electronics

MDPI

Article

Optimizing an Autonomous Robot's Path to Increase Movement Speed

Damian Gorgoteanu [1], Cristian Molder [1], Vlad-Gabriel Popescu [1], Lucian Ștefăniță Grigore [1] and Ionica Oncioiu [2,*]

[1] Center of Excellence in Robotics and Autonomous Systems—CERAS, "Ferdinand I" Military Technical Academy, 050141 Bucharest, Romania; damian.gorgoteanu@mta.ro (D.G.); cristian.molder@mta.ro (C.M.); gabriel.popescu@mta.ro (V.-G.P.); lucian.grigore@mta.ro (L.Ș.G.)

[2] Department of Informatics, Faculty of Informatics, Titu Maiorescu University, 040051 Bucharest, Romania

* Correspondence: ionica.oncioiu@prof.utm.ro; Tel.: +40-372-710-962

Abstract: The goal of this study is to address the challenges associated with identifying and planning a mobile land robot's path to optimize its speed in a stationary environment. Our focus was on devising routes that navigate around obstacles in various spatial arrangements. To achieve this, we employed MATLAB R2023b for trajectory simulation and optimization. On-board data processing was conducted, while obstacle detection relied on the omnidirectional video processing system integrated into the robot. Odometry was facilitated by engine encoders and optical flow sensors. Additionally, an external video system was utilized to verify the experimental data pertaining to the robot's movement. Last but not least, the algorithms and hardware equipment used enabled the robot to go along the path at greater speeds. Limiting the amount of time and energy required to travel allowed us to avoid obstacles.

Keywords: path planning; mobile robot; omnidirectional camera; optical flux sensors

Citation: Gorgoteanu, D.; Molder, C.; Popescu, V.-G.; Grigore, L.Ș.; Oncioiu, I. Optimizing an Autonomous Robot's Path to Increase Movement Speed. *Electronics* **2024**, *13*, 1892. https://doi.org/10.3390/electronics13101892

Academic Editors: Zhongguo Li and Zhongchao Liang

Received: 6 April 2024
Revised: 30 April 2024
Accepted: 9 May 2024
Published: 11 May 2024

1. Introduction

Planning the trajectory of an autonomous mobile robot is rather challenging because the existing planning methods require long reaction times [1]. This can clearly be seen in situations in which the number of robots increases because, at present, there is little knowledge of scalability regarding the determination of trajectories when the obstacles are not static.

Although the endowment of robots with the capability to carry out autonomous movements and actions is advantageous, it must be noted that route planners are required [2]. These planners [3–5] allow robots to make decisions about how they move from one location to another. This requires the use of route planning algorithms based on several criteria: requirements imposed by users [6–8], the multitude of obstacles that may appear (static or dynamic) [9–12], the energy resources available [13,14], the on-board capabilities for processing sensor data [15,16], and the communication system [17,18].

The principles of the autonomous navigation of mobile robots involve finding the best route, i.e., finding the optimal path [19]. Therefore, the initial state and the final state must be taken into account. Optimization refers to the fact that algorithms minimize one or more objective functions. For example, in the case of robots that intervene in human search and rescue missions, the objective function is determined by the priorities set by the operator [20]. Another objective can be generated if the robot must perform missions that require mapping over large spaces; the criterion can be determined through the management of the energy resource [21]. Otherwise, if the robot has to travel along a route with obstacles, a criterion can be the shortest path between two landmarks (obstacles) [22,23].

The robot's locomotion system and the characteristics of the terrain are the elements that must be taken into account when the planner is considering the limitations that affect

the robot's progress [24–26]. These elements are likely to introduce the most limitations when identifying optimal travel paths.

Autonomous land mobile robots rely on localization and mapping models to navigate their surroundings. The process of localization involves breaking down the kinematics and dynamics problems into discrete points along the path, adhering to specific constraints [27,28].

One of the methods used is the collocation method, which is based on the differential equations that describe the dynamic behavior of the robots to generate their trajectories [29,30]. In this case, trajectory generation constitutes an optimal control problem where allocation is performed only at the collocation points. To generate a good trajectory, there must be a large enough number of these points, because there is a possibility of collision when the robot moves from one reference point to another; however, the number of points should not be so large that it would substantially increase the computation time.

Another alternative approach to trajectory planning involves two distinct modes of planning: deliberative and reactive. Deliberative methods require more computational resources and time, as they need to consider all possible options and their combinations [31–34]. These methods are suitable for static environments. On the other hand, reactive methods rely on instinctive problem-solving and generate immediate movements. While reactive decision-making allows for fast route planning, it does not guarantee accuracy or optimal routes. The robot may get stuck and oscillate, sometimes moving in a loop. However, reactive algorithms excel in their quick calculations and adaptability to the environment. The hybrid use of the two algorithms may represent a solution, with the deliberative algorithm being the one that coordinates the reactive one.

Different approaches to route planning can be categorized as online or offline methods, depending on the scope of the workspace. Online planning is associated with local environments, while offline planning is associated with global environments [35–37]. To navigate effectively, it is crucial to have a method for mapping the workspace. There are various mapping methods available, including metric, topological, and semantic approaches.

Although there are tools and methodologies for constructing metric maps and, to some extent, topological graphs, environmental information must be provided to the robot in an appropriate format by the operator or a specialist [38–40]. This is not satisfactory if symbolic and semantic knowledge are acquired incrementally or if this process is an interpretation of a metric map. Neither the occupancy grid nor the topological graph provides a good means of adding semantic knowledge [41]. Since it has a fine discretization, which is good for a rich description of structural details, the occupancy grid is not easily translated into a symbolic representation, as is necessary for a form of high-level spatial reasoning [42].

Metric or quantitative mapping is based on measurements of the environment and includes dimensional quantities (length, width). Trajectory planning in metric spaces allocates (x,y) coordinates, and the movements are simple: move forward, turn left/right, etc. Metric mapping does not provide complete information on the existence or nonexistence of obstacles, and only (x,y) coordinates are obtained. These data are interpreted as obstacles to be avoided based on a predetermined algorithm. The sensors used are proximity sensors, including ultrasonic, infrared, LiDAR, and video cameras. At the heart of metric mapping is an artificial neural network that transforms measured values into occupancy values. Once trained, the network generates values that can be interpreted as occupancy values [43].

Topological mapping is more strongly based on landmarks; the map is a representation of the global position, as it is centered on the robot using topological organization [44]. Topological navigation and mapping are not based on precise measurements; they involve creating maps where the position of landmarks is essential but the distance between them is not. Topological mapping is qualitative, unlike metric mapping, which is quantitative. Topological maps represent a type of connectivity in the form of a graphical structure where "vertices" represent distinct points/places and "edges" represent routes [45]. Depending on the algorithms used, topological maps supplement their information with data from metric mappings. Topological navigation, in addition to path length estimations, introduces an odometry error model [46].

Semantic mapping is a process of data acquisition, data processing, and data representation. The process consists of adding semantic meanings to the mapped features. Semantic data sources are based on the following processes: extracting features from sensor data, incorporating data through HRI (human–robot interaction), and extracting data from a database representing model concepts. All this is represented in a structured manner (an organization of geometric data), so that the robot is able to understand the environment. The semantic mapping process is based on three acquisition methods: perception, HRI, and reasoning. The geometric representation is obtained using SLAM (simultaneous localization and mapping) methods, which consist of the simultaneous estimation of the state of the robot and the realization of the semantic model.

The aim of the present work is to identify the optimal solution in terms of size, computing power, the need for sensors, and the communication system for the realization of a terrestrial mobile robot, so that we can optimize its movement speed in a static environment. We also aimed to increase the efficiency with which the movement trajectory of mobile robots can be planned from the initial position to the final position in an unknown environment with obstacles. To achieve this, it was assumed that the designed/realized robot has a limited capacity in terms of sensors and computing capacity. We improved the performance of the omnidirectional vision system in order to increase the reaction speed of the robot [47].

The main objectives to achieve the proposed goal were as follows:

- Conduct research and critical analyses of models and systems used by other similar robots;
- Research and conduct a bibliographic analysis related to the testing of theoretical and experimental research procedures that lead to relevant conclusions regarding the robot's levels of mobility and stability;
- Conducting research on the use of omnidirectional cameras for terrain mapping;
- Researching directions for software and hardware optimization in order to optimize the travel speed, based on an analysis of data obtained from an omnidirectional camera;
- Improving the algorithms regarding the movement of the robot with the help of optical flow sensors;
- Improving the characteristics of the sensor system based on the omnidirectional camera;
- Implementing a mesh communication system;
- Achieving trajectory optimization when the robot encounters obstacles.

To optimize the trajectory, predetermined routes were iterated multiple times to accurately determine the robot's current position and its proximity to obstacles. The method proposed involves analyzing the robot's potential movement in specific directions using analytical geometry, utilizing data from proximity and optical sensors. The main goal was to select an intermediate state and adjust the trajectory based on sensor measurements. This approach enabled the robot to navigate through unfamiliar environments while maintaining a safe distance from obstacles.

The aim of this research paper was to develop and test a mobile robot with omnidirectional visual perception, with the following objectives:

- The design and development of a NEXUS 2 mobile robot platform for its integration into a robot swarm;
- Conducting a comparative analysis of displacement sensors and their calibration;
- Conducting tests to determine a viable travel data fusion solution;
- Identifying the solution to generate a map based on the images acquired with the omnidirectional camera;
- Testing the integrated system using the Jetson Nano development platform;
- Conducting system optimization for real-time work;
- Optimizing data transmission through the communication system.

The most important contributions of this paper are as follows:

- Studying existing robot architectures that are capable of being integrated into, for example, robot swarms;

- Identifying the advantages and disadvantages of different platforms, while also analyzing the purchase price;
- Achieving the design and realization of a new platform with a modular architecture that allows for independent operations with a basic system controlled by the PIC18LF46K22 microcontroller;
- Designing and implementing an on-board computing system of the SBC type;
- Designing and realizing a positioning system based on the fusion of data from speed and optical flow transducers;
- Designing and making a communication system based on a mesh network;
- Designing and making an omnidirectional vision system with a convex mirror for determining the positions of obstacles;
- Realizing a monobloc architecture that facilitates a simple tiered design, on which the main video processing unit is located;
- Creating hardware support for the communication system, which allows for the transmission of data to a base station and, in the future, to other robots within a swarm of robots.

The paper is structured as follows: Section 2 discusses how the robot was built, the kinematics of the propulsion system, and the omnidirectional image sensor. Section 3 presents aspects related to the process of conducting a simulation of the image processing algorithm, the principles of image binarization, the experimental results and measurements related to the displacement error, the testing of the data transmission algorithm, and the path planning algorithm. In Section 4, the conclusions are outlined along with potential avenues for further development.

2. Materials and Methods

2.1. The NEXUS-2 Robot

The NEXUS-2 robot was designed and built with a modular chassis (Figure 1), which can be adapted for different situations. One of the requirements was that this robot could, in the future, be integrated into a swarm of robots.

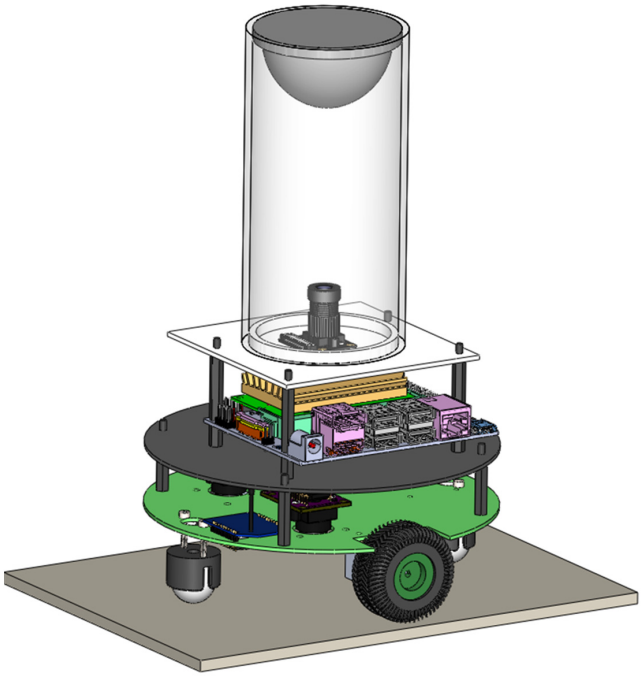

Figure 1. Robot NEXUS-2 version 2.

The modular structure consists of two main elements:

- A basic module with reduced computing power, but which is able to exist and function independently and is capable of executing tasks autonomously and also when commanded;
- A video processing system based on a Jetson-Nano-type development board, a board that is integrated with the base module to realize the movement and to analyze the movement data.

The hardware platform (Figures 2 and 3) of the robot consists of the following elements:

- The computing unit, for which the basic component is a PIC18LF46K22 microcontroller that has the role of managing all the sensors present on the module and ensuring the movement of the platform. Microcontroller programming was conducted using MicroE's MikroC PRO for PIC;
- The power supply module, which includes a lithium–polymer-type battery and two voltage regulators. The voltage generated by the battery of 3.7 V is converted into a voltage of 5 V using a voltage-raising circuit, which is then applied to the current motors continuously; the circuit used to achieve this was the LM2577-Adj, which has an operating frequency of 52 kHz and a maximum current of 3A;
- The propulsion module, which is composed of two direct current brush motors with a mechanical reducer and a transducer for determining displacement;
- The interface module, which consists of an OLED screen controlled via SPI with a resolution of 128 × 64 pixels;
- Sensors: an IR distance sensor, GP2Y0A21YK; an optical flow sensor, ADNS 3080;
- The communication module, which is composed of a radio transmitter and receiver device based on the ZigBee radio wireless protocol, DigiXBee.

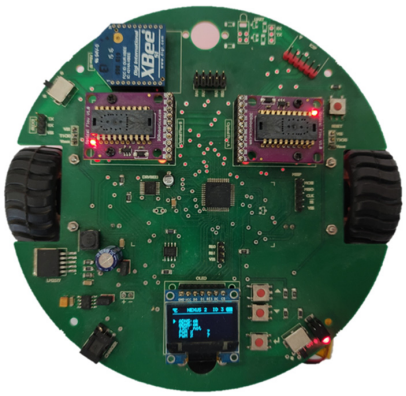

Figure 2. NEXUS-2 robot hardware.

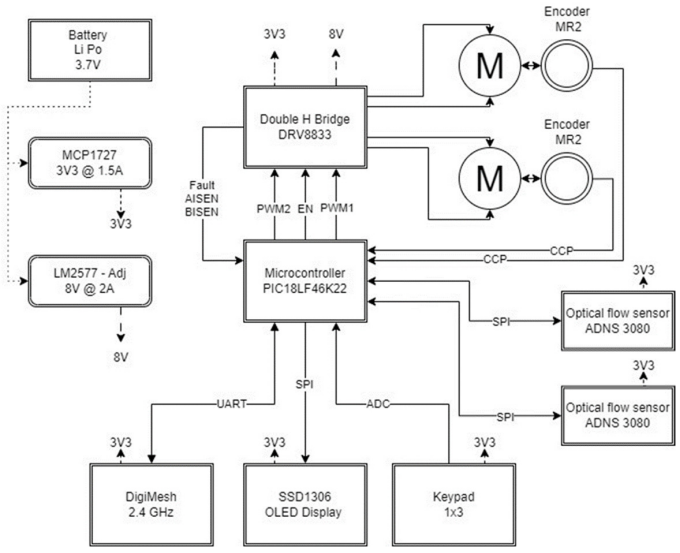

Figure 3. Block diagram of the basic module.

2.2. Kinematics of the NEXUS-2 Robot

As the robot is small, it was preferable for the motors to use a direct current with brushes without an iron core and to have a mechanical reducer and a position transducer.

The classical kinematic model of a wheeled vehicle [48–51] is produced based on the following assumptions:

- Dynamics elements are neglected: forces, moments, and friction;
- The coordinate system is in the plane $\{2D \mid Z = 0\}$,
- where D—the reference system in two dimensions,
- $Z = 0$—coordinate after Z if we are talking about the three dimensions XYZ;
- The lateral dimension of the vehicle is considered to be zero;
- The wheels are symmetrical.

During planning, the position of the robot is determined by considering the connection between the overall reference frame of the plan and the local reference frame (Figure 4). The global reference frame defines the position of point P using the x and y coordinates, while the angular disparity between the global and local reference frames is indicated by θ.

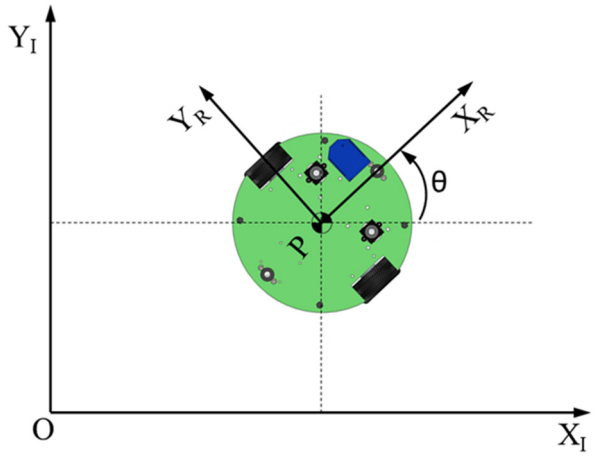

Figure 4. The global reference frame and the local reference frame of the robot.

According to Figure 4, we can describe the position of the robot as a vector:

$$\xi_I = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}. \tag{1}$$

To analyze the movement of the robot by breaking it down into individual motions, it is essential to align the robot with the axes of the global reference frame while considering its local reference system. This alignment is accomplished through the utilization of the orthogonal rotation matrix:

$$R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2}$$

The matrix in (2) can be used to map the motion into the global reference frame $\{X_I, Y_I\}$ to movement in terms of the local reference $\{X_R, Y_R\}$. This operation is given by $R(\theta) \cdot \dot{\xi}_I$, because the calculation depends on the following value:

$$\dot{\xi}_R = R\left(\frac{\pi}{2}\right) \cdot \dot{\xi}_I = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{y} \\ -\dot{x} \\ \dot{\theta} \end{bmatrix}. \tag{3}$$

The analyzed robot has a differential drive on each wheel. Given a point centered between the two drive wheels, each wheel is at a distance of l from point P. Given the robot elements (r, l, θ) and the rotational speed of each wheel $(\dot{\varphi}_1, \dot{\varphi}_2)$ and a kinematic model, which allows for the determination of the absolute speed of the robot with respect to the global reference frame, we determine that (1) becomes:

$$\dot{\xi}_I = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = f(r, l, \theta, \dot{\varphi}_1, \dot{\varphi}_2). \tag{4}$$

The two wheels are fixed and do not have a vertical axis of rotation for steering. The angle between the wheels and the chassis remains constant, allowing only forward and backward movement along the wheel's plane. Figure 5 provides a depiction of the typical fixed wheel, with its position indicated in relation to the robot's local reference frame $\{X_R, Y_R\}$.

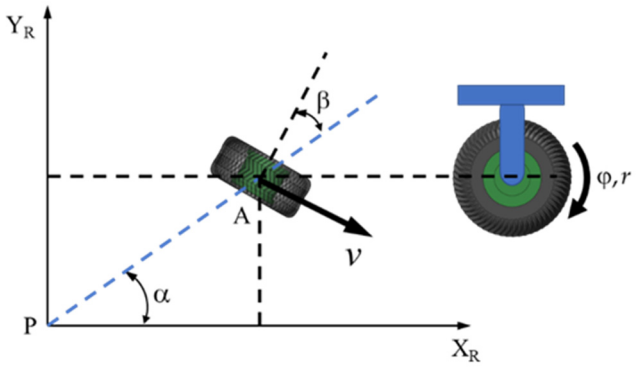

Figure 5. A fixed standard wheel and its parameters.

The rolling constraint for this wheel requires that all movement along the direction of the wheel plane be accompanied by the appropriate amount of rolling motion of the wheel, so that there is pure rolling at the point of contact with the tread:

$$\begin{bmatrix} \sin(\alpha + \beta) & -\cos(\alpha + \beta) & (-l) \cdot \cos\beta \end{bmatrix} \cdot R(\theta) \cdot \dot{\xi}_I - r \cdot \dot{\varphi} = 0. \tag{5}$$

The first term of the sum describes the total motion along the plane of the wheel. The three elements of the left vector $\left(\dot{x}, \dot{y}, \dot{\theta}\right)$ represent mappings from each of their contributions for motion along the plane of the wheel. Deadline $R(\theta) \cdot \dot{\xi}_I$ is used to transform the motion parameters $\dot{\xi}_I$, which are in the global reference frame $\{X_I, Y_I\}$ in motion parameters in the local reference frame $\{X_R, Y_R\}$. This is necessary because all other parameters in Equation (α, β, l) are relative to the robot's local reference frame. This movement along the plane of the wheel must be equal, according to this constraint, to the movement achieved by turning the wheel $(r \cdot \varphi)$.

Thus, in order to meet the slip constraint for this wheel, it is necessary for the component of wheel movement perpendicular to the wheel's plane to be null:

$$\begin{bmatrix} \cos(\alpha + \beta) & \sin(\alpha + \beta) & l \cdot \sin\beta \end{bmatrix} \cdot R(\theta) \cdot \dot{\xi}_I = 0. \tag{6}$$

Due to the position of the sensors relative to the robot's center of rotation, it was necessary to determine the robot's displacement. To achieve this, it was necessary to transform the coordinates from the real coordinate system to the robot's coordinate system and then to the sensor's coordinate system (Figure 6).

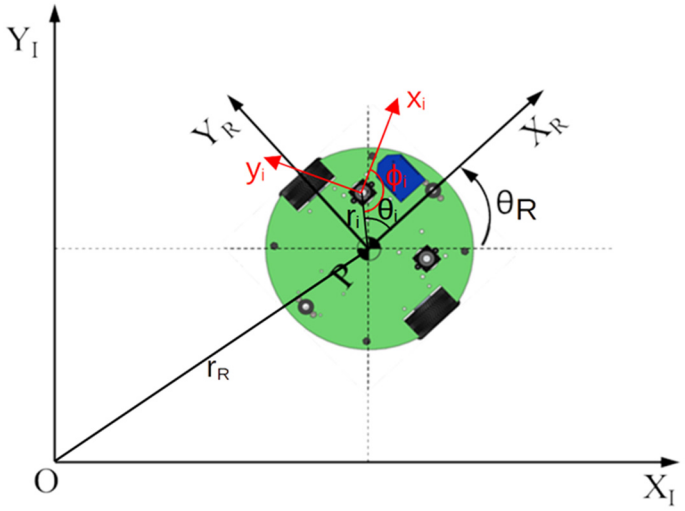

Figure 6. Coordinate system of the ADNS 3080 flow sensors.

The first is the absolute coordinate system, which is designated by the Cartesian coordinates $\{X_I, Y_I\}$. The second is the reference system of the robot, which is specified by $\{X_R, Y_R\}$. The final reference system is aligned with the measuring axes of each sensor and is designated $\{x_i, y_i\}$.

If the robot performs a translation ΔX_R, the sensor moves with ΔX_R, but, because of its relative position to the center of the robot, the measured values on the axes $\{x_i, y_i\}$ are as follows:

$$\begin{cases} x_i = \Delta X_R \cdot \sin(\theta_i + \phi_i) \\ y_i = \Delta X_R \cdot \cos(\theta_i + \phi_i) \end{cases}. \tag{7}$$

If the robot performs the translation ΔY_R, then we obtain the following:

$$\begin{cases} x_i = -\Delta Y_R \cdot \cos(\theta_i + \phi_i) \\ y_i = \Delta Y_R \cdot \sin(\theta_i + \phi_i) \end{cases}. \tag{8}$$

If the robot performs a rotation around its axis by an angle $\Delta\omega$, where ω is represented in radians, then the displacement of the sensor is $\Delta\omega \cdot r_i$:

$$\begin{cases} x_i = -\Delta\omega \cdot r_i \cdot \cos\phi_i \\ y_i = \Delta\omega \cdot r_i \cdot \sin\phi_i \end{cases}. \tag{9}$$

By summing relations (7), (8), and (9), we obtain the total displacement of the sensor on the two axes:

$$\begin{cases} x_i = \Delta X_R \cdot \sin(\theta_i + \phi_i) - \Delta Y_R \cdot \cos(\theta_i + \phi_i) - \Delta\omega \cdot r_i \cdot \cos\phi i \\ y_i = \Delta X_R \cdot \cos(\theta_i + \phi_i) + \Delta Y_R \cdot \sin(\theta_i + \phi_i) + \Delta\omega \cdot r_i \cdot \sin\phi i \end{cases}. \tag{10}$$

2.3. The Image Sensor

The image sensor is actually an omnidirectional camera, which provides a field of view greater than 1800. The camera used is of the catadioptric type, which combines a standard camera with a convex mirror (Figure 7), so that a 3600 image is obtained.

Figure 8 [52] presents an image made using the catadioptric system used in the creation of the robot; it also shows the perspective of the image via a cylindrical projection on the image.

Figure 7. The catadioptric system of the NEXUS 2 robot.

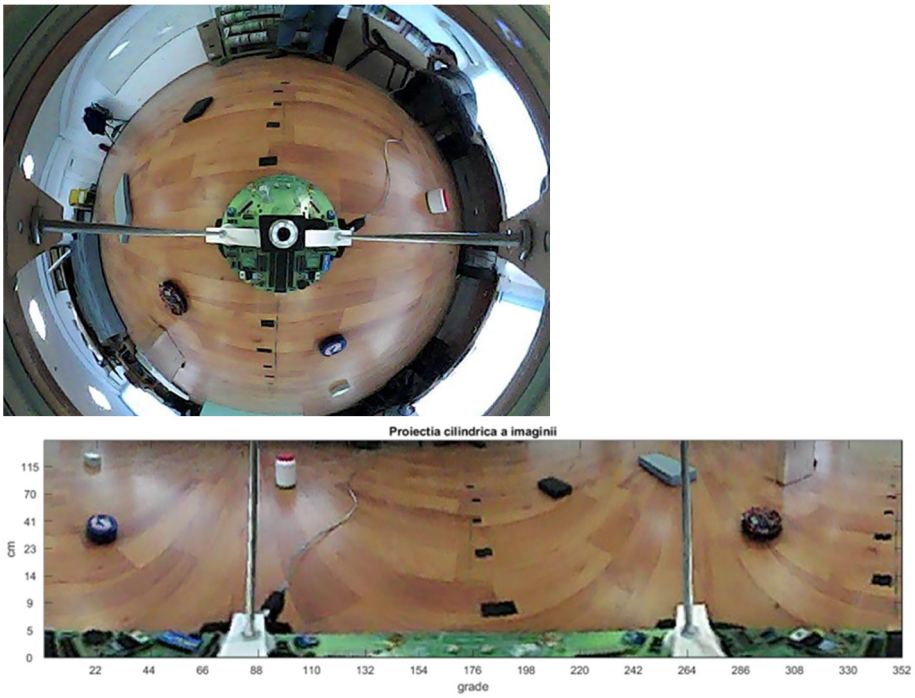

Figure 8. Catadioptric image and the same image displayed on a cylindrical surface.

Due to the uncertainties involved in manufacturing and assembling an omnidirectional camera, the resulting parameters deviate from the desired values. Therefore, it is important to estimate the actual parameters, either for accurate modeling or simply to control the quality of the manufacturing process. For this purpose, the calibration of the omnidirectional vision system was carried out.

In this paper, we explore the implementation of an omnidirectional vision system that utilizes both a camera and a convex mirror. A unique property of convex mirrors is their ability to distort the image, causing distant objects to appear closer together in the captured image. However, one challenge that arises is the lack of knowledge regarding the construction parameters of round convex mirrors. Therefore, image distortions must be estimated in different ways. Even when the mirror parameters have already been determined, the computational requirements of the mirror deformation are too complex to implement using on-board hardware.

The module consists of an SBC (single-board computer) development board, a catadioptric omnidirectional vision system for mapping the area around the robot, and a battery for powering the circuit.

The single-board computer (SBC) was chosen after conducting a price comparison between five SBCs. The last two boards were purchased and tested: the Raspberry Pi 3 model B+ and Odroid XU4. The testing of the two SBCs was achieved using the GIMP program [53] to test the processing speeds of different image processing algorithms. Despite the Samsung Exynos5 Octa ARM Cortex™-A15 Quad 2Ghz and Cortex™-A7 Quad 1.3GHz processors having better sensors than Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz, Odroid XU4 proved slower in the image processing tests compared to the Raspberry Pi 3 B+. Thus, for the creation of the NEXUS 2 robot, Raspberry Pi 3 B+ was used as the processing system [54].

Later, after carrying out the real tests, we identified a need for higher computing power, so the Raspberry Pi 3 B+ was replaced by the Jetson Nano. In terms of computing power, the Jetson Nano surpasses the Raspberry Pi 3. The Jetson Nano is equipped with a 64-bit quad-core ARM Cortex-A57 processor, while the Raspberry Pi 3 has a 32-bit quad-core ARM Cortex-A53 processor. The Jetson Nano is designed to deliver higher performance in artificial intelligence and machine learning applications.

Comparing the two SBCs in terms of the video processor, the Jetson Nano includes an NVIDIA GPU of CUDA architecture, which provides more graphic processing power than the Raspberry Pi 3 B+. This makes the Jetson Nano more efficient in graphics-intensive applications such as 3D simulations or complex video applications.

The processing module makes a map of the obstacles based on the information received from the displacement sensors and the information extracted from the images acquired by the omnidirectional camera located on the robot. It also transmits the processed information regarding the identified obstacles to the neighboring robots and performs the command of the microcontroller located on the base module.

3. Results

3.1. Simulation of the Image Processing Algorithm

The simulation of the algorithm had several purposes:

- Creating a calculation algorithm necessary to identify obstacles;
- Determining the position of obstacles relative to the robot;
- Checking the spherical mirror equation;
- Determining the maximum distance at which objects can be identified;
- Generating an obstacle map;
- Determining the errors caused by the omnidirectional vision system by comparing the real map with the map obtained by the algorithm.

In the first stage, in order to simulate the image processing algorithm, it was necessary to create the input database consisting of images. To achieve this, a model of the robot was created; then, 20 images were acquired with the help of a webcam-type video camera on the route shown in Figure 9. All simulations were executed in the Matlab program.

The acquisition system was moved along the straight path shown in Figure 10, performing image acquisitions every 10 cm to acquire the system positioning data necessary to generate the global image of the robot's path. To test several solutions to achieve obstacle identification, obstacles of different colors and heights were used. In the second stage, after creating the image (Figure 10a), a filter was applied to select the area of interest to facilitate the extraction of the background of the image (Figure 10b).

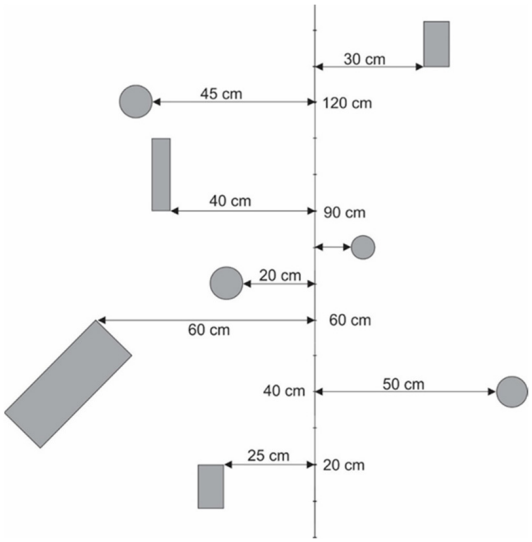

Figure 9. Map of the route to be taken by the robot: virtual model.

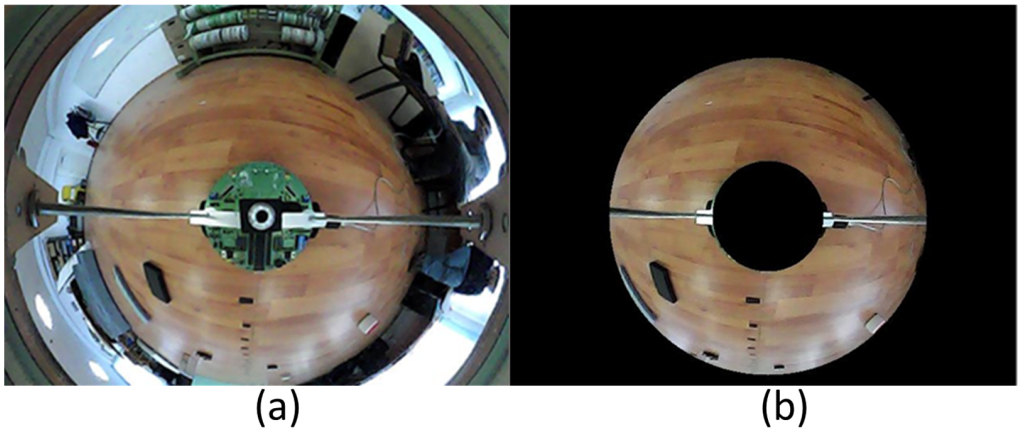

Figure 10. (**a**) The image acquired by the video camera; (**b**) a selection filter for the area of interest is applied to the image.

3.2. Image Binarization

An adaptive threshold was used for the image binarization in order to reduce the complexity of the solution and the computing power so that we achieved a good processing speed. Regarding the adaptive threshold method, the Otsu method [55–59] consists of a segmentation whereby, for each pixel in the image, depending on its value and specific parameters, the component of which that pixel is a part is determined. After the threshold has been calculated and applied, we obtain the image seen in Figure 11; here, the binary image is presented, meaning that all black areas are considered to be obstacles.

After image binarization, the image correction process takes place; this is followed by obstacle map generation.

To generate the route, the binary images are superimposed according to the position at which they were acquired, because, in this way, the accuracy with which the obstacles can be positioned increases by summing the same obstacle from several images in the phase. Obstacles are identified according to their distance from the robot in one or more images; by summing the images, we obtain a highly precise idea of the size of the obstacle and there is a high probability that the obstacle is not just a system error.

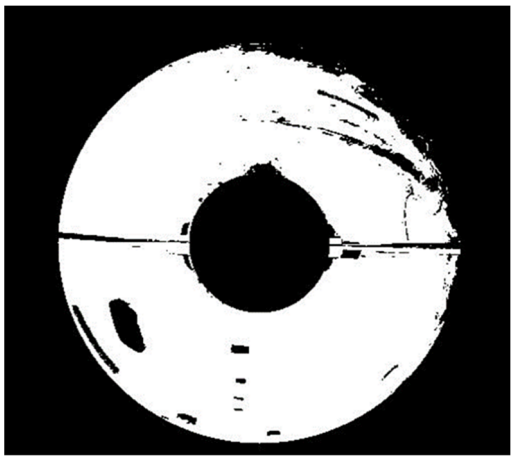

Figure 11. The binarized image after applying filter i.

After the obstacle map was made at the level of the robot, the problem of transmitting the data to the rest of the robots in the swarm was addressed. Using a mesh network was the only flexible and robust alternative to the robot's communication system, but it comes with a major drawback: namely, the small amount of data that can be transmitted by each robot. Because of this, in order to be able to transmit the data obtained by processing the images acquired by the robot, it was necessary to create a protocol that uses a small amount of data.

In order to minimize the number of points on the outline of the obstacles that have already been transformed into real coordinates, we identified as a possible solution the approximation of the shape of the obstacles with some ellipses, because ellipses are the most common non-linear shapes in natural and man-made objects. This solution also has the disadvantage of increasing the areas that are considered obstacles during the image processing.

The main purpose of the simulation was to determine the percentage of difference between the real route and the approximated route with ellipses. To validate this simulation, we used the intersection and union of the two resulting images. The simulation was carried out in the MATLAB program. The procedure involved inserting the route image and binarizing it to identify obstacles.

3.3. Experimental Determination of the Robot Displacement Error

Given that the purpose of the robot is to map an unknown area, it is necessary that the data taken by the robot be accurate from the point of view of the robot and its movement. As such, the data from two different transducers were used: one magneto-resistive transducer, which was used to measure the speed of the motors, and an ADNS 3080 optical flow transducer.

To ensure that the data regarding the real movement of the robot were as accurate as possible, a graph paper sheet was used. In this way, its positioning error was precisely determined.

Before undertaking the test, a calibration of the sensors was performed. The data obtained from the sensors were read by the microcontroller (Table 1) and then transmitted to a laptop via the XBee module.

The trajectory generation also requires the robot's displacement angle, which contains the sum of the robot's previous displacements.

For ADNS 3080 optical sensors, the interpretation of the data is complicated, especially due to their position relative to the robot's center of rotation. Figure 12 shows the coordinate axes of the sensors relative to the center of rotation of the robot.

Table 1. Data from optical flow sensors that measure the robot's movement.

Iteration	Sensor ADNS 3080 nr.1			Sensor ADNS 3080 nr.2			Translator 1	Translator 2
	X Displacement	Y Displacement	Squali	X Displacement	Y Displacement	Squali		
1	0	0	124	0	0	116	0	0
2	0	0	124	0	0	115	0	0
3	−1	0	124	3	0	112	0	1
4	−13	0	123	15	0	108	3	3
5	−20	0	124	19	0	111	4	4
6	−24	0	124	23	0	115	5	5
7	−24	0	124	23	0	123	6	5

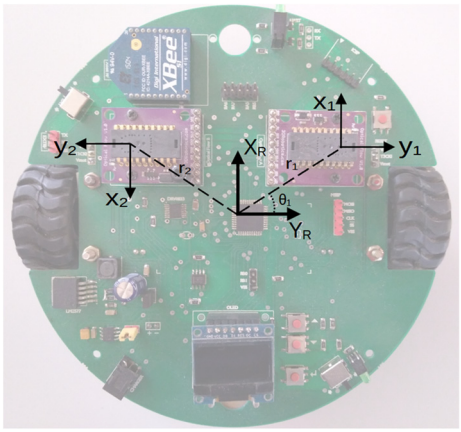

Figure 12. Coordinate axes of optical flow sensors.

In order to calculate the robot's displacement, the transformation matrix A of the sensor's coordinates is required. For NEXUS 2 robots, r1 and r2 have the same actual size 42 mm, and angle θ1 is 0.615 radians. Thus, the matrix A has the following form:

$$A = \begin{bmatrix} 1 & 0 & -109.62 \\ 0 & 1 & 155.18 \\ -1 & 0 & 109.62 \\ 0 & -1 & 155.62 \end{bmatrix}. \quad (11)$$

3.4. Experimental Determinations Based on the Jetson Nano Platform

The move to dynamic testing mainly consisted of adapting the code from Matlab to Python. This required installing an OpenCV image processing library in Python. An important element of the Jetson Nano development board (Figure 13) is the CSI (camera serial interface) connector to which the Raspberry Pi-v2 camera is connected.

For image acquisition, the convex lens was mounted on a plexiglass cylindrical support; thus, the supports in Figure 13 were eliminated because they introduced an area of indeterminacy in the final image and also successfully reduced the vibrations. Moreover, the mirror was placed at a distance of 10 cm from the video camera because the video system does not have a zoom function.

To adapt the processing board to the Jetson Nano platform, serial data transmission was achieved, because changes were made to the data transmission module between the SBC and the microcontroller. Along the same route, in the new configuration, the robot managed to acquire 64 images, which meant the speed was 10 times higher (Figure 14).

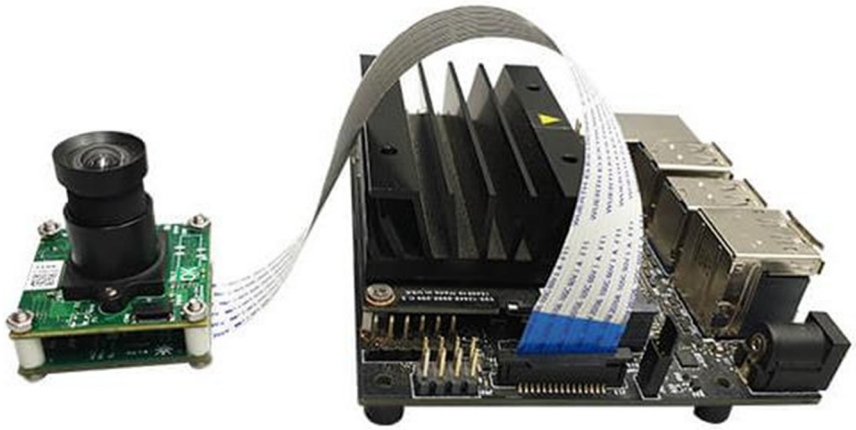

Figure 13. NEXUS 2 robot with the Jetson Nano platform and Raspberry Pi-v2 camera.

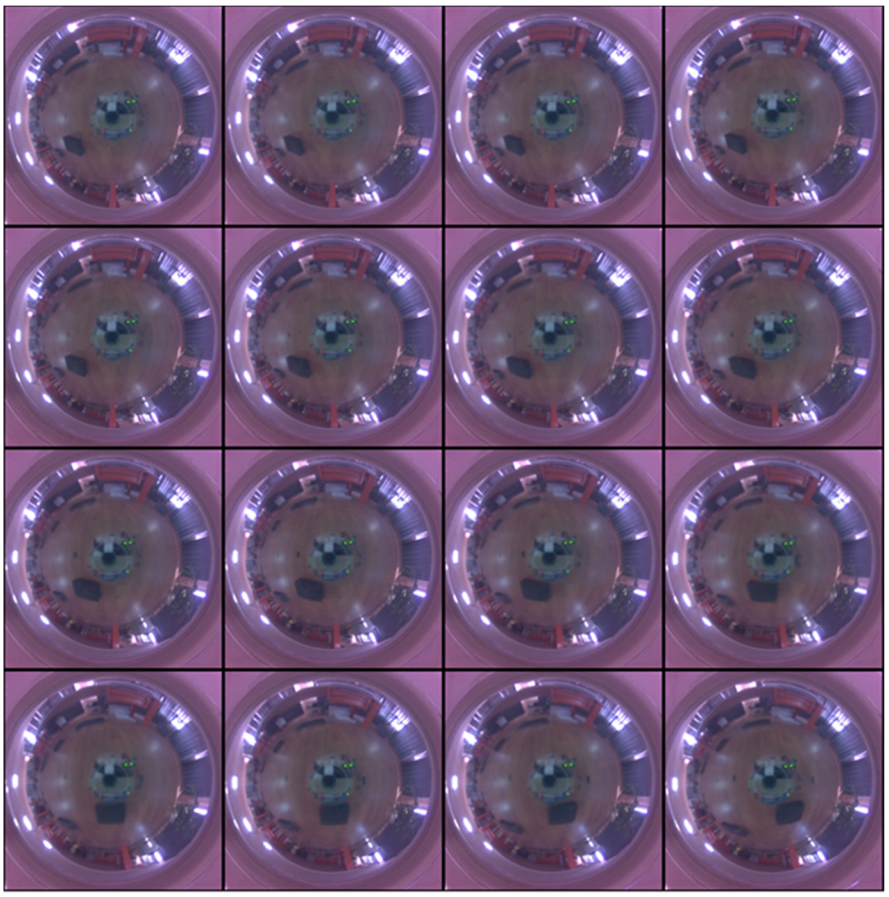

Figure 14. Images acquired by Jetson Nano platform with Raspberry Pi-v2 camera.

By processing each image and by superimposing them based on the positioning data transmitted by the microcontroller, the route in Figure 15 was produced. This represents the route followed by the robot, which is superimposed on it in red; the positions and the real shape of the obstacles are represented.

The purpose of this test was to verify that the image processing algorithm works for a real scenario and that it can accurately generate the path followed by the robot.

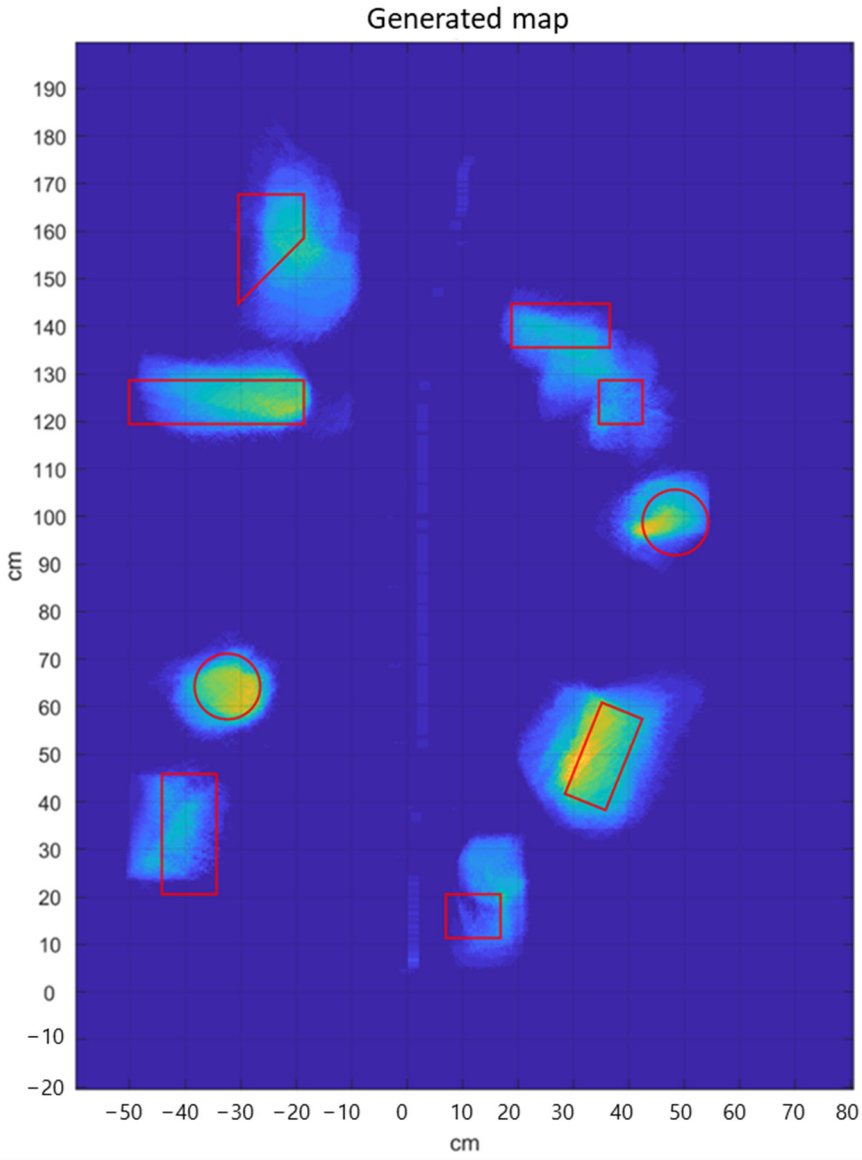

Figure 15. Route map created by the robot.

3.5. Testing the Data Transmission Algorithm

The testing of the data transmission algorithm has the purpose of finalizing and certifying the functionality of the integrated systems of the realized robot, NEXUS 2.

Through the fusion of the data from the optical flow sensors and the brush DC motor transducers, the robot's movement was obtained. By testing and rebuilding the image processing algorithm, the actual positions of the obstacles were obtained and, through the communication system, the data were transmitted to the laptop.

For each acquired image shown in Figure 16, its processing was carried out in order to identify the obstacles and to transform the position of the obstacles into real coordinates by applying the mirror distortion equation; finally, an approximation of the obstacle with an ellipse was created.

After obtaining the ellipses of all obstacles observed in the image, the displacement data from the microcontroller were added. At this moment, the data package to be transmitted was complete, as shown in Figure 17.

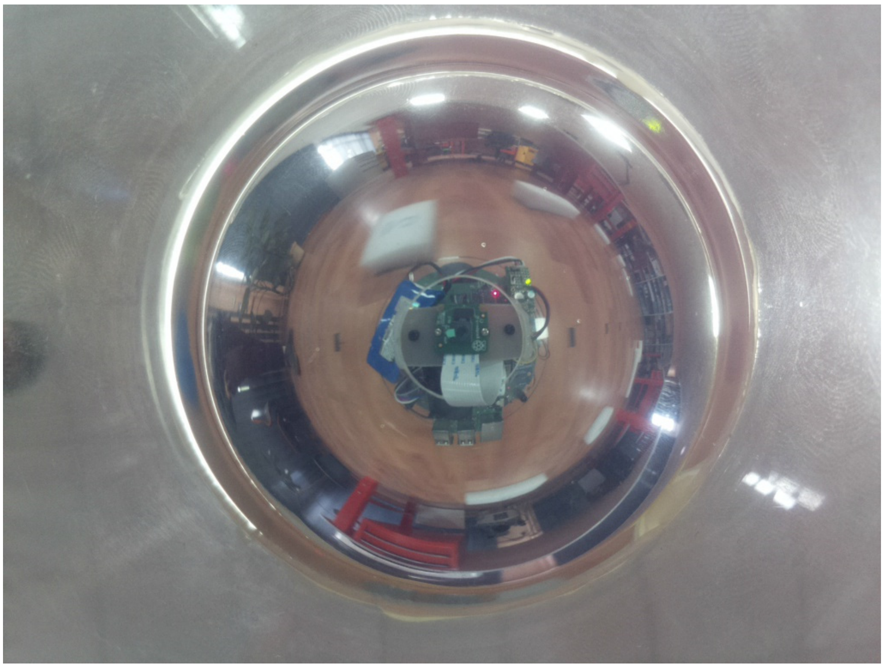

Figure 16. The image processed in real time with the ellipses formed around the obstacles.


```
 14 0 9 347 [[  0.      275.8     235.         6.035    74.56    115.25 ]
   [   0.       0.       0.       0.       0.       0.    ]
   [   0.       0.       0.       0.       0.       0.    ]
   [   0.       0.       0.       0.       0.       0.    ]
   [   0.       0.       0.       0.       0.       0.    ]
   [   0.       0.       0.       0.       0.       0.    ]
   [   0.       0.       0.       0.       0.       0.    ]
   [   0.       0.       0.       0.       0.       0.    ]
   [   0.       0.       0.       0.       0.       0.    ]
   [   0.       0.       0.       0.       0.       0.    ]
   [   0.       0.       0.       0.       0.       0.    ]
   [   0.       0.       0.       0.       0.       0.    ]
   [   0.       0.       0.       0.       0.       0.    ]
   [   0.       0.       0.       0.       0.       0.    ]
   [   0.       0.       0.       0.       0.       0.    ]
   [   0.       0.       0.       0.       0.       0.    ]
   [   0.       0.       0.       0.       0.       0.    ]
   [   0.       0.       0.       0.       0.       0.    ]
   [   0.       0.       0.       0.       0.       0.    ]
   [   0.       0.       0.       0.       0.       0.    ]]
```


Figure 17. The data packet transmitted by the NEXUS 2 robot.

A data package of this kind also contains the robot's position coordinates, as shown in Figure 17. This package contains:

- Processed image number 14;
- The coordinates of the robot's position x = 0, y = 9 and the movement angle of 347 degrees;
- The points needed to generate an ellipse corresponding to an obstacle: the current number of the ellipse, the coordinates of the center of the ellipse, the dimensions of the axes of the ellipse and its rotation angle, as shown in Table 2.

Table 2. The coordinates of the ellipses.

Ellipse	x_0	y_0	Small Axis	Big Axis	Φ*Grade*
o	275.8	235	6.035	74.56	115.25

During the movement, the robot sent data packets that included the position of the ellipses. These packets were received by a computer with the help of a module simulating another robot. An application was made in Matlab, and the map traveled by the robot was recreated based on the data from the packages. This map is shown in Figure 18, where the white ellipses are the obstacles generated using the application and the image has been overlaid on top of the actual obstacle map.

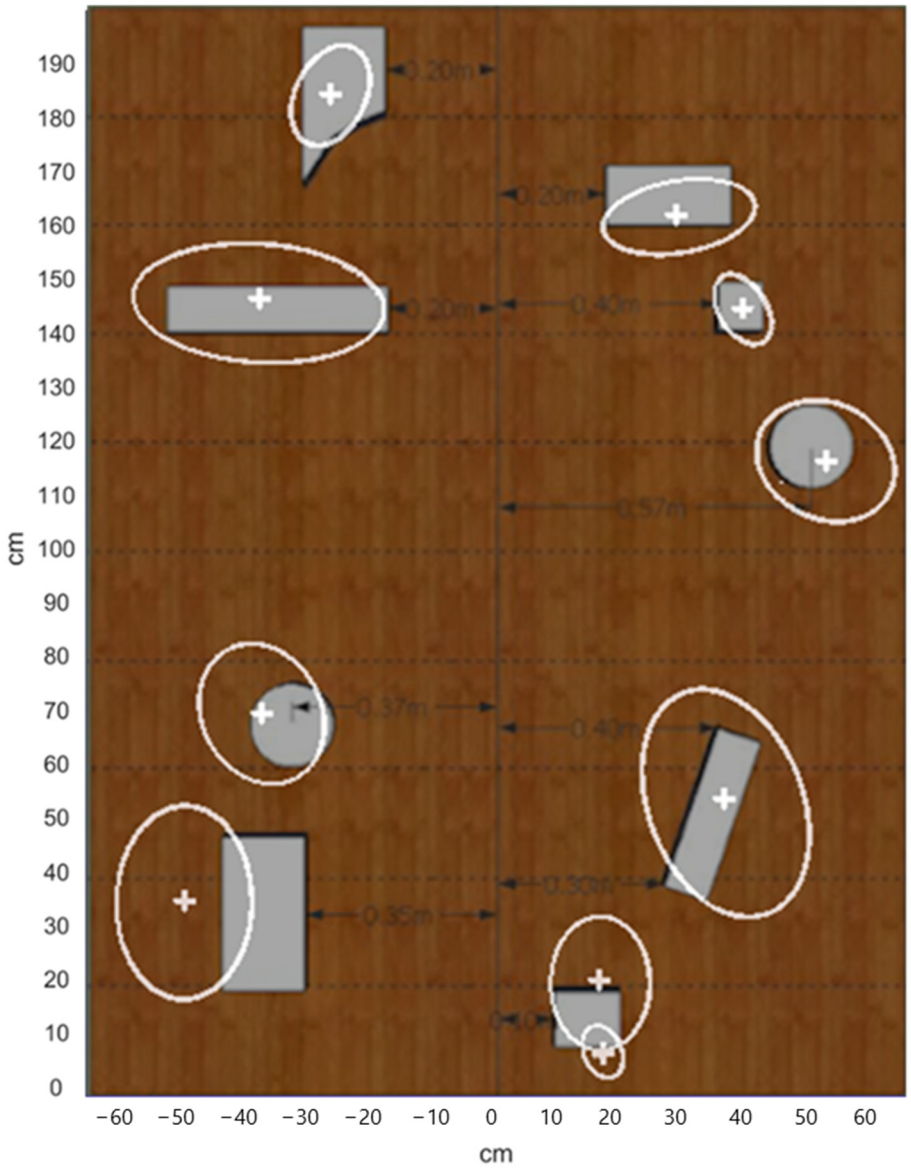

Figure 18. The map traveled by the robot was recreated based on the data in the packets.

The positioning errors between real obstacles and generated ellipses involve a multitude of factors:

- The oscillations of the spherical mirror due to vibrations caused by displacement;
- The approximation of the image transformation equation;
- The approximation performed to determine the outline of the obstacles.

3.6. Path Planning Algorithm

The conventional framework serves as the foundation for developing a system that enables the robot to independently navigate from its starting position to a desired endpoint. The user provides a specific task, which is then processed by the planner. Utilizing a

model of the environment, the planner generates a path that ensures there are no collisions. This path is subsequently transformed into a viable trajectory. The motion controller then utilizes feedback obtained from position and velocity sensors to accurately follow the trajectory [60].

When the robot detects a change in the environment during its journey, it automatically halts and initiates the process of devising a new path. The decision to follow this approach is based on the time it takes to plan a new path, although the duration of computation can vary depending on the intricacy of motion planning.

To decrease the time needed, an alternative approach involves incorporating reactive behavior into the controller, enabling real-time obstacle avoidance. However, there is a risk of the robot straying from its intended path. In such cases, the controller lacks the ability to utilize the initial path data to reach its objective.

The trajectory is depicted as a series of line segments, resulting in numerous disruptions in the movement direction. In order for the robot to adhere to this trajectory, it is necessary for it to halt completely at each of these interruptions, leading to a substantial increase in the time needed to reach the objective.

The process of path planning involves creating a geometric path that connects an initial point to a final point, while also passing through predetermined intermediate points. This can be carried out either in the robot's operating space or in joint space. On the other hand, trajectory planning algorithms add time information to a given path's geometry. It is important to note that the route planning algorithm relies on the inclusion of time.

To optimize the route, the process begins by referring to a map (Figure 15) and aiming to locate the most cost-effective route from the robot's present position to a designated area, while taking into account the limitations imposed by the robot's kinematics and the terrain [61,62]. The resulting grid map is composed of three distinct layers. The first layer indicates whether a cell is obstructed by an obstacle or accessible; the second layer provides information on the quality of the surface; and the third layer assigns a height value to each cell.

The position of the robot is represented by a two-dimensional vector and its orientation. The height of the robot is not taken into account, as it is assumed to be equal to the height of the cell.

When performing the optimization algorithm, "Algorithm A*" is applied [63]; this algorithm is limited to discrete spaces. In the case of a simple four- or eight-connection network, this allows the vehicle to rotate in place, which is not possible in the presence of nonholonomic constraints (Figure 19).

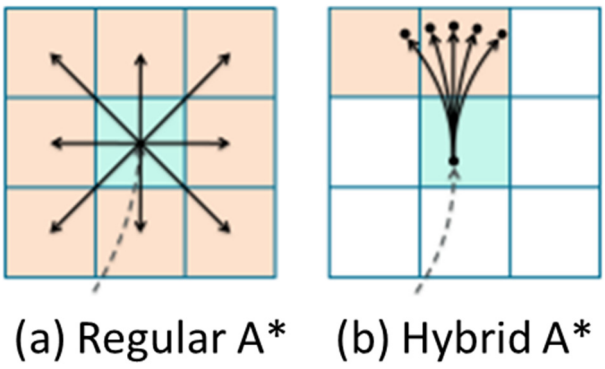

Figure 19. Simulating route planning based on possible neighbors for a cell.

The Hybrid A* algorithm is an extension of the regular A* algorithm, and seeks to overcome its shortcomings.

The obstacle map (Figure 20) resulted from the binarization of Figure 15.

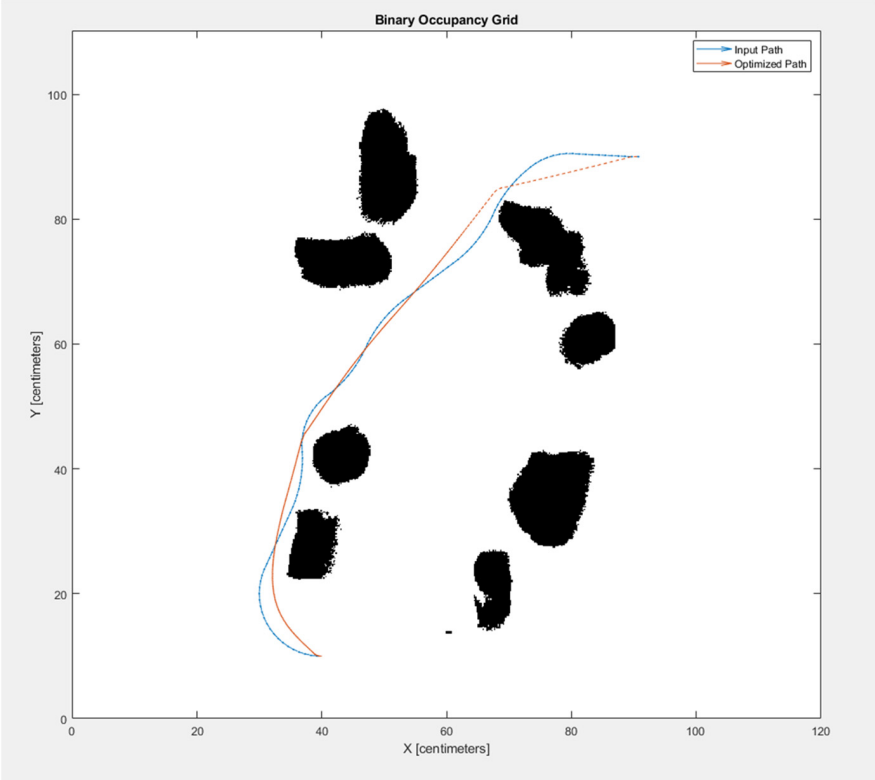

Figure 20. Route planning optimization: binary occupancy grid.

The application of the "Hybrid Algorithm A*" (Figure 21) for planning the route of an autonomous mobile robot in a structured indoor environment was conducted; the robot's kinematic constraints and the working surface conditions were taken into account.

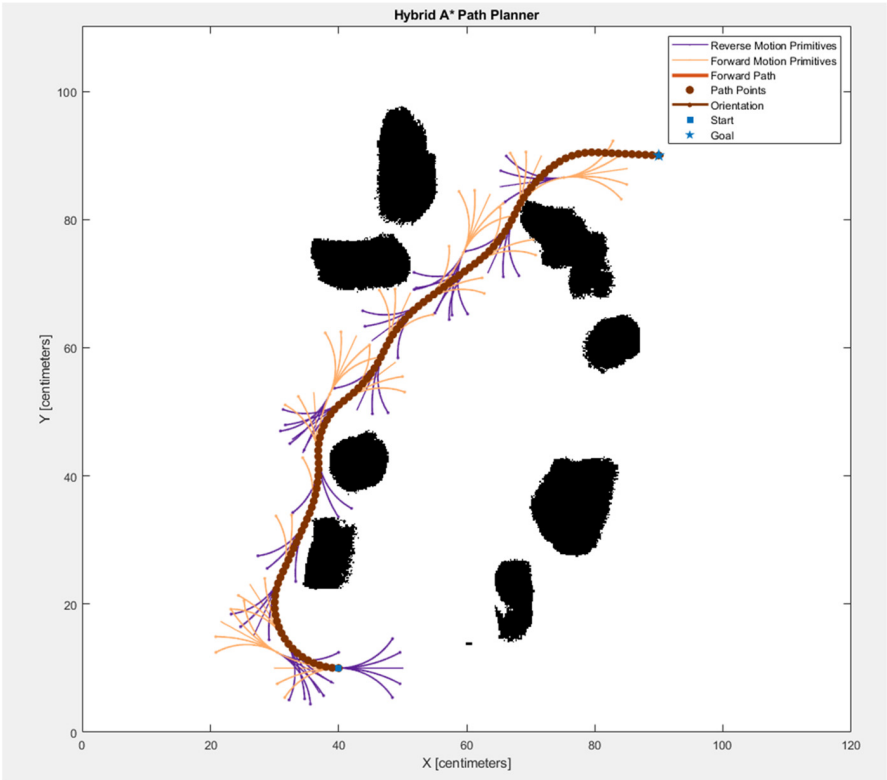

Figure 21. Route planning optimization: application of the "Hybrid Algorithm A*".

This algorithm is able to explicitly plan a route starting from one waypoint and moving to another waypoint. The paths generated were those traveled by the robot.

4. Conclusions

During the evaluation of displacement error in optical flow sensors and speed transducers for brushed DC motors, an unforeseen outcome emerged: the optical flow sensors exhibited higher levels of error compared to the speed transducers. The primary cause of these errors was attributed to the utilization of lenses to extend the distance between the sensors and the surface. Consequently, it was determined that data from the optical flow sensors would only be considered valid when they detected a substantial increase in motor current consumption.

The improvement of the image processing algorithm had the aim of increasing the processing speed so that the system could perform real-time processing. By changing the resolution of the acquired images and by introducing a separate acquisition thread, it was possible to increase the processing speed. However, a disadvantage was also found: namely, a deformation of the acquired images when using a Plexiglas tube to support the mirror. Removing the raw image transformation and replacing it with only the points on the outline of the obstacles increased the processing speed.

Using the Jetson Nano required changing the data transmission protocol between the microcontroller and the SBC from SPI to UART, as the Jetson Nano does not have such a communication protocol implemented. Tests with the Jetson Nano were able to demonstrate the system's ability to map the obstacles it encounters.

The final test consisted of remote transmission via the XBee communication module to a laptop of data packets containing the obstacle ellipses data. The received data were displayed using an application made in MATLAB; thus, the integral testing of the system was achieved.

Beginning with the objectives of the research work and accounting for the fact that the final robot is the result of three development phases, we note the following areas for further development:

- The integration of accumulators in the basic platform by creating a 3D-printed chassis;
- The realization of the integrated SLAM algorithm on all robots in the swarm;
- The integration of distance sensors on the robot's wiring;
- The integration of a socket for changing the microcontroller;
- Improving the processing module by using a Jetson Xavier NX module;
- Producing a cooperation algorithm for robots, in order to create a swarm of robots;
- Replacing the omnidirectional vision system with a stereo vision camera;
- Creating a new four-wheel-drive system.

In summary, this paper—in which the design is the main contribution—aims to optimize the movement speed of a mobile robot with omnidirectional visual perception in a static environment. This research is in the first phase, in which a system of two collaborative robots, equipped with a sensory system that includes an omnidirectional camera, has been created. The current stage of the research complements the data presented at other conferences on the development of SLAM and path planning [4,39,42,44,61]. Practically, from the point of view of (Simultaneous Localization and Mapping—SLAM) this work focused on metric mapping, and the data processing on board the robot was also successful for the current configuration. The simulation results are close to the experimental ones. In further development of this research, comparative studies need to be carried out to identify the best algorithm for achieving robot movement in a structured indoor environment. In addition, the results will be verified and cross-checked with SLAM using a 360-degree laser scanner to improve robot autonomy.

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

Funding: This research received no external funding.

Data Availability Statement: Data is contained within the article.

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

References

1. Vamvoudakis, K.G.; Jagannathan, S. *Control of Complex Systems. Theory and Applications*; Butterworth-Heinemann-Elsevier: Woburn, MA, USA, 2016; p. 762, ISBN 978-0-12-805246-4. [CrossRef]
2. Sánchez-Ibáñez, J.R.; Pérez del Pulgar, C.J.; Garcia-Cerezo, A. Path Planning for Autonomous Mobile Robots: A Review. *Sensors* **2021**, *21*, 7898. [CrossRef] [PubMed]
3. Zhang, H.; Lin, W.; Chen, A. Path Planning for the Mobile Robot: A Review. *Symmetry* **2018**, *10*, 450. [CrossRef]
4. Zhang, F.; Li, T.; Xue, T.; Zhu, Y.; Yuan, R.; Fu, Y. An Improved Dynamic Window Approach Integrated Global Path Planning. In Proceedings of the 2019 IEEE International Conference on Robotics and Biomimetics (ROBIO), Dali, China, 6–8 December 2019; pp. 2873–2878, ISBN 978-1-7281-6321-5. [CrossRef]
5. Dahalan, A.A.; Saudi, A.; Sulaiman, J. Enhancing Autonomous Guided Vehicles with Red-Black TOR Iterative Method. *Mathematics* **2023**, *11*, 4393. [CrossRef]
6. Patle, B.K.; Babu, G.; Pandey, A.; Parhi, D.R.K.; Jagadeeesh, A. A review: On Path Planning Strategies for Navigation of Mobile Robot. *Def. Technol.* **2019**, *15*, 582–606. [CrossRef]
7. Hoy, M.; Matveev, A.S.; Savkin, A.V. Algorithms for Collision-free Navigation of Mobile Robots in Complex Cluttered Environments: A Survey. *Robotica* **2015**, *33*, 463–497. [CrossRef]
8. Aria, E.; Olstam, J.; Schwietering, C. Investigation of Automated Vehicle Effects on Driver's Behavior and Traffic Performance. *Transp. Res. Procedia* **2016**, *15*, 761–770. [CrossRef]
9. Ajeil, F.H.; Ibraheem, I.K.; Azar, A.T.; Humaidi, A.J. Autonomous Navigation and Obstacle Avoidance of an Omnidirectional Mobile Robot Using Swarm Optimization and Sensors Deployment. *Int. J. Adv. Robot. Syst.* **2020**, *17*, 15. [CrossRef]
10. Khan, S.; Guivant, J. Fast Nonlinear Model Predictive Planner and Control for an Unmanned Ground Vehicle in the Presence of Disturbances and Dynamic Obstacles. *Sci. Rep.* **2022**, *12*, 12135. [CrossRef] [PubMed]
11. Patle, B.K.; Pandey, A.; Jagadeesh, A.; Parhi, D.R. Path Planning in Uncertain Environment by Using Firefly Algorithm. *Def. Technol.* **2018**, *14*, 691–701. [CrossRef]
12. Menezes Morato, M.; Normey-Rico, J.; Sename, O. Model Predictive Control Design for Linear Parameter Varying Systems: A Survey. *Annu. Rev. Control* **2020**, *49*, 64–80. [CrossRef]
13. Mikolajczyk, T.; Mikolajewski, D.; Klodowski, A.; Łukaszewicz, A.; Mikolajewska, E.; Paczkowski, T.; Macko, M.; Skonia, M. Energy Sources of Mobile Robot Power Systems: A Systematic Review and Comparison of Efficiency. *Appl. Sci.* **2023**, *13*, 7547. [CrossRef]
14. Farooq, M.U.; Eizad, A.; Bae, H.-K. Power Solutions for Autonomous Mobile Robots: A Survey. *Robot. Auton. Syst.* **2023**, *159*, 104285. [CrossRef]
15. Zhao, S.; Hwang, S.-H. ROS-Based Autonomous Navigation Robot Platform with Stepping Motor. *Sensors* **2023**, *23*, 3648. [CrossRef]
16. Baek, E.-T.; Im, D.-Y. ROS-Based Unmanned Mobile Robot Platform for Agriculture. *Appl. Sci.* **2022**, *12*, 4335. [CrossRef]
17. Haxhibeqiri, J.; Jarchlo, E.A.; Moerman, I.; Hoebeke, J. Flexible Wi-Fi Communication Among Mobile Robots in Indoor Industrial Environments. *Hindawi Mob. Inf. Syst.* **2018**, *2018*, 3918302. [CrossRef]
18. Shepard, J.; Kitts, C. A Multi-Layer, Multi-Robot Control Architecture for Long-Range, Dynamic Communication Links. In *Multi-Robot Systems-New Advances*; Küçük, S., Ed.; IntechOpen: Rijeka, Croatia, 2023. [CrossRef]
19. Liu, L.; Wang, X.; Yang, X.; Liu, H.; Li, J.; Wang, P. Path planning techniques for mobile robots: Review and prospect. *Expert Syst. Appl.* **2023**, *227*, 120254. [CrossRef]
20. Choi, S.H.; Zhu, W.K. Performance Optimisation of Mobile Robots for Search-and-Rescue. *Appl. Mech. Mater.* **2012**, *232*, 403–407. [CrossRef]
21. Van, N.T.T.; Tiene, N.M.; Luong, N.C.; Duyen, H.T.K. Energy Consumption Minimization for Autonomous Mobile Robot: A Convex Approximation Approach. *J. Robot. Control* **2023**, *4*, 403–412. [CrossRef]
22. Grisales-Ramirez, E.; Osorio, G. Multi-Objective Combinatorial Optimization Using the Cell Mapping Algorithm for Mobile Robots Trajectory Planning. *Electronics* **2023**, *12*, 2105. [CrossRef]
23. Toscano-Moreno, M.; Mandow, A.; Martínez, M.A.; García-Cerezo, A. DEM-AIA: Asymmetric Inclination-Aware Trajectory Planner for Off-Road Vehicles with Digital Elevation Models. *Eng. Appl. Artif. Intell.* **2023**, *121*, 105976. [CrossRef]
24. Wong, J.Y. *Theory of Ground Vehicles*, 3rd ed.; John Willey & Sons: Hoboken, NJ, USA, 2001; ISBN 0-471-35461-9.

25. Ciobotaru, T. Semi-Empiric Algorithm for Assessment of the Vehicle Mobility. *Leonardo Electron. J. Pract. Technol.* **2009**, *8*, 19–30.
26. Wong, J.Y.; Chiang, C.F. A General Theory for Skid Steering of Tracked Vehicles. *Proc. Inst. Mech. Eng. Part D J. Automob. Eng.* **2001**, *215*, 343–355. [CrossRef]
27. Li, X.A.; Sun, K.; Guo, C.; Liu, H. Hybrid Adaptive Disturbance Rejection Control for Inflatable Robotic Arms. *ISA Trans.* **2022**, *126*, 617–628. [CrossRef] [PubMed]
28. Li, X.A.; Yue, H.; Yang, D.; Sun, K.; Liu, H. A Large-Scale Inflatable Robotic Arm Toward Inspecting Sensitive Environments: Design and Performance Evaluation. *IEEE Trans. Ind. Electron.* **2023**, *70*, 12486–12499. [CrossRef]
29. Rendón, S.V. Trajectory Planning Based on Collocation Methods for Multiple Aerial and Ground Autonomous Vehicles. Ph.D. Thesis, Escuela Técnica Superior de Ingeniería Universidad de Sevilla, Seville, Spain, 2015.
30. Yue, X.; Wang, X.A.; Dai, H. Simple Time Domain Collocation Method to Precisely Search for the Periodic Orbits of Satellite Relative Motion. *Hindawi Math. Probl. Eng.* **2014**, *2014*, 854967. [CrossRef]
31. Wijanto, E. Design of Deliberative and Reactive Hybrid Control System for Autonomous Stuff-Delivery Robot Rover. *Elkomik J. Tek. Energi Elektr. Teknk Telekommunikasi Tek. Elektron.* **2023**, *11*, 15. [CrossRef]
32. Ingrand, F.; Ghallab, M. Deliberation for Autonomous Robots: A Survey. *Artif. Intell.* **2017**, *247*, 10–44. [CrossRef]
33. Evans, J.; Patrón, P.; Smith, B.; Lane, D.M. Design and Evaluation of a Reactive and Deliberative Collision Avoidance and Escape Architecture for Autonomous Robots. *Auton. Robot.* **2008**, *24*, 247–266. [CrossRef]
34. Apoorva, A.; Gautam, R.; Kala, R. Motion Planning for a Chain of Mobile Robots Using A* and Potential Field. *Robotics* **2018**, *7*, 20. [CrossRef]
35. Raja, P.; Pugazhenthi, S. Optimal Path Planning of Mobile Robots: A review. *Int. J. Phys. Sci.* **2012**, *7*, 1314–1320. [CrossRef]
36. Zvi, S. Off-Line and On-Line Trajectory Planning. *Mech. Mach. Sci.* **2015**, *29*, 29–62. [CrossRef] [PubMed]
37. Llopis-Albert, C.; Rubio, F.; Valero, F. Optimization Approaches for Robot Trajectory Planning. *Multidiscip. J. Education. Soc. Technol. Sci.* **2018**, *5*, 16. [CrossRef]
38. Dubois, R.; Eudes, A.; Frémont, V. Sharing Visual-Inertial Data for Collaborative Decentralized Simultaneous Localization and Mapping. *Robot. Auton. Syst.* **2022**, *148*, 103933. [CrossRef]
39. Yue, Y.; Zhao, C.; Li, R.; Yang, C.; Zhang, J.; Wen, M.; Wang, Y.; Wang, D. A Hierarchical Framework for Collaborative Probabilistic Semantic Mapping. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 15 September 2020; pp. 9659–9665. [CrossRef]
40. Gemignani, G.; Capobianco, R.; Bastianelli, E.; Bloisi, D.D.; Iocchi, L.; Nardi, D. Living with Robots: Interactive Environmental Knowledge Acquisition. *Robot. Auton. Syst.* **2016**, *76*, 1–16. [CrossRef]
41. Achour, A.; Al-Assaad, H.; Dupuis, Y.; El Zaher, M. Collaborative Mobile Robotics for Semantic Mapping: A Survey. *Appl. Sci.* **2022**, *12*, 316. [CrossRef]
42. Bastianelli, E.; Bloisi, D.D.; Capobianco, R.; Cossu, F.; Gemignani, G.; Iocchi, L.; Nardi, D. On-Line Semantic Mapping. In Proceedings of the 2013 16th International Conference on Advanced Robotics (ICAR), Montevideo, Uruguay, 25–29 November 2013; pp. 1–6. [CrossRef]
43. Bîtea, M.A. Theoretical and Experimental Analysis and Synthesis of a Mobile Autonomous Mechatronic System. Ph.D. Thesis, Polytechnic University of Timisoara, Timişoara, Romania, 2012.
44. Liu, Y.; Petillot, Y.; Lane, D.; Wang, S. Global Localization with Object-Level Semantics and Topology. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 4909–4915. [CrossRef]
45. Huang, W.H.; Beevers, K.R. Topological Mapping with Sensing-Limited Robots. In *Algorithmic Foundations of Robotics VI*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 235–250, ISBN 978-3-540-25728-8. [CrossRef]
46. Ravikumar, T.M.; Saravanan, R.; Nirmal, N. Modeling and Optimization of Odometry Error in a TwoWheeled Differential Drive Robot. *Int. J. Sci. Res. Publ.* **2013**, *3*, 9.
47. Dang, T.-V.; Bui, N.-T. Multi-Scale Fully Convolutional Network-Based Semantic Segmentation for Mobile Robot Navigation. *Electronics* **2023**, *12*, 533. [CrossRef]
48. Dénes, T.; Gábor, S. Contact Patch Memory of Tyres Leading to Lateral Vibrations of Four-Wheeled Vehicles. *Philos. Trans. R. Soc. A Math. Phisycal Eng. Sci.* **2013**, *371*, 20120427. [CrossRef] [PubMed]
49. Kozłowski, K.; Pazderski, D. Modeling and Control of a 4-Wheel Skid-Steering Mobile Robot. *Int. J. Appl. Math. Comput. Sci.* **2004**, *14*, 477–496.
50. Alexa, O.; Ciobotaru, T.; Grigore, L.Ș.; Grigorie, T.L.; Ștefan, A.; Oncioiu, I.; Priescu, I.; Vlădescu, C. A Review of Mathematical Models Used to Estimate Wheeled and Tracked Unmanned Ground Vehicle Kinematics and Dynamics. *Mathematics* **2023**, *11*, 3735. [CrossRef]
51. Hachem, M.; Borrell, A.M.; Sename, O.; Atoui, H.; Morato, M. ROS Implementation of Planning and Robust Control Strategies for Autonomous Vehicles. *Electronics* **2023**, *12*, 3680. [CrossRef]
52. Siciliano, B.; Khatib, O. *Handbook of Robotics*, 2nd ed.; Springer: Würzburg, Germany, 2016; p. 2304, ISBN 978-3-319-32550-7.
53. GIMP—GNU Image Manipulation Program. Available online: https://www.gimp.org/ (accessed on 14 November 2023).
54. Gorgoteanu, D.; Molder, C. Electric Powered Miniature Vehicle for Multi-Agent Network Testbed. In Proceedings of the 2019 Electric Vehicles International Conference (EV), Bucharest, Romania, 11 November 2019; pp. 1–5. [CrossRef]
55. Sezgin, M.; Sankur, B. Survey over Image Thresholding Techniques and Quantitative Performance Evaluation. *J. Electron. Imaging* **2004**, *13*, 146–168. [CrossRef]

56. Otsu, N. A Threshold Selection Method from Gray-Level Histograms. *IEEE Trans. Syst. Man Cybern.* **1979**, *9*, 62–66. [CrossRef]
57. Michalak, H.; Okarma, K. Improvement of Image Binarization Methods Using Image Preprocessing with Local Entropy Filtering for Alphanumerical Character Recognition Purposes. *Entropy* **2019**, *21*, 562. [CrossRef] [PubMed]
58. Image Thresholding. Available online: https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html (accessed on 15 November 2023).
59. Huang, D.-Y.; Wang, C.-H. Optimal Multi-Level Thresholding using a Two-Stage Otsu Optimization Approach. *Pattern Recognit. Lett.* **2009**, *30*, 275–284. [CrossRef]
60. Qiao, L.; Luo, X.; Luo, Q. An Optimized Probabilistic Roadmap Algorithm for Path Planning of Mobile Robots in Complex Environments with Narrow Channels. *Sensors* **2022**, *22*, 8983. [CrossRef]
61. Petereit, J.; Emter, T.; Frey, C.W.; Kopfstedt, T.; Beutel, A. Application of Hybrid A* to an Autonomous Mobile Robot for Path Planning in Unstructured Outdoor Environments. In Proceedings of the ROBOTIK 2012, 7th German Conference on Robotics, Munich, Germany, 21–22 May 2012; pp. 1–6.
62. Dolgov, D.; Thrun, S.; Montemerlo, M.; Diebel, J. *Practical Search Techniques in Path Planning for Autonomous Driving*; American Association for Artificial Intelligence: Palo Alto, CA, USA, 2008; Available online: https://ai.stanford.edu/~ddolgov/papers/dolgov_gpp_stair08.pdf (accessed on 11 September 2023).
63. Hart, P.; Nilsson, N.; Raphael, B. A formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. Syst. Sci. Cyberntics* **1968**, *4*, 100–107. [CrossRef]