

Article

# Model-Predictive Control for Omnidirectional Mobile Robots in Logistic Environments Based on Object Detection Using CNNs

Stefan-Daniel Achirei <sup>1,\*</sup>, Razvan Mocanu <sup>2</sup>, Alexandru-Tudor Popovici <sup>1</sup> and Constantin-Catalin Dosoftci <sup>2</sup><sup>1</sup> Department of Computer Engineering, “Gheorghe Asachi” Technical University of Iasi, 700050 Iasi, Romania<sup>2</sup> Department of Automatic Control and Applied Informatics, “Gheorghe Asachi” Technical University of Iasi, 700050 Iasi, Romania

\* Correspondence: stefan-daniel.achirei@academic.tuiasi.ro

**Abstract:** Object detection is an essential component of autonomous mobile robotic systems, enabling robots to understand and interact with the environment. Object detection and recognition have made significant progress using convolutional neural networks (CNNs). Widely used in autonomous mobile robot applications, CNNs can quickly identify complicated image patterns, such as objects in a logistic environment. Integration of environment perception algorithms and motion control algorithms is a topic subjected to significant research. On the one hand, this paper presents an object detector to better understand the robot environment and the newly acquired dataset. The model was optimized to run on the mobile platform already on the robot. On the other hand, the paper introduces a model-based predictive controller to guide an omnidirectional robot to a particular position in a logistic environment based on an object map obtained from a custom-trained CNN detector and LIDAR data. Object detection contributes to a safe, optimal, and efficient path for the omnidirectional mobile robot. In a practical scenario, we deploy a custom-trained and optimized CNN model to detect specific objects in the warehouse environment. Then we evaluate, through simulation, a predictive control approach based on the detected objects using CNNs. Results are obtained in object detection using a custom-trained CNN with an in-house acquired data set on a mobile platform and in the optimal control for the omnidirectional mobile robot.

**Keywords:** omnidirectional mobile robots; object detection; convolutional neural networks; depth sensing; computer vision; discretized-time model; predictive control algorithm; navigation



**Citation:** Achirei, S.-D.; Mocanu, R.; Popovici, A.-T.; Dosoftci, C.-C. Model-Predictive Control for Omnidirectional Mobile Robots in Logistic Environments Based on Object Detection Using CNNs. *Sensors* **2023**, *23*, 4992. <https://doi.org/10.3390/s23114992>

Academic Editors: Ikhlas Abdel-Qader and Enrico Meli

Received: 22 March 2023

Revised: 6 May 2023

Accepted: 19 May 2023

Published: 23 May 2023



**Copyright:** © 2023 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

The mobile robots sector has seen a global rise over the past decade. Industrial mobile robots are becoming more advanced to achieve higher levels of autonomy and efficiency in various industries [1]. These robots are equipped with sophisticated sensors, such as Light Detection and Ranging (LiDAR), stereo cameras, Inertial Measurement Unit (IMU), and a global positioning system or indoor positioning system, to gather information about the work environment and make well-informed decisions [2]. This is made possible by using complex algorithms for path planning, obstacle avoidance, and task execution. Furthermore, autonomous mobile robots, grouped in fleets, are often integrated with cloud-based technologies for remote monitoring and control, allowing for greater flexibility and scalability in their deployment.

Path planning is a crucial aspect of mobile robotics navigation because of the need to perform a task by moving from one point to another while avoiding obstacles and satisfying more constraints, among which are time, the level of autonomy given by the energy available, and significantly, maintaining safety margins regarding human operators and transported cargo. Mobile robot navigation is still one of the most researched topics of today, addressing two main categories: classical and heuristic navigation. In the variety of classical

approaches, the most well-known algorithms, characterized by limited intelligence in [3], are cell decomposition, roadmap approach, and artificial potential field (APF). Heuristic approaches are more intelligent, including but not limited to the main components of computational intelligence (i.e., fuzzy logic, neural networks, and genetic algorithms). Researchers investigate solutions based on the particle swarm optimization algorithm, the FireFly algorithm, and the artificial BEE colony algorithm [4]. Combining classical and heuristic approaches, known as hybrid algorithms, offers better performances, especially for navigation in complex, dynamic environments [3].

In dynamic operational environments, the increased flexibility of autonomous mobile robots compared to automated guided vehicles is an added advantage due to decreased infrastructure setup and maintenance costs. Supplementary, the omnidirectional mobile robots (OMR), compared to other traction and steering systems (e.g., differential drive and Ackerman), provide three independent degrees of freedom (longitudinal and lateral translation, together with in-place rotation), motions that can be combined within certain speed and acceleration limits without producing any excessive wear on the ground contact surfaces. On the other hand, to obtain precise motion for the OMR, certain constraints apply to their suspension system and the smoothness of the ground surface.

Considering the computational requirements criteria, the planning technology of a mobile robot is divided into offline planning and online planning [5]. In offline planning, the path for the robot is pre-computed and stored in the robot's memory. The robot then follows the pre-computed path to reach its destination. This approach is suitable for deterministic environments with a priori information. When the mobile robot navigates and performs tasks in a dynamic and uncertain environment, it is necessary to use the online planning approach. The robot computes its path in real time based on its current location and the information obtained from its perception module.

Independent of the type of path planning algorithm, the OMR structure is beneficial because it better resembles the material point model used for simplifying the modeling of robots in motion planning simulations. In [6–9], a four-wheel's dynamic and kinematic modeling, OMR was studied using the Lagrange framework. Sliding mode control allows robust control for OMRs employing mecanum-wheels and rejects disturbances caused by unmodeled dynamics [10–12]. The nonholonomic model of the wheel was used to develop the dynamic equation of an OMR with four mecanum wheels [13]. The kinematic model of a three-wheeled mobile robot was used to create a predictive control model and filtered Smith predictor for steering the robot along predetermined paths [14]. A reduced dynamic model of the robot is the basis for developing a nonlinear model-predictive controller for trajectory tracking of a mecanum wheeled OMR [15]. A constrained quadratic programming problem is formulated towards optimizing the trajectory of a four-wheel omnidirectional robot [16]. Dynamic obstacles are considered in the work of the authors [17], whereas the numerical implementation presented in [18] is based on a three-wheeled omnidirectional robot. Distributed predictive control on a cooperative paradigm is discussed for a coalition of robots [19]. A nonlinear predictive control strategy with a self-rotating prediction horizon for OMR in uncertain environments is discussed. The appropriate prediction horizon was selected by incorporating the effects of moving velocity and road curvature on the system [20]. Adaptive model-predictive control, with friction compensation and incremental input constraints, is presented for an omnidirectional mobile robot [21]. Wrench equivalent optimality is used in a model-predictive control formulation to control a cable-driven robot [22]. Authors discuss an optimal controller to control the robot's motion on a minimum energy trajectory [23]. Recently, potential field methods have been used mainly due to their naturally inspired logic. These methods are also widely used in omnidirectional mobile robots due to their simplicity and performance in obstacle avoidance [24,25]. Timed elastic-band approaches utilize a predictive control strategy to steer the robot in a dynamic environment to tackle real-time trajectory planning tasks [26]. Because the optimization is confined to local minima, the original timed elastic-band planner may cause a route through obstacles. Researchers proposed an improved strategy for producing alternate

sub-optimal trajectory clusters based on unique topologies [27]. To overcome the mismatch problem between the optimization graph and grid-based map, the authors suggested an egocentric map representation for a timed elastic band in an unknown environment [28]. These path-planning methods are viable and pragmatic, and acquiring a desired path in various scenarios is generally possible. Yet, these approaches could have multiple drawbacks, such as a local minimum, a low convergence rate, a lack of robustness, substantial computation, and so on. Additionally, in logistic environments where OMR robots are equipped with conveyor belts to transport cargo, it is essential to guarantee low translational and rotational accelerations for the safety of the transported cargo. Therefore, we propose a nonlinear predictive control strategy on a reduced model where we can include maximum acceleration and velocities of the wheels within inequality constraints derived from the obstacle positions obtained from environment perception sensors (i.e., LiDAR and video camera). To tackle the problem of local minima, we propose a variable cost function based on the proximity of obstacles ahead to balance the global objectives.

#### *Object Detection for Mobile Platforms*

Deep neural networks specifically created to analyze organized arrays of data (i.e., images) are known as convolutional neural networks, often called CNNs or ConvNets. CNNs offered solutions to computer vision challenges that are difficult to handle using conventional methods. They quickly advance to the state-of-the-art in areas such as semantic segmentation, object detection, and image classification. They are widely used in computer vision because they can quickly identify image patterns (such as lines, gradients, or more complex objects such as eyes and faces). CNNs are convolutional-layered feed-forward neural networks. CNNs attempt to mimic the structure of the human visual cortex with these specific layers.

Localization of object instances in images is implied by object detection. Object recognition generally assigns a class to the identified objects from a previously learned class list. Although object detection operates at the bounding-box level, it has no notion of different classes. The phrase “object detection” now encompasses both activities, even though they were initially two distinct jobs. So, before continuing, let’s be clear that object detection includes both object localization and object recognition.

Object detection and recognition is an essential field of study in the context of autonomous systems. The models can be broadly divided into one-stage and two-stage detectors. One-stage detectors are designed to detect objects in a single step, making them faster and more suitable for real-time applications, such as path planning based on object detection for a moving system. On the other hand, two-stage detectors use a two-step process, first proposing regions of interest and then looking for objects within those areas. This approach excludes irrelevant parts of the image, and the process is highly parallelizable. However, it comes at the cost of being slower than one-stage detectors.

To meet the constraints of the Nvidia Jetson mobile platforms considered for the OMR, lightweight neural networks were investigated for object detection and recognition. Among the models evaluated, YoloV5 [29], SSD-Mobilenet-v1 [30], SSD-Mobilenet-v2-lite [31] and SSD-VGG16 [32] were trained and tested. Earlier, the YoloV4 [33] model had already made significant improvements over the previous iteration by introducing a new backbone architecture and modifying the neck of the model, resulting in an improvement of mean average precision (mAP) by 10% and an increase in FPS by 12%. Additionally, the training process has been optimized for single GPU architectures, like the Nvidia Jetson family, commonly used in embedded and mobile systems.

A particular implementation is YoloV5 [29], which differs from other Yolo implementations using the PyTorch framework [34] rather than the original Yolo Darknet repository. This implementation offers a wide range of architectural complexity, with ten models available, starting from the YoloV5n (nano), which uses only 1.9M parameters, up to the YoloV5x6 (extra large), which uses 70 times as many parameters (140M). The lightest models are recommended for Nvidia Jetson platforms.

Recently, an increasing interest has been in developing mobile device object detection and recognition algorithms. A popular approach is using the Single Shot Detector (SSD) [35] neural network, a one-step algorithm. To further improve the efficiency of the SSD algorithm on mobile devices, researchers have proposed various modifications to the SSD architecture, such as combining it with other neural network architectures. One such modification is the use of the SSD-MobileNet and SSD-Inception architectures, which combine the SSD300 [35] neural network with various backbone architectures, such as MobileNet [30] or Inception [36]. These architectures, such as the Nvidia Jetson development platforms, are recognized for their real-time object detection capabilities on mobile devices.

These methods for object detection perform very well in general detection tasks. Yet, there must be more datasets and pretrained models for objects specific to the OMR environment, such as fixed or mobile conveyors, charging stations, other OMRs, etc. We have acquired our dataset and deployed domain-specific models for object detection in the OMR environment. We summarize the main contributions of this paper to the field of object detection and OMR control in logistic environments below:

- Acquisition of a data set for object detection in the OMR environment;
- Investigation of domain-specific models for object detection and providing a model to be used in an OMR environment;
- Deployed an image acquisition and object detection module fit for the real-time task of OMR control;
- Proposed a joint perception&control strategy based on a non-linear model-predictive control;
- Avoid local minima by using switched cost function weights to navigate around obstacles while still achieving the overall objective of decreasing travel distance;
- Guarantee maximum wheel speed and acceleration through the constrained non-linear MPC in order to ensure safe transportation of cargo;

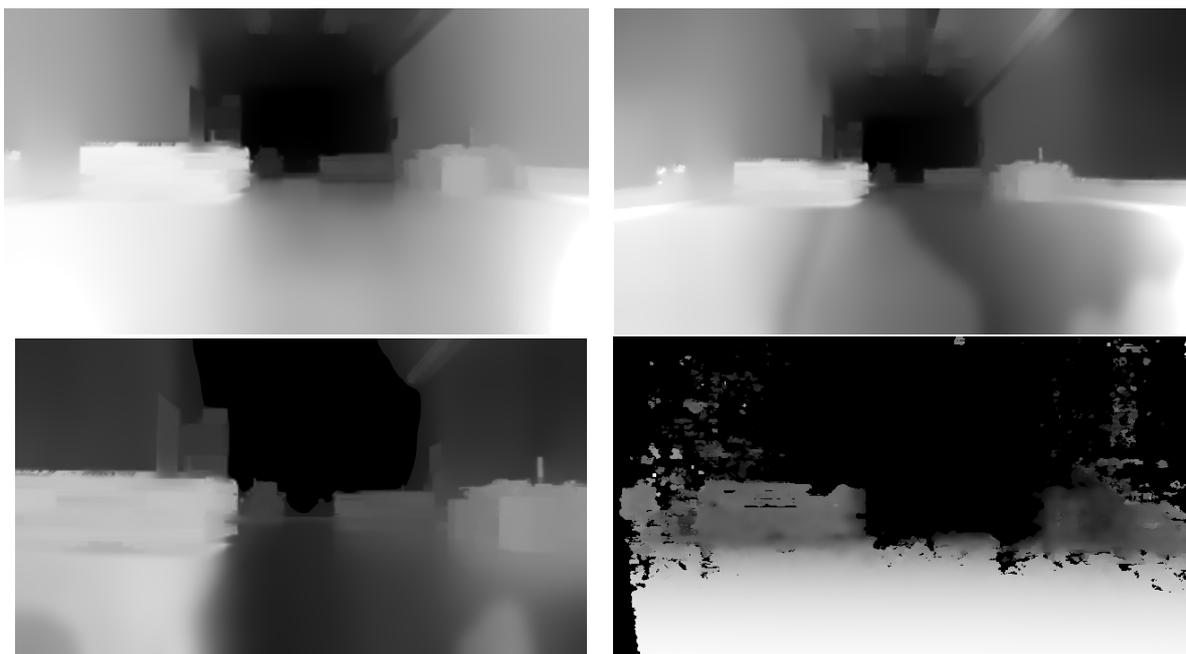
The rest of the paper is organized as follows: Section 2 discusses object detection in the context of OMR's logistic environment. First, some equipment experiments were conducted on the image acquisition sensor and the processing unit. We also describe the object detection dataset creation and object mapping in 2D and 3D perspectives. Section 3 is dedicated to the modeling and control of the OMR. We introduce the mathematical model used for developing the control strategy, followed by formulating the optimization problem considering the environmental objects. In the last two sections, we discuss the object detection results and the simulation of the control algorithm, conclude, and emphasize future work goals.

## 2. Object Detection for Omnidirectional Mobile Robots

### 2.1. Image Acquisition and Processing Unit

For the image acquisition unit, we analyzed four depth cameras. Depth information is needed to accurately place the detected objects on the 2D and 3D maps of the environment. The predictive control task relies on object maps. The most important features considered for the experiments were the correctness of the depth information and the integration of the camera with the Nvidia Jetson platforms, which are already in use on the Omnidirectional Robot.

All Zed cameras perform well in indoor environments, but, as can be seen in Figure 1, the far-depth information provided by Zed 2i is significantly better. Depth information is completely missing after 10 m for Intel RealSense. The best depth information is given by Zed 2i; it also has the largest FoV. Based on the image acquisition experiments performed in the OMR environment, Zed 2i was chosen to be integrated into the robot.



**Figure 1.** Depth information for ZED 1 (top left), ZED 2i (top right), ZED mini (bottom left), and Intel RealSense D435i (bottom right).

Nvidia Jetson system-on-chip platforms are already used on the OMR. Some experiments evaluated the computational capabilities, detection precision, and the dependency between inference time and resolution. Localization is very important in our defined use cases for the OMR environment. MS COCO dataset [37] was used for object detection evaluation across different lightweight neural networks such as Mobilenet [30,31] and Yolo [29,33] which are suitable for mobile platforms.

The neural networks used for the first experiment are optimized using TensorRT to run on Jetson mobile platforms. In Table 1, we can see the run-time measurements for the selected models from the SSD family. The same solution takes considerably more time to run on the Jetson Nano.

**Table 1.** Object detection evaluation of the SSD model family.

Architecture	FPS on Jetson Nano	FPS on Jetson Xavier AGX
SSD–Mobilenet–v1	10	83
SSD–Mobilenet–v2	7	61
SSD–Inception–v2	6	42

A second experiment aims to see how the processing time evolves depending on the image resolution. Table 2 presents the results in terms of FPS on a test subset from Cityscapes data set [38,39]. The results emphasize that the inference time depends on the size of the images provided at input. Thus, the higher the image resolution, the slower the model. Jetson Xavier AGX is 4 to 6 times faster than Jetson Nano, depending on the model and the input resolution.

**Table 2.** Inference Time vs. Image Resolution.

Architecture	Resolution	FPS on Jetson Nano	FPS on Jetson Xavier AGX
SSD–Mobilenet–v1	2048 × 1024	15 fps	91 fps
	1024 × 512	23 fps	102 fps
SSD–Mobilenet–v2	2048 × 1024	12 fps	74 fps
	1024 × 512	18 fps	76 fps
SSD–Inception–v2	2048 × 1024	11 fps	63 fps
	1024 × 512	15 fps	63 fps

Following the analysis of hardware equipment and the experimental measurements, the Jetson architecture chosen to be integrated into the proposed solution for object detection in the OMR environment that meets the minimum requirements is the Nvidia Jetson Xavier AGX.

## 2.2. The Omnidirectional Robot Object Detection Dataset (OROD)

Enabling the efficient operation of autonomous robots is crucial for accurately detecting and recognizing objects specific to the OMR environment. Using the ZED 2i camera, we have acquired a new dataset for the object detection task that contains objects specific to the omnidirectional robot environment. The “Omnidirectional Robot Object Detection (OROD)” dataset includes charging stations, construction cones, mobile conveyors, and different types of fixed conveyors. The images in the dataset were captured using the camera mounted on an omnidirectional robot and were annotated with bounding boxes of objects. The dataset is intended to evaluate the performance of object detection algorithms in an omnidirectional robot environment.

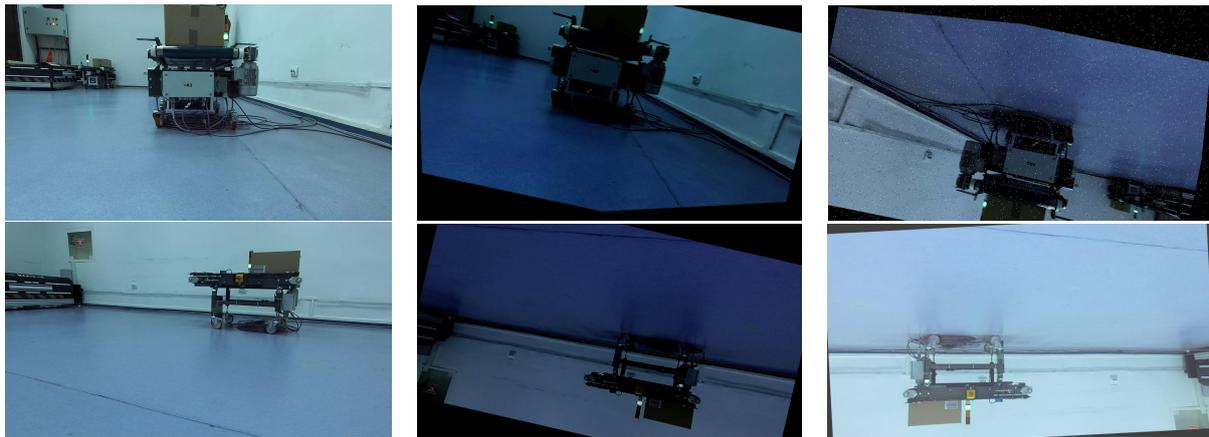
The OROD dataset contains 1343 images, each labeled with the objects of interest in the scene. The images were collected in different environments, such as industrial warehouses and logistics centers, to reflect the various scenarios in which an omnidirectional robot operates. Additionally, the data set includes images with varying lighting conditions, occlusions, and different orientations of the objects to represent real-world challenges in object detection. The training subset was augmented for better results by applying flip, rotation, zoom, hue, saturation, blur, noise, etc. The original and augmented data sets were split according to the figures from Table 3. Examples of the augmented images can be visualized in Figure 2. The dataset augmentation process did not change the initial class distribution; it scaled by 3.

**Table 3.** OROD train-val-test split.

	Annotated Frames before/after Augmentation	Percentage before/after Augmentation
train-initial	940/2816	70/88
validation	269/269	20/8
test	134/134	10/4

The OROD dataset is the first to focus specifically on object detection in the context of an omnidirectional robot environment. It is intended to serve as a reference for evaluating the performance of object detection algorithms in this context and to promote research in this field.

The augmented data set and the raw dataset, both with YOLO annotations, are publicly available at <https://universe.roboflow.com/gheorghe-asachi-technical-university-of-iasi/rmoa>, accessed on 20 May 2023.

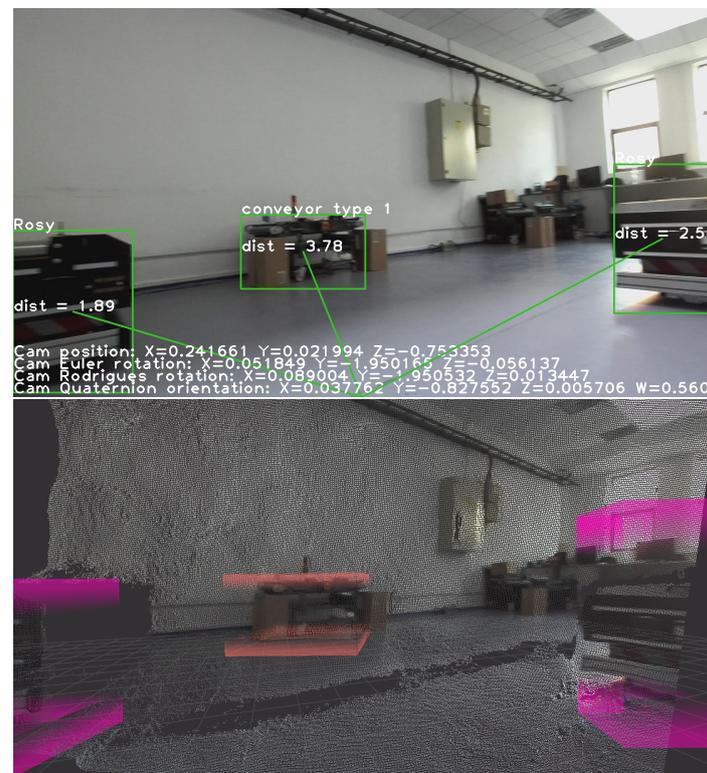


**Figure 2.** Acquired frame (column 1) and augmentation results (columns 2 and 3).

### 2.3. Detected Objects in 3D and Mapping

All objects detected by the custom-trained model, along with the distances to them, are visible on the left side of Figure 3. Their 3D position is also exemplified on the right side. The distance between the scene object and the camera is measured from the back of the left eye of the camera and is given in meters.

In the context of an OMR moving through its environment, an important feature is to continuously be aware of its position and rotation relative to the starting point, the charging station, in our case.



**Figure 3.** Object detection and distance estimation in meters (top) and 3D point cloud mapping (bottom).

Examples of the OMR position and orientation are listed on the bottom of the frames in Figure 3. As a benefit of the IMU integration with Zed 2i, we can obtain the camera position, rotation, and quaternion orientation. In addition to the ZED 2i camera, the OMR is equipped with two LiDAR sensors for a 360-degree map. At this stage, the LiDAR data are empirically merged with the detected objects to obtain a bird's-eye view map of the

entire environment. In Figure 4, we can see the obtained map of the environment with the detected objects shown in Figure 3.

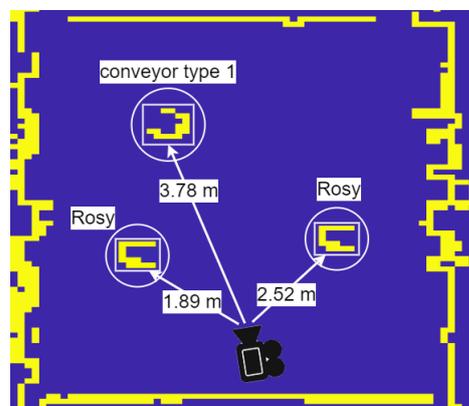


Figure 4. Bird's eye view mapping of detected objects.

### 3. Model-Predictive Motion Control of OMR

We derive the motion control strategy of the OMR based on a non-linear optimization algorithm as the core of the motion controller. We define in Section 3.1 the mathematical model used in the predictions step of the controller. The continuous time equations are discretized by the Euler method to realize the numerical implementation. Then, we define the physical constraints of the robot's actuators (i.e., omnidirectional wheel speed and acceleration) and the geometrical constraints of the objects (i.e., circumscribed circles of objects). We formulate the optimization problem considering the global objective of navigating on the shortest path, avoiding obstacles, and limiting the movement of the OMR within actuator limits.

#### 3.1. Mathematical Model of 3D of Omnidirectional Robot

In this section, we define the discrete mathematical model used in the model-predictive controller to generate short-term paths and control the robot's movement along the predicted trajectory. Equation (1) depicts the inverse kinematics matrix representation:

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = J \begin{bmatrix} v_x \\ v_y \\ \Omega \end{bmatrix} \quad (1)$$

where  $v_x$  and  $v_y$  are the longitudinal and lateral velocities of the OMR, respectively.  $\Omega$  defines the angular speed along the normal axis,  $\omega_j, j = \overline{1..4}$  are the individual wheels' angular velocities, while  $J$  is the inverse kinematic Jacobian matrix of the OMR defined in (2) [1]:

$$J = \frac{1}{R} \begin{bmatrix} 1 & 1 & -(l_x + l_y) \\ 1 & -1 & -(l_x + l_y) \\ 1 & 1 & (l_x + l_y) \\ 1 & -1 & (l_x + l_y) \end{bmatrix} \quad (2)$$

The forward kinematics of the 3DOF system are obtained from the lateral, longitudinal, and rotation velocities:

$$\begin{bmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \\ \frac{d\theta}{dt} \end{bmatrix} = \frac{R}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ \frac{-1}{l_x+l_y} & \frac{-1}{l_x+l_y} & \frac{1}{l_x+l_y} & \frac{1}{l_x+l_y} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} \quad (3)$$

where  $x, y, \theta$  are plane coordinates and robot orientation, respectively. Moreover,  $R$  is the wheel radius,  $l_x$  defines the distance from the GC to the front axle, while  $l_y$  defines the half distance between the left and right wheels.

Pragmatically, it can be considered that deviations from the nominal kinematic model act on the system input. Therefore, we can design an input disturbance observer to compensate for unmodeled dynamics and disturbances. Let us define the disturbance acting on the system input as  $F = [f_1, f_2, f_3, f_4]^t$ , where the additive terms  $F$  act on the system inputs. The observer is designed considering the inverse kinematics of the process. An additional pole is added for the realizability of the observer. We define  $Q$  as a passive (i.e., unitary gain) first-order low-pass filter diagonal matrix. We define the estimated input disturbance as:  $\hat{F} = [\hat{f}_1, \hat{f}_2, \hat{f}_3, \hat{f}_4]^t = -Q[\omega_1, \omega_2, \omega_3, \omega_4]^t + QJ[v_x, v_y, \theta]^t$ . Therefore, the plant model becomes:

$$\begin{bmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \\ \frac{d\theta}{dt} \end{bmatrix} = \frac{R}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ \frac{-1}{l_x+l_y} & \frac{-1}{l_x+l_y} & \frac{1}{l_x+l_y} & \frac{1}{l_x+l_y} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} + \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \quad (4)$$

The discretized-time model of (3) is obtained by backward rectangle area approximation (i.e., Euler method). Therefore, the system Equation (3) can be re-written in the state space framework  $\dot{X} = AX + (J_+)\omega$ , where the state transition matrix is null, the state vector is  $X = [x \ y \ \theta]^t$  while the input matrix  $J_+$  is defined as  $J_+ = (J^T J)^{-1} J^T$ . Thus, we obtain the discretized-time model of the OMR in global coordinates:

$$X_{k+1} = I_3 X_k + (J_+) T_s \omega_k \quad (5)$$

where  $I_3 \in \mathbb{R}^{3 \times 3}$  unity matrix,  $X_{k+1} = [x_k \ y_k \ \theta_k]^t$  is the state vector at iteration  $k + 1$ ,  $T_s$  is the sampling time and  $\omega_k = [\omega_{1k} \ \omega_{2k} \ \omega_{3k} \ \omega_{4k}]^t$  is the input vector.

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} + T_s \frac{R}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ \frac{-1}{l_x+l_y} & \frac{-1}{l_x+l_y} & \frac{1}{l_x+l_y} & \frac{1}{l_x+l_y} \end{bmatrix} \begin{bmatrix} \omega_{1k} \\ \omega_{2k} \\ \omega_{3k} \\ \omega_{4k} \end{bmatrix} \quad (6)$$

To improve controller behavior w.r.t to deviations of the model and input perturbation, the extended discretized model can be used for states and output predictions within the MPC solver:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} + T_s \frac{R}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ \frac{-1}{l_x+l_y} & \frac{-1}{l_x+l_y} & \frac{1}{l_x+l_y} & \frac{1}{l_x+l_y} \end{bmatrix} \begin{bmatrix} \omega_{1k} \\ \omega_{2k} \\ \omega_{3k} \\ \omega_{4k} \end{bmatrix} + T_s \begin{bmatrix} f_{1k} \\ f_{2k} \\ f_{3k} \\ f_{4k} \end{bmatrix} \quad (7)$$

Table 4 contains the parameters of the mobile robot and the sampling time considered for the time-discretization of the process.

**Table 4.** OMR parameters.

Parameter	Value
Wheel radius ( $R$ )	0.076 [m]
Distance from GC to front axle ( $l_x$ )	0.294 [m]
Half distance between left and right wheels ( $l_y$ )	0.2 [m]
Sampling time ( $T_s$ )	0.02 [s]

### 3.2. OMR Motion Optimization Problem

In the optimization problem, we aim to find the solution at time  $kT_s$ , comprised of actuator commands  $\omega_j^{(i,k)}$  for  $j = \overline{1..4}$ ,  $i \in \{1 \dots H\}$  satisfying actuator physical constraints,

the geometric constraints, and to fulfill the global objective of traveling the shortest distance and avoiding the detected obstacles. Therefore, we formulate the optimization problem as follows. Find,

$$\begin{aligned}
 & \min_{x(\cdot|k), y(\cdot|k), \theta(\cdot|k), d_p(\cdot|k)} J_k(X, X_r, \alpha) \\
 & \text{s.t. } -\omega_{UB} \leq \omega_{j,j=\overline{1..4}} \leq \omega_{UB} \\
 & \quad -a_{UB} \leq \dot{\omega}_{j,j=\overline{1..4}} \leq a_{UB} \\
 & \quad C_o < 0
 \end{aligned} \tag{8}$$

where  $J_k$  is the cost function defined in (9),  $x(\cdot|k), y(\cdot|k), \theta(\cdot|k)$  are the solutions of the optimization problem;  $\omega_{UB}$  and  $a_{UB}$  are the upper bounds of the angular velocity and acceleration of the wheels, respectively.  $C_o$  is the geometric constraints vector and is defined in (22).

The cost function  $J_k$  is defined by:

$$\begin{aligned}
 J_k(X, X_r, \alpha) = & \frac{1}{2} \sum_{i=0}^{H-1} [w_x(\alpha)(x_r^{(i|k)} - x^{(i|k)})^2 + w_y(\alpha)(y_r^{(i|k)} - y^{(i|k)})^2 + w_\theta(\theta_r^{(i|k)} - \theta^{(i|k)})^2 + \\
 & + w_{Tx}(x_r^{(H-1|k)} - x^{(i|k)})^2 + w_{Ty}(y_r^{(H-1|k)} - y^{(i|k)})^2] + w_p(\alpha) \sum_{i=0}^{H-1} (d_p(X, X_0, X_f)^{(i|k)}) \tag{9}
 \end{aligned}$$

where  $X_r \in R^{H \times 3}$  is the reference trajectory matrix of the OMR over the prediction horizon  $H$ :

$$X_r = \begin{bmatrix} x_r^{(0,k)} & y_r^{(0,k)} & \theta_r^{(0,k)} \\ x_r^{(1,k)} & y_r^{(1,k)} & \theta_r^{(1,k)} \\ \dots & \dots & \dots \\ x_r^{(H-1,k)} & y_r^{(H-1,k)} & \theta_r^{(H-1,k)} \end{bmatrix} \tag{11}$$

$d_p(X_0, X_f)$  is the length of the projection of the OMR geometric center over the ideal straight path connecting the starting (i.e., initial) node with the final node and is defined in (12):

$$d_p(X, X_0, X_f) = \sqrt{|L_1^2 - [(L_1^2 + L_3^2 - L_2^2) / (2L_3)]^2|} \tag{12}$$

with

$$L_1 = \sqrt{(x - x_0)^2 + (y - y_0)^2} \tag{13}$$

$$L_2 = \sqrt{(x - x_r^{(H-1)})^2 + (y - y_r^{(H-1)})^2} \tag{14}$$

$$L_3 = \sqrt{(x_0 - x_r^{(H-1)})^2 + (y_0 - y_r^{(H-1)})^2} \tag{15}$$

where  $L_1, L_2$ , and  $L_3$  define the L2-norms between the OMR position, initial, and resting positions, while  $X_0 = [x_0, y_0, \theta_0]^t$  and  $X_f = [x_r^{(H-1)}, y_r^{(H-1)}, \theta_r^{(H-1)}]^t = [x_f, y_f, \theta_f]^t$  are the initial and final resting positions. In the cost function, we aim to penalize by weights  $w_x(\alpha)$  and  $w_y(\alpha)$  the deviation from the reference trajectory  $x_r^{(i)}, y_r^{(i)}, i = \overline{0..H-1}$  defined by (25); by weight  $w_\theta$  it is penalized the deviation from the desired orientation of the OMR. The set-point orientation  $\theta_r^{(i)}, i = \overline{0..H-1}$  is such that the OMR remains with the frontal part facing the destination location. By  $w_{Tx}$ , we penalize the terminal cost of  $x_r$  and  $y_r$  to reduce the steady-state error. Therefore,  $w_{Tx} > w_x$  and  $w_{Ty} > w_y$ ;  $w_p(\alpha)$  is a weight with two discrete states, and its value is a function of  $\alpha$  which depends on the proximity (tolerance) of the closest object and is defined by (23).

The actuator constraints of the OMR are defined as:

$$-\omega_{UB} \leq \omega_{j,j=\overline{1..4}} \leq \omega_{UB} \tag{16}$$

$$-a_{UB} \leq \dot{\omega}_{j,j=\overline{1..4}} \leq a_{UB} \tag{17}$$

We define the physical-space constraints from the coordinates of the objects and their known sizes as:

$$-(x - x_o)^2 - (y - y_o)^2 + r_o^2 \leq 0 \tag{18}$$

where  $x_o = [x_{o_1} \dots x_{o_{n_o}}]^t$  and  $y_o = [y_{o_1} \dots y_{o_{n_o}}]^t$  are the coordinates  $r_o = [r_{o_1} \dots r_{o_{n_o}}]^t$  is the radius of the circle circumscribed about the polygon defining the object. Index  $n_o$  refers to *Number-of-Objects*, while, index  $o$  signifies the word *Objects*, also, index  $a$  refers to word *Actuators*. The global coordinates of the obstacles are obtained from the local coordinates of the OMR according to the equation below:

$$\begin{bmatrix} x_o \\ y_o \end{bmatrix} = \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} \left[ \begin{bmatrix} x_l \\ y_l \end{bmatrix} - \begin{bmatrix} x \\ y \end{bmatrix} \right] \tag{19}$$

where  $\phi$  is the angle from the global system’s abscissa to the local system’s abscissa.  $x_l$  and  $y_l$  are the local coordinates of the detected objects, and  $x$  and  $y$  are the global coordinates of the local system’s origin. From inequality constraints (16) and (18), we obtain a concatenated vector of inequality constraints denoted by  $C_k \in \mathbb{R}^{(n_a \cdot n_w \cdot H + n_o \cdot H) \times 1} \leq 0$ :

$$C_k = [C_a^t, C_o^t]^t \in \mathbb{R}^{(n_a \cdot n_w \cdot H + n_o \cdot H) \times 1} \tag{20}$$

where  $C_a \in \mathbb{R}^{n_a \cdot n_w \cdot H \times 1}$ ,  $C_o \in \mathbb{R}^{n_o \cdot H \times 1}$  are defined below:

$$C_a^{(k)} = \begin{bmatrix} |\omega_1^{(0|k)}| - \omega_{UB} \\ |\omega_1^{(1|k)}| - \omega_{UB} \\ \vdots \\ |\omega_1^{(H-1|k)}| - \omega_{UB} \\ |\omega_2^{(0|k)}| - \omega_{UB} \\ \vdots \\ |\omega_4^{(H-1|k)}| - \omega_{UB} \\ |\dot{\omega}_1^{(0|k)}| - a_{UB} \\ \vdots \\ |\dot{\omega}_4^{(H-1|k)}| - a_{UB} \end{bmatrix} \leq 0 \tag{21}$$

$$C_o^{(k)} = \begin{bmatrix} -(x^{(0|k)} - x_{o_1})^2 - (y^{(0|k)} - y_{o_1})^2 + r_{o_1}^2 \\ -(x^{(1|k)} - x_{o_1})^2 - (y^{(1|k)} - y_{o_1})^2 + r_{o_1}^2 \\ \vdots \\ -(x^{(H-1|k)} - x_{o_1})^2 - (y^{(H-1|k)} - y_{o_1})^2 + r_{o_1}^2 \\ -(x^{(0|k)} - x_{o_2})^2 - (y^{(0|k)} - y_{o_2})^2 + r_{o_2}^2 \\ \vdots \\ -(x^{(H-1|k)} - x_{o_{n_o}})^2 - (y^{(H-1|k)} - y_{o_{n_o}})^2 + r_{o_{n_o}}^2 \end{bmatrix} \leq 0 \tag{22}$$

In the previous equations,  $n_a = 2$  defines the number of constraints regarding actuators, and it is two because we included two types of actuator restrictions: angular speed and angular acceleration.

We approximate numerically  $\dot{\omega}_j$  by  $\dot{\omega}_j \approx \frac{(\omega_j^{(i)} - \omega_j^{(i-1)})}{T_s}$  where  $T_s$  is the sampling time.

In the cost function (9), we propose that  $w_p(\alpha)$ ,  $w_x(\alpha)$  and  $w_y(\alpha)$  are switched between their two states based on the value of  $\alpha = \max_{1 \leq j \leq n_o \cdot H} C_{o_j}$  which, practically, determines the minimum proximity to an obstacle from the object list. Therefore,

$$\begin{aligned} w_p(\alpha) &= \begin{cases} w_{p1}, & \text{if } \alpha > tol \\ w_{p2}, & \text{if } \alpha \leq tol \end{cases} \\ w_x(\alpha) = w_y(\alpha) &= \begin{cases} w_{xy1}, & \text{if } \alpha > tol \\ w_{xy2}, & \text{if } \alpha \leq tol \end{cases} \end{aligned} \quad (23)$$

In the previous equation,  $tol$  defines the avoidance tolerance.

The set-point orientation over the control horizon  $H$  is defined as:

$$\theta_r^{(i)} = \text{atan}_2(y_r^{(H-1)} - y, x_r^{(H-1)} - x) \frac{360}{2\pi}, \text{ for } i = \overline{0..H-1} \quad (24)$$

and the reference trajectory is given by a first-order static function where the slope  $\lambda$  and the bias  $\rho$  are given by:

$$\begin{aligned} \lambda &= \begin{cases} \frac{y_f - y}{x_f - x}, & \text{if } x_f \neq x \\ 0, & \text{otherwise} \end{cases} \\ \rho &= \begin{cases} y_f - \frac{y_f - y}{x_f - x} x_f, & \text{if } x_f \neq x \\ 0, & \text{otherwise} \end{cases} \end{aligned}$$

$$\begin{aligned} x_r^{(i)} &= \frac{x_f - x}{H} i + x, i = \overline{0..H-2} \\ x_r^{(H-1)} &= x_f \\ y_r^{(i)} &= \lambda x_f^{(i)} + \rho, i = \overline{0..H-2} \\ y_r^{(H-1)} &= y_f \end{aligned} \quad (25)$$

### 3.3. Control Algorithm—One Step Optimization

The control algorithm core is the sequential quadratic optimizer with a constraint tolerance of  $1.0 \times 10^{-3}$  and an optimality tolerance of  $1.0 \times 10^{-4}$  deduced heuristically through multiple experiments. Under this parametrization, the behavior is fairly robust and predictable with respect to the initial robot position, final resting position, varying size obstacles, wheel speeds, and acceleration. The object lists consist of a matrix of object positions obtained from the perception module. In order to determine the radius of the obstacles, we use the Moor–Neighbour tracing algorithm with Jacob’s stopping criteria, which provides the contour of the objects from LIDAR data. Beyond providing LIDAR data, CNN can provide estimates of object radius with higher precision based on the object class. In order to reduce the computation time, the optimization problem is reformulated at each sampling time, and we consider in the optimization only those objects within a maximum radius ( $d_{max}$ ) relative to the OMR’s geometric center. The avoidance radius for each object is determined from the actual object radius with an additional tolerance according to OMR’s dimensions. The reference orientation  $\theta_r$ , and reference trajectory  $(x_r, y_r)$  are determined at each sample time since the OMR position evolves from one pose to another, constantly changing the heading to the final resting position. The first computed command over the predicted horizon is applied to the process inputs. We summarize the control algorithm steps for one sampling time  $T_s$  in Algorithm 1.

**Algorithm 1** Main control algorithm**Inputs:**

Desired setpoint  $X_f$  from mission planner,  $X_f \leftarrow [x_f \ y_f \ \theta_f]^t$ ;

Initial position  $X_0$  from perception module,  $X_0 \leftarrow [x_0 \ y_0 \ \theta_0]^t$ ;

**Outputs:**

Actuator commands over horizon  $H$ ,  $\omega_j^{(i,k)}$ ,  $j = \overline{1..4}$ ,  $i = \overline{0..H-1}$

Predicted path over horizon  $H$ ,  $x^{(i,k)}$ ,  $y^{(i,k)}$ ,  $\theta^{(i,k)}$ ,  $i = \overline{0..H-1}$

**Runtime**

Acquire object list data: positions  $(x_o, y_o)$ , radius  $(r_o)$  from perception module;

Detect object boundaries from LiDAR data using Moore-Neighbor tracing algorithm with Jacob's stopping criteria [40]

$[B, L] = \text{bwboundaries}(\text{LiDAR data})$ ; (Matlab specific function)

$l = 0$ ;

**for**  $k \in \{1 \dots \text{length}(B)\}$  **do**

Object boundary  $\leftarrow B\{k\}$ ;  $B$  is Matlab cell data-type, therefore brackets are '{}' for indexing

Ignore objects composed of a very small or very high number of pixels (usually are artifacts or room boundaries)

**if**  $\text{Boundary}_{\text{Min}} \leq \text{numel}(\text{Object boundary}) / 2 \leq \text{Boundary}_{\text{Max}}$  **then**

$l \leftarrow l + 1$ ;

If the number of objects exceeds buffer size ( $\text{MaxNoObjs}$ ), an error will be thrown, and optimization will not be started

**if**  $l > \text{MaxNoObjs}$  **then**

$l \leftarrow -1$ ;

**break**;

$\overline{xy} \leftarrow \text{mean}(\text{Object boundary}) \in \mathbb{R}^{2 \times 1}$  Matlab specific function to determine mean value over each line of a matrix.

$x_o(l) \leftarrow \overline{xy}[2]$ ;

$y_o(l) \leftarrow \overline{xy}[1]$ ;

$r_o(l) \leftarrow \max(|\max(\text{Objectboundary}) - \min(\text{Objectboundary})|)$ ; Matlab specific functions to determine min, max values of matrix rows; or  $r_o$  provided by CNN subsystem;

$\text{noObjs} \leftarrow l$ ; No. of all objects detected in the map;

Determine relevant objects (within specified proximity  $d_{\text{max}}$ );

**for**  $k \in \{1 \dots \text{noObjs}\}$  No. of all objects **do**

Calculate distance to each relevant object:

$d_o \leftarrow \sqrt{(x_o[k] - x)^2 + (y_o[k] - y)^2}$ ;

**if**  $d_o \leq d_{\text{max}}$  **then**

Update object radius to include tolerance w.r.t to OMR dimensions

$r_o[k] \leftarrow r_o[k] + \max(l_x, l_y)$ ;

$n_o \leftarrow n_o + 1$ ;

Calculate reference trajectory  $x_r^{(i)}$ ,  $y_r^{(i)}$  according to Equation (25);

Calculate reference angle  $\theta_r^{(i)}$ ,  $i = \overline{0..H-1}$  according to Equation (24);

Input data to optimizer: Sampling time:  $T_s$ ; Object list:  $x_o, y_o, r_o$ ; Number of objects  $n_o$ ; Initial resting point  $X_0$ ; Final resting point  $X_f$ ; Run-time reference trajectory  $X_r$ ; Previous optimized commands  $\omega$ ;

Optimize;

Save in buffer the optimized commands;

Provide to the process inputs the first (i.e.,  $i = 0$ ) command from the- control buffer;

Figure 5 depicts the control structure consisting of two main subsystems: Environment perception, Model-Predictive Controller, and the interconnection with the psychological process.

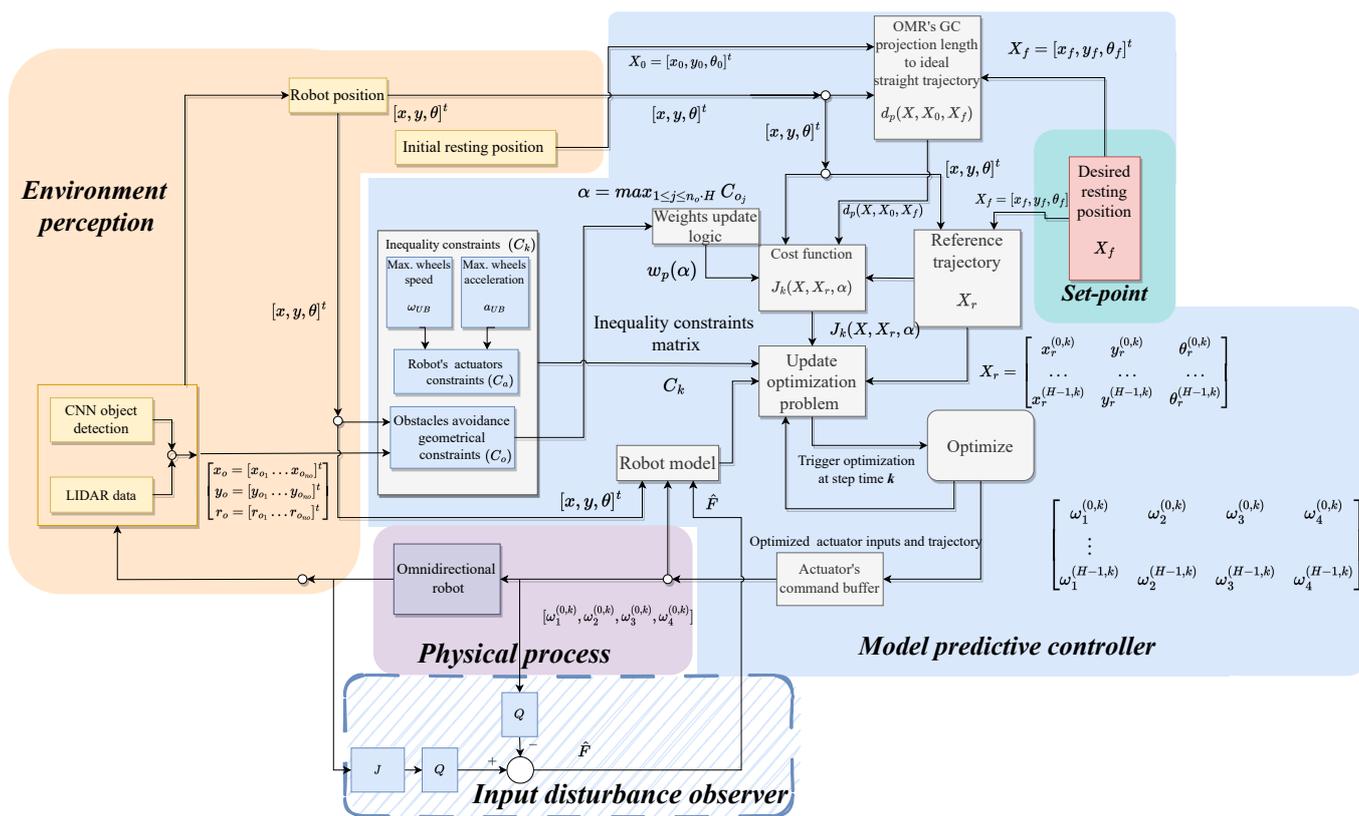


Figure 5. Illustrative block diagram of the control structure.

Figure 6 illustrates the main coordinates and notations used throughout the optimization problem. The projection  $d_p$  from the robot CG to the imaginary straight path connecting the initial  $X_0$  and final  $X_f$  resting locations is noticeable. Moreover, the L2-norms used in calculating the cost function,  $L_1$ ,  $L_2$ , and  $L_3$  define the distances between the OMR, initial, and resting positions.

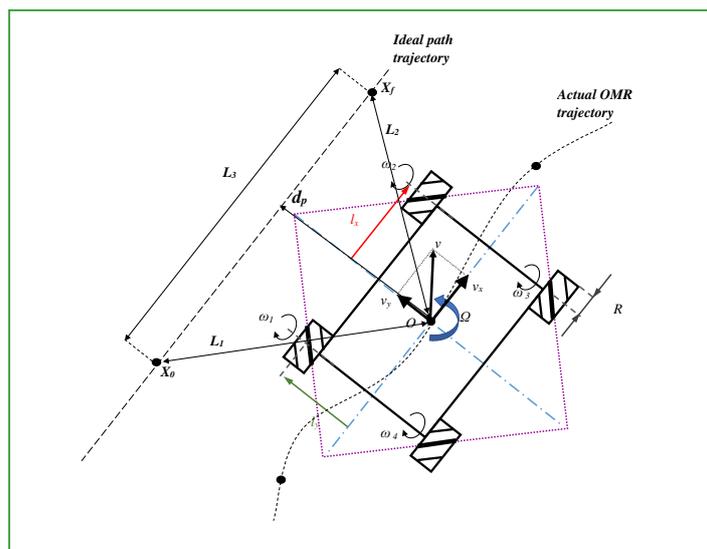


Figure 6. Coordinates system for control algorithm illustrating the used notations.

Table 5 contains the parameters of the model-predictive controller, including the penalizing factor of the cost function, the proximity threshold ( $tol$ ) for switching cost function weights, the radius w.r.t to OMR's CG to and the prediction horizon.

**Table 5.** Control parameters.

Parameter	Value
Cost weight 1 of reference trajectory ( $w_{xy1}$ )	0.6
Cost weight 1 of projection length to ideal path ( $w_{p1}$ )	0.01
Cost weight 2 of reference trajectory ( $w_{xy2}$ )	0.05
Cost weight 2 of projection length to ideal path ( $w_{p2}$ )	2.0
Cost weight of orientation angle ( $w_{\theta}$ )	0.3
Terminal cost weight of reference trajectory ( $w_{Tx}$ )	0.8
Terminal cost weight of reference trajectory ( $w_{Ty}$ )	0.8
Threshold for switching cost weights $tol$	−0.1 [m]
Maximum distance from CG to objects ( $d_{max}$ )	2.5 [m]
Prediction horizon ( $H$ )	10 [samples] ( $T_{horizon} = 0.2$ s)
Map cell size	10 [cm]

## 4. Results

### 4.1. Object Detection Results

The performance of the selected object detection solutions (ssd-mobilenet-v1, ssd-mobilenet-v2-lite, ssd-vgg16, and YoloV5) was evaluated on a testing subset with image resolutions varying between  $720 \times 404$  and  $2048 \times 1024$  pixels. The neural networks were tested on the Nvidia Jetson AGX mobile platform with the same input.

All models are optimized for Jetson Xavier AGX with the TensorRT framework from CUDA for Nvidia cards. The run time of the three selected architectures from the SSD family and the five main YoloV5 [29] is presented in Table 6. Architectures with fewer parameters performed better in terms of frames per second. Being the lightest model, the Nano YoloV5 is six times faster than the Extra Large model, the largest we considered for the Jetson platform. This highlights the importance of considering the specific hardware platform and the model's complexity for deploying object detection algorithms on mobile robots.

The two largest YoloV5 models did not bring any improvements for the overall precision and the precision per class compared to the Medium architecture; therefore, they were not considered for Table 6. A comparison between the precision of the models can be made based on the figures presented in Table 7. All architectures were trained for 150 epochs to evaluate the mean Average Precision. SSD Mobilenet v2 lite and SSD VGG16 reach a similar mAP@0.5 of 98–99%, while SSD Mobilenet v1 has a lower precision on the test subset, 86%.

Based on the results from Tables 6 and 7, we can draw the conclusion that the best model for our OMR object detection use cases is YoloV5 Medium which has a mAP comparable to SSD-VGG16, with the benefit of being twice as fast. Detection examples with the neural network models tested in the OMR environment are shown in Figure 7.

**Table 6.** Inference time.

Architecture	FPS on Jetson Xavier AGX	Number of Parameters
SSD Mobilenet v1	120	4.2M
SSD Mobilenet v2 lite	130	3.4M
SSD VGG16	50	35M
yoloV5 Nano	270	1.9M
yoloV5 Small	225	7.2M
yoloV5 Medium	109	21.2M
yoloV5 Large	72	46.5M
yoloV5 Extra Large	44	86.7M

Table 7. Average precision on test subset.

Model	Overall mAP@0.5	Conveyor Type 1	Conveyor Type 2	Per Class mAP@0.5		
				Rosy	Charging Station	Cone
SSD Mobilenet v1	0.860	0.909	0.891	0.816	0.717	0.969
SSD Mobilenet v2 lite	0.989	0.998	0.980	0.974	0.998	0.992
SSD VGG16	0.997	0.998	0.996	0.990	0.998	0.998
YoloV5 Nano	0.989	0.993	0.985	0.992	0.982	0.995
YoloV5 Small	0.992	0.995	0.989	0.994	0.99	0.995
YoloV5 Medium	0.994	0.995	0.992	0.995	0.991	0.995

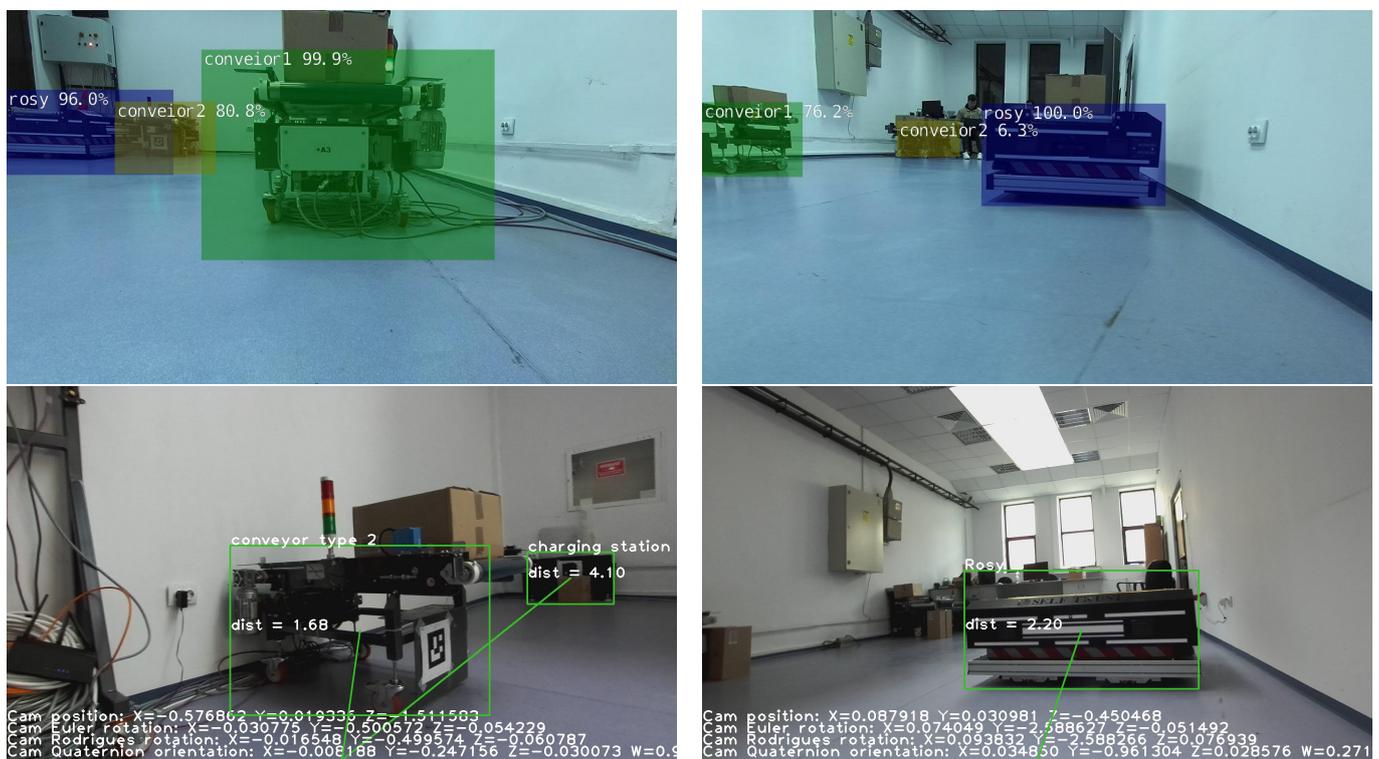


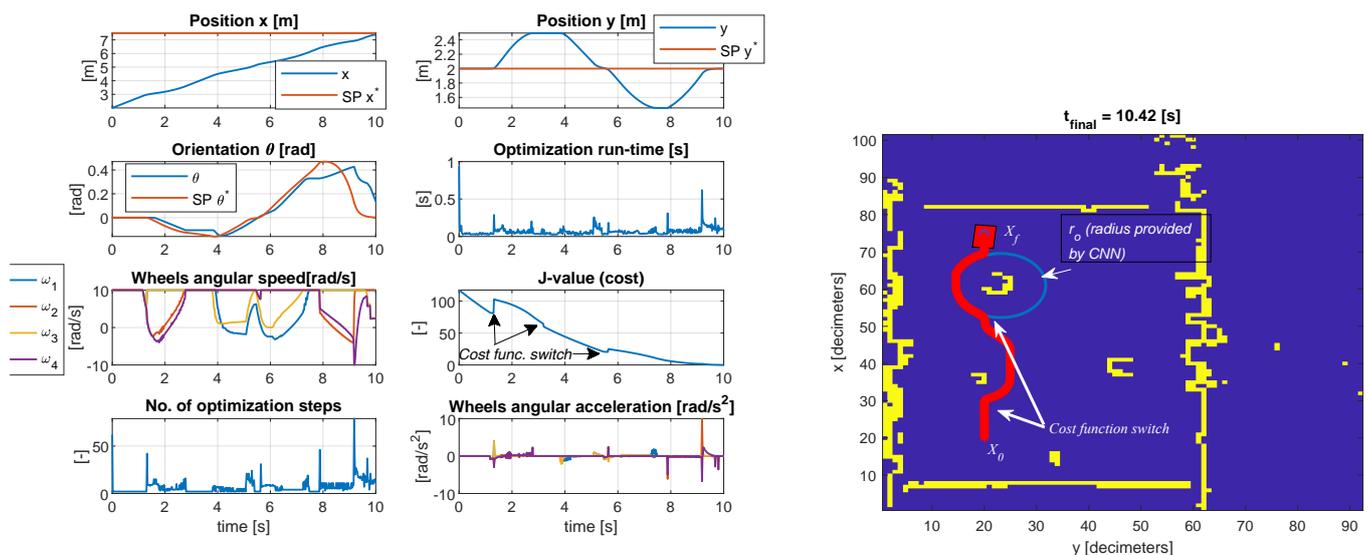
Figure 7. Detected objects with SSD architectures (1st row) and with YOLOv5 architecture (2nd row).

#### 4.2. Simulation Results

To evaluate the control performances, we considered scenarios where the initial and final positions varied throughout the room so that obstacles blocked the OMR path. We perform numerical simulations on real data acquired from the perception module. We evaluate the steady-state error, the possible constraint violations, the cost function, and the optimization run-time.

In the first test case considered in Figure 8, the final resting position  $X_f$  is reached after avoiding the two obstacles on the circumference of virtual circles centered around the objects. The inequality geometric inequality constrained  $C_o < 0$ , and the actuator constraints are satisfied  $C_a < 0$  with an acceptable tolerance. Generally, the tolerance is within the expected margin of  $1.0 \times 10^{-3}$ . The steady-state error of the controlled position  $(x, y)$  is less than 1% as measured around moment  $t = 10.2$  s. The transient time is limited by the upper and lower bounds of the wheel speed, in this case,  $\pm 10$  rad/s. The orientation  $\theta$  changes at each sample time as the vehicle travels towards  $X_f$ . Hence, the tracking is decent, with a peak error of 17 degrees noticeably on the roundabouts of the objects since the optimizer is more constrained. The cost function decreases as the

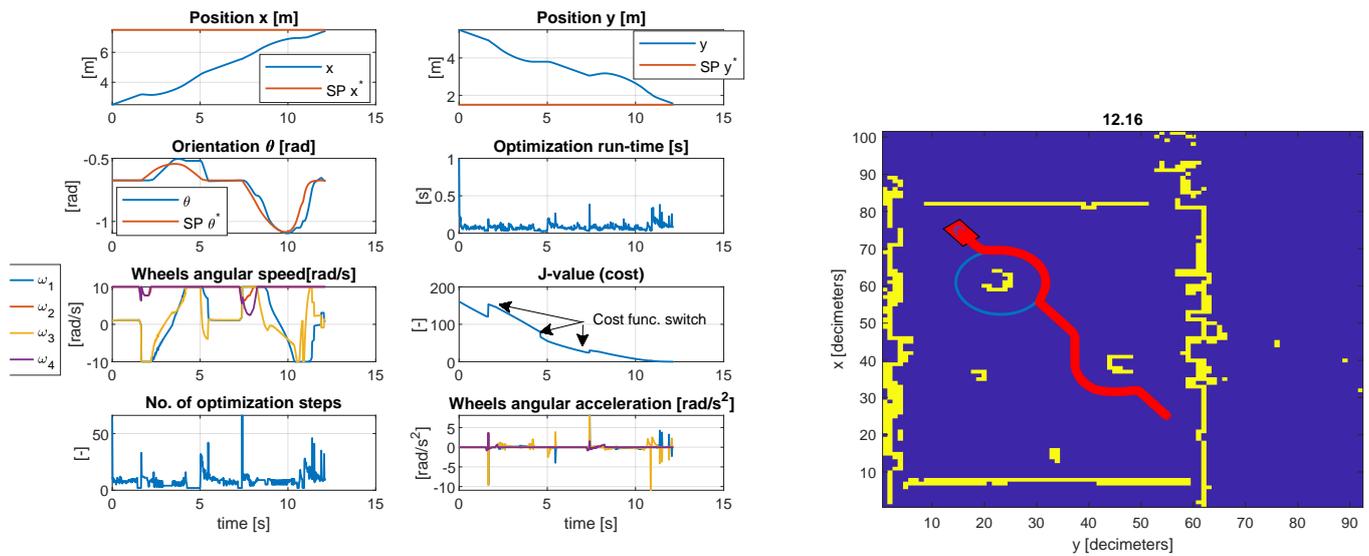
vehicle evolves across the map. In the proximity of an object, the cost function is purposely increased to avoid local minima by amplifying the deviation from the reference trajectory and decreasing the penalizing weight for the projection to the ideal path to allow solutions on the circumference of the encircled object. The maximum number of iterations was 79 with a run-time of 0.8945 s, and the minimum number of iterations was 2 with a run-time of 0.0204 s (CPU Intel i7 7500u, dual-core, 7th generation). The mean number of iterations was 6.526, with an execution time of 0.0647 s. It must be mentioned that the run-time is less relevant since in MEX mode (Matlab executable), the run-time can be reduced considerably (in MEX mode, the average run-time was 0.0507 s, while in normal mode 0.0647 s). The execution time is platform dependent.



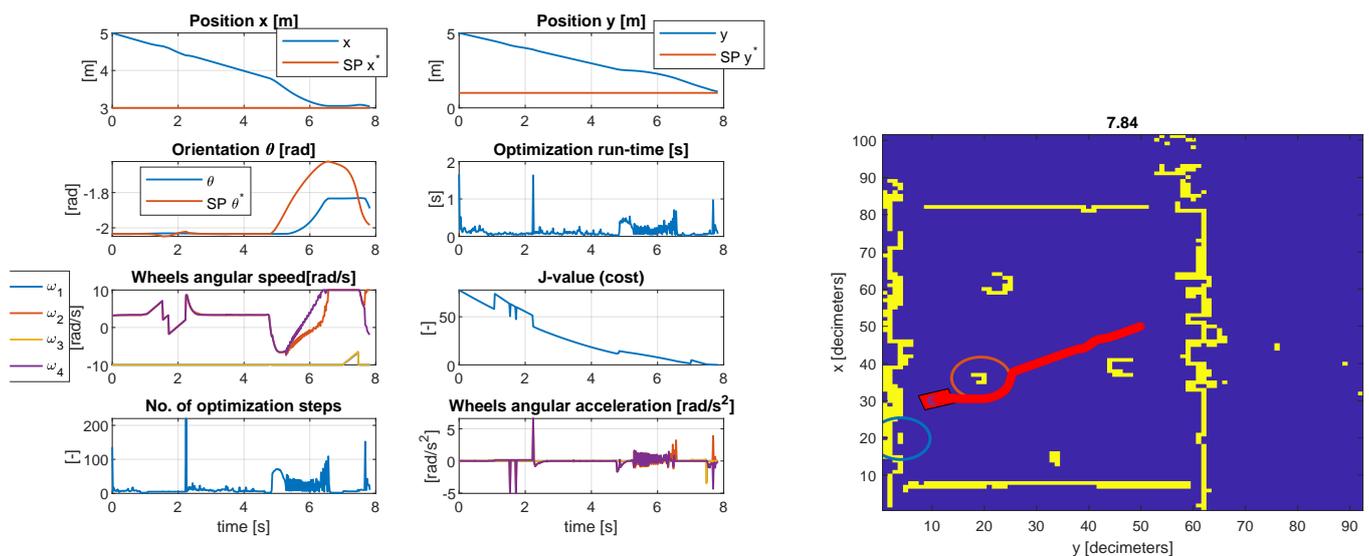
**Figure 8.** Simulation results of the model–predictive controller with LiDAR data and simulation of camera detection (test case I).

In the second scenario presented in Figure 9, the behavior is similar concerning the constraint tolerances. The violation of the object boundaries is within the expected limit, and the steady-state error of the controlled pose  $(x, y, \theta)$  is less than 1%. In this case, the actuator constraints limit the transient time,  $\pm 10$  rad/s. The maximum number of iterations was 66 with a run-time of 0.9923 s, and the minimum number of iterations was 2 with a run-time of 0.022 s (same CPU as mentioned in test case I). The mean number of iterations was 8.5658, with a mean execution time of 0.0814 s. In MEX mode, the maximum run-time was 0.5681 s, the minimum 0.0039 s, and the average 0.0356 s. Generally, the behavior is as expected, and the run-time proves the applicability of the control structure.

Similar behavior is obtained in the third test case presented in Figure 10, but the maximum run-time is slightly higher at 1.6 s, the maximum number of iterations is 210, and the minimum is 2. The minimum run-time was 0.0191 s. However, the average run-time in MEX mode is 0.0358 s with a maximum of 0.3176 s (instead of 1.6 s as in normal mode) and a minimum of 0.0043 s.



**Figure 9.** Simulation results of the model–predictive controller with LiDAR data and simulation of camera detection (test case II).



**Figure 10.** Simulation results of the model–predictive controller with LiDAR data and simulation of camera detection (test case III).

## 5. Conclusions and Future Work

The use of CNNs for object detection in mobile robot navigation provides benefits such as accuracy, robustness, and adaptability, which are desirable for the navigation of mobile robots in a logistic environment.

The paper proves the use of an object detector for a better understanding of the OMR working environment. To overcome this challenge, we also acquired a dataset for domain-specific object detection that was made public. It contains all objects of interest for the working environment, such as fixed or mobile conveyors, charging stations, other robots, and boundary cones. The results show a detection accuracy of 99% using the selected lightweight model, which was optimized to run on the available mobile platform already installed on the OMR at about 109 frames per second. The detection results offer a better understanding of the LiDAR map by assigning a name to obstacles and objects within the working environment, allowing the control model constraints to be adjusted on the fly.

This paper also demonstrates the model-predictive control of the OMR in logistic environments with actuator and geometric constraints. We avoid local minima by using variable cost function weights to navigate around obstacles while still achieving the overall objective of reducing travel distance. The execution runtime of the optimizer allows for practical implementation while the control performance is within the expected margin.

Future work is also expected to involve the deployment of the OMR controller and testing in a controlled environment and then in an automated logistic warehouse. One of the short-term goals is to collect and annotate more instances of domain-specific objects so that the intraclass variety is better covered and the detector can extrapolate on new data.

**Author Contributions:** Conceptualization, S.-D.A., R.M., A.-T.P. and C.-C.D.; methodology, S.-D.A., R.M., A.-T.P. and C.-C.D.; software, S.-D.A. and R.M.; validation, S.-D.A., R.M., A.-T.P. and C.-C.D.; formal analysis, S.-D.A. and R.M.; investigation, S.-D.A.; resources, S.-D.A.; data curation, S.-D.A.; writing—original draft preparation, S.-D.A. and R.M.; writing—review and editing, S.-D.A., R.M., A.-T.P. and C.-C.D.; visualization, S.-D.A. and R.M.; supervision, A.-T.P. and C.-C.D.; funding acquisition, S.-D.A. and R.M. All authors have read and agreed to the published version of the manuscript.

**Funding:** This paper was realized with the support of “Institutional development through increasing the innovation, development, and research performance of TUIASI–COMPETE”, project funded by contract no. 27PFE/2021, financed by the Romanian Government.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The obtained data set for object detection is publicly available.

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

## Abbreviations

The following abbreviations are used in this manuscript:

AP	Average Precision
APF	Artificial Potential Field
AR	Average Recall
CNN	Convolutional Neural Network
CPU	Central Processing Unit
FN	False Negative
FoV	Field of View
FP	False Positive
FPS	Frames Per Second
GPU	Graphics Processing Unit
GT	Ground Truth
IMU	Inertial Measurement Unit
IoU	Intersection over Union
IR	Infrared
LiDAR	Light Detection and Ranging
mAP	mean Average Precision
mAR	mean Average Recall
MEX	Matlab Executable
MS COCO	Microsoft Common Objects in COntext
OMR	Omnidirectional Mobile Robots
OROD	Omnidirectional Robot Object Detection
SP	Set Point
SSD	Single Shot Detector
TN	True Negative
TP	True Positive
TPU	Tensor Processing Unit
YOLO	You Only Look Once the algorithm

## References

1. Dosoftei, C.C.; Popovici, A.T.; Sacaleanu, P.R.; Gherghel, P.M.; Budaciu, C. Hardware in the Loop Topology for an Omnidirectional Mobile Robot Using Matlab in a Robot Operating System Environment. *Symmetry* **2021**, *13*, 969. [[CrossRef](#)]
2. Rubio, F.; Valero, F.; Llopis-Albert, C. A review of mobile robots: Concepts, methods, theoretical framework, and applications. *Int. J. Adv. Robot. Syst.* **2019**, *16*, 1729881419839596. [[CrossRef](#)]
3. Patle, B.; Babu L, G.; Pandey, A.; Parhi, D.; Jagadeesh, A. A review: On path planning strategies for navigation of mobile robot. *Def. Technol.* **2019**, *15*, 582–606. [[CrossRef](#)]
4. Ngwenya, T.; Ayomoh, M.; Yadavalli, S. Virtual Obstacles for Sensors Incapacitation in Robot Navigation: A Systematic Review of 2D Path Planning. *Sensors* **2022**, *22*, 6943. [[CrossRef](#)] [[PubMed](#)]
5. Wang, Z.; Tian, G. Hybrid offline and online task planning for service robot using object-level semantic map and probabilistic inference. *Inf. Sci.* **2022**, *593*, 78–98. [[CrossRef](#)]
6. Tătar, M.O.; Popovici, C.; Măndru, D.; Ardelean, I.; Pleșa, A. Design and development of an autonomous omni-directional mobile robot with Mecanum wheels. In Proceedings of the 2014 IEEE International Conference on Automation, Quality and Testing, Robotics, Cluj-Napoca, Romania, 22–24 May 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 1–6.
7. Sheikhlar, A.; Fakharian, A. Adaptive optimal control via reinforcement learning for omni-directional wheeled robots. In Proceedings of the 2016 4th International Conference on Control, Instrumentation, and Automation (ICCIA), Qazvin, Iran, 27–28 January 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 208–213.
8. Popovici, A.T.; Dosoftei, C.C.; Budaciu, C. Kinematics Calibration and Validation Approach Using Indoor Positioning System for an Omnidirectional Mobile Robot. *Sensors* **2022**, *22*, 8590. [[CrossRef](#)] [[PubMed](#)]
9. Peng, T.; Qian, J.; Zi, B.; Liu, J.; Wang, X. Mechanical design and control system of an omni-directional mobile robot for material conveying. *Procedia CIRP* **2016**, *56*, 412–415. [[CrossRef](#)]
10. Lu, X.; Zhang, X.; Zhang, G.; Jia, S. Design of adaptive sliding mode controller for four-Mecanum wheel mobile robot. In Proceedings of the 2018 37th Chinese Control Conference (CCC), Wuhan, China, 25–27 July 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 3983–3987.
11. Sun, Z.; Xie, H.; Zheng, J.; Man, Z.; He, D. Path-following control of Mecanum-wheels omnidirectional mobile robots using nonsingular terminal sliding mode. *Mech. Syst. Signal Process.* **2021**, *147*, 107128. [[CrossRef](#)]
12. Lu, X.; Zhang, X.; Zhang, G.; Fan, J.; Jia, S. Neural network adaptive sliding mode control for omnidirectional vehicle with uncertainties. *ISA Trans.* **2019**, *86*, 201–214. [[CrossRef](#)]
13. Zimmermann, K.; Zeidis, I.; Abdelrahman, M. Dynamics of mechanical systems with mecanum wheels. In *Applied Non-Linear Dynamical Systems*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 269–279.
14. Santos, J.; Conceição, A.G.; Santos, T.L. Trajectory tracking of omni-directional mobile robots via predictive control plus a filtered smith predictor. *IFAC-PapersOnLine* **2017**, *50*, 10250–10255. [[CrossRef](#)]
15. Conceição, A.S.; Oliveira, H.P.; e Silva, A.S.; Oliveira, D.; Moreira, A.P. A nonlinear model predictive control of an omni-directional mobile robot. In Proceedings of the 2007 IEEE International Symposium on Industrial Electronics, Vigo, Spain, 4–7 June 2007; IEEE: Piscataway, NJ, USA, 2007; pp. 2161–2166.
16. Wang, D.; Wei, W.; Yeboah, Y.; Li, Y.; Gao, Y. A robust model predictive control strategy for trajectory tracking of omni-directional mobile robots. *J. Intell. Robot. Syst.* **2020**, *98*, 439–453. [[CrossRef](#)]
17. Teatro, T.A.V.; Eklund, J.M.; Milman, R. Nonlinear model predictive control for omnidirectional robot motion planning and tracking with avoidance of moving obstacles. *Can. J. Electr. Comput. Eng.* **2014**, *37*, 151–156. [[CrossRef](#)]
18. Wang, C.; Liu, X.; Yang, X.; Hu, F.; Jiang, A.; Yang, C. Trajectory tracking of an omni-directional wheeled mobile robot using a model predictive control strategy. *Appl. Sci.* **2018**, *8*, 231. [[CrossRef](#)]
19. Rosenfelder, M.; Ebel, H.; Eberhard, P. Cooperative distributed nonlinear model predictive control of a formation of differentially-driven mobile robots. *Robot. Auton. Syst.* **2022**, *150*, 103993. [[CrossRef](#)]
20. Zhang, H.; Wang, S.; Xie, Y.; Wu, H.; Xiong, T.; Li, H. Nonlinear Model Predictive Control of an Omnidirectional Mobile Robot with Self-tuned Prediction Horizon. In Proceedings of the 2022 IEEE 17th Conference on Industrial Electronics and Applications (ICIEA), Chengdu, China, 16–19 December 2022.
21. Ren, C.; Li, C.; Hu, L.; Li, X.; Ma, S. Adaptive model predictive control for an omnidirectional mobile robot with friction compensation and incremental input constraints. *Trans. Inst. Meas. Control* **2022**, *44*, 835–847. [[CrossRef](#)]
22. Santos, J.C.; Gouttefarde, M.; Chemori, A. A nonlinear model predictive control for the position tracking of cable-driven parallel robots. *IEEE Trans. Robot.* **2022**, *38*, 2597–2616. [[CrossRef](#)]
23. Kim, H.; Kim, B.K. Minimum-energy trajectory generation for cornering with a fixed heading for three-wheeled omni-directional mobile robots. *J. Intell. Robot. Syst.* **2014**, *75*, 205–221. [[CrossRef](#)]
24. Ge, S.S.; Cui, Y.J. New potential functions for mobile robot path planning. *IEEE Trans. Robot. Autom.* **2000**, *16*, 615–620. [[CrossRef](#)]
25. Li, Z.; Zhao, S.; Duan, J.; Su, C.Y.; Yang, C.; Zhao, X. Human cooperative wheelchair with brain–machine interaction based on shared control strategy. *IEEE/ASME Trans. Mechatron.* **2016**, *22*, 185–195. [[CrossRef](#)]
26. Rösmann, C.; Hoffmann, F.; Bertram, T. Timed-elastic-bands for time-optimal point-to-point nonlinear model predictive control. In Proceedings of the 2015 European Control Conference (ECC), Linz, Austria, 15–17 July 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 3352–3357.

27. Rösmann, C.; Hoffmann, F.; Bertram, T. Integrated online trajectory planning and optimization in distinctive topologies. *Robot. Auton. Syst.* **2017**, *88*, 142–153. [[CrossRef](#)]
28. Smith, J.S.; Xu, R.; Vela, P. egoteb: Egocentric, perception space navigation using timed-elastic-bands. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 2703–2709.
29. Glenn, J.; Alex, S.; Jirka, B. *YOLOv5*; Zenodo: Geneva, Switzerland, 2020.
30. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4510–4520. [[CrossRef](#)]
31. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:1704.04861.
32. Liu, S.; Deng, W. Very deep convolutional neural network based image classification using small training sample size. In Proceedings of the 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR), Kuala Lumpur, Malaysia, 3–6 November 2015; pp. 730–734. [[CrossRef](#)]
33. Bochkovskiy, A.; Wang, C.; Liao, H.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934.
34. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Proceedings of the Advances in Neural Information Processing Systems*; Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2019; Volume 32.
35. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.E.; Fu, C.; Berg, A.C. SSD: Single Shot MultiBox Detector. *arXiv* **2015**, arXiv:1512.02325.
36. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 1–9. [[CrossRef](#)]
37. Lin, T.; Maire, M.; Belongie, S.J.; Bourdev, L.D.; Girshick, R.B.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common Objects in Context. *arXiv* **2014**, arXiv:1405.0312.
38. Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; Schiele, B. The Cityscapes Dataset for Semantic Urban Scene Understanding. *arXiv* **2016**, arXiv:1604.01685.
39. Cordts, M.; Omran, M.; Ramos, S.; Scharwächter, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; Schiele, B. The Cityscapes Dataset. In Proceedings of the CVPR Workshop on the Future of Datasets in Vision, Boston, MA, USA, 11 June 2015.
40. Seo, J.; Chae, S.; Shim, J.; Kim, D.; Cheong, C.; Han, T.D. Fast contour-tracing algorithm based on a pixel-following method for image sensors. *Sensors* **2016**, *16*, 353. [[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.