0.945
μ YOLOv4		0.916	0.932	0.937	0.941	0.946	0.914	0.931
μ DAL (YOLOv4)		0.926	0.934	0.948	0.936	0.962	0.946	0.944
σ YOLOv4		0.012	0.007	0.012	0.007	0.002	0.028	0.012
σ DAL (YOLOv4)		0.009	0.042	0.003	0.020	0.009	0.022	0.013

In Figure 10, ADAM was determined to be the best optimizer for the proposed navigation system and was thus used in all the remaining experiments. After successful detection and localization, the beginning position was established, as shown in Figure 9 in Section 4.

Practical training results of YOLOv4 based on those shown in Figure 10 require 2500 iterations, so the final results are almost the same. SDGM, for example, has small learning loss results, but in specific iterations (1000–1500), it increases while the noise decreases. Meanwhile, the RMSProp optimizer has lower performance than ADAM. The ADAM optimizer has the maximum learning loss compared to the other two. However, the noise in the ADAM optimizer still needs to be cleaner than that SGDM optimizer. This condition is sufficient for detection in indoor environments with various variations of obstacles/targets: orientation, zooming, image angle, and lighting.

Before discussing the detection results as in Table 3, it is better to explain the YOLOv4 concept in carrying out detection. The first technical preprocessing for the dataset is where the data is labeled and then stored in the bounding box in matrix form. After that, it is necessary to carry out training with several settings: of course, by setting the training options (ADAM, RMSProp, and SGDM), setting the initial learning rate to 0.00001, and by setting the squared gradient decay factor (SGDF) to 0.99, the maximum epochs to 20, and the mini-batch size to 64, respectively. If the learning rate is too low, training may take longer. However, training may provide a suboptimal or distorted output if the learning rate is too high.

In evaluating detection results, they are greatly influenced by the input image; viewing angle, brightness, initial mini-batch size, learning rate, and maximum epochs will all significantly impact detection accuracy and training duration. When the long training process has been completed, a training loss graph will appear as in Figure 10, and the detector will be obtained with each optimizer. Before the optimizer appears, training loss will be visible; essentially, the smaller the value, the better the results. This learning loss depends on the initial learning rate determined. Hasil training YOLOv4 tercipta tiga detektor untuk masing kelas dataset sport cone, water container, dan cupboard. The YOLOv4 training results produce three detectors for each class of the sport cone, water container, and cupboard dataset. All images detected by DAL and YOLOv4 are captured with shift left, right, forward, and backward conditions. It would have yet to think about whether the image is captured by the AGV by pivoting and moving diagonally closer or further away.

After studying several empirical results of applying DAL to conventional wheeled AGVs, it was realized that the movement of industrial AGVs allows them to move not only forward and backward but also to pivot and move diagonally back and forth. This movement brings image consequences that are difficult to detect even by YOLOv4. Jika hal itu diterapkan pada AGV beroda omni atau mecanum maka kecepatan belok diperkirakan semikin sulit meski menggukanan DAL yang sama. If this is applied to an AGV with omni or mecanum wheels, turning speed is estimated to be even more challenging, even if the same DAL is used. This challenge goes to prove that DAL still has weaknesses, especially in connecting images with the AGV slippage.

5.3. DAL Navigation Experiments

Table 4 evaluates the successful task completion performance of the AGV given the use of only YOLOv4 (upper rows) and YOLOv4 and DAL (lower rows). Two groups of attempts were completed for each implementation, with 58 trials each time. In evaluating the performance, "success" was defined as a homing time of $\leq$0.40 m, while failure was determined otherwise. Overall, the results presented in Table 4 show that the inclusion of DAL in the navigation scheme increases the mean success probability from 0.902 to 0.952.

Table 4. Performance effects of DAL vs. YOLOv4.

Methods	Attempt Number			Rate of		
	Success	Failures	∑ Cycle	Prob.	Ave.	Time (m)
YOLOv4	54 49	4 9	n/a n/a	0.931 0.874	0.902	0.40
∑	103	13	-	-	-	-
DAL	57 50	1 8	63 72	0.982 0.923	0.952	0.36
∑	107	9	135	-	-	-

The experimental trials provide useful insights into the effectiveness of the YOLOv4 detector. Employing a data-driven methodology is essential in this approach. The selection of the detector type is determined using raw data derived from the input image and YOLOv4 detection. Furthermore, the system demonstrates a rapid response time of just 0.36 min. This enables DAL to swiftly analyze the next frame to improve localization accuracy. Making incorrect navigation choices can lead to route complications and require additional time for correction. Next, the fourth column for YOLOv4 does not use cycles, so it conveys not available (n/a).

This DAL navigation experiment has not made comparisons with other methods. In this paper, the confirmation carried out is between DAL-YOLOv4 and YOLOv4 only. This comparison was carried out to determine the time consumption and average execution by the system. As in Table 4, the system was tested for two large trials on YOLOv4, the

first of 54 and the second of 49 attempts, while DAL had 57 and 50 attempts, respectively. Both produce rating rates of 0.902 and 0.952 for YOLOv4 and DAL. Kemudian pada kolom keempat untuk YOLOv4 while tentu tidak menggunakan siklus, sehingga tertulis n/a (not avilable).

Based on Table 4, the uniqueness of DAL can be seen from the existence of cycles. This cycle continues testing until the AGV navigation goal is achieved. The passing score β determines AGV achievement of targets; in this DAL cycle, β is set at 0.87 by splitting into Algorithms 1 and 2, and so detects 0.04 faster than YOLOv4. Looking more deeply, DAL architecturally uses YOLOv4 in it, but empirically, it is almost as fast because the DAL principle with the four stages of planning, acting, observing, and reflecting can carry out work from various parts. For example, the data will be stored in the reflecting section if the first detection has been carried out. If one day, YOLOv4 fails or poorly detects, the reflecting data can be used. Meanwhile, YOLOv4 works in three stages: backbone, neck, and head (YOLOv3), as shown in Figure 8.

In the first experiment, the DAL took 63 cycles for 57 successful attempts, meaning there were six extra cycles for each navigation process. In the second DAL experiment, there were 72 cycles for 50 experiments or a surplus of 22 cycles. This surplus could come from one trial containing more than one cycle. The ideal conditions between experiments and cycles are the same; in other words, DAL does not repeat the process. However, it seems impossible that the ratio between attempts and cycles is 1:1.

6. Conclusions

This study develops a visual navigation system for indoor AGVs, and the algorithm results can demonstrate its robustness. The built navigation system, with the DAL architecture consisting of a combination of YOLOv4, SURF, kNN, and AL, was able to complete visual navigation tasks in an average of 0.36 min, 0.04 min faster than the system that only uses YOLOv4. The speed of DAL is generally contributed to by the robustness and localization accuracy in reading each obstacle by SURF and kNN. DAL has a particular cycle and will end depending on the beta setting value. Of 107 navigation attempts, nine failed and required 135 cycles; however, the time was still 0.04 faster than YOLOv4. The experimental findings reveal that navigation with DAL allows AGVs to complete indoor driving tasks with an average success rate of 0.952. However, this research is still limited to conventional wheeled robot applications, even though industrial AGVs are dominated by mecanum or omni wheels. Future research plans will consider AGVs that use omni or mecanum wheels to perform more complex turning speed tactics with the same DAL. This challenge is to prove that DAL can work optimally to minimize slippage.

Author Contributions: Conceptualized the research, M.-S.W.; methodology and software used, M.M. and A.A.; investigation and validation, M.-S.W.; formal analysis, M.M.; resources, A.A.; data curation, A.A. and M.M.; writing—preparing the original draft, M.M.; writing—reviewing and editing the manuscript, A.A., M.M. and M.-S.W.; supervised the analysis, reviewed the manuscript, M.-S.W.; visualization and project administration, A.A. and M.M.; funding acquisition, M.-S.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Higher Education Sprout Project of the Ministry of Education, Taiwan, and Ministry of Science and Technology, grant number NSTC 112-2221-E-218-014.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available in this article.

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

References

1. Digani, V.; Sabattini, L.; Secchi, C. A Probabilistic Eulerian Traffic Model for the Coordination of Multiple AGVs in Automatic Warehouses. *IEEE Robot. Autom. Lett.* **2016**, *1*, 26–32. [CrossRef]
2. Liu, Y.; Ma, X.; Shu, L.; Hancke, G.P.; Abu-Mahfouz, A.M. From Industry 4.0 to Agriculture 4.0: Current Status, Enabling Technologies, and Research Challenges. *IEEE Trans. Ind. Inform.* **2021**, *17*, 4322–4334. [CrossRef]

3. Santos, J.; Rebelo, P.M.; Rocha, L.F.; Costa, P.; Veiga, G. A* Based Routing and Scheduling Modules for Multiple AGVs in an Industrial Scenario. *Robotics* **2021**, *10*, 72. [CrossRef]
4. Li, Z.; Liu, J.; Huang, Z.; Peng, Y.; Pu, H.; Ding, L. Adaptive Impedance Control of Human–Robot Cooperation Using Reinforcement Learning. *IEEE Trans. Ind. Electron.* **2017**, *64*, 8013–8022. [CrossRef]
5. Liu, S.; Xiong, M.; Zhong, W.; Xiong, H. Towards Industrial Scenario Lane Detection: Vision-Based AGV Navigation Methods. In Proceedings of the 2020 IEEE International Conference on Mechatronics and Automation (ICMA), Beijing, China, 13–16 October 2020; pp. 1101–1106. [CrossRef]
6. Yang, Z.; Yang, X.; Wu, L.; Hu, J.; Zou, B.; Zhang, Y.; Zhang, J. Pre-Inpainting Convolutional Skip Triple Attention Segmentation Network for AGV Lane Detection in Overexposure Environment. *Appl. Sci.* **2022**, *12*, 10675. [CrossRef]
7. Matos, D.; Costa, P.; Lima, J.; Costa, P. Multi AGV Coordination Tolerant to Communication Failures. *Robotics* **2021**, *10*, 55. [CrossRef]
8. Chowdhury, M.E.H.; Khandakar, A.; Ahmed, S.; Al-Khuzaei, F.; Hamdalla, J.; Haque, F.; Reaz, M.B.I.; Al Shafei, A.; Al-Emadi, N. Design, Construction and Testing of IoT Based Automated Indoor Vertical Hydroponics Farming Test-Bed in Qatar. *Sensors* **2020**, *20*, 5637. [CrossRef] [PubMed]
9. Lottes, P.; Behley, J.; Milioto, A.; Stachniss, C. Fully Convolutional Networks with Sequential Information for Robust Crop and Weed Detection in Precision Farming. *IEEE Robot. Autom. Lett.* **2018**, *3*, 2870–2877. [CrossRef]
10. Tokekar, P.; Hook, J.V.; Mulla, D.; Isler, V. Sensor Planning for a Symbiotic UAV and UGV System for Precision Agriculture. *IEEE Trans. Robot.* **2016**, *32*, 1498–1511. [CrossRef]
11. Qadeer, N.; Shah, J.H.; Sharif, M.; Khan, M.A.; Muhammad, G.; Zhang, Y.-D. Intelligent Tracking of Mechanically Thrown Objects by Industrial Catching Robot for Automated In-Plant Logistics 4.0. *Sensors* **2022**, *22*, 2113. [CrossRef]
12. 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]
13. Sheng, W.; Thobbi, A.; Gu, Y. An Integrated Framework for Human–Robot Collaborative Manipulation. *IEEE Trans. Cybern.* **2015**, *45*, 2030–2041. [CrossRef] [PubMed]
14. Bozek, P.; Karavaev, Y.L.; Ardentov, A.A.; Yefremov, K.S. Neural network control of a wheeled mobile robot based on optimal trajectories. *Int. J. Adv. Robot. Syst.* **2020**, *17*, 172988142091607. [CrossRef]
15. Urban, R.; Štroner, M.; Kuric, I. The use of onboard UAV GNSS navigation data for area and volume calculation. *Acta Montan. Slovaca* **2020**, *25*, 361–374. [CrossRef]
16. Feng, S.; Sebastian, B.; Ben-Tzvi, P. A Collision Avoidance Method Based on Deep Reinforcement Learning. *Robotics* **2021**, *10*, 73. [CrossRef]
17. Huang, X.; Wu, W.; Qiao, H.; Ji, Y. Brain-Inspired Motion Learning in Recurrent Neural Network With Emotion Modulation. *IEEE Trans. Cogn. Dev. Syst.* **2018**, *10*, 1153–1164. [CrossRef]
18. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
19. Caglayan, A.; Can, A.B. Volumetric Object Recognition Using 3-D CNNs on Depth Data. *IEEE Access* **2018**, *6*, 20058–20066. [CrossRef]
20. Zhang, D.; Li, J.; Xiong, L.; Lin, L.; Ye, M.; Yang, S. Cycle-Consistent Domain Adaptive Faster RCNN. *IEEE Access* **2019**, *7*, 123903–123911. [CrossRef]
21. Zhang, Y.; Song, C.; Zhang, D. Deep Learning-Based Object Detection Improvement for Tomato Disease. *IEEE Access* **2020**, *8*, 56607–56614. [CrossRef]
22. Josef, S.; Degani, A. Deep Reinforcement Learning for Safe Local Planning of a Ground Vehicle in Unknown Rough Terrain. *IEEE Robot. Autom. Lett.* **2020**, *5*, 6748–6755. [CrossRef]
23. Yang, H.; Chen, L.; Chen, M.; Ma, Z.; Deng, F.; Li, M.; Li, X. Tender Tea Shoots Recognition and Positioning for Picking Robot Using Improved YOLO-V3 Model. *IEEE Access* **2019**, *7*, 180998–181011. [CrossRef]
24. Fang, W.; Wang, L.; Ren, P. Tinier-YOLO: A Real-Time Object Detection Method for Constrained Environments. *IEEE Access* **2020**, *8*, 1935–1944. [CrossRef]
25. Divyanth, L.G.; Soni, P.; Pareek, C.M.; Machavaram, R.; Nadimi, M.; Paliwal, J. Detection of Coconut Clusters Based on Occlusion Condition Using Attention-Guided Faster R-CNN for Robotic Harvesting. *Foods* **2022**, *11*, 3903. [CrossRef] [PubMed]
26. Du, Y.-C.; Muslikhin, M.; Hsieh, T.-H.; Wang, M.-S. Stereo Vision-Based Object Recognition and Manipulation by Regions with Convolutional Neural Network. *Electronics* **2020**, *9*, 210. [CrossRef]
27. Cheng, R.; Wang, K.; Bai, J.; Xu, Z. Unifying Visual Localization and Scene Recognition for People With Visual Impairment. *IEEE Access* **2020**, *8*, 64284–64296. [CrossRef]
28. Chalup, S.K.; Murch, C.L.; Quinlan, M.J. Machine Learning With AIBO Robots in the Four-Legged League of RoboCup. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **2007**, *37*, 297–310. [CrossRef]
29. Ali, M.A.H.; Baggash, M.; Rustamov, J.; Abdulghafor, R.; Abdo, N.A.-D.N.; Abdo, M.H.G.; Mohammed, T.S.; Hasan, A.A.; Abdo, A.N.; Turaev, S.; et al. An Automatic Visual Inspection of Oil Tanks Exterior Surface Using Unmanned Aerial Vehicle with Image Processing and Cascading Fuzzy Logic Algorithms. *Drones* **2023**, *7*, 133. [CrossRef]
30. Semwal, A.; Lee, M.M.J.; Sanchez, D.; Teo, S.L.; Wang, B.; Mohan, R.E. Object-of-Interest Perception in a Reconfigurable Rolling-Crawling Robot. *Sensors* **2022**, *22*, 5214. [CrossRef]

31. Singh, D. Fast-BoW: Scaling Bag-of-Visual-Words Generation. In Proceedings of the 2018 British Machine Vision Conference, Newcastle, UK, 2–6 September 2018.
32. Feng, M.; Wang, Y.; Liu, J.; Zhang, L.; Zaki, H.F.M.; Mian, A. Benchmark Data Set and Method for Depth Estimation From Light Field Images. *IEEE Trans. Image Process.* **2018**, *27*, 3586–3598. [CrossRef]
33. Dornaika, F.; Horaud, R. Simultaneous robot-world and hand-eye calibration. *IEEE Trans. Robot. Autom.* **1998**, *14*, 617–622. [CrossRef]
34. Ibrahim, Y.; Wang, H.; Liu, J.; Wei, J.; Chen, L.; Rech, P.; Adam, K.; Guo, G. Soft errors in DNN accelerators: A comprehensive review. *Microelectron. Reliab.* **2020**, *115*, 113969. [CrossRef]
35. Muslikhin; Horng, J.-R.; Yang, S.-Y.; Wang, M.-S. Self-Correction for Eye-In-Hand Robotic Grasping Using Action Learning. *IEEE Access* **2021**, *9*, 156422–156436. [CrossRef]
36. Chen, P.-J.; Yang, S.-Y.; Chen, Y.-P.; Muslikhin, M.; Wang, M.-S. Slip Estimation and Compensation Control of Omnidirectional Wheeled Automated Guided Vehicle. *Electronics* **2021**, *10*, 840. [CrossRef]
37. Adam, S.; Busoniu, L.; Babuska, R. Experience Replay for Real-Time Reinforcement Learning Control. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **2012**, *42*, 201–212. [CrossRef]
38. Sanchez, A.G.; Smart, W.D. Verifiable Surface Disinfection Using Ultraviolet Light with a Mobile Manipulation Robot. *Technologies* **2022**, *10*, 48. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.