applied sciences

MDPI

Article

Design, Development, and Experimental Verification of a Trajectory Algorithm of a Telepresence Robot

Ali A. Altalbe [1,2,*], Aamir Shahzad [3] and Muhammad Nasir Khan [4,*]

[1] Department of Computer Science, Prince Sattam Bin Abdulaziz University, Al-Kharj 11942, Saudi Arabia
[2] Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia
[3] Mechanical Engineering Department, The University of Lahore, Lahore 54000, Pakistan
[4] Electrical Engineering Department, Government College University Lahore, Lahore 54000, Pakistan
* Correspondence: a.altalbe@psau.edu.sa (A.A.A.); dr.ir.mnkhan@gcu.edu.pk (M.N.K.)

Featured Application: Healthcare, remote sensing, and military.

Abstract: Background: Over the last few decades, telepresence robots (TRs) have drawn significant attention in academic and healthcare systems due to their enormous benefits, including safety improvement, remote access and economics, reduced traffic congestion, and greater mobility. COVID-19 and advancements in the military play a vital role in developing TRs. Since then, research on the advancement of robots has been attracting much attention. Methods: In critical areas, the placement and movement of humans are not safe, and researchers have started looking at the development of robots. Robot development includes many parameters to be analyzed, and trajectory planning and optimization are among them. The main objective of this study is to present a trajectory control and optimization algorithm for a cognitive architecture named auto-MERLIN. Optimization algorithms are developed for trajectory control. Results: The derived work empirically tests the solutions and provides execution details for creating the trajectory design. We develop the trajectory algorithm for the clockwise direction and another one for the clockwise and counterclockwise directions. Conclusions: Experimental results are drawn to support the proposed algorithm. Self-localization, self-driving, and right and left turn trajectories are drawn. All of the experimental results show that the designed TR works properly, with better accuracy and only a slight jitter in the orientation. The jitter is found due to the environmental factor caught by the sensors, which can be filtered easily. The results show that the proposed approach is less complex and provides better trajectory planning accuracy.

Keywords: autonomous vehicle (AV); healthcare; telepresence; trajectory; speed and sensors

check for updates

Citation: Altalbe, A.A.; Shahzad, A.; Khan, M.N. Design, Development, and Experimental Verification of a Trajectory Algorithm of a Telepresence Robot. *Appl. Sci.* **2023**, *13*, 4537. https://doi.org/10.3390/app13074537

Academic Editor: Yutaka Ishibashi

Received: 29 December 2022
Revised: 28 March 2023
Accepted: 29 March 2023
Published: 3 April 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

In the last few decades, humans have started looking towards greater mobility in the tasks that machines perform. Since that era, researchers have been getting involved in the development of telepresence robots (TRs) [1–4]. MERLIN (Mobile Experimental Robot for Locomotion and Intelligent Navigation) was developed on the same idea of mobility. In the modern era, human–robot interaction is increasing and demanding in many application areas, including healthcare systems and military applications. This is due to the advancement and capability of robots to perform complex tasks in dangerous and prohibited environments. The evolution of the digital era and smart robotic designs are continuously simplifying daily routine tasks with fast response and precision. Researchers are working very hard to design such robots, but still, the robots have many limitations in performing various functions. On the other hand, it has become necessary for humans to involve robots in tasks from remote locations or in harmful situations, e.g., COVID-19. Robots can execute these schedules tasks instead of human beings, who can instead sit in remote locations [1–4].

Mobile robots are also extensively used in other applications, such as entertainment, audio–video communication, and remote monitoring. However, the design of many physical parameters—including the maneuverability, controllability, and stability of human–robot interaction—still needs attention [5]. Telepresence robots (TRs) have the facility of a video conferencing display to keep in contact with the robot continuously. The application of TRs is found in many areas, including remote offices [2–5], online education for disabled students [6,7], and shopping plazas [8]. The employment of TRs is not easy, even though a continuous social presence receives attention daily [5,8,9]. In hazardous or remote locations, precise navigation to avoid obstacles is another challenge, limited due to the camera image resolution [10,11]. To overcome these challenges, researchers work day and night on control mechanisms.

The robot is controlled using the remote controller, which is connected to the device and communicates through the remote location. Although the use of robots is still increasing, the stability of the control systems should be designed appropriately so that the journey will be free of collision [12]. Keeping in mind the development and application of robots in [12–15], we developed a speed control method with additional sensors, which can provide information to control the robot's speed by detecting obstacles and determining another safe path. Such a useful design for robots, which controls the fast and slow motion of the robot through a dense and narrow passage, is currently in demand. Much work has been conducted previously in developing the industrial robots reported in [16–18]. However, to the best of our knowledge, distance-based speed control has not yet been developed. This motivated us to design a controller for deploying robots in healthcare and military systems.

The main contribution of this paper is to analyze the trajectory and stabilize the nonlinear behavior by designing optimized control system algorithms. This paper presents and illustrates some ideas that could contribute to the genetic basis of the architecture, allowing us to consider the previous requirements as much as possible. The idea of the proposed work was to design such a robot that can plan to accomplish the specified tasks. To fulfill this requirement, the focus of the current research was on the devising of trajectory planning and driving new algorithms. Then, experimental verification was performed to determine the accuracy of the algorithms, along with noise analysis.

A chassis of an HPI Savage 2.1 monster truck was developed to carry out the experimental work, as shown in Figure 1. The vehicle was equipped with Ackermann steering, with a four-wheel drive capacity. The force is propagated on all four wheels in the same direction. This shows that simultaneously turning is impossible with the coupled position and orientation control.

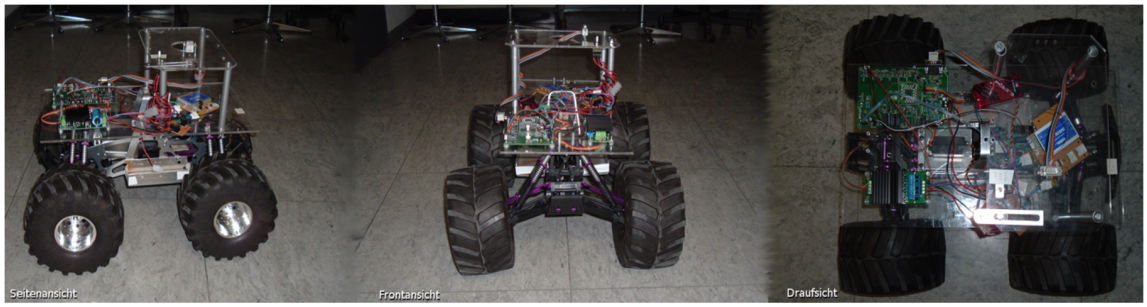


Figure 1. HPI Savage 2.1 monster truck with robot attachment.

The purpose of designing the monster truck was to obtain results on the trajectory, navigate a rough surface, detect obstacles, and find a smooth path by avoiding them. It was necessary to design the control electronics by utilizing the proportional–integral–derivative (PID) controller to fulfill the requirements. Still, because of the low space, the information about the controller is not given in this article. The stability of the TR was achieved by using three powerful direct-current (DC) motors of 7.2 volts, and the model was equipped

with a HiTec HS-5745MG servo motor for steering [19]. The optical position encoder M101B MEGATRON Elektronik AG & Co. [20] was used to determine the speed and direction.

In the era of automation, the demand for telepresence robots (TRs) in every field of life is increasing daily. The role of sensor selection and integration is the thematic area. The crucial point is to use various sensors—such as cameras, Lidar, radar, and ultrasonic sensors—to detect the best path and avoid obstacles [21]. The second area of research is the optimization of control algorithms. Scientists are introducing the latest methods to determine the path planning and trajectory of the robots accurately. These schemes are efficient in responding to the changes that occur in the environment. The methods are developed to record the data using the sensors to estimate the robot's state, and then to plan and execute safe trajectories [22].

The third research area is the human–robot interaction (HRI). A teleoperator often operates TRs, and these teleoperators do not have much experience with robotics. There is much deficiency in optimizing and minimizing this research gap. Scientists are investigating how to make the control interfaces for TRs more intuitive and easier to use, so as to enable more effective human–robot collaboration [23]. Many approaches are given in this section, which have already been utilized to assist in TRs' teleoperation. For a more detailed discussion of TRs and their relationship with communication delay, refer to [24].

In summary, current research is on the trajectory planning and experimental verification of TRs to avoid obstacles in an optimal manner. This is the main contribution of the proposed work. The objective is to improve the reliability and robustness of the obstacle detection and avoidance system, develop advanced control algorithms that can quickly and accurately respond to environmental changes, and make the control interfaces user-friendly [25].

Moreover, the kinematic and dynamic constraints require further optimization. Numerical optimization creates trajectories based on differentiated cost functions. Based on the trajectory planning parameters (e.g., position, speed, acceleration, and thrust) continuous trajectories are generated with the help of an optimization algorithm. On the other hand, further optimization will make the algorithm more complex, which adds limitations. The authors of [26] employed a deep deterministic policy gradient to learn self-driving in order to avoid obstacles with primitive actions. However, this approach has many limitations in terms of less precision in complex environments. The first example is jerky behavior; the second is that it becomes difficult to determine the cost function. In [27], the authors proposed real-time trajectory planning with less complexity and great precision.

Trajectory planning with Ackermann steering is a process that involves calculating a path for a vehicle to follow from its current position to a desired end position while taking into account the Ackermann steering geometry. Ackermann steering is a type of steering mechanism used in vehicles that allows the wheels to turn at different angles, depending on the desired turning radius. The trajectory planning process for a vehicle with Ackermann steering can be divided into several steps, which may include the following:

- Specify the initial and final conditions, such as the vehicle's starting position, orientation, and velocity, and the desired final position, orientation, and velocity.
- Definition of constraints and objectives, such as the maximum steering angle of the wheels, the allowable acceleration and deceleration, and the optimization criteria.
- Generation of a candidate trajectory that satisfies the constraints and objectives, using techniques such as model predictive control, polynomial curve fitting, or optimization algorithms.
- Evaluation and refinement of the trajectory, considering factors such as tire slip, lateral and longitudinal forces, and vehicle dynamics.
- Execution of the trajectory, using a feedback control system to track the planned trajectory and adjust for any deviations or disturbances.
- In trajectory planning with Ackermann steering, it is important to consider the steering system's limitations and constraints, such as the maximum steering angle and the turning radius. The trajectory planning algorithm must ensure that the vehicle can

follow the planned trajectory without exceeding these limits or causing any instability or loss of control.

- Overall, trajectory planning with Ackermann steering is a complex process that requires a deep understanding of vehicle dynamics, control theory, and optimization algorithms. It is an important tool for achieving accurate and efficient motion control in autonomous vehicles, robotic systems, and other applications requiring precise and agile motion.

2. Background

TRs' importance is vital and has remained a topic of interest among scientists over the last few decades. Implementing TRs in many application areas—e.g., industry, healthcare, and academia—is challenging. Much research has been carried out to achieve this task, but each approach has limitations. In the recent era, human beings' social lives have become dependent on technology. Although technologies have greatly improved the lifestyles of human beings, along with their workplaces and social gatherings, more investigations could be conducted to improve customer satisfaction and perform systematic analysis. TRs and autonomous vehicles (AVs) might be attractive alternatives in the human social ecosystem.

In [1], a remote manipulator was implemented, considered to be the pioneer in robotic arms. Implementing these TRs is useful in hazardous or COVID-19 environments that are inaccessible to humans [23]. In [24], the researchers developed TRs for offices, healthcare systems, and nursing homes. Another useful application is augmented virtual reality, which is useful to simulate the feeling of a human–robot interaction environment. In [25], immersed virtual reality was developed to provide guidelines for user design. Many challenges remain, including the adjustable height [26–30], motion along the slope surface, system stability, and low-speed control [12,31–39].

The well-known application areas of mobile robots include ocean exploration, approaching the moon, implementation in nuclear plants [40,41] and, recently, during COVID-19 [10,42,43]. It is often hard to conduct repairs in such scenarios; therefore, the alternate approach of a mobile robot to perform these tasks from remote locations is quite attractive. Moreover, the negotiation with the end consumer is condensed to mission provisions, and then automation can minimize mobile robots' communications with experts. There are many approaches that have been developed by researchers, especially for deciding and controlling the robots [34,35]. Another approach to defining the trajectory control using the software architecture of TRs was presented by [20,44–46]. Telepresence robots are utilized in many applications and have shown tremendous results in human–robot interaction [47,48]. A search approach was proposed by [49]. It is easy to satisfy the environmental constraints, but with low dynamic probability and high computational complexity. The authors of [50] proposed a lattice trajectory technique requiring non-uniform sampling. Another approach suggested by [51] uses the interpolation method, but the results are not optimal, and the track is not smooth.

A more general architecture of TRs is presented in [24,25]. A TR comprises both software and hardware architectures. It contains the hardware components, e.g., biosensors, which obtain information from the patient and send it to the consultant at a remote location using the available communications technology [6,10,36]. It also contains components that can produce the control inputs so that the robot can move in a stable position using the actuator connected to it. These actuators control the hardware actions, e.g., motion, speed, and position [37–39]. The present work focuses on the identification and stabilization of the TR and the development of microcontroller-based architectures. The fundamental responsibility of the design is to control the driving behavior and avoid obstacles throughout the whole journey of the TR. The design also contains the module with the best driving path, called the decision-making module. The function of this module is to provide the best path and ensure safe driving with obstacle avoidance control. The term "maneuver" is most likely utilized in the literature to describe path planning. Still, to ensure more clarity and consistency, the term "behaviour" is employed to label the whole journey in the present

research article. According to the activities generated by the mini-computer in computing, the other independent attributes—such as position, trajectory, orientation, and speed—are considered. More general techniques are presented in [52–55].

To determine the trajectory, much research has been carried out on the geometric and kinematic, dynamic, optimal, adaptive, model-based, and classical controllers [41,46,50]. The authors of [15] presented comprehensive surveys, and the motion prediction and risk estimator analysis were reported in [51]. A detailed overview of the control methods handling the uncertainties that occur due to the unseen environment is given in [41,45,53–55]. It is easy to conclude that each study works on a single component—such as motion planning, behavior planning, and trajectory control—rather than an integrated system [36,52]. The authors of [56] modeled the motion information for pedestrians and used the dimensional feature space to measure the positional relationship. A method of predicting the future positional information of vehicles and other mobile targets was developed in [57], and another intelligent approach was presented in [58].

3. System Design and Implementation

This section deals with the planning of a trajectory that the mobile robot can follow to reach the target point Z with the target orientation φZ. The point S is the starting point, the angle φS is the starting orientation, and r is the radius of a curve. The radii of the starting and target circles must be identical for the subsequent two geometric derivations of trajectory planning. This means that the steering angle of the vehicle's steering axle is the same in both cases. A common tangent connects the two circles. This ensures that the vehicle leaves or enters the circles with the correct orientation and does not have to be rotated around its center of gravity at this point, which would be impossible anyway due to the Ackermann drive.

3.1. Trajectory Planning (Clockwise)

When planning the trajectory, a distinction must first be made as to whether the start and finish circles are traversed in the same or opposite directions. First, the case where the circles are traversed in the same direction is considered. The geometric conditions for clockwise trajectory planning are shown in Figure 2. Clockwise means that the circles are traversed clockwise. However, shorter trajectories may arise due to a counterclockwise rotation. The controller must determine this on a case-by-case basis.

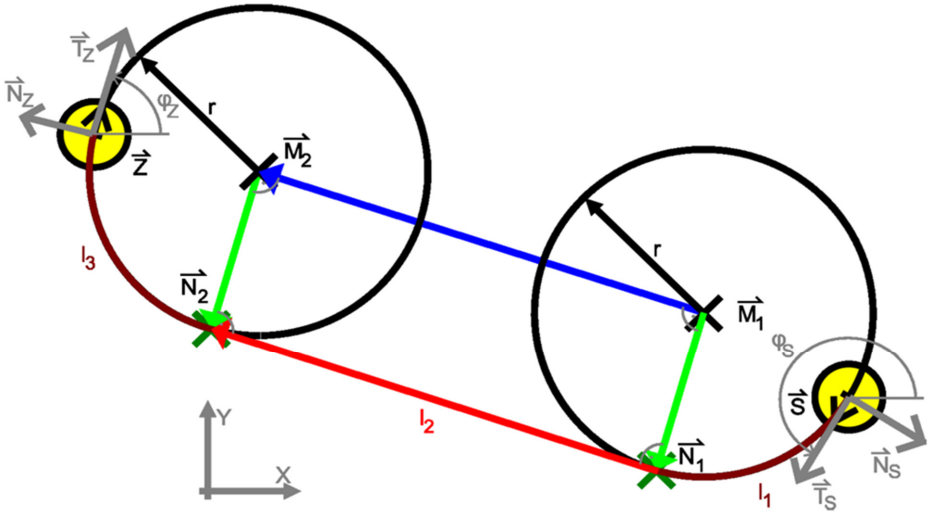


Figure 2. Sketch of the trajectory (clockwise).

The circle's radius r is specified before the calculation. Normally, a radius is selected that is slightly larger than the minimum radius that can be driven, so that the orientation control still has some reserve to compensate for disturbances.

In the following considerations, the x-axis is defined as the reference axis. This means that the angle of this axis is 0°. However, it should be noted that this does not correspond to the software implementation, as the reference axis is the y-axis because it points north. However, it can be calculated back and forth between these reference systems by simply adding or subtracting 90°.

First, the start and end points, the orientations at these points, and the drivable circle radius r are known. The centers of the start and finish circles must first be determined. The orientations φ describe the tangential direction of the associated circle in the associated points. Each of these points lies on the corresponding circle. The normal direction of the circle can now be determined from the tangential direction by rotating the tangent by 90°. The conditions are derived here at the starting point S. The tangent direction at this point is as follows:

$$\vec{T_S} = (\cos\phi_S \quad \sin\phi_S) \tag{1}$$

To maintain the normal, the tangent is rotated 90° in the mathematically positive sense (counterclockwise). This is a 2D rotation matrix $R(\varphi)$.

$$R_{(\phi)} = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} \tag{2}$$

Results for a rotation angle of $\varphi = 90°$:

$$R_{(90°)} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \tag{3}$$

Applying this matrix to the tangent at the starting point, we obtain the normal

$$\vec{N_S} = \vec{T_S} \cdot R_{(90°)} = (\cos\phi_S \quad \sin\phi_S) \cdot \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} = (-\sin\phi_S \quad \cos\phi_S) \tag{4}$$

The normal N_s has the length 1, i.e., it is already normalized. Now, the circle will go through to the right, which corresponds to a mathematically negative direction, so the circle's center point M_1 of the reset circuit is in exactly the opposite direction of a normal vector N_s at a distance of radius r from the starting point, and S is calculated. Accordingly, the circle center point M_1 of the reset circuit is as follows:

$$\vec{M_1} = \vec{S} - r \cdot \vec{N_S} = (S_x \quad S_y) + r \cdot (\sin\phi_S \quad -\cos\phi_S) \tag{5}$$

All of these considerations apply by analogy to the target circle. There is the center of the circle at a right turn to M_2:

$$\vec{M_2} = \vec{Z} - r \cdot \vec{N_Z} = (Z_x \quad Z_y) + r \cdot (\sin\phi_Z \quad -\cos\phi_Z) \tag{6}$$

Now, the connection vector between the two circle center points M_1 and M_2 is required. This is shown in the picture in blue and is calculated as follows:

$$\vec{M_d} = \vec{M_2} - \vec{M_1} = \vec{Z} - \vec{S} + r \cdot (\sin\phi_Z - \sin\phi_S \quad \cos\phi_S - \cos\phi_Z) \tag{7}$$

$$\vec{M_d} = (Z_x - S_x + r \cdot (\sin\phi_Z - \sin\phi_S) \quad Z_y - S_y + r \cdot (\cos\phi_S - \cos\phi_Z)) \tag{8}$$

The normal vectors N_1 and N_2 are both orthogonal to the vector connecting the two circle centers, and since they are unit vectors, they each have a length of 1. Therefore, they are both the same. They can be determined over a 90° rotation in the mathematically positive sense of the connection vector M_d and subsequent normalization.

$$\vec{N_1} = \vec{N_2} = \frac{\vec{M_d}}{|\vec{M_d}|} \cdot R_{(90^\circ)} \tag{9}$$

$$\vec{N_1} = \vec{N_2} = \frac{\begin{pmatrix} Z_x - S_x + r \cdot (\sin\phi_Z - \sin\phi_S) & Z_y - S_y + r \cdot (\cos\phi_S - \cos\phi_Z) \end{pmatrix}}{\sqrt{(Z_x - S_x + r \cdot (\sin\phi_Z - \sin\phi_S))^2 + (Z_y - S_y + r \cdot (\cos\phi_S - \cos\phi_Z))^2}} \cdot \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \tag{10}$$

$$\vec{N_1} = \vec{N_2} = \frac{\begin{pmatrix} S_y - Z_y + r \cdot (\cos\phi_Z - \cos\phi_S) & Z_x - S_x + r \cdot (\sin\phi_Z - \sin\phi_S) \end{pmatrix}}{\sqrt{(Z_x - S_x + r \cdot (\sin\phi_Z - \sin\phi_S))^2 + (Z_y - S_y + r \cdot (\cos\phi_S - \cos\phi_Z))^2}} \tag{11}$$

With the connecting vector M_d around the circle radius r in the above-determined normal direction N, we can obtain the exact part of the desired trajectory on which the vehicle travels straight ahead. This line segment is nestled at the beginning and end of the two circles at the start and goal circle. The straight line represents one of the two common tangents of two circles. However, only the tangent can be calculated here in a forward direction to enter and leave, because both run by a right curve.

The other tangent would have to be traversed in the reverse direction, which is undesirable. It is worth mentioning again that both parties in the same direction must be traversed by the method shown here, so that the vectors spanning M_d, N_1, and N_2 form a rectangle. Next, a quality criterion for the observed line can be determined. For this purpose, the length of the trajectory is considered. Since the trajectory is composed of three sections, these sections are first considered individually. This begins with the circular area on the home circle. The required length of the arc length l_1 corresponds to the starting circuit from the starting point S of the vehicle to the exit point B_1. It should be noted that the circle is traversed by a right-hand bend in the mathematically negative sense. Therefore, the angle is swept by the arc $\varphi_S - \varphi_{M_d}$. The angle φ_{M_d} is the pitch angle of the vector M_d and is calculated as follows:

$$\phi_{M_d} = atan\left(M_{dy}, M_{dx}\right) \tag{12}$$

It must finally be ensured that the swept angle $\varphi_S - \varphi_{M_d}$ is in the range between 0 and 360 degrees. To determine the length l_1,

$$l_1 = r \cdot \left[(\phi_S - \phi_{M_d}) mod(2\pi)\right] \tag{13}$$

The length of l_2 is the straight line connecting the two circles to the length of the corresponding vector M_d.

$$l_2 = |\vec{M_d}| \tag{14}$$

Now, even the length l_3 is missing on the target circle. This is calculated analogously to the length l_1 on the home circle, where the swept angle for a right turn is $\varphi_{M_d} - \varphi_Z$.

$$l_3 = r \cdot \left[(\phi_{M_d} - \phi_Z) mod(2\pi)\right] \tag{15}$$

The trajectory length l is now the sum of the last three computed lengths.

$$l = l_1 + l_2 + l_3 = r \cdot [(\phi_S - \phi_{M_d}) mod(2\pi) + (\phi_{M_d} - \phi_Z) mod(2\pi)] + |\vec{M_d}| \tag{16}$$

The trajectory has been derived for a right turn in the above derivation. The angle in the mathematically negative sense of rotation is painted over, so the circle centers are opposite to the normal vectors. In contrast, a left-hander will go through the angle in the mathematically positive sense, so the circle's center point here is in the direction of the normal vectors. Moreover, the sign of the angular change is important in calculating the quality criterion. The derivation of a left-hander should not be performed, because it can be carried out similarly to the derivation of the right-hander. Instead, a sequence of steps

are now specified, as are all of the necessary points, and the trajectory length is calculated automatically. Algorithm 1:

Algorithm 1 Selection of Trajectory (Smallest Length)

1: *Initialization* : Given the coordinates and the orientation of the starting point S_x, S_y, and φ_S and the target point Z_x, Z_y, and φ_Z, in addition to moving the curve, the radius r is given. Steps 1 through 5 must be carried out for both a left turn and a right turn. The upper operator describes the case of multiple operators and the equation for a curve to the left, and the bottom one is the equation for a right turn,

2: Determination of the circle center of the circle, starting with the coordinates M_{1x} and M_{1y}.

$$M_{1x} = S_x \mp r \cdot \sin(\phi_S) \quad (17)$$

$$M_{1y} = S_y \pm r \cdot \cos(\phi_S) \quad (18)$$

3: Determination of the center of the goal circle with coordinates M_{2x} and M_{2y}.

$$M_{2x} = Z_x \mp r \cdot \sin(\phi_Z) \quad (19)$$

$$M_{2y} = Z_y \pm r \cdot \cos(\phi_Z)$$

4: Determination of the vector connecting the coordinates M_d, M_{dx}, and M_{dy} between the two circle center points M_1 and M_2, and its magnitude $|Md|$, φ_{M_d} and direction.

$$M_{dx} = Z_x - S_x \pm r \cdot (\sin\phi_S - \sin\phi_Z) \quad (20)$$

$$M_{dy} = Z_y - S_y \pm r \cdot (\cos\phi_Z - \cos\phi_S) \quad (21)$$

$$| M_d | = \sqrt{M_{dx}^2 + M_{dy}^2} \quad (22)$$

$$\phi_{M_d} = atan2\left(M_{dy}, M_{dx}\right) \quad (23)$$

5: Determination of the B_1 exit point from the start circle with coordinates B_{1x}, B_{1y}, and B_2 of the entry point into the goal circle with coordinates B_{2x} and B_{2y}.

$$B_{1x} = M_{1x} \pm \tfrac{r}{|M_d|} \cdot M_{dy} \quad (24)$$

$$B_{1y} = M_{1y} \mp \tfrac{r}{|M_d|} \cdot M_{dx} \quad (25)$$

$$B_{2x} = M_{2x} \pm \tfrac{r}{|M_d|} \cdot M_{dy} \quad (26)$$

$$B_{2y} = M_{2y} \mp \tfrac{r}{|M_d|} \cdot M_{dx} \quad (27)$$

6: Determination of the trajectory length l.

$$l = r \cdot \left[(\pm\phi_{M_d} \mp \phi_S) mod(2\pi) + (\pm\phi_Z \mp \phi_{M_d}) mod(2\pi)\right] + | M_d | \quad (28)$$

7: Selection of the trajectory with the smallest length l

3.2. Trajectory Planning (Counterclockwise)

However, there are cases in which the trajectory planning already considered delivers suboptimal results, since it only allows a common direction of rotation for the two circles used. As a result, large correction paths can become necessary with certain pose constellations, which take up much space for driving. To avoid this problem, the trajectory planning for opposite rotation directions of the circles must now be considered. Almost ideal paths can be calculated by combining both methods for trajectory planning. Figure 3 shows the geometric conditions for a trajectory where the start circle is clockwise and the target circle is counterclockwise. Figure 3 can be used to derive the basis for planning.

The determination of the circle centers M_1 and M_2 is almost identical to the previous derivation with the same direction of the circle. The only difference is that, in this case, the target circle is run through to the left; therefore, the center point for a left turn must also be determined. This only changes a few signs. The determination of the connecting line M_d also remains the same, *i*; that is, it is still the connecting vector between the starting circle center M_1 and the ending circle center M_2. A significant change occurs when looking at the connecting line (here: red) between the two circles. It no longer runs parallel to the connecting vector M_d. However, together with the two normal vectors N_1 and N_2, they span two right-angled triangles. The two interior angles κ of the two triangles at the common point of contact are the vertex angles of two intersecting straight lines. Thus, they are the same size. Since every triangle has a right angle, and the sum of the three interior angles is the same, the two angles marked with μ must also be the same. Since the adjacent sides of the two angles μ also have the length r and are, therefore, of the same size, the two triangles are equal. Therefore, the connecting vector M_d is cut exactly in the

middle by the red connecting line. The hypotenuses of the two triangles thus have the length $M_d/2$. The angle μ can be calculated using trigonometry on a right-angled triangle.

$$\mu = \arccos\left(\frac{r}{\frac{|\vec{M_d}|}{2}}\right) = \arccos\left(\frac{2 \cdot r}{|\vec{M_d}|}\right) \tag{29}$$

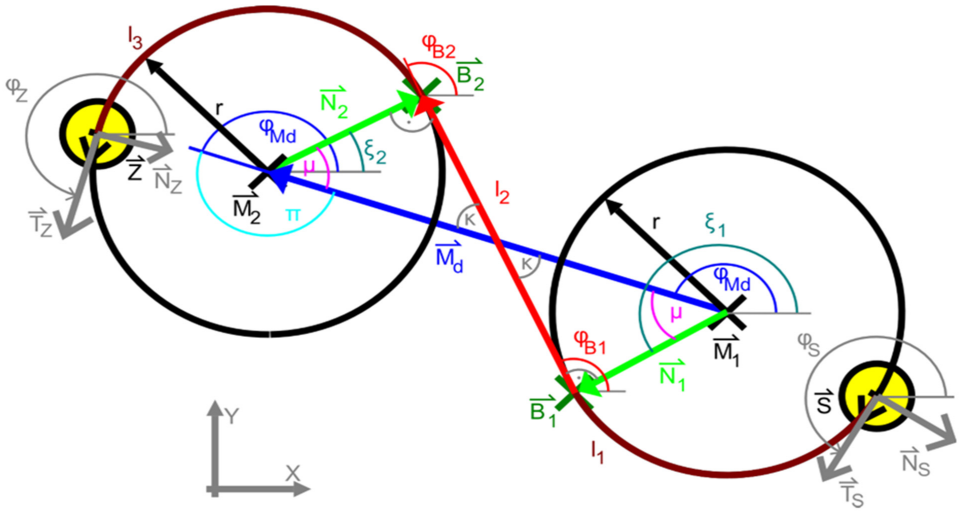


Figure 3. Circle trajectory with alternating flow direction (right- and left-handed).

It should be noted that for a valid solution, the arccosine argument must be between -1 and 1. Otherwise, there is no real solution for the angle μ, and the given geometric problem is unsolvable. This can happen if, for example, the two circles overlap.

An auxiliary angle ξ_1 is now introduced, which is not shown in Figure 3. In this case, it is the sum of the angle φM_d of the connection vector M_d and the angle μ.

$$\xi_1 = \phi_{M_d} + \mu = \phi_{M_d} + \cos\left(\frac{2 \cdot r}{|\vec{M_d}|}\right) \tag{30}$$

The direction of the normal vector N_1 can now be determined with this auxiliary angle, since this angle rotates it exactly ξ_1 from the x-axis of the absolute coordinate system. The normal vector N_2 is the inverse vector of the vector N_1.

$$\vec{N_1} = (\cos\xi_1 \quad \sin\xi_1) = -\vec{N_2} \tag{31}$$

The points B_1 and B_2 are away from the respective circle center by the distance r in the corresponding normal direction. Thus, for these points,

$$\vec{B_i} = \vec{M_i} + r \cdot \vec{N_i} \tag{32}$$

These points represent the circle's entry or exit point. The orientation that the robot should have in these points is given by the angles φB_1 and φB_2. These two angles are equal because they are both the tangent angle of the red connecting line. The two angles are rotated by $\pi/2$ in the mathematically negative sense against the normal vector N_1, which has the direction ξ_1. This results in φ_{Bi} for the tangent angle:

$$\phi_{Bi} = \xi_1 - \frac{\pi}{2} = \phi_{Md} + \mu - \frac{\pi}{2} \tag{33}$$

Alternatively, the tangent angles φ_{Bi} can also be determined using the target circle. The angles μ and φ_{M_d} remain the same. If we look at the angles in the target circle in Figure 3, we can set up the following summation equation:

$$\phi_{Md} + \pi + \mu = \xi_2 + 2\pi \tag{34}$$

where π represents a semicircle in radians. If we solve the equation for ξ_2, we can get

$$\xi_2 = \phi_{Md} + \mu - \pi \tag{35}$$

The tangent angles φ_{Bi} are rotated by $\pi/2$ in the mathematically positive sense by the angle ξ_2. Thus,

$$\phi_{Bi} = \phi_{Md} + \mu - \pi + \frac{\pi}{2} = \phi_{Md} + \mu - \frac{\pi}{2} \tag{36}$$

which corresponds exactly to Equation (33). This means that we obtain the same result with both methods of calculation.

Now, all that is missing is the determination of the quality criterion, i.e., to evaluate the trajectory length. However, the determination is again analogous to the previous derivation of circular cycles in the same direction. For this reason, the procedure is not described again here. It should also be noted that the derivation for the inverse case of the directions of rotation is analogous, but some signs are reversed. For completeness, Algorithm 2 is developed for both directions of rotation.

Algorithm 2 Trajectory (for both directions of rotation)

1: *Initialization* : The coordinates and the orientation of the starting point S_x, S_y, and φS and the target point Z_x, Z_y, and φZ are given. In addition, the curve radius r to be driven is given. Steps 1 to 6 must be carried out for both a left/right turn and a right/left turn. In the case of multiple operators, the upper operator describes the equation for a left/right curve, while the lower one describes the equation for a right/left curve,

2: Determination of the center of the starting circle with the coordinates M_{1x} and M_{1y}.

$$M_{1\mathrm{x}} = S_x \mp r \cdot \sin(\phi_S) \tag{37}$$
$$M_{1\mathrm{y}} = S_y \pm r \cdot \cos(\phi_S) \tag{38}$$

3: Determination of the center of the goal circle with the coordinates M_{2x} and M_{2y}.

$$M_{2\mathrm{x}} = Z_x \pm r \cdot \sin(\phi_Z) \tag{39}$$
$$M_{2\mathrm{y}} = Z_y \mp r \cdot \cos(\phi_Z)$$

4: Determination of the vector connecting the coordinates M_d, M_{dx}, and M_{dy} between the two circle center points M_1 and M_2, and its magnitude $|Md|$, φM_d and direction.

$$M_{dx} = M_{2\mathrm{x}} - M_{1\mathrm{x}} \tag{40}$$
$$M_{dy} = M_{2\mathrm{y}} - M_{1\mathrm{y}} \tag{41}$$
$$\mid M_d \mid = \sqrt{M_{dx}^2 + M_{dy}^2} \tag{42}$$
$$\phi_{M_d} = atan2\left(M_{dy}, M_{dx}\right) \tag{43}$$

5: If the result for the angle μ is not real, the calculation for the given direction of rotation can be aborted here, since the problem cannot be solved.

$$\mu = \mathrm{acos}\left(\frac{2 \cdot r}{|\vec{M}_d|}\right) \tag{44}$$

Determination of the angles μ and ξ_1, as well as the normal vector N_1 with the coordinates N_{1x} and N_{1y}.

$$\xi_1 = \phi_{M_d} \mp \mu \tag{45}$$
$$N_{1\mathrm{x}} = \cos\xi_1 \tag{46}$$
$$N_{1\mathrm{y}} = \sin\xi_1 \tag{47}$$

6: Determination of the exit point B_1 from the starting circle with the coordinates B_{1x} and B_{2x}, and the entry point B_2 in the target circle with the coordinates B_{2x} and B_{2y}, as well as the distance $|B_{1\to2}|$ between the two points.

$$B_{1\mathrm{x}} = M_{1\mathrm{x}} + \tfrac{r}{|M_d|} \cdot M_{dy} \tag{48}$$
$$B_{1\mathrm{y}} = M_{1\mathrm{y}} + \tfrac{r}{|M_d|} \cdot M_{dx} \tag{49}$$
$$B_{2\mathrm{x}} = M_{2\mathrm{x}} - \tfrac{r}{|M_d|} \cdot M_{dy} \tag{50}$$
$$B_{2\mathrm{y}} = M_{2\mathrm{y}} - \tfrac{r}{|M_d|} \cdot M_{dx} \tag{51}$$
$$\mid B_{1\to2} \mid = \sqrt{\left(B_{2\mathrm{x}} - B_{1\mathrm{x}}\right)^2 + \left(B_{2\mathrm{y}} - B_{1\mathrm{y}}\right)^2} \tag{52}$$

7: Step 6: Determination of the trajectory length l(quality criterion).

$$l = r \cdot \left[\left(\pm\xi_1 \pm \tfrac{\pi}{2} \mp \phi_S\right) mod(2\pi) + \left(\pm\xi_1 \pm \tfrac{\pi}{2} \mp \phi_Z\right) mod(2\pi)\right] + \mid B_{1\to2} \mid \tag{53}$$

8: Selecting the trajectory with the smallest length l.

Finally, after applying Algorithms 1 and 2, the optimal trajectory can be obtained for each method. The optimal trajectory, i.e., the shortest overall length, can now be determined using the quality criterion.

4. Hardware Implementation

The robot's actuators are controlled at the hardware control level. These are the drive train on the one hand and the steering on the other. The idea of this level is to control the actuators in such a way that they behave almost ideally from the point of view of the higher layers. For example, the robot should drive at a defined speed. To ensure this, the rotary encoder in the drive train must be evaluated, and the deviation of its signal from the setpoint must be included in the manipulated variable, i.e., the voltage of the drive motor. Basic control elements such as PT_1 delay elements and PID controllers are required for the controls. These are also defined at this level but can be used by higher levels. In the actual hardware control, only the speed of the drive train is currently controlled to a set point. For this purpose, deriving the position value of the optical incremental rotary encoder in the drive train serves as the actual value supplier. However, since it is subject to strong quantization noise, the value is first smoothed by a PT_1 element. Figure 4 shows the signal flow diagram of the powertrain control.

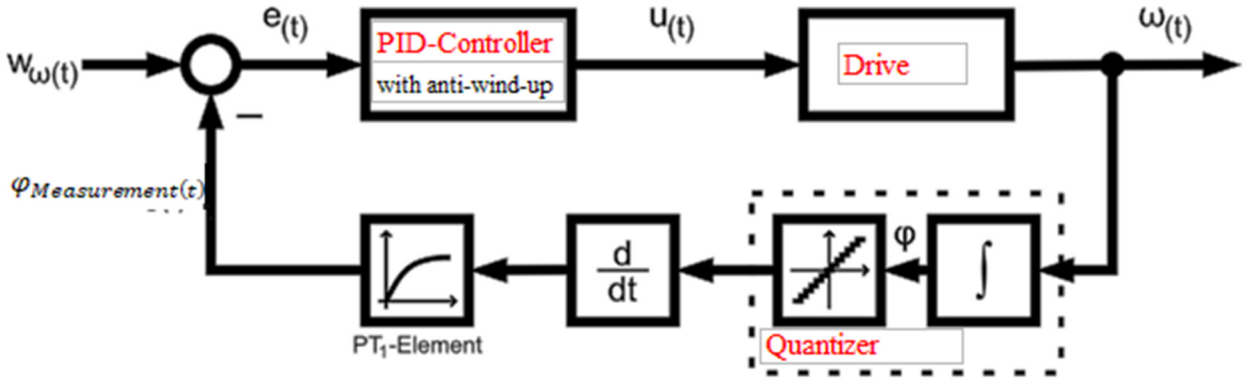


Figure 4. The speed control circuit of the drive train.

The PID controller shown is a PID controller with anti-windup feedback. The controlled system is the complete drive train, consisting of the motor bridge, the drive motors, the gearbox and the tires, and the inertia of the entire robot. The motor bridge is connected to the controller. The actual speed $\omega(t)$ is first integrated into a position by the incremental encoder and then quantized to integer values. This quantized position signal must be differentiated again to obtain the most original speed. Since the signal is very noisy due to the quantization, it must first be filtered. For this purpose, a PT_1 element is inserted in the software filter for the noisy measured values. The control error $e(t)$ is formed from the difference between the target value $w_{\omega(t)}$ and the filtered measurement signal $\omega_{measurement(t)}$.

The steering angle is not regulated because there is no steering angle sensor. Therefore, the steering is only controlled. However, it should be noted here that the relationship between the steering value (i.e., the numerical value that sets the steering position in the microcontroller program) and the deflection of the wheels is nonlinear. However, it can be assumed that this relationship is monotonic, which means that an increase in the steering value also means an increase in the wheel deflection.

4.1. State Control Level

The robot's position and orientation control are carried out using the state control level. The robot should approach certain points under boundary conditions with a defined orientation. A mathematical determination of the points does not occur here, but they are made available by a higher authority. Obstacles are also taken into account at this level.

Various functions are available for the robot's different operating modes (e.g., manual mode, trajectory mode). At the state control level, it must be noted that the robot's state (i.e., its position and orientation) is controlled here. This has nothing to do with state controllers in the classic sense. Digital PID controllers or PI controllers are still used here as controllers.

4.1.1. Manual Operation

In manual mode, the set value for the drive speed is first calculated from the parameters transmitted by the mini-computer. Then, this manipulated variable and the steering value that is also transmitted are transferred to the hardware control level.

4.1.2. Trajectory Mode

The process in trajectory mode is significantly more complicated than in manual mode. A simple parallel simulation of the trajectory run is performed in the microcontroller to generate the target values for position and orientation. The drive speed is changed in a specified trapezoidal or triangular profile. This speed is integrated into the target position. The desired speed profile is mathematically

$$\omega_{(t)} = \begin{array}{cc} \dot{\omega} \cdot t & t \leq t_{AD} \\ \dot{\omega} \cdot t_{AD} & t_{AD} < t \leq t_{SD} \\ \dot{\omega} \cdot (t_{SD} - t) & t > t_{SD} \end{array} \tag{54}$$

The target position is the integration of the target speed

$$\phi_{(t)} = \int_{\tau=0}^{t} \omega_{(\tau)} d\tau. \tag{55}$$

If the simulation has reached the target, the position value is frozen. Suppose that the robot's position only deviates by a maximum error when the position value is frozen; in this case, the trajectory is considered to have been completed successfully, and the robot reports this to the mini-computer, which then transmits new target coordinates if necessary.

4.2. Constant of Proportionality

This chapter briefly explains how the constant of proportionality for the conversion between the unit encoder pulses (EncImp) and meters (m) was determined. The robot drove straight ahead in trajectory mode at 10,000 EncImp. The start and end points were marked, and the route was measured afterwards. The proportionality was determined as follows:

$$x_{[\text{EncImp}]} = k \cdot x_{[\text{m}]} \tag{56}$$

where the proportionality constant k can be determined. The length of the distance traveled, measured with a ruler, was 5.8 m in the test. The constant of proportionality k was determined accordingly using

$$k = \frac{x_{[\text{EncImp}]}}{x_{[\text{m}]}} = \frac{10,000\text{EncImp}}{5.8\text{m}} = 1724\frac{\text{EncImp}}{\text{m}}. \tag{57}$$

5. Results and Discussion

5.1. Driving Straight Ahead

Next, partial trajectories are examined. First, straight-ahead driving is considered, with both the position and the orientation being controlled. Figure 5 shows the distance covered and the orientation of the robot. Figure 6 shows the position of the robot in the X–Y plane. In the graphs, it should be noted that the measured values were not necessarily recorded evenly. Therefore, the orientation curve is slightly jagged. During the test drive in the laboratory, the robot covered a distance of 4 m in a northeast direction. The orientation for this direction is around $-57°$.

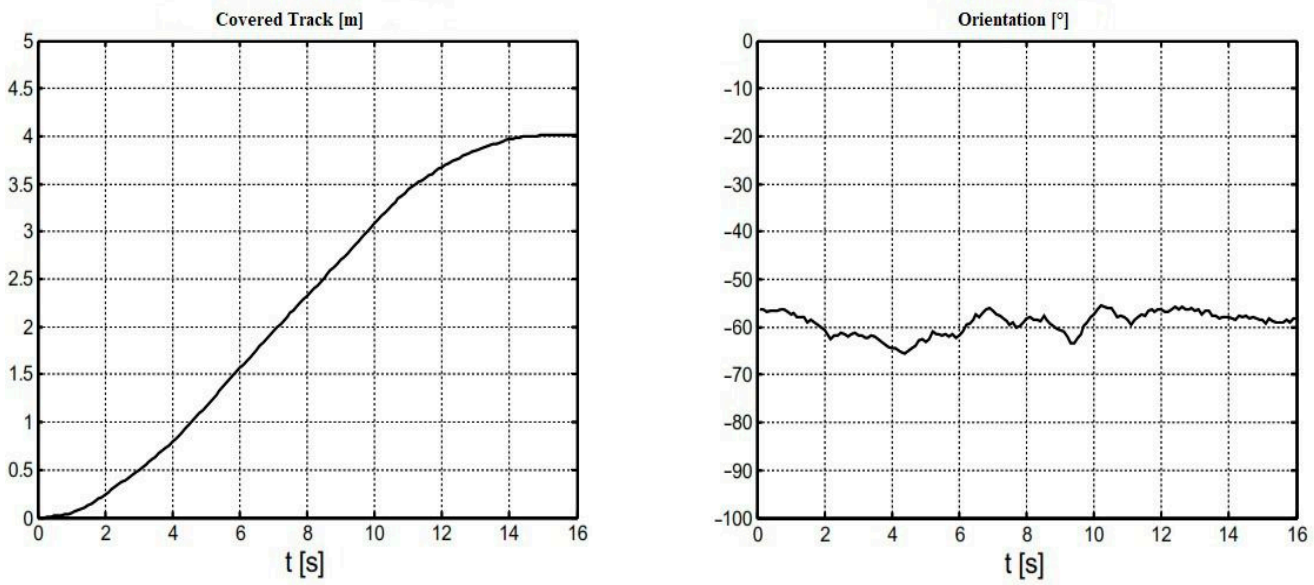


Figure 5. Distance covered and orientation of the robot.

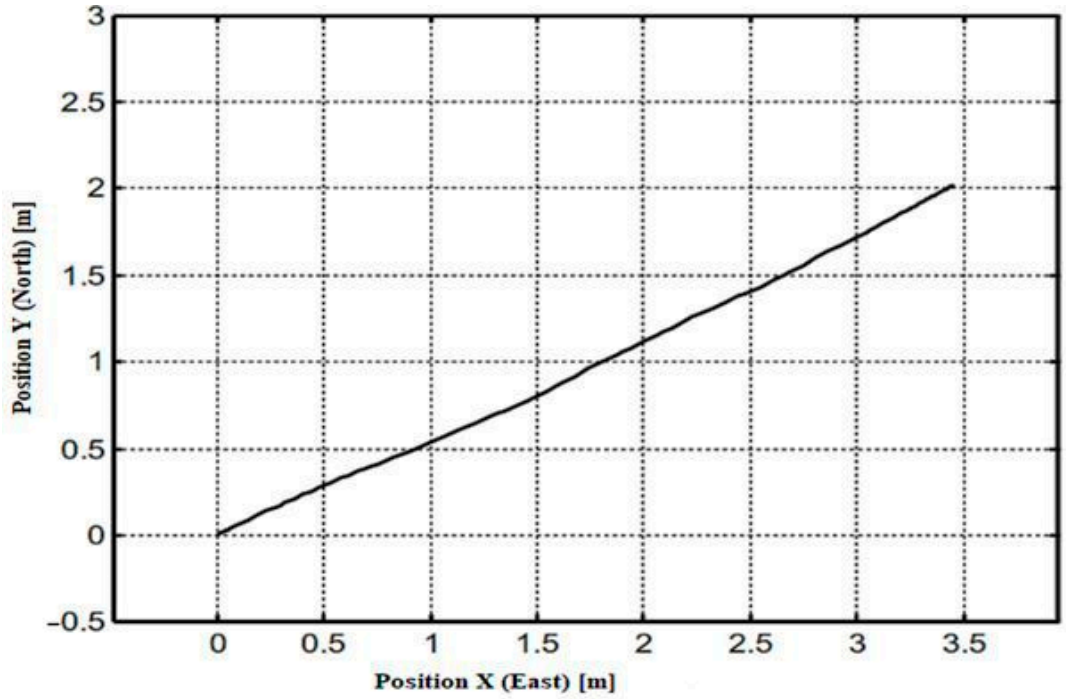


Figure 6. Position of the robot in the X-Y plane.

The graph of the distance traveled shows the typical S-curve that a trapezoidal speed profile creates. The end value of 4 m = 6896 EncImp is reached very well. The orientation fluctuates around the $-57°$ mark, i.e., the target value. All in all, the deviation from the target value is quite small, at around 10 degrees. In particular, the steering is nonlinear, the zero point is not exactly correct and, to make matters worse, it is stiff. This, of course, makes it extremely difficult to regulate the orientation. However, looking at the trajectory driven, we can see that the robot moved in a very straight line. The length of the trajectory also corresponds to the desired 4 m. The controllers have thus proven their ability in this scenario.

5.2. Right Turn

In the next scenario, simple cornering is considered. This is a right-hand curve, with an angle of 84° (−64° . . . −148°), swept over a distance of 3 m = 5172 EncImp. This roughly corresponds to one-quarter of a circle. Figure 7 shows the distance covered, the target, and the actual value of the robot's orientation. Figure 8, on the other hand, shows the driven trajectory in the X–Y plane—the circle on which the robot moves was reconstructed from three randomly selected trajectory points.

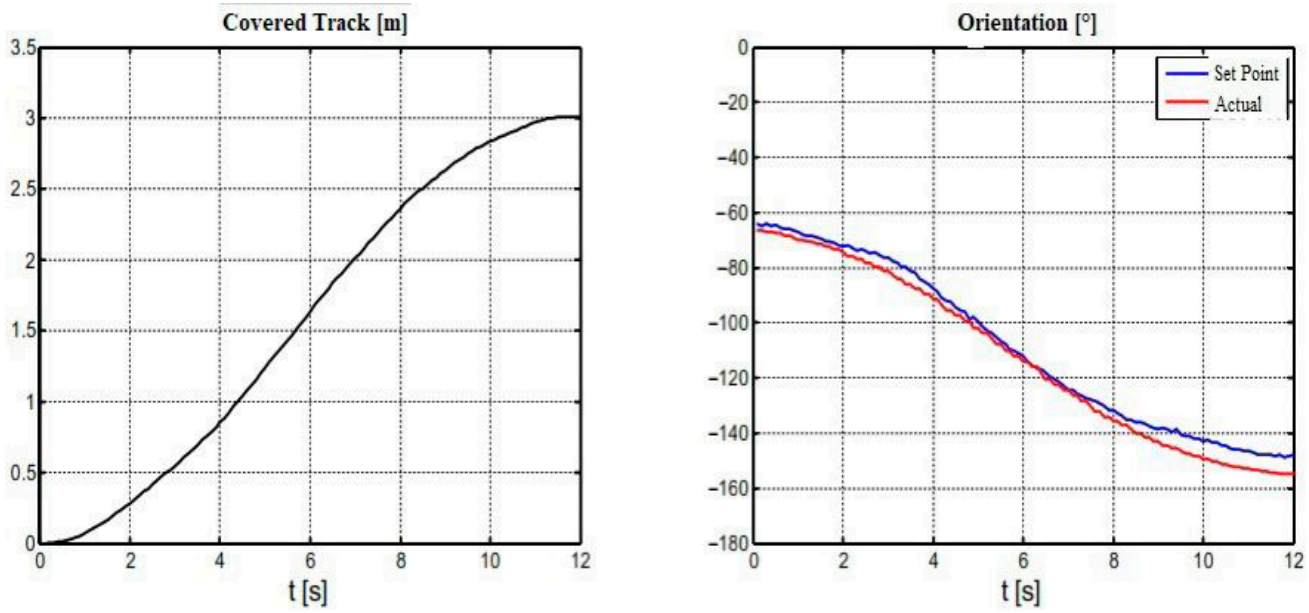


Figure 7. Target and actual value of the robot's orientation.

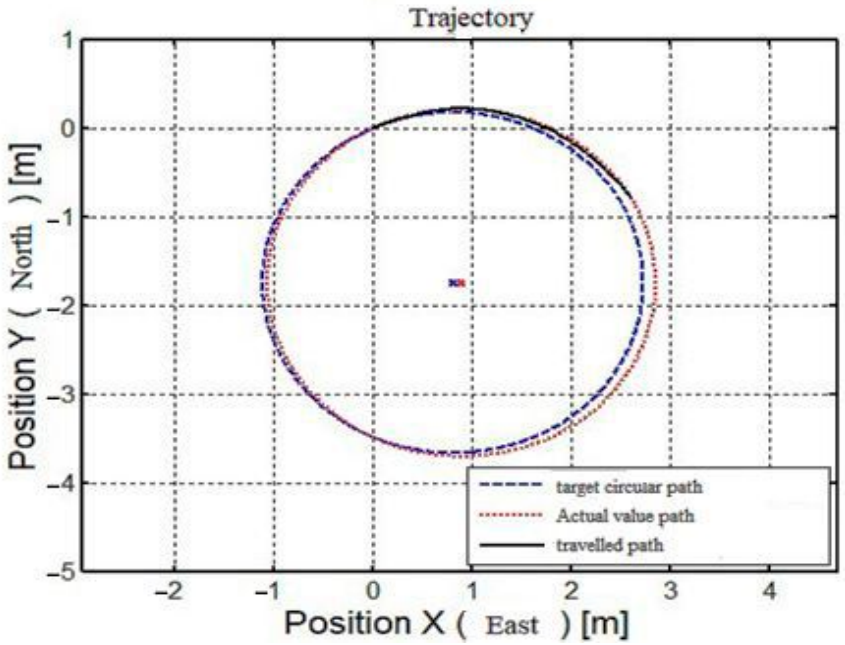


Figure 8. Driven trajectory in the X-Y plane.

If we look at the graph of the distance covered, there is no significant difference from driving straight ahead. The same S-shape can be seen again. The orientation, on the other hand, looks very different. If we first look at the target value, we can guess that it changes

linearly with the distance covered. It has the same S pattern as the position graph. The actual value curve follows the setpoint curve quite well.

If we look at the trajectory that was driven (black line), we can see that the robot followed a very nice circular path. The subsequently calculated red circle, calculated from three randomly selected trajectory points, serves as a comparison in the diagram. It deviates only slightly from the blue target circle. Thus, the controllers of the robot seem to work well in principle.

5.3. Self-Localization

After automatically calibrating the magnetometer, the robot was driven in a circle across the corridor of the EE building of the University of King Abdulaziz. One test was conducted clockwise, and another was conducted counterclockwise. A Google Maps aerial photograph was used to record the positions of the resulting trajectories. The trajectory was recorded by considering the aspect ratio and orientation of the building. The result of the self-localization is shown in Figure 9.

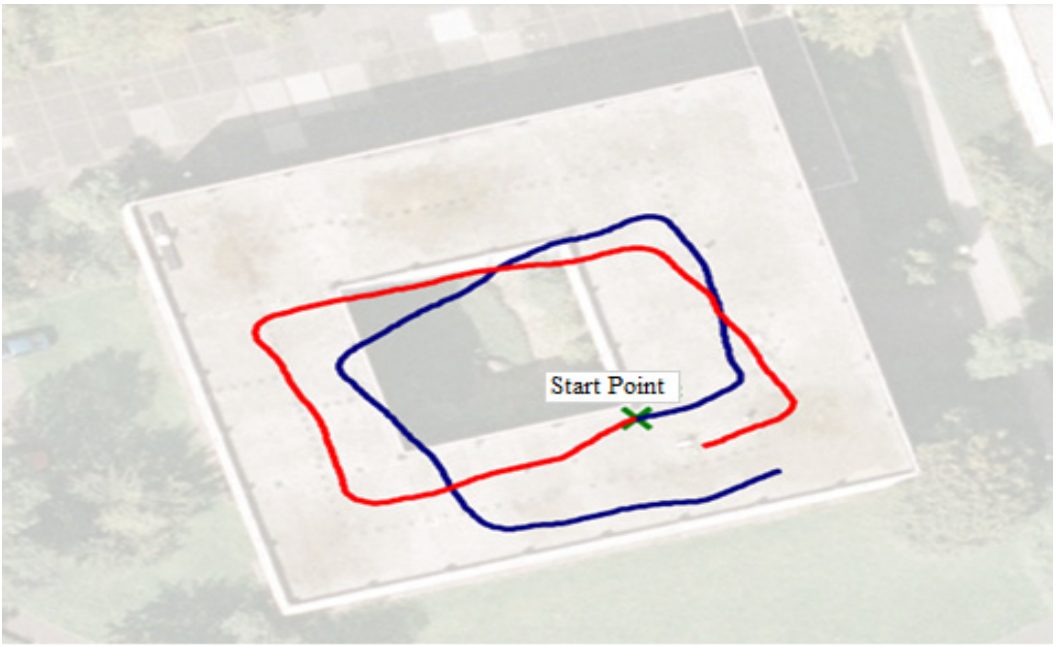


Figure 9. Self-estimates of robot position.

Figure 9 shows red and blue colors, and each color represents a trajectory direction-wise—either clockwise or anticlockwise. The red measurement is recorded for the clockwise trajectory, and the blue color trajectory denotes the counterclockwise direction. From the results, it can be seen that there is a small deviation in each trajectory. This deviation is due to the accumulation of past values in the present value. To perform further analysis, the error accumulation keeps increasing, and the results deviate from the present value. However, the error can be further reduced through more calibration and control, but the residual value improvement is not that much, and the computational complexity continues increasing. The results also show that the first three tracks are exactly parallel to one another, and a slight deviation is found in the fourth track, which can be improved.

As mentioned above, there are many causes of errors in the trajectories. The first reason could be poor magnetometer calibration. The variation in the calibration is due to the rotation of the angle or other interferences. The interference could be due to other sources of interference in the building, which can cause deviation. Examples of these sources could include the transmission lines, waves from undesired interfering sources, and the railing stairs. The experimental data are given in Table 1.

Table 1. Experimental data for different trials.

Examples	Trial 1	Trial 2	Trial 3	Mean
	Distance (m)/Time (s)	Distance (m)/Time (s)	Distance (m)/Time(s)	Distance (m)/Time (s)
1st round	8.67 m/92 s	9.79 m/98 s	9.13 m/95 s	9.2 m/95 s
2nd round	9.89 m/90 s	9.13 m/87 s	8.13 m/88 s	9.0 m/88 s
3rd round	9.13 m/98 s	9.05 m/89 s	9.05 m/75 s	9.0 m/87 s
4th round	8.44 m/87 s	9.84 m/72 s	9.94 m/61 s	9.3 m/73 s

The results validate the usability of our approach and show auto-MERLIN to be a ready robot for short- and long-term tasks, showing better results than using a default system—particularly when deployed in highly interactive scenarios. The results show that the proposed approach is less complex and provides better trajectory planning accuracy. An experiment was conducted to compare the proposed approach with the A-star algorithm. The results showed that the proposed system better covers the distance from the doctor's office to patient room A. The cumulative distance covered by each approach is noted in Table 2.

Table 2. Comparison of A-star with the proposed approach.

Algorithms	Examples	Trial 1	Trial 2	Trial 3	Mean
		Distance (m)/Time (s)	Distance (m)/Time (s)	Distance (m)/Time (s)	Distance (m)/Time (s)
A-star	1st round	8.85 m/91 s	9.79 m/99 s	9.15 m/95 s	9.25 m/95 s
	2nd round	9.95 m/90 s	9.15 m/87 s	8.15 m/88 s	9.10 m/88 s
Developed	1st round	8.15 m/84 s	8.85 m/89 s	7.75 m/85 s	8.25 m/86 s
	2nd round	7.85 m/86 s	7.55 m/87 s	8.15 m/88 s	7.85 m/87 s

Gazebo is an open-source 3D robotics simulator that can be used to simulate the different environments over which robots navigate. A Gazebo simulation framework was used with the robot operating system (ROS) integration for Lidar sensor configuration, and the results are shown in Figure 10. The model was trained in four steps using our suggested multistage training-mechanism-based technique. Utilizing the Gazebo simulation framework with the help of ROS integration, we created four training scenarios, as shown in Figure 10. Each of the simulation environments corresponds to one of the training phases. Our customized TR was chosen to train our model, and its starting position was predetermined. In this simulation environment, the navigation success rate (NSR) was evaluated. The NSR is the TR's success rate in navigating toward its destination without collisions. The estimated NSR was calculated as an average of 500 navigations. The navigation of the TR was successfully calculated, and the experiment is shown in Figure 10. In each of the given frames, an environment is shown and the robot's navigation can be observed. In the scenarios, the robot was successfully maneuvered, and 3D results were obtained with the help of Lidar.

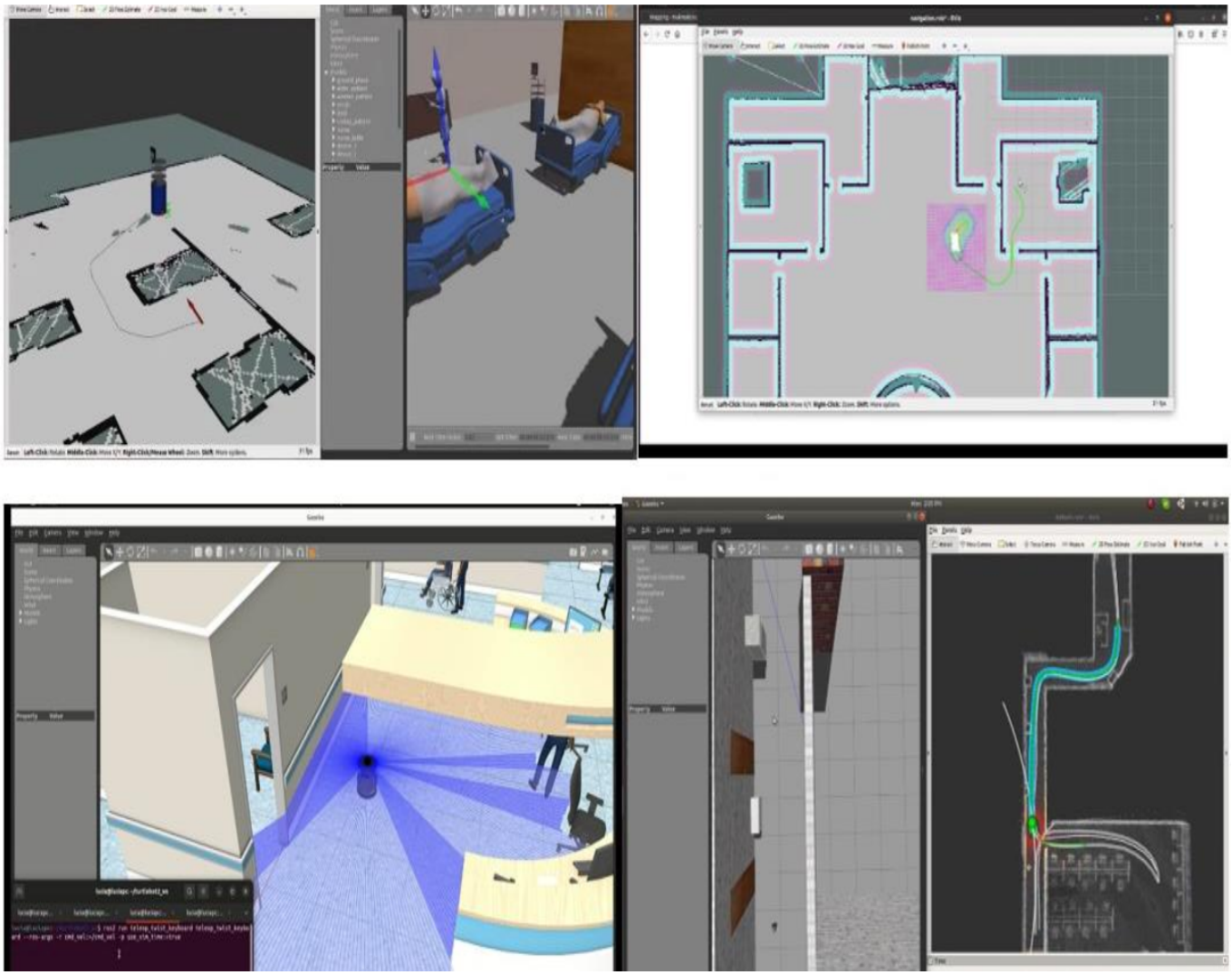


Figure 10. Different environmental simulations using Gazebo software.

6. Conclusions

A control board and a steering servo motor voltage regulator are also required to maneuver the robot successfully. The purpose of this research was to develop a well-maintained telepresence robot (TR) that can be used in healthcare environments. The system identification model was used to design the control parameters of the TR. A perfect design for the TR was proposed, with the development of trajectory algorithms, and a full analysis was provided. Theoretical models of the robot's mechanics, with trajectory control, were also created. These include the digital variants of basic control engineering elements, analog-to-digital converter evaluation, and characteristic curve linearization. In the case of the robot, the high-level controller is a mini-computer running the application program. This, of course, was also developed, but it was not included in the present paper, so as to save space. To this end, controlled test drives were undertaken, and the results were evaluated. The settings for the robot could then be determined from these findings.

Finally, the ideally adjusted robot was checked for its desired behavior. A few tests were conducted to validate the proposed design. A complex path is driven, with self-localization, left and right turns, and observing the robot's behavior. The simulation results showed that the TR's trajectory was well maintained during a full course. Theoretical models of the robot's mechanics and the associated control were also created. These included the digital variants of basic control engineering elements, analog-to-digital converter evaluation, and characteristic curve linearization.

The robot's development includes many parameters to be analyzed, along with the trajectory planning and optimization to be developed. It presents obstacle avoidance, propagating through the optimal path, and shows suitable trajectory planning using the developed algorithms. We have presented a trajectory control and optimization algorithm for a cognitive architecture named MERLIN. Optimization algorithms for both directions were developed for trajectory control, and the research empirically tested these algorithms over the developed monster truck model. There was only a slight jitter in the orientation due to the environmental instability, which can be easily filtered out. The experimental results showed that the model could follow the trajectory with good precision and could continue on its path for 25 s in the event of communication delay or disconnection.

Author Contributions: Conceptualization, A.A.A. and A.S.; methodology, M.N.K.; software, M.N.K.; validation, A.A.A., A.S., and M.N.K.; formal analysis, M.N.K.; investigation, A.S.; resources, A.S.; writing—original draft preparation, M.N.K.; writing—review and editing, A.S.; supervision, M.N.K.; funding acquisition, A.A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Deanship of Scientific Research at Prince Sattam bin Abdulaziz University under the research project (PSAU/2023/01/23001).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors acknowledge the support of the Ministry of Education in Saudi Arabia, as well as the Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia.

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

References

1. Dai, Y.; Xiang, C.; Zhang, Y.; Jiang, Y.; Qu, W.; Zhang, Q. A Review of Spatial Robotic Arm Trajectory Planning. *Aerospace* **2022**, *9*, 361. [CrossRef]
2. Hirzinger, G.; Brunner, B.; Dietrich, J.; Heindl, J. ROTEX-the First Remotely Controlled Robot in Space. In Proceedings of the 1994 IEEE International Conference on Robotics and Automation, San Diego, CA, USA, 8–13 May 1994; IEEE: Piscataway, NJ, USA, 1994; pp. 2604–2611.
3. Luo, J.; Yu, M.; Wang, M.; Yuan, J. A Fast Trajectory Planning Framework with Task-Priority for Space Robot. *Acta Astronaut.* **2018**, *152*, 823–835. [CrossRef]
4. Menasri, R.; Nakib, A.; Daachi, B.; Oulhadj, H.; Siarry, P. A Trajectory Planning of Redundant Manipulators Based on Bilevel Optimization. *Appl. Math. Comput.* **2015**, *250*, 934–947. [CrossRef]
5. Guillén-Climent, S.; Garzo, A.; Muñoz-Alcaraz, M.N.; Casado-Adam, P.; Arcas-Ruiz-Ruano, J.; Mejías-Ruiz, M.; Mayordomo-Riera, F.J. A Usability Study in Patients with Stroke Using MERLIN, a Robotic System Based on Serious Games for Upper Limb Rehabilitation in the Home Setting. *J. Neuroeng. Rehabil.* **2021**, *18*, 41. [CrossRef] [PubMed]
6. Karimi, M.; Roncoli, C.; Alecsandru, C.; Papageorgiou, M. Cooperative Merging Control via Trajectory Optimization in Mixed Vehicular Traffic. *Transp. Res. Part C Emerg. Technol.* **2020**, *116*, 102663. [CrossRef]
7. Kitazawa, O.; Kikuchi, T.; Nakashima, M.; Tomita, Y.; Kosugi, H.; Kaneko, T. Development of Power Control Unit for Compact-Class Vehicle. *SAE Int. J. Altern. Powertrains* **2016**, *5*, 278–285. [CrossRef]
8. Rodríguez-Lera, F.J.; Matellán-Olivera, V.; Conde-González, M.Á.; Martín-Rico, F. HiMoP: A Three-Component Architecture to Create More Human-Acceptable Social-Assistive Robots. *Cogn. Process.* **2018**, *19*, 233–244. [CrossRef]
9. Narayan, P.; Wu, P.; Campbell, D.; Walker, R. An Intelligent Control Architecture for Unmanned Aerial Systems (UAS) in the National Airspace System (NAS). In Proceedings of the AIAC12: 2nd Australasian Unmanned Air Vehicles Conference; Waldron Smith Management: Bristol, UK, 2007; pp. 1–12.
10. Laengle, T.; Lueth, T.C.; Rembold, U.; Woern, H. A Distributed Control Architecture for Autonomous Mobile Robots-Implementation of the Karlsruhe Multi-Agent Robot Architecture (KAMARA). *Adv. Robot.* **1997**, *12*, 411–431. [CrossRef]
11. de Oliveira, R.W.; Bauchspiess, R.; Porto, L.H.; de Brito, C.G.; Figueredo, L.F.; Borges, G.A.; Ramos, G.N. A Robot Architecture for Outdoor Competitions. *J. Intell. Robot. Syst.* **2020**, *99*, 629–646. [CrossRef]
12. Atsuzawa, K.; Nilwong, S.; Hossain, D.; Kaneko, S.; Capi, G. Robot Navigation in Outdoor Environments Using Odometry and Convolutional Neural Network. In Proceedings of the IEEJ International Workshop on Sensing, Actuation, Motion Control, and Optimization (SAMCON), Chiba, Japan, 4–6 March 2019.

13. Cuesta, F.; Ollero, A.; Arrue, B.C.; Braunstingl, R. Intelligent Control of Nonholonomic Mobile Robots with Fuzzy Perception. *Fuzzy Sets Syst.* **2003**, *134*, 47–64. [CrossRef]
14. Ahmadzadeh, A.; Jadbabaie, A.; Kumar, V.; Pappas, G.J. Multi-UAV Cooperative Surveillance with Spatio-Temporal Specifications. In Proceedings of the 45th IEEE Conference on Decision and Control, San Diego, CA, USA, 13–15 December 2006; IEEE: Piscataway, NJ, USA, 2006; pp. 5293–5298.
15. Anavatti, S.G.; Francis, S.L.; Garratt, M. Path-Planning Modules for Autonomous Vehicles: Current Status and Challenges. In Proceedings of the 2015 International Conference on Advanced Mechatronics, Intelligent Manufacture, and Industrial Automation (ICAMIMIA), Surabaya, Indonesia, 15–17 October 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 205–214.
16. Alami, R.; Chatila, R.; Fleury, S.; Ghallab, M.; Ingrand, F. An Architecture for Autonomy. *Int. J. Robot. Res.* **1998**, *17*, 315–337. [CrossRef]
17. Lee, H.; Seo, H.; Kim, H.-G. Trajectory Optimization and Replanning Framework for a Micro Air Vehicle in Cluttered Environments. *IEEE Access* **2020**, *8*, 135406–135415. [CrossRef]
18. Hu, R.; Zhang, Y. Fast Path Planning for Long-Range Planetary Roving Based on a Hierarchical Framework and Deep Reinforcement Learning. *Aerospace* **2022**, *9*, 101. [CrossRef]
19. Hitec HS-5745MG Servo Specifications and Reviews. Available online: https://servodatabase.com/servo/hitec/hs-5745mg (accessed on 28 November 2022).
20. Optical Encoder M101 | MEGATRON. Available online: https://www.megatron.de/en/products/optical-encoders/optoelectronic-encoder-m101.html (accessed on 28 November 2022).
21. Zhu, H.; Brito, B.; Alonso-Mora, J. Decentralized probabilistic multi-robot collision avoidance using buffered uncertainty-aware Voronoi cells. *Auton. Robot.* **2022**, *46*, 401–420. [CrossRef]
22. Batmaz, A.U.; Maiero, J.; Kruijff, E.; Riecke, B.E.; Neustaedter, C.; Stuerzlinger, W. How automatic speed control based on distance affects user behaviours in telepresence robot navigation within dense conference-like environments. *PLoS ONE* **2020**, *15*, e0242078. [CrossRef]
23. Xia, P.; McSweeney, K.; Wen, F.; Song, Z.; Krieg, M.; Li, S.; Yu, X.; Crippen, K.; Adams, J.; Du, E.J. Virtual Telepresence for the Future of ROV Teleoperations: Opportunities and Challenges. In Proceedings of the SNAME 27th Offshore Symposium, Houston, TX, USA, 22 February 2022.
24. Shen, S.; Michael, N.; Kumar, V. Autonomous multi-floor indoor navigation with a computationally constrained MAV. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 20–25.
25. Dong, Y.; Pei, M.; Zhang, L.; Xu, B.; Wu, Y.; Jia, Y. Stitching videos from a fisheye lens camera and a wide-angle lens camera for telepresence robots. *Int. J. Soc. Robot.* **2022**, *14*, 733–745. [CrossRef]
26. Zong, X.; Xu, G.; Yu, G.; Su, H.; Hu, C. Obstacle Avoidance for Self-Driving Vehicle with Reinforcement Learning. *SAE Int. J. Passeng. Cars-Electron. Electr. Syst.* **2017**, *11*, 30–39. [CrossRef]
27. Fawad, N.; Khan, M.N.; Altalbe, A. Intelligent Time Delay Control of Telepresence Robots Using Novel Deep Reinforcement Learning Algorithm to Interact with Patients. *Appl. Sci.* **2023**, *13*, 2462.
28. Estlin, T.A.; Volpe, R.; Nesnas, I.; Mutz, D.; Fisher, F.; Engelhardt, B.; Chien, S. Decision-Making in a Robotic Architecture for Autonomy. *Decis.-Mak. A Robot. Archit. Auton.* **2001**, *2001*, 92152–97383.
29. Kress, R.L.; Hamel, W.R.; Murray, P.; Bills, K. Control Strategies for Teleoperated Internet Assembly. *IEEE/ASME Trans. Mechatron.* **2001**, *6*, 410–416. [CrossRef]
30. Goldberg, K.; Siegwart, R. Beyond Webcams: An Introduction to Online Robots. MIT Press: Cambridge, MA, USA, 2002; ISBN 0262072254.
31. Brito, C.G. de Desenvolvimento de Um Sistema de Localização Para Robôs Móveis Baseado Em Filtragem Bayesiana Não-Linear. 2017. Bachelor's Thesis, Universidade de Bras'ılia, Brasilia, Brazil, 2018.
32. Rozevink, S.G.; van der Sluis, C.K.; Garzo, A.; Keller, T.; Hijmans, J.M. HoMEcare ARm RehabiLItatioN (MERLIN): Telerehabilitation Using an Unactuated Device Based on Serious Games Improves the Upper Limb Function in Chronic Stroke. *J. NeuroEng. Rehabil.* **2021**, *18*, 48. [CrossRef] [PubMed]
33. Schilling, K. Tele-Maintenance of Industrial Transport Robots. *IFAC Proc. Vol.* **2002**, *35*, 139–142. [CrossRef]
34. Garzo, A.; Arcas-Ruiz-Ruano, J.; Dorronsoro, I.; Gaminde, G.; Jung, J.H.; Téllez, J.; Keller, T. MERLIN: Upper-Limb Rehabilitation Robot System for Home Environment. In Proceedings of the International Conference on NeuroRehabilitation, Online, 13–16 October 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 823–827.
35. Ahmad, A.; Babar, M.A. Software Architectures for Robotic Systems: A Systematic Mapping Study. *J. Syst. Softw.* **2016**, *122*, 16–39. [CrossRef]
36. Sharma, O.; Sahoo, N.C.; Puhan, N.B. Recent Advances in Motion and Behavior Planning Techniques for Software Architecture of Autonomous Vehicles: A State-of-the-Art Survey. *Eng. Appl. Artif. Intell.* **2021**, *101*, 104211. [CrossRef]
37. Ziegler, J.; Werling, M.; Schroder, J. Navigating Car-like Robots in Unstructured Environments Using an Obstacle Sensitive Cost Function. In Proceedings of the 2008 IEEE Intelligent Vehicles Symposium, Eindhoven, The Netherlands, 4–6 June 2008; IEEE: Piscataway, NJ, USA, 2008; pp. 787–791.
38. González-Santamarta, M.Á.; Rodríguez-Lera, F.J.; Álvarez-Aparicio, C.; Guerrero-Higueras, Á.M.; Fernández-Llamas, C. MERLIN a Cognitive Architecture for Service Robots. *Appl. Sci.* **2020**, *10*, 5989. [CrossRef]

39. Shao, J.; Xie, G.; Yu, J.; Wang, L. Leader-Following Formation Control of Multiple Mobile Robots. In Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation Intelligent Control, Limassol, Cyprus, 27–29 June 2005; IEEE: Piscataway, NJ, USA, 2005; pp. 808–813.
40. Faisal, M.; Hedjar, R.; Al Sulaiman, M.; Al-Mutib, K. Fuzzy Logic Navigation and Obstacle Avoidance by a Mobile Robot in an Unknown Dynamic Environment. *Int. J. Adv. Robot. Syst.* **2013**, *10*, 37. [CrossRef]
41. Favarò, F.; Eurich, S.; Nader, N. Autonomous Vehicles' Disengagements: Trends, Triggers, and Regulatory Limitations. *Accid. Anal. Prev.* **2018**, *110*, 136–148. [CrossRef]
42. Gopalswamy, S.; Rathinam, S. Infrastructure Enabled Autonomy: A Distributed Intelligence Architecture for Autonomous Vehicles. In Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV), Suzhou, China, 26–30 June 2018; IEEE, 2018; pp. 986–992.
43. Allen, J.F. Towards a General Theory of Action and Time. *Artif. Intell.* **1984**, *23*, 123–154. [CrossRef]
44. Hu, H.; Brady, J.M.; Grothusen, J.; Li, F.; Probert, P.J. LICAs: A Modular Architecture for Intelligent Control of Mobile Robots. In Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots, Pittsburgh, PA, USA, 5–9 August 1995; IEEE: Piscataway, NJ, USA, 1995; Volume 1, pp. 471–476.
45. Alami, R.; Chatila, R.; Espiau, B. Designing an Intelligent Control Architecture for Autonomous Robots. In Proceedings of the ICAR; 1993; Volume 93, pp. 435–440.
46. Khan, M.N.; Hasnain, S.K.; Jamil, M.; Imran, A. *Electronic Signals and Systems: Analysis, Design and Applications*; Rivers: Gistrup, Denmark, 2022.
47. Kang, J.-M.; Chun, C.-J.; Kim, I.-M.; Kim, D.I. Channel Tracking for Wireless Energy Transfer: A Deep Recurrent Neural Network Approach. *arXiv* **2018**, arXiv:1812.02986.
48. Zhao, W.; Gao, Y.; Ji, T.; Wan, X.; Ye, F.; Bai, G. Deep Temporal Convolutional Networks for Short-Term Traffic Flow Forecasting. *IEEE Access* **2019**, *7*, 114496–114507. [CrossRef]
49. Schilling, K.J.; Vernet, M.P. Remotely Controlled Experiments with Mobile Robots. In Proceedings of the Thirty-Fourth Southeastern Symposium on System Theory (Cat. No. 02EX540), Huntsville, AL, USA, 19 March 2002; IEEE: Pittsburgh, PA, USA, 2002; pp. 71–74.
50. Moon, T.-K.; Kuc, T.-Y. An Integrated Intelligent Control Architecture for Mobile Robot Navigation within Sensor Network Environment. In Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No. 04CH37566), Sendai, Japan, 28 September–2 October 2004; IEEE: Pittsburgh, PA, USA, 2004; Volume 1, pp. 565–570.
51. Lefèvre, S.; Vasquez, D.; Laugier, C. A Survey on Motion Prediction and Risk Assessment for Intelligent Vehicles. *ROBOMECH J.* **2014**, *1*, 1–14. [CrossRef]
52. Behere, S.; Törngren, M. A Functional Architecture for Autonomous Driving. In Proceedings of the First International Workshop on Automotive Software Architecture, Montreal, QC, Canada, 4 May 2015; pp. 3–10.
53. Carvalho, A.; Lefévre, S.; Schildbach, G.; Kong, J.; Borrelli, F. Automated Driving: The Role of Forecasts and Uncertainty—A Control Perspective. *Eur. J. Control* **2015**, *24*, 14–32. [CrossRef]
54. Liu, P.; Paden, B.; Ozguner, U. Model Predictive Trajectory Optimization and Tracking for On-Road Autonomous Vehicles. In Proceedings of the 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 4–7 November 2018; IEEE: Pittsburgh, PA, USA, 2018; pp. 3692–3697.
55. Weiskircher, T.; Wang, Q.; Ayalew, B. Predictive Guidance and Control Framework for (Semi-) Autonomous Vehicles in Public Traffic. *IEEE Trans. Control Syst. Technol.* **2017**, *25*, 2034–2046. [CrossRef]
56. Inkyu, C.; Song, H.; Yoo, J. Deep learning based pedestrian trajectory prediction considering location relationship between pedestrians. In Proceedings of the 2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Okinawa, Japan, 11–13 February 2019; IEEE: Pittsburgh, PA, USA, 2019; pp. 449–451.
57. Liang, Z.; Liu, Y.; Al-Dubai, A.Y.; Zomaya, A.Y.; Min, G.; Hawbani, A. A novel generation-adversarial-network-based vehicle trajectory prediction method for intelligent vehicular networks. *IEEE Internet Things J.* **2020**, *8*, 2066–2077.
58. Liang, Z.; Bi, Z.; Hawbani, A.; Yu, K.; Zhang, Y.; Guizani, M. ELITE: An Intelligent Digital Twin-Based Hierarchical Routing Scheme for Softwarized Vehicular Networks. *IEEE Trans. Mob. Comput* **2022**. [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.