mathematics

Article

Dynamical Analysis of a Navigation Algorithm

Mireya Cabezas-Olivenza [1], Ekaitz Zulueta [1,*], Ander Sánchez-Chica [1], Adrian Teso-Fz-Betoño [1] and Unai Fernandez-Gamiz [2]

[1] System Engineering and Automation Control Department, University of the Basque Country (UPV/EHU), Nieves Cano, 12, 01006 Vitoria-Gasteiz, Spain; mireya.cabezas@ehu.eus (M.C.-O.); ander.sanchez@ehu.eus (A.S.-C.); ateso001@ehu.eus (A.T.-F.-B.)

[2] Department of Nuclear and Fluid Mechanics, University of the Basque Country (UPV/EHU), Nieves Cano, 12, 01006 Vitoria-Gasteiz, Spain; unai.fernandez@ehu.eus

* Correspondence: ekaitz.zulueta@ehu.eus

Abstract: There is presently a need for more robust navigation algorithms for autonomous industrial vehicles. These have reasonably guaranteed the adequate reliability of the navigation. In the current work, the stability of a modified algorithm for collision-free guiding of this type of vehicle is ensured. A lateral control and a longitudinal control are implemented. To demonstrate their viability, a stability analysis employing the Lyapunov method is carried out. In addition, this mathematical analysis enables the constants of the designed algorithm to be determined. In conjunction with the navigation algorithm, the present work satisfactorily solves the localization problem, also known as simultaneous localization and mapping (SLAM). Simultaneously, a convolutional neural network is managed, which is used to calculate the trajectory to be followed by the AGV, by implementing the artificial vision. The use of neural networks for image processing is considered to constitute the most robust and flexible method for realising a navigation algorithm. In this way, the autonomous vehicle is provided with considerable autonomy. It can be regarded that the designed algorithm is adequate, being able to trace any type of path.

Keywords: navigation; localization; SLAM; computer vision; neural network; semantic segmentation; Lyapunov; AGV; path planning; path following

check for updates

Citation: Cabezas-Olivenza, M.; Zulueta, E.; Sánchez-Chica, A.; Teso-Fz-Betoño, A.; Fernandez-Gamiz, U. Dynamical Analysis of a Navigation Algorithm. *Mathematics* **2021**, *9*, 3139. https://doi.org/10.3390/math9233139

Academic Editors: Irina Cristea, Yuriy Rogovchenko, Justo Puerto, Gintautas Dzemyda and Patrick Siarry

Received: 15 November 2021
Accepted: 4 December 2021
Published: 6 December 2021

1. Introduction

In industrial applications, the current demand is to have an intelligent navigation system for mobile robots. Those systems must include navigation and localization methods, both of which are implemented in automated guided vehicles (AGVs). Nevertheless, the study of those methods has usually been carried out in independent ways. The consideration of the study as separate techniques in not a disadvantage, but rather a division of problems. The unification of these systems is major research; however, while performing, the AGVs must integrate all of them. This paper is focused on giving more robustness to this type of autonomous vehicle. With that purpose, the following topics specify the analyses that have been performed for the various techniques that form the whole AGV system.

1.1. Localization

One of the indispensable systems of autonomous guided vehicles is the one that determines their positioning, defined by the vector $\vec{pose}$. With this information, it is possible to generate a map of the environment where the AGV is located and to determine where the vehicle is placed on the map. This is known as simultaneous localization and mapping (SLAM) and it is important to execute it in real-time.

A traditional approach to the actual problem is to use the wheel odometry as Kilic et al. [1] studied, combined with an inertial navigation system, where measurements are taken by an encoder. The present work rejects this method due to the slippage that the wheels suffer during the movement, being necessary data to be predicted. It also penalizes accuracy. According to Chen [2], it is recognized that these aspects can be clarified with a Kalman filter.

The Kalman filter can also be suitable for use in conjunction with matching techniques. According to the study carried out by Cho et al. [3], two matching methods can be employed in combination with that filter. Firstly, the geometric method, and secondly, a method based on the point-to-line matching of the iterative closest point (ICP) algorithm. In this way, the geometric method is applied to predict, and the ICP is used for correcting the estimated position. This requires prior information concerning the environment in which the AGV is located. The unique use of geometric methods is also possible, as Shamsfakhr et al. [4] demonstrate in their developed algorithm. Geometric pattern registration is performed based on the segmentation of the real laser range data and simulated laser data. Looking for the critical points of both and with the discrepancy between them, it is possible to achieve a robust and computationally efficient algorithm for determining the $\overrightarrow{pose}$ in real-time.

The iterative closest point (ICP) persist in being a recurrent approach to solving the localization problem. In addition, knowing that LiDAR sensors do not require external spatial infrastructure, they can be utilized for SLAM (see Naus et al. [5]). Employing architecture to reduce iterations (wP-ICP) coupled with LiDAR sensors, Wang et al. [6] achieve a reduction of computational effort. In this way, it can be better managed in real-time. It is equally possible to generate more 3D point sets by focusing on the geometry of the environment as Senin et al. [7] do, obtaining better results. Using different sensors such as INS and GPS coupled with LiDAR, Gao et al. [8] achieved localization in indoor and outdoor areas. However, it is impossible to distinguish areas with unevenness, solved with KITTI arrays as Kim et al. [9] summarized.

Avoiding the application of a GPS antenna, one approach to localize AGVs is to use trilateration, through a number of signals that can estimate their distance to the vehicle. For algorithms that need more speedup and high efficiency, Sadeghi Bigham et al. [10] prove that by focusing on orthogonal polygons the n/2 landmarks are sufficient to solve the localization problem. So that trilateration of an orthogonal n-gon can be performed. Further to the concept of using landmarks, it is possible to merge them with computer vision to be detected. As Yap et al. [11] explain, the distance between the landmarks and the AGV is estimated with an algorithm based on two landmarks according to the idea of the intersection of two circles.

The noise that can be generated by the sensing devices used must also be taken into account, so the choice of the sensors implemented in the AGV is crucially important. A cost-effective option represents the use of RGB sensors, which process images to extract features by finding similarities between frames (see Gao and Zhang [12]). Deriving the ICP algorithm with all necessary sensing elements and constructing random point maps, develops the algorithm for SLAM independent of sensor type as Clemens et al. [13] discussed. In this way, noise becomes a measure for determining uncertainty.

The use of an algorithm based on Bayesian filtering has also given good results in the localization of mobile robots, without the need for prior information about the environment as Gentner et al. [14] explained. In this way, it is possible to obtain the positioning of the AGV in the SLAM. Continuing with the use of filters, the particle filter has remained a recurring method when trying to solve the localization problem. With it, it is possible to estimate the state of an environment over time. The precision is related to the number of particles, but it should be noted that increasing the number of particles penalizes the computational cost (see Yang and Wu [15]).

For that reason, a method of optimizing the cells produced in the SLAM is necessary. Because of this optimization, Zhang et al. [16] demonstrated that there is a difficulty in distinguishing between areas with the same appearance. Therefore, the reference has to be taken as a global localization. However, at large distances, the accuracy decreases. Carrera Villacres et al. [17] described that focusing the problem on a deep learning model provides a satisfactory result, with the need to include new filters to reduce failures. In addition, combining the particle filter with a vector field histogram (VFH) provides a way to circumvent obstacles (see Wang [18]).

Integrated into a particle filter, Tao et al. [19] include a novel ratio frequency identification (RFID) based method. By combining the information from two RFID signals, this strategy is able to predict the $\overrightarrow{pose}$. In addition, in AGVs where magnetic field lines are involved, it is equally possible to use a for-field active RFID locating method, providing higher accuracy, more stable movements and a smaller fluctuating rate. (see Lu et al. [20]).

1.2. Path Following

Another faculty that AGVs must possess is the ability to navigate in diverse environments. This is based on immediate sensor readings, providing the mobile robot with the capacity to decide and circumvent obstacles. The dynamic properties of the vehicle must also be taken into account. The most commonly used method for trajectory tracking is the dynamic window approach (DWA). By hybridizing with a teaching-learning based technique, it is possible to achieve the endpoint to be reached by the AGV, as Kashyap et al. studied [21]. This additionally provides the ability to avoid obstacles without stopping. In succession, it is possible to combine it with a real-time motion planning method, giving the AGV the possibility to gain a high speed (see Brock and Khatib [22]).

By mixing Dijkstra's algorithm and the DWA, it is possible to attain the desired position with the information provided by a SLAM system, as Liu et al. [23] summarised. Another fusion option is discussed by Dobrevski et al. [24] in their work, where they manage a convolutional neural network to select the parameters of the DWA algorithm. This provides a combination between data-driven learning and the dynamic model of the mobile robot. It is substantial to take into account the dynamic properties due to the constraints imposed by the AGV itself on velocity and acceleration, as pointed out by Fox et al. [25].

As Wang et al. [26] studied, the problem can be divided into two layers. On one hand, decision-making is performed for the path, and on the other hand, trajectory tracking is carried out. In the second layer, it is possible to employ the virtual force field (VFF) to detect objects. Similarly, it is possible to combine it with a potential force field (PFF) to build a viable means of navigation (see Burgos et al. [27]). Another algorithm used to solve the path following is the vector field histogram (VFH), as done by Borenstein et al. [28], where they employ histogram grids as a model to generate a map of the environment. In this way, it is possible to obtain the AGV control commands. There is a modification of the latter method called the traversability field histogram (TFH), which is independent of the instantaneous vehicle speed, as Ye [29] noted. In this way, distant obstacles can be prevented from impairing optimal path following.

From another perspective, the use of a lateral control for path following resolution represents an interesting method, performing an identification of the closest point between the trajectory and the AGV. One of the algorithms that can conclude this is the Stanley. It performs a discrete prediction model of the subsequent states of the mobile robot. In the study by AbdElmoniem et al. [30] a combination with a LiDAR is used, creating a local path planner and being able to complete collision-free navigation.

1.3. Path Planning

Path planning is one of the most complex areas of mobile robotics, where it is necessary to calculate the trajectory to be followed by the AGV. In this case, the most conventional technique is Particle Swarm Optimization (PSO). By improving the inertia weights with a linear variation, the algorithm can be prevented from falling into a local minimum, achieving a higher convergence speed as Fei et al. [31] demonstrated. In this way, it is possible to obtain the optimal trajectory in any environment, with better path length than with the A* algorithm on 2D maps.

Authors such as Liu et al. [32] explained that the A* algorithm can be used as a map modelling method to procure path planning. In many cases, considering only the optimal distance to the destination point is not enough, so it is meaningful to evaluate also the shortest time to that point. The A* algorithm, depending on these attributes, can search for one or another trajectory as Cheng et al. [33] explained. Combining this algorithm with the Rapidly-Exploring Radom Tree (RRT) achieves good efficiency (see da Silva Costa et al. [34]). A modification of the RTT itself gets a new path planning diagram, in which the trajectory is found instantly, as the study of Wang et al. [35] develop. Another design represents the one by Wen and Rei [36], called Smoothly RRT, where the optimization strategy focuses on the maximum curve of the trajectory, achieving a higher exploration speed.

The Wavefront algorithm is equally implemented to calculate the path, employing it to obtain the closest front points. In this way, it is possible to select the optimal point based on the motion requirements. Therefore, the Wavefront algorithm can search for additional paths, if necessary, as Tang et al. [37] summarized. The generalized Wavefront algorithm is also discussed. Multiple sets of target points, multilevel grid costs and geometric expansions around obstacles are combined, and with this information, the path is optimized, recognizing a safe and smooth trajectory (see Sifan et al. [38]).

A less conventional, but also interesting technique to perform trajectory planning represents the use of neural networks. Using the Q-learning algorithm with reinforcement learning can support the features of the environment, as Sdwk et al. [39] do. Advancing neural network training provides an optimal path. In addition, with an incremental training method, where algorithms are first evaluated, a more pleasing ultimate design of the deep learning algorithm can be obtained (see Gao et al. [40]).

Operating the Resnet-50 network, a path planning algorithm based on deep reinforcement learning has been created. In this way, the parameters of the deep Q-network are trained, solving the path planning problem, as Zheng et al. [41] demonstrated. Another option is to implement a convolutional neural network (CNN) that segments an image to condition the navigation zone, proposed by Teso-Fz-Betoño et al. [42]. The study manages a residual neural network that participates in the learning of the Resnet-18 network. As follows, it is possible to perform semantic segmentation for AGV navigation by selecting the mask of the navigation area.

Focusing on computer vision, there are networks established specifically for path planning, such as PilotNet, which can detect lanes using cameras and apply vehicle following algorithms to gain the direction, as discussed by Olgun et al. [43]. This is merely effective for single-lane trajectories. LaneNet is, furthermore, a network that applies computer vision, detecting lane markings and lane locations, and being able to create maps and paths using semantic segmentation (see Azimi et al. [44]).

The fundamental objective of the present study is to ensure the stability of an indoor navigation algorithm in a 2D environment for the AGV shown in Figure 1. Based on Stanley's algorithm, a lateral control adapted to this autonomous vehicle is proposed. The algorithm calculates the velocity and rotation angle commands to perform the movement, applying different mathematical operations. In this manner, lateral control and longitudinal control should coexist.

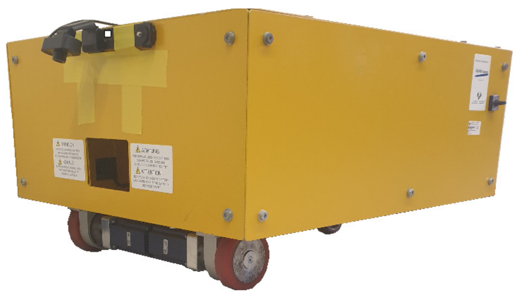


Figure 1. The AGV for which the algorithm is proposed.

The main goal of the current work is to implement the algorithm in the AGV and be able to perform collision-free navigation. This issue is achieved by using computer vision and a neural network that segments the environment to generate a path.

2. Materials and Methods

This article presents an algorithm for indoor navigation of AGVs and the study of its stability. With the idea of the Stanley algorithm (see Hoffmann et al. [45]), this work proposes a modification because of the use of the AGV shown in Figure 1. This autonomous vehicle allows the adjustment of the angular and linear velocity criteria.

Accordingly, in this paper, the algorithm focuses on the lateral control of the AGV, acting mainly on the rotational speed $\dot{\theta}$ (rad/s). It is necessary to observe that the alignment error φ (rad) remains the difference between the vehicle angle θ (rad) and the path curvature φ_{path} (rad). The positioning error noted as e (m), refers to the minimum distance between the autonomous vehicle and the closest point of the trajectory in reference to the vehicle as represented in Figure 2.

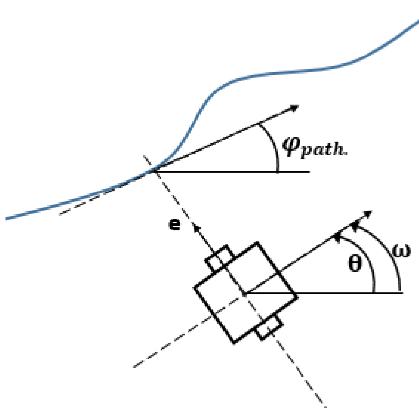


Figure 2. Schematic representation of the lateral control proposal.

The side wheels of the AGV used, rotate on the same axis, as represented in Figure 2. Note that ω should take into account both the positioning error noted as e and the alignment error noted as φ. The alignment error remains a revealed fact, so the positioning error is the variable, where two opposite cases are assumed. If e is remarkably large, it is of interest to situate the AGV perpendicular to the path, in order to get closer to it, so the value of φ is be $\frac{\pi}{2}$. In the case that e is significantly smaller, there is no alignment error, so it is of interest to keep both angles with the same value. The curve given by the values of φ is an Arctangent function, which also appears in Stanley's algorithm.

In this manner, it is proposed to implement Stanley's algorithm as shown in the following equations. Knowing that θ is derivable in time, the value that ω should have is proposed in Equation (1).

$$\frac{d\theta}{dt} = \omega = K_1(\varphi_{\text{setpoint}}(e) - \left(\theta - \varphi_{\text{path}}\right) \tag{1}$$

The introduction of the parameter $\varphi_{\text{setpoint}}$ (rad) is necessary, to adapt the range of values of φ. Its value is defined in Equation (2):

$$\varphi_{setpoint}(e) = arctg(K_2 \cdot e) \tag{2}$$

In this way, it is proposed to have two functions depending on the constants K_1 (s^{-1}) and K_2 (1/m), which are to be determined. It can be appreciated that the constant K_1 is the one related to the alignment error φ and K_2 controls the value of the positioning error e.

Some other equations also need to be accounted for. $\dot{x}$ (m/s) and $\dot{y}$ (m/s) represent the linear velocities in the directions of the axes, as indicated in Equations (3) and (4).

$$\dot{x} = V \cos\theta \tag{3}$$

$$\dot{y} = V \sin\theta \tag{4}$$

As mentioned throughout this work, the objectives of the lateral control are that the positioning error tends to be 0 and that θ equals φ_{path}.

A study is required to demonstrate the proposed design is stable for all types of paths. For this purpose, the Lyapunov function will be used. Through the study, it is equally possible to dictate the value of the constants K_1 and K_2 in Equations (1) and (2). The Lyapunov energy equation noted as L (m^2) is proposed based on the previous equations in Equation (5).

$$\begin{cases} L = e^2 > 0 \\ \frac{dL}{dt} < 0 \end{cases} \tag{5}$$

For the first approach, the linear velocity V (m/s) is assumed to be constant.

The autonomous vehicle is placed with a random $\overrightarrow{pose} = [x,\ y,\ \theta]$ value, and the constants K_1 and K_2 are applied to the lateral control, analysing how the AGV behaves as a function of these coefficients. To simplify the analysis, a trajectory of the form $Ax + By + C = 0$ will be considered.

Distinction of the Positioning Error Sign

To implement the lateral control design, it is necessary to contemplate the sign of e because it is not taken into account in either of Equation (1) or Equation (2). It is necessary to differentiate on which side of the path the AGV is located because depending on this; it must be steered with a positive or negative sign.

Hence, to consider the sign, the positioning error e is to be taken as a vector $\overrightarrow{\mathrm{e}}$. The vector $\overrightarrow{\mathrm{N}}$ is the one that represents in which direction the AGV will follow the path. Finally, α is the angle formed between these two vectors, as illustrated in Figure 3.

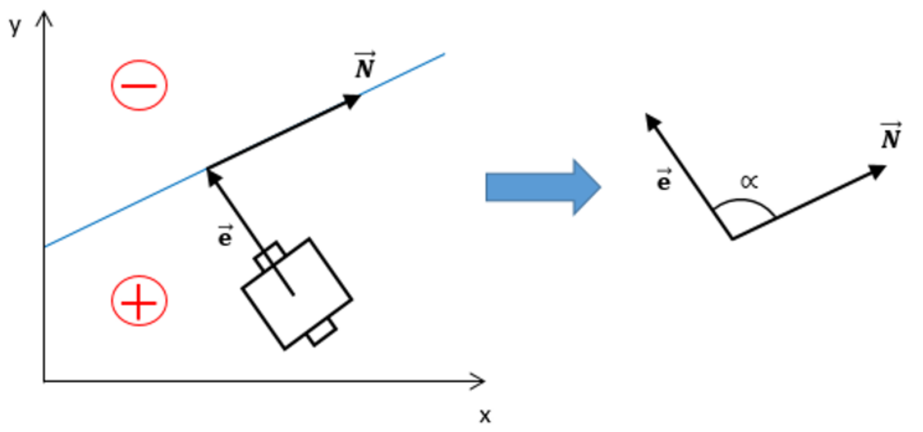


Figure 3. Positioning error vector and trajectory direction vector.

Figure 3 also shows in which region the positioning error is considered positive and in which negative. Due to that consideration, the direction of $\overrightarrow{\mathrm{e}}$ is conditioned by that sign, representing the course the AGV needs to take to reduce the positioning error. Like so, if the AGV is on the right side of the path, the value of φ needs to be increased. If the AGV is on the left side, the value of the alignment error has to be reduced, supporting negative values. To resolve it, the vector product of $\overrightarrow{\mathrm{e}}$ and $\overrightarrow{\mathrm{N}}$ must be performed.

If $\left|\vec{N}\right|$ and $\left|\vec{e}\right|$ take the value 1, an analysis can be performed as a function of α. In case α is positive, the positioning error has to be positive as well. In the opposite case, that is when α is negative, the positioning error has to bear a negative sign. Knowing the path is defined, it is possible to acquire the values of the points $P1(x, y)$ and $P2(x, y)$.

$P1$ refers to the point on the path most adjacent to the AGV. The position of the autonomous vehicle is noted as $p(x, y)$. In the case of $P2$, it refers to the nearest next point from the AGV after $P1$. Thus, the values of the vectors $\vec{e}$ and $\vec{N}$ can be determined as presented in Equations (6) and (7).

$$\vec{N} = \frac{\vec{P2} - \vec{P1}}{\|\vec{P2} - \vec{P1}\|} \tag{6}$$

$$\vec{e} = \vec{P1} - \vec{p} \tag{7}$$

With this knowledge, in the line of code that calculates the positioning error, it is necessary to apply the approximation of Equation (8) to consider the sign of e.

$$e \sin \propto = T = N(1)e(2) - N(2)e(1) \tag{8}$$

Considering the path form and the previous equations, the positioning error is defined in Equation (9).

$$e = \frac{|Ax + By + C|}{\sqrt{A^2 + B^2}} = (sign(e))distmin.\left(\vec{p}, path\right) \tag{9}$$

3. Proposal Explanation

A combination of neural networks and hardware devices, such as the AGV itself, is employed. Regarding the hardware, the use of a Beckhoff PLC (C6925) and its automation software allows the drivers to be managed on a real-time industrial platform. This platform is highly robust and widely used therein type of industrial applications. In addition, the employment of Matlab R2019b provides the advantage of utilizing a platform that allows very rapid development of control engineering algorithms. From a sensor point of view, and in the present case, the use of a vision-based navigation system that recognises lanes, the advantage resides in the fact that it is a very rapid way of implementing path-following systems.

3.1. Necessary Data Acquisition

To obtain the trajectory, the example of the study by Teso-Fz-Betoño et al. [42] is followed. A convolutional neural network (CNN) is managed to perform semantic segmentation of an image. It is classified into several masks, generating a vector of the interest points from the mask corresponding to the navigation area. This vector with the position of the path in pixels x and y represents the information provided to the navigation algorithm. The scenario comprises a room with a yellow line representing the trajectory. The neural network has to detect this line, which will be followed by the AGV. This is concluded by employing a camera. The processing of the image is represented in Figure 4.

Figure 4a shows the image captured by the AGV, with the yellow line to be followed. After obtaining the image, the convolutional neural network performs semantic segmentation, obtaining two independent masks. The shaded mask constitutes the part that is not of interest to the autonomous vehicle, so the shinier one attends the important one, as illustrated in Figure 4b. In addition, the image is cropped at the lower part to remove the portion of the AGV that is captured by the camera.

From the clearest mask, the midpoints represented by red crosses are gained, as shown in Figure 4c. Thus, the trajectory to be followed by the AGV is obtained. Conclusively, the points approached with the semantic segmentation are connected to forge the path as represented in light blue in Figure 4d. Due to the pronounced curves generated in this path, an interpolation is performed to acquire a smoother trajectory, coloured with dark blue. The AGV follows that final trajectory.

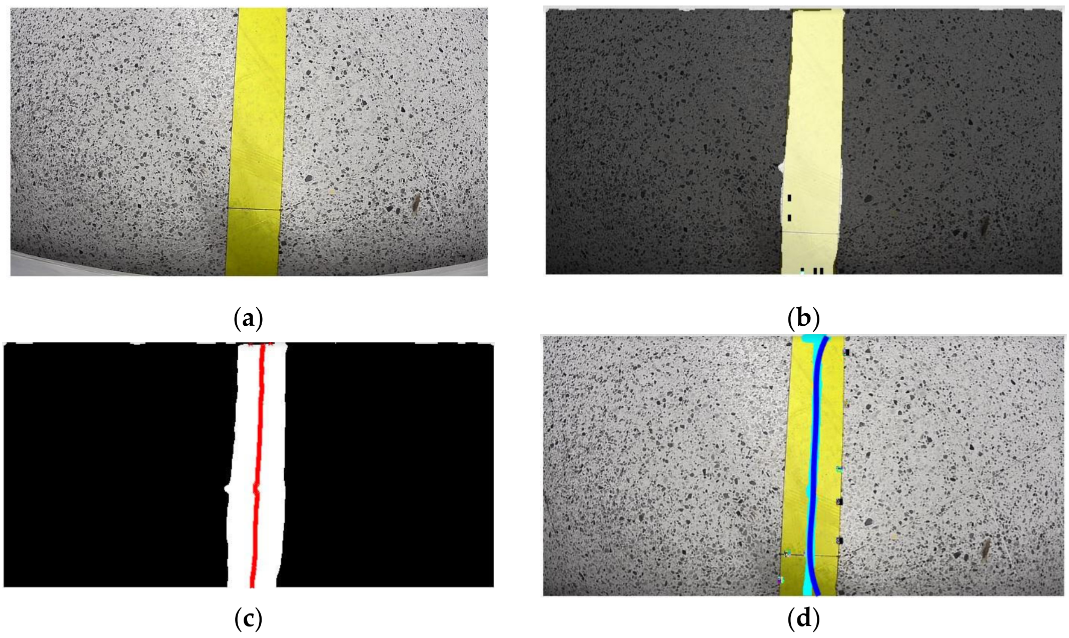


Figure 4. Process of obtaining trajectory: (**a**) Image that AGV takes of the path; (**b**) Semantic segmentation of the path; (**c**) Medium point of the navigable mask; (**d**) Obtained path.

With the image taken of the path, the AGV can acquire the information of the positioning error and the angle of the trajectory as graphed in Figure 5. Note that the measurement is produced considering the location of the AGV as the position of the camera. The camera is placed at the leading centre of the autonomous vehicle.

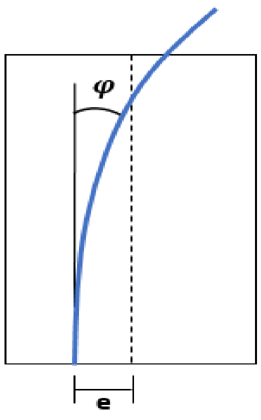


Figure 5. Localization data for the AGV from image.

This approach provides all the data necessary for the navigation resolution.

3.2. *Concurrency in the Approach*

The designed algorithm has fast dynamics. This implies that regardless of the initial position of the AGV as a function of e, it is necessary to obtain an θ_{optimun} (rad) for the autonomous vehicle. This value is not the same as θ, and this is where K_1 comes in. θ_{optimun} refers to the sum between φ_{path} and $\varphi_{\text{setpoint}}(\text{e})$. Accordingly, because of the fast dynamic, the AGV orients itself on the way of the trajectory rapidly, being able to consider that the path has no inclination. This is depicted in Figure 6. In the case of the positioning error, it is not possible to make an approximation, and the path will be considered to be angled. In conclusion, it can be declared that the time constants of the orientation loop are smaller than those of the displacement. Ergo, two situations are envisaged as represented in Figure 6a,b.

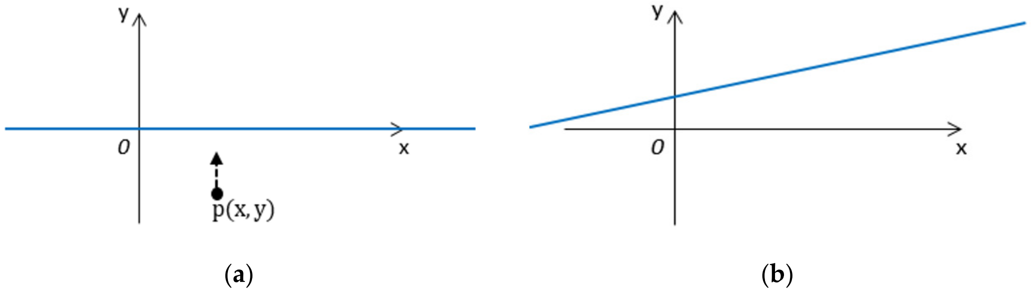


Figure 6. Path approach: (**a**) Situation where φ_{path} is zero; (**b**) Situation where φ_{path} is not zero.

With this approach, it is possible to analyse the stability of the system and to get the K_1 and K_2 values.

4. Value of K_1

4.1. System Stability Study

Lyapunov stability analysis is performed as mentioned above. Conventional techniques, for instance "frecuency" methods, are generally carried out on cases where the dynamical model of the system is linear. In the present study, the dynamical model is non-linear and the use of Lyapunov provides a generalist manner of assuring the stability of a dynamical system.

The first scenario is where can be assumed that the angle of the trajectory is zero, as illustrated in Figure 6a. Considering that assumption, the terms A and C of the path equation disappear, the path being By = 0. Hence, the study is simplified.

The initial consideration is when the AGV is away from the trajectory, so the positioning error value is obtained directly as indicated in Equation (10).

$$e = -y \tag{10}$$

The objective is to guarantee that the system is stable, being L consistently positive. Therefore, considering the previous equation, it can be formulated in Equation (11).

$$L = e^2 = y^2 \tag{11}$$

As the value of y is squared, it is confirmed that L is always positive, so the difficulty now lies in the second expression of Equation (5). In this manner, the derivate must be considered taking into account the Equation (12).

$$\frac{dL}{dt} = 2 \cdot e \cdot \dot{e} = 2 \cdot y \cdot \dot{y} \tag{12}$$

Additionally, Equation (1) can be restated considering $\varphi_{\text{path}} = 0$, as shown in Equation (13).

$$\dot{\theta} = K_1 \left(\varphi_{setpoint}(e) - \theta \right) \tag{13}$$

As mentioned, assuming that the dynamic of Equation (13) is extremely fast, it is possible to obtain the value of the angle θ, because of the rapid tendency of the AGV will have, positioning with the orientation of Figure 6a. Then θ is defined in Equation (14).

$$\theta = \varphi_{setpoint}(e) = arctg(K_2 \cdot e) \tag{14}$$

Equation (4) can directly be raised anew, bearing in mind Equations (10) and (14).

$$\dot{y} = V\sin(-arctg(K_2 \cdot y)) = -V\sin(arctg(K_2 \cdot y)) \tag{15}$$

To obtain the value of $\dot{L}$, the Equation (12) can be complemented with that seen in Equation (15).

$$\dot{L} = 2y(-V\sin(arctg(K_2 \cdot y))) \tag{16}$$

By the way, Equation (2) is designed knowing that K_2 will always be positive, so the following remarks can be made.

$$sign(arctg(K_2 \cdot y)\ ) = sign(y) \tag{17}$$

$$sign(\sin(arctg(K_2 \cdot y))) = sign(arctg(K_2 \cdot y)) \tag{18}$$

This leads to the following conclusion.

$$sign(y) = sign(\sin(arctg(K_2 \cdot y))) \tag{19}$$

With this information, analysing Equation (19) and knowing the positioning error is negative as stated in Equation (10), it can be guaranteed that the Lyapunov function is fulfilled.

4.2. Procurement of Value of K_1

For the system to be stable when the AGV is distant from the trajectory, it has been assumed that the dynamics are so fast. Accordingly, the AGV is oriented perpendicular to the path. In addition, it will approach rapidly, producing a minor positioning error, which is considered to be zero. Carrying on with that consideration, therein case, only the angle of the AGV can be taken into account, reformulating Equation (13).

$$\dot{\theta} = K_1(0 - \theta) = -K_1\theta \tag{20}$$

In this situation, it is necessary to study again the stability of the system. Recalling the Lyapunov system of Equations (11) and (12), it will not be a problem to confirm that L is by squaring. The problem is again in the derivative of the energy. The study examines the case where the AGV is under the trajectory, so it is recognized that in that area θ will allow positive values. So, if e is null, it is also comprehensible that θ tends to be equal to φ_{path}. Therefore, one can formulate the integral of Equation (20), which remains a linear system.

$$\theta = \frac{\pi}{2} exp^{-K_1 \cdot \tau} \tag{21}$$

Equation (12) requires the value of y and $\dot{y}$, which can be obtained from Equation (4) by substituting Equation (21).

$$\dot{y} = V\sin\left(\frac{\pi}{2} exp^{-K_1\tau}\right) \tag{22}$$

$$y = V\int_0^t \sin\left(\frac{\pi}{2} exp^{-K_1\tau}\right) d\tau + y(0) \tag{23}$$

From Equation (22) it can be deduced that the value is always negative because the function sin is always between 0 and π. In the case of Equation (23), the initial condition is also always negative, so it is necessary to ensure that the integral never obtains a value greater than y(0), to confirm the stability. So it is necessary to guarantee that the positioning error never changes sing, whereby the value of K_1 can be known. In Equations (24) and (25) the integral is noted as I.

$$y = VI(K_1,\ t) + y(0) < 0 \tag{24}$$

$$VI(K_1, t) < -y(0) \tag{25}$$

Thus, the linear velocity and K_1 are related. Analysing Equations (23) and (25), the following conclusions can be drawn. If the value of K_1 is very aggressive, the exponential tends to 0 quickly obtaining a sinusoidal function with value 0 and making the expression vanish very briefly. This makes it independent of the value V that is set. On the contrary, if K_1 is small, the velocity is limited. Otherwise, an undesired oscillation system would appear.

This system is implemented in Matlab Ver. R2019b (The Matworks Inc., Madrid, Spain), obtaining the plots revealed in Figure 7. In Figure 7a the values of V = 100 m/s and K_1 = 1 s^{-1} are set for plotting. It can be visualized how the value of the integral (red line) takes time to fade out, in the order of seconds. In this case, the value of y admits an extremely significant positive value which does not ensure stability as it cannot be guaranteed to be lower than y(0). In the plot of Figure 7b, V = 10 m/s and K_1 = 1000 s^{-1} are set. The value of the integral disappears instantly, ensuring the stability of the system and regardless of the velocity value set.

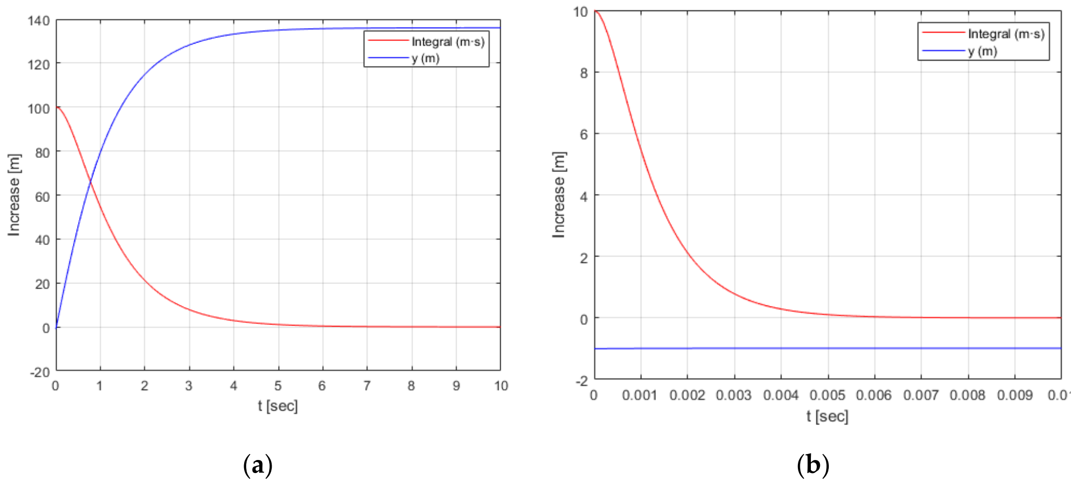


Figure 7. Results of simulation: (**a**) Plot when V is high and K_1 low; (**b**) Plot when V is low and K_1 high.

With this analysis, the need for longitudinal control is acknowledged to ensure that the system is always stable. As 10 m/s is a reasonable speed value for the AGV used, the value of K_1 is set to 1000 s^{-1}. This value refers to the gain of the alignment error control loop.

5. Value of K_2

5.1. System Stability Study

As previously indicated in the stability analysis of the K_1 value, once again a non-linear system is observed, so Lyapunov is employed in order to ensure stability.

The second scenario to be investigated is when the angle of the trajectory needs to be taken into account, as shown in Figure 6b. As already known, the system has highly fast dynamics so Equation (1) is adapted because the AGV assumes the desired direction very quickly. As K_1 is already defined, it can be ignored.

$$\dot{\theta} = \varphi_{setpoint}(e) + \varphi_{path}(e) \tag{26}$$

Therein situation, φ_{path} must also attend a function that depends on the positioning error, due to the position (x, y) of the AGV. Depending on this, the most adjacent point of the path will vary. In this case, the positioning error is calculated as in Equations (27) and (28).

$$(x_{near}, y_{near}) = Argmin\left(\|(x, y) - \left(x_{path}, y_{path}\right)\|\right) \tag{27}$$

$$\varphi_{path}(e) = \varphi_{path}(x_{near}, y_{near}) \tag{28}$$

Resolving this analytically can be complex. Remembering the Lyapunov system of Equation (5), the e can be defined as in Equation (9). However, this time it is necessary to calculate the parameters A, B and C, considering the position of the AGV because the sign of e depends on that.

$$\left\{(A,\ B,\ C) = f\left((x_{near}, y_{near}), \vec{p}\right)\right. \tag{29}$$

Due to the added difficulty, the systems must be expounded in a discrete form. An optimization algorithm is proposed in which e is calculated for every point in a bounded area. With this information and fixing θ, K_2 is varied, allowing its value to be dictated. The following system is considered at instant t.

$$\theta(x, y, K_2) = f(e, K_2) + \varphi_{path}(e) \tag{30}$$

$$e(x(t), y(t)) = e(t) \tag{31}$$

Knowing that e and θ depend on the position of the AGV and K_2, it is possible to determine some expression at t + dt, taking into account Equations (3) and (4).

$$x(t + dt) = x + V\cos\theta \tag{32}$$

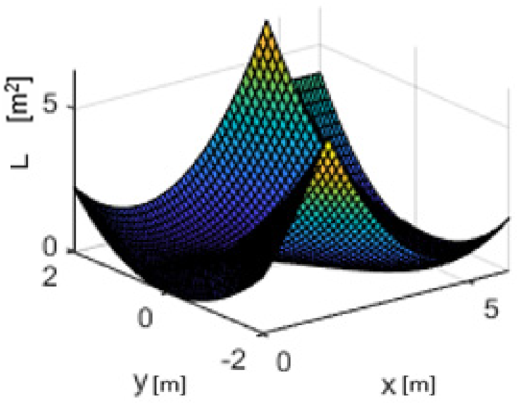
$$y(t + dt) = y + V\sin\theta \tag{33}$$

$$e(x(t + dt), y(t + dt)) = e(t + dt) \tag{34}$$

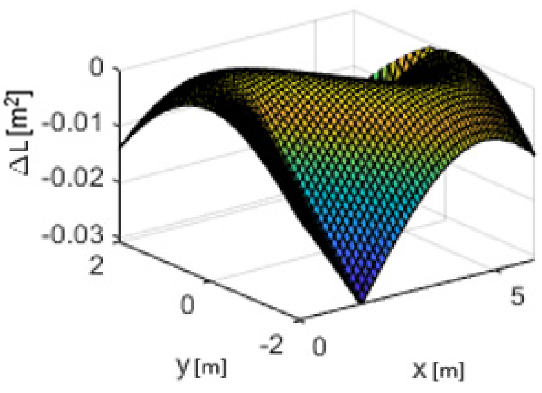
$$\Delta e = e(t + dt) - e(t) \tag{35}$$

Equation (35) has to be negative to ensure stability. The problem is discontinuities can be generated. This issue occurs when the closest point of the path at t is not the same at t + dt or does not continue in the corresponding direction. Instead of considering the whole e (from AGV to the path), it is analysed in sections, guaranteeing that Equation (35) is negative. Hence, the value of Δe is to be taken in absolute values to demonstrate the system is stable. A simulation is performed on all $\vec{p}$ of a bounded area to visualize this phenomenon. It is affected by a sinusoidal trajectory as it is more in line with reality (curves and straight lines).

Figure 8a shows that the Lyapunov energy function is consistently positive over the whole space. L represents a (m^2) value. The derivative of L in Figure 8b is negative throughout the space considered, coinciding with the value of the variation of the positioning error. Therefore, the stability of the system is confirmed.



(**a**)



(**b**)

Figure 8. Results of the simulation: (**a**) The values of L in all the space; (**b**) The values of ΔL in all the space.

5.2. *Optimization of Value of K_2*

Once it is comprehended that the system is stable, a code has been generated that tests all the values of K_2 in a range for the whole amount $\vec{p}$. By setting a value of K_2 it is possible to visualize the evolution of Δe, and the optimal value can be acquired. In Equations (36) and (37), a cost function is proposed that depends on the mean square error, denoted as J.

$$\| e \| = \sqrt{(x - x_{near})^2 + (y - y_{near})^2} \tag{36}$$

$$J = \frac{1}{t}\left(\int_0^t \| e\left(K_2, \vec{p},\ path\right) \| \, dt\right) = \frac{1}{N}\sum_{k=0}^{k=N} e^2(k) \tag{37}$$

This provides the average of J, as a function of the initial position of the AGV.

$$E_{\vec{p}}\left(K_2, \vec{p},\ path\right) = \bar{J}\left(K_2, \vec{p},\ path\right) \tag{38}$$

The path persists in attending to a non-variable parameter, just like $\vec{p}$, so to vary J the entirely dependent value is K_2, formulated in Equation (39).

$$K_2 = Argmin\left(\bar{J}\right) \tag{39}$$

The optimization strategy in this test is to perform an exhaustive analysis of all possible combinations of the space in order to get the most optimal result. This approach is simulated, resulting in the K_2 values presented in Figure 9.

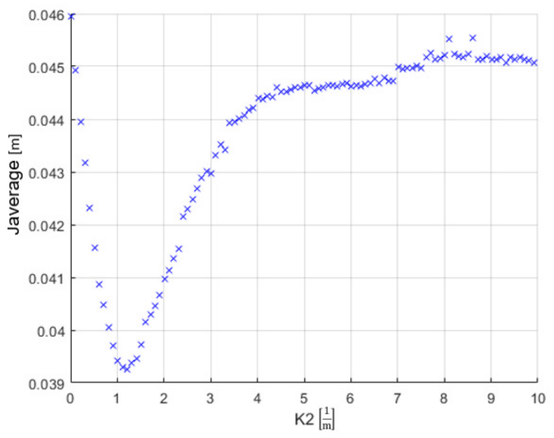


Figure 9. The optimal value of K_2.

This figure also shows that there is an optimal value that minimizes J. Note that K_2 attends the parameter that acts on the rate of evolution of the $\varphi_{\text{setpoint}}$. Accordingly, this value is set as K_2 = 1.21 (1/m).

6. Longitudinal Control Algorithm

As already noted, the trajectory represents an acknowledged fact, so it is possible to identify the state of a point at t + 1. It is equally recognized that there are curves in the path so each point must provide a tangential acceleration, denoted as a_T (m/s^2) and a normal acceleration, denoted as a_N (m/s^2). The latter can be defined as in Equation (40), from which ρ (m) can be known.

$$a_N = \frac{V^2}{\rho} = \omega^2\rho = V\omega \tag{40}$$

In the same way, it is decided that a_N needs to be under maximum speeding up, called a_{Nmax} (m/s^2), as in Equation (41).

$$\parallel a_N \parallel \leq a_{Nmax} \tag{41}$$

Due to Equation (41), it is possible to achieve the linear velocity from Equation (40).

$$V = \frac{a_N}{\omega} \tag{42}$$

Keep in mind that the angular velocity is given by K_1 and K_2, so a maximum velocity can be fixed as well, as in Equation (43).

$$V_{max}{}' = \frac{a_{Nmax}}{\omega} \tag{43}$$

Therefore, by taking Equation (43) it is possible to enhance the value of the linear velocity of the AGV, allowing it to be determined by the positioning error.

$$V = \min\left(\frac{a_{Nmax}}{\omega},\ V_{max} = f(e)\right) \tag{44}$$

Equation (44) contemplates the velocity policy designed in a previous section. Depending on e the AGV adjusts the speed, but it will also be subject to the curvature of the trajectory. It is substantial to know the angle that the path will occupy at the continuous instant, being able to predict the V at t + 1. In that manner, the AGV will have knowledge of if it is close to a curve, allowing it to reduce the velocity. Therefore, with this idea of prediction, the vectors that compose the acceleration are proposed, perceiving the relation of Equation (45), where a represents the acceleration (m/s^2).

$$a^2 = a_N{}^2 + a_T{}^2 \rightarrow a_T = \sqrt{a^2 - a_N{}^2} \tag{45}$$

With this information, it is possible to propose the velocity at t + 1 for the AGV.

$$V(t+1) = V(t) + \Delta t{\cdot}a_T \tag{46}$$

In Equation (46) it is viable to substitute a_T, as seen in Equation (40).

$$V(t+1) = V(t) + \Delta t\sqrt{a^2 - V(t)^2\omega(t)^2} \tag{47}$$

Knowing all the variables of Equation (47), it is possible to obtain the value of the maximum speed at t + 1, denoted as $V_{max}{}''$. Taking into account Equation (44), Equation (48) is defined.

$$V(t+1) = \min\left(V_{max}{}'',\ V_{max}{}' = f(e)\right) \tag{48}$$

To such a degree, if the AGV maintains a significantly high angular velocity, the linear speed is reduced. The parameter $V_{max}{}'$ is provided by the AGV, representing the maximum nominal velocity.

$$V(t+1) = f(e,\ \omega(t), V(t),\ a) \tag{49}$$

Ultimately, e is contemplated, attaching importance to the trajectory execution speed.

7. Results

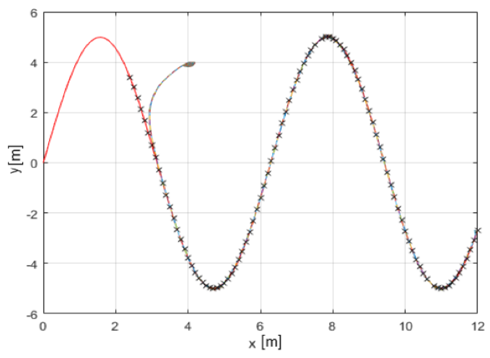
Beforehand, the algorithm is proved in simulation, analysing the compliance drop the various objectives. As designed, the lateral control and the longitudinal control work together to allow proper trajectory tracking. In the simulations, it is observed that regardless of the values of the $\overrightarrow{pose}$, the autonomous vehicle redirects and reaches the path. As the AGV approaches, it adjusts itself to be able to develop over the trajectory and not overshoot it, obtaining correct trajectory tracking.

At the beginning of the simulation, the algorithm can determine the closest position of the trajectory based on the AGV. Therefore, the first objective is to attain that point. In this situation, the value of the positioning error is considerable. Regardless of the initial θ, the AGV is required to take an angle of $\frac{\pi}{2}$ (rad) to correct the value of e and reach the trajectory quickly. When the AGV assumes $\theta = \frac{\pi}{2}$ rad, the forward motion begins. With the decrease in the positioning error, the value of the vehicle angle starts adapting, adjusting to fit the trajectory angle and generating a curve. The linear velocity also starts increasing. As the AGV approaches the path, the closest point is necessarily unmaintained.

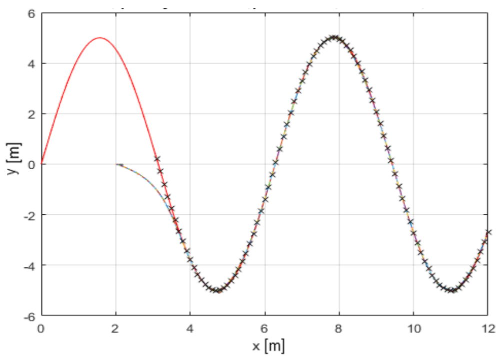
While the positioning error decreases, the alignment error has to do so as well. Because of that, and as the closest point is changing, the value of φ is adapting.

When the AGV is on a straight trajectory, both error values are approximately null, making accurate tracking of the trajectory achievable. In this case, the speed of the movement is limited by the V_{nom} of the AGV. In the case of a curve in the path, it can be observed that the autonomous vehicle reduces its speed to prevent deviating from the predetermined path. Simultaneously, it is making a constant redirection to adhere to the reduced value of alignment error.

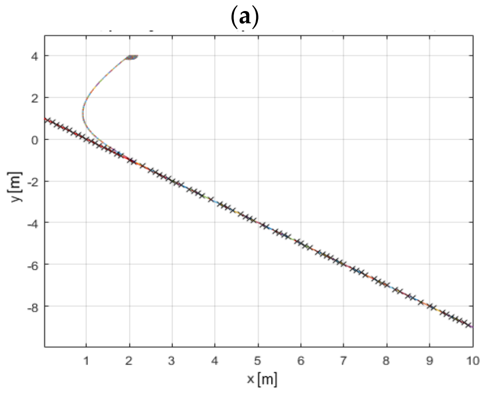
In this way, it is proved that the designed algorithm produces a satisfactory result due to the good following of any type of trajectory as can be seen in Figure 10. It exhibits diverse types of paths and $\overrightarrow{pose}$s of the AGV. The red line represents the established trajectory. The black crosses are the nearest point calculated by the algorithm. The coloured line represents the route followed by the AGV. It can be perceived that irrespective of the positioning error, the navigation algorithm performs well in all cases. In Figure 10a,b, the established path is sinusoidal where it can be seen how the AGV can reach the trajectory and adapt to it in both cases. In Figure 10c,d a purely linear trajectory is considered where the AGV also tracks the path well. At long last, a fully curved trajectory is presented in Figure 10e,f. Once more, the following is performed accurately.



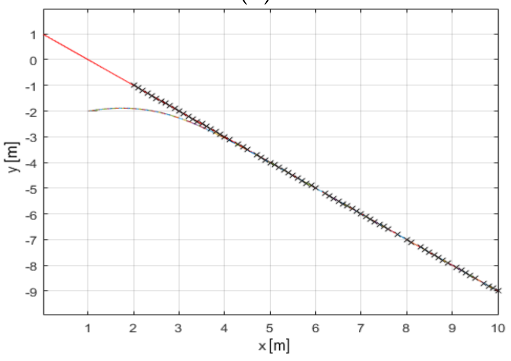
(**a**)



(**b**)



(**c**)



(**d**)

Figure 10. *Cont.*

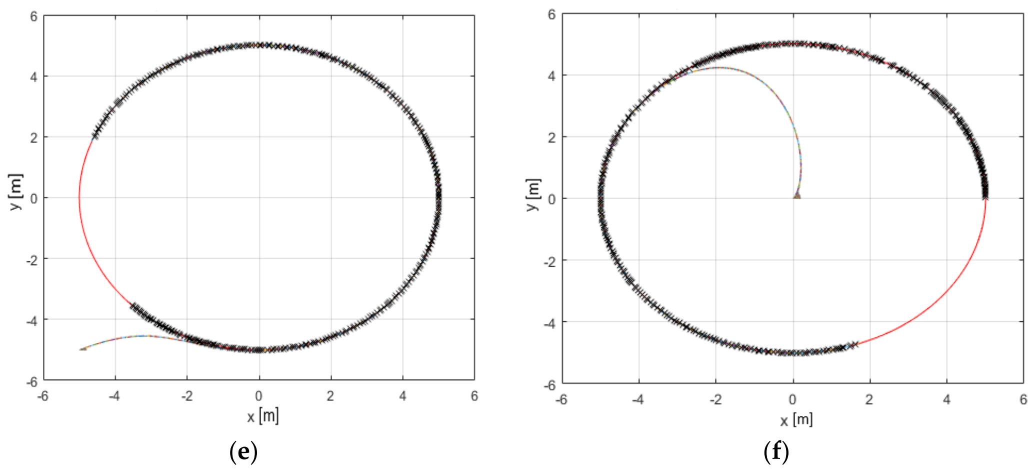


Figure 10. Different path simulation: (**a**) Sinusoidal with AGV at the right side; (**b**) Sinusoidal with AGV at the left side; (**c**) Linear with AGV at the right side; (**d**) Linear with AGV at the left side; (**e**) Circular with AGV outside; (**f**) Circular with AGV inside.

Additionally, it can be marked from all the graphs in Figure 10 that the tracking of the trajectory is correct independently of the positioning of the AGV and therefore, of the sign of *e*.

It is equally necessary to test whether the algorithm as a whole is suitable for the real AGV. To perform this, two instances are created in MATLAB. One of them processes the image, and the other one executes the navigation algorithm. These are communicated by ROS nodes, to give concurrency to the execution. The movement of the AGV is achieved with a PLC. It is observed that the AGV can follow the whole route correctly.

In the interest of clarifying the execution times of the algorithm, it should be marked that the processor employed is Intel® Core™ i9-9880H CPU @ 2.30 GHz 3.30 GHz. The RAM memory of the computer on which the algorithm is executed is 16 GB and it has an NVIDIA Quadro T1000 graphics card.

Under these conditions, the neural network execution time is 144ms. This time is incorporated into the total period of the ROS publishing node, which is the one that manages the images and takes 186 ms to send the trajectory data vector, measured as an average of 1550 executions.

Regarding the subscriber node, that is the one that executes the control and sends the commands to the PLC, it takes 136 ms on average in 5510 iterations.

8. Discussion

In considering the advantages of this study, comparisons with other techniques commonly used in the control of AGVs are mentioned.

On the one hand, one of the most frequently used sensor techniques in this type of vehicles is LiDAR. These devices are highly effective when it comes to localizing ad receiving data related to the environment in which the AGV is located. In studies such as the one performed by Quan and Chen [46], these devices are employed in conjunction with the odometry of the wheels in order to localize an autonomous guided vehicle. Despite their extensive use, these sensors do not provide the necessary robustness for this type of systems. In the present paper, this robustness is consistently achieved.

On the other hand, in the industrial field, it has been frequent to employ philo-guided vehicles (see Chet et al. [47]). These vehicles use electromagnetism to perform navigation, providing the necessary robustness and accuracy. However, it is not a flexible solution. The AGV and navigation system presented in this work have the benefit of achieving minimum cost when implementing a fixed trajectory. By the use of tape of a determined color, any type of path can be established without the need for expensive and specific infrastructure.

Furthermore, with the re-training of the neural network the path can be adapted to any color or operating area.

To conclude, it is possible to get the positioning error committed with this navigation algorithm in a specific trajectory, as graphited in Figure 11. The figure, therefore, shows that in curved areas, like in the beginning and the end iterations, the positioning error increases, because of the location of the wheels and the camera in the AGV itself. However, in the central iterations, it can be appreciated that in straight areas the positioning error is close to zero.

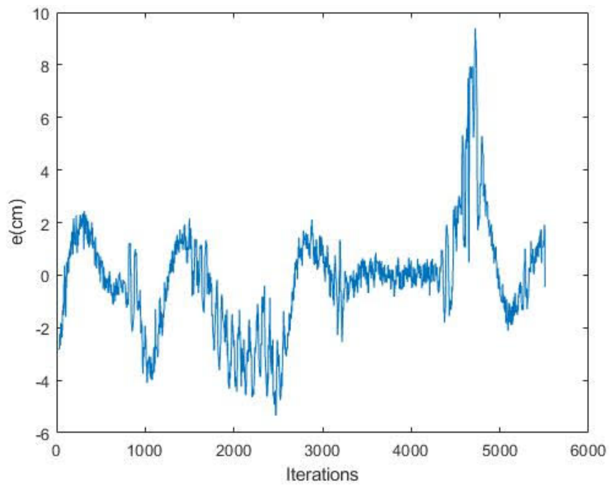


Figure 11. Positioning error values in a specific trajectory.

As demonstrated by Hoffmann et al. [45] in the Stanley algorithm, a typical RMS cross-track error of less than 0.1 m is obtained. In the case of this modification of the algorithm, the RMS of the positioning error is around 0.02 m.

9. Conclusions

The present work focuses on the search for solutions for the navigation and localization of an AGV, because of the need to discover robust techniques, achieving greater precision and reliability.

Fundamentally, it is demonstrated that the proposed algorithm modification is stable. As mentioned in the previous section, the advantages over conventional techniques can be recognized. This algorithm is remarkably simple, requiring no previous training to perform properly. When changing path, no retraining is necessary, it merely requires the colour of the trajectory to match.

In terms of navigation, starting from a base such as the Stanley algorithm and presenting it another perspective comprehends its complexity. In addition, demonstrating stability and ensuring the given solution is adequate is uneasy. For this, other alternatives have to be tested until a solution is obtained that proves its robustness. Therein way, the values of the constants can be demonstrated and a sense for them, as well. This issue has been one of the most arduous tasks of this work.

The main objective set in this work represents the stability analysis of the modification of a navigation algorithm capable of performing lateral control and longitudinal control. This has been achieved, obtaining satisfactory results. As mentioned, the difficulty was in the stability analysis, but due to the results, its proper performance has been demonstrated.

Continuing with localization, instead of designing a new algorithm, some data derived from the other algorithms are employed. In this case, the necessary parameters can be obtained from the navigation and path planning codes, without using sensors.

Sensors remain the conventional method for calculating the trajectory. In this work, it is completed with artificial vision and a neural network, a less common method but with remarkably pleasing results as well. The latter algorithm was considered with Hough transforms, for example, but concluded that the use of a neural network was more appropriate. Nevertheless, in some areas, further study is required to achieve more accuracy. These areas are related to light shimmers generated in the image that create indistinctness.

Furthermore, it has been possible to decouple the problem of data acquisition from the problem of navigation. Consequently, until a new image is received, predictions are used.

Overall, the work accomplishes the objective of ensuring stability of an algorithm for the free navigation of an industrial autonomous vehicle.

As a prolongation of this research, attempts will be made to enhance the architecture of the convolutional neural network with the aim of achieving higher speed rates.

For future work, the execution of the algorithm in real-time but with more excessive speed may reveal a lack in the concurrency of the ROS nodes. An analysis of resource consumption can in addition be effected.

On the other hand, numerous tests can be done with the neural network, changing the configuration or performing different learning, to improve the accuracy of the navigable path. This can give an idea of the characteristics that the CNN may need for this application.

An explicability analysis using the LIME technique can clarify whether the semantic segmentation errors are related to the similarities between the images. In all images, the trajectory is mostly linear and close to the centre of the image. This analysis provides insight into what the neural network relies on to classify the parts of the image. It will also be interesting to investigate the interpretation of deep learning.

In addition, if the semantic segmentation has mask discontinuities to compute the trajectory, it is necessary to consider alternatives in the algorithm to finalize the navigation.

As for the algorithm used, it is necessary to incorporate the case where there is no trajectory and how to stop the AGV's movement. In the simulation, the AGV attains the final of the trajectory and turns π (rad), following the path in reverse. In the real AGV, when it does not visualize any more trajectory, it starts to turn on its own to detect the yellow line again. Therefore, there exists an understandable need for a stopping policy that contemplates different situations.

Author Contributions: Conceptualization, M.C.-O. and E.Z.; methodology, M.C.-O.; software, M.C.-O. and E.Z.; validation, E.Z. and U.F.-G.; formal analysis, E.Z. and A.T.-F.-B.; investigation, M.C.-O. and A.S.-C.; resources, U.F.-G.; writing—original draft preparation, M.C.-O. and E.Z.; writing—review and editing, U.F.-G.; All authors have read and agreed to the published version of the manuscript.

Funding: The current study has been sponsored by the Government of the Basque Country-ELKARTEK21/10 KK-2021/00014 ("Estudio de nuevas técnicas de inteligencia artificial basadas en Deep Learning dirigidas a la optimización de procesos industriales") research program.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

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

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

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

References

1. Kilic, C.; Ohi, N.; Gu, Y.; Gross, J. Slip-Based Autonomous ZUPT through Gaussian Process to Improve Planetary Rover Localization. *IEEE Robot. Autom. Lett.* **2021**, *6*, 4782–4789. [CrossRef] [PubMed]
2. Chen, S.Y. Kalman filter for robot vision: A survey. *IEEE Trans. Ind. Electron.* **2012**, *59*, 4409–4420. [CrossRef]
3. Cho, H.; Kim, E.K.; Kim, S. Indoor SLAM application using geometric and ICP matching methods based on line features. *Rob. Auton. Syst.* **2018**, *100*, 206–224. [CrossRef]
4. Shamsfakhr, F.; Sadeghi Bigham, B. GSR: Geometrical scan registration algorithm for robust and fast robot pose estimation. *Assem. Autom.* **2020**, *40*, 801–817. [CrossRef]

5. Naus, K.; Marchel, Ł. Use of a weighted ICP algorithm to precisely determine USV movement parameters. *Appl. Sci.* **2019**, *9*, 3530. [CrossRef]
6. Wang, Y.T.; Peng, C.C.; Ravankar, A.A.; Ravankar, A. A single LiDAR-based feature fusion indoor localization algorithm. *Sensors* **2018**, *18*, 1294. [CrossRef]
7. Senin, N.; Colosimo, B.M.; Pacella, M. Point set augmentation through fitting for enhanced ICP registration of point clouds in multisensor coordinate metrology. *Robot. Comput. Integr. Manuf.* **2013**, *29*, 39–52. [CrossRef]
8. Gao, Y.; Liu, S.; Atia, M.M.; Noureldin, A. INS/GPS/LiDAR integrated navigation system for urban and indoor environments using hybrid scan matching algorithm. *Sensors* **2015**, *15*, 23286–23302. [CrossRef]
9. Kim, H.; Song, S.; Myung, H. GP-ICP: Ground Plane ICP for Mobile Robots. *IEEE Access* **2019**, *7*, 76599–76610. [CrossRef]
10. Sadeghi Bigham, B.; Dolatikalan, S.; Khastan, A. Minimum landmarks for robot localization in orthogonal environments. *Evol. Intell.* **2021**, *1*, 1–4. [CrossRef]
11. Yap, Y.Y.; Khoo, B.E. Landmark-based Automated Guided Vehicle Localization Algorithm for Warehouse Application. *Pervasive Health Pervasive Comput. Technol. Healthc.* **2019**, 47–54. [CrossRef]
12. Gao, X.; Zhang, T. Robust RGB-D simultaneous localization and mapping using planar point features. *Rob. Auton. Syst.* **2015**, *72*, 1–14. [CrossRef]
13. Clemens, J.; Kluth, T.; Reineking, T. β-SLAM: Simultaneous localization and grid mapping with beta distributions. *Inf. Fusion* **2019**, *52*, 62–75. [CrossRef]
14. Gentner, C.; Jost, T.; Wang, W.; Zhang, S.; Dammann, A.; Fiebig, U.C. Multipath Assisted Positioning with Simultaneous Localization and Mapping. *IEEE Trans. Wirel. Commun.* **2016**, *15*, 6104–6117. [CrossRef]
15. Yang, P.; Wu, W. Efficient particle filter localization algorithm in dense passive RFID tag environment. *IEEE Trans. Ind. Electron.* **2014**, *61*, 5641–5651. [CrossRef]
16. Zhang, Q.B.; Wang, P.; Chen, Z.H. An improved particle filter for mobile robot localization based on particle swarm optimization. *Expert Syst. Appl.* **2019**, *135*, 181–193. [CrossRef]
17. Carrera Villacres, J.L.; Zhao, Z.; Braun, T.; Li, Z. A Particle Filter-Based Reinforcement Learning Approach for Reliable Wireless Indoor Positioning. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 2457–2473. [CrossRef]
18. Wang, L. Automatic control of mobile robot based on autonomous navigation algorithm. *Artif. Life Robot.* **2019**, *24*, 494–498. [CrossRef]
19. Tao, B.; Wu, H.; Gong, Z.; Yin, Z.; Ding, H. An RFID-Based Mobile Robot Localization Method Combining Phase Difference and Readability. *IEEE Trans. Autom. Sci. Eng.* **2021**, *18*, 1406–1416. [CrossRef]
20. Lu, S.; Xu, C.; Zhong, R.Y. An Active RFID Tag-Enabled Locating Approach with Multipath Effect Elimination in AGV. *IEEE Trans. Autom. Sci. Eng.* **2016**, *13*, 1333–1342. [CrossRef]
21. Kashyap, A.K.; Parhi, D.R.; Muni, M.K.; Pandey, K.K. A hybrid technique for path planning of humanoid robot NAO in static and dynamic terrains. *Appl. Soft Comput. J.* **2020**, *96*, 106581. [CrossRef]
22. Brock, O.; Khatib, O. High-speed navigation using the global dynamic window approach. In Proceedings of the 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C), Detroit, MI, USA, 10–15 May 1999; Volume 1, pp. 341–346. [CrossRef]
23. Liu, L.S.; Lin, J.F.; Yao, J.X.; He, D.W.; Zheng, J.S.; Huang, J.; Shi, P. Path Planning for Smart Car Based on Dijkstra Algorithm and Dynamic Window Approach. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 8881684. [CrossRef]
24. Dobrevski, M.; Skocaj, D. Adaptive dynamic window approach for local navigation. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October–24 January 2021; pp. 6930–6936. [CrossRef]
25. Fox, D.; Burgard, W.; Thrun, S. The dynamic window approach to collision avoidance. *IEEE Robot. Autom. Mag.* **1997**, *4*, 23–33. [CrossRef]
26. Wang, T.; Yan, X.; Wang, Y.; Wu, Q. A distributed model predictive control using virtual field force for multi-ship collision avoidance under COLREGs. In Proceedings of the 2017 4th International Conference on Transportation Information and Safety (ICTIS), Banff, AB, Canada, 8–10 August 2017; pp. 296–305. [CrossRef]
27. Burgos, E.; Bhandari, S. Potential flow field navigation with virtual force field for UAS collision avoidance. In Proceedings of the 2016 International Conference on Unmanned Aircraft Systems ICUAS 2016, Arlington, VA, USA, 7–10 June 2016; pp. 505–513. [CrossRef]
28. Borenstein, J.; Koren, Y. The Vector Field Histogram—Fast obstacle avoidance for mobile robots. *IEEE J. Robot. Autom.* **1991**, *7*, 278–288. [CrossRef]
29. Ye, C. Navigating a mobile robot by a traversability field histogram. *IEEE Trans. Syst. Man, Cybern. Part B Cybern.* **2007**, *37*, 361–372. [CrossRef]
30. AbdElmoniem, A.; Osama, A.; Abdelaziz, M.; Maged, S.A. A path-tracking algorithm using predictive Stanley lateral controller. *Int. J. Adv. Robot. Syst.* **2020**, *17*, 1–11. [CrossRef]
31. Fei, W.; Ziwei, W.; Meijin, L. Robot Path Planning Based on Improved Particle Swarm Optimization. In Proceedings of the 2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering ICBAIE 2021, Nanchang, China, 26–28 March 2021; Volume 21, pp. 887–891. [CrossRef]

32. Liu, Z.; Liu, H.; Lu, Z.; Zeng, Q. A Dynamic Fusion Pathfinding Algorithm Using Delaunay Triangulation and Improved A-Star for Mobile Robots. *IEEE Access* **2021**, *9*, 20602–20621. [CrossRef]
33. Cheng, L.; Liu, C.; Yan, B. Improved hierarchical A-star algorithm for optimal parking path planning of the large parking lot. In Proceedings of the ICIA 2014—IEEE International Conference on Information and Automation, Hailar, China, 28–30 July 2014; pp. 695–698. [CrossRef]
34. da Silva Costa, L.; Tonidandel, F. DVG+A* and RRT Path-Planners: A Comparison in a Highly Dynamic Environment. *J. Intell. Robot. Syst.* **2021**, *101*, 1–20. [CrossRef]
35. Wang, J.; Li, B.; Meng, M.Q.H. Kinematic Constrained Bi-directional RRT with Efficient Branch Pruning for robot path planning. *Expert Syst. Appl.* **2021**, *170*, 114541. [CrossRef]
36. Wei, K.; Ren, B. A method on dynamic path planning for robotic manipulator autonomous obstacle avoidance based on an improved RRT algorithm. *Sensors* **2018**, *18*, 571. [CrossRef]
37. Tang, C.; Sun, R.; Yu, S.; Chen, L.; Zheng, J. Autonomous Indoor Mobile Robot Exploration Based on Wavefront Algorithm. *Lect. Notes Comput. Sci.* **2019**, *11744*, 338–348.
38. Wu, S.; Du, Y.; Zhang, Y. Mobile Robot Path Planning Based on a Generalized Wavefront Algorithm. *Math. Probl. Eng.* **2020**, *2020*, 6798798. [CrossRef]
39. Sdwk, U.; Edvhg, S.; Ohduqlqj, R.Q.; Dqj, L.; Lqj, L.X.; Hiilflhqf, F.; Wkh, R.I.; Dojrulwkp, S.; Ri, J.; Vlqfh, U.; et al. Mobile robot path planning based on Q-learnig algorithm*. In Proceedings of the 2019 WRC Symposium on Advanced Robotics and Automation (WRC SARA), Beijing, China, 21–22 August 2019; pp. 160–165.
40. Gao, J.; Ye, W.; Guo, J.; Li, Z. Deep reinforcement learning for indoor mobile robot path planning. *Sensors* **2020**, *20*, 5493. [CrossRef]
41. Zheng, K.; Gao, J.; Shen, L. UCAV Path Planning Algorithm Based on Deep Reinforcement Learning. *Lect. Notes Comput. Sci.* **2019**, *11902*, 702–714.
42. Teso-Fz-Betoño, D.; Zulueta, E.; Sánchez-Chica, A.; Fernandez-Gamiz, U.; Saenz-Aguirre, A. Semantic segmentation to develop an indoor navigation system for an autonomous mobile robot. *Mathematics* **2020**, *8*, 855. [CrossRef]
43. Olgun, M.C.; Baytar, Z.; Akpolat, K.M.; Koray Sahingoz, O. Autonomous vehicle control for lane and vehicle tracking by using deep learning via vision. In Proceedings of the 2018 6th International Conference on Control Engineering and Information Technology, CEIT 2018, Istanbul, Turkey, 25–27 October 2018; pp. 25–27. [CrossRef]
44. Azimi, S.M.; Fischer, P.; Korner, M.; Reinartz, P. Aerial LaneNet: Lane-Marking Semantic Segmentation in Aerial Imagery Using Wavelet-Enhanced Cost-Sensitive Symmetric Fully Convolutional Neural Networks. *IEEE Trans. Geosci. Remote Sens.* **2019**, *57*, 2920–2938. [CrossRef]
45. Hoffmann, G.M.; Tomlin, C.J.; Montemerlo, M.; Thrun, S. Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing. In Proceedings of the 2007 American Control Conference, New York, NY, USA, 9–13 July 2007; pp. 2296–2301. [CrossRef]
46. Quan, S.; Chen, J. AGV localization based on odometry and LiDAR. In Proceedings of the 2019 2nd World Conference on Mechanical Engineering and Intelligent Manufacturing (WCMEIM), Shanghai, China, 22–24 November 2019; pp. 483–486. [CrossRef]
47. Chen, X.; Lin, W.; Liu, J.; Guan, L.; Zheng, Y.; Gao, F. Electromagnetic Guided Factory Intelligent AGV. In Proceedings of the 2016 3rd International Conference on Mechatronics and Information Technology, Shenzhen, China, 9–10 April 2016; pp. 1–6. [CrossRef]