





Article

A Free Navigation of an AGV to a Non-Static Target with Obstacle Avoidance

Daniel Teso-Fz-Betoño ^{1,*} , Ekaitz Zulueta ¹, Unai Fernandez-Gamiz ² , Iñigo Aramendia ² 
and Irantzu Uriarte ³ 

¹ System Engineering and Automation Control Department, University of the Basque Country (UPV/EHU), Nieves Cano, 12, 01006 Vitoria-Gasteiz, Spain; ekaitz.zulueta@ehu.eus

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 (U.F.-G.); inigo.aramendia@ehu.eus (I.A.)

³ Department of Mechanical Engineering, University of the Basque Country (UPV/EHU), Nieves Cano, 12, 01006 Vitoria-Gasteiz, Spain; irantzu.uriarte@ehu.eus

* Correspondence: daniel.teso@ehu.eus

Received: 8 November 2018; Accepted: 23 January 2019; Published: 1 February 2019



Abstract: The industry is changing in order to improve the economy sector. This is the reason why technology is improving and developing new devices. The autonomous guided vehicle with free navigation is a new machine, which uses different techniques to move such as mapping, localization, path planning, and path following. In this paper, a path following is proposed. The path following is called moving to a point, which uses the proportional distance between the target and the autonomous guided vehicles (AGV) to calculate the velocity and direction. If some obstacles appear in the trajectory, however, the vehicle stops. Instead of stopping the machine, by using moving to a point logic, an obstacle avoidance function will be implemented. In this implementation, different parameters can be configured, such as: security distance, which determinates when the obstacle avoidance must correct the pose; and proportional values, which modify the velocity and steering commands. It is also compared to a dynamic window approach (DWA) obstacle avoidance solution. Additionally, the AGV navigates to a non-static target with a path following algorithm.

Keywords: Industry 4.0; Cobots; autonomous guided vehicles; free navigation; path following; moving to a point; obstacle avoidance; dynamic window approach; indoor mobile robots

1. Introduction

Nowadays, the industry is changing its organizational structure. This change is known as Industry 4.0, which is intended to improve the economy sector. Companies will become more intelligent by using diverse techniques [1]. To make this possible, new programmable logic controller (PLC) devices will be introduced, which will transfer huge calculation and information capabilities, and will also be used to introduce more robots in production lines etc. Apart from introducing new devices, there will also be increased connectivity into the same information system. As a result, production can be adapted to demand and become more efficient [2]. In the end, the aim is to have a greater production capability, better quality, and optimization in production resources.

Robotics is changing the production concept and adapting to this new era. Today, robots and humans work in the same areas. Djuric et al. [3] commented that these robots are known as Cobots. Thanks to power and force limitations, the employees can stay together in the same working area. According to Cherubini et al. [4], when they are cooperating, ergonomic concerns can be reduced, due to the physical job and cognitive loading. It also provides improvements on safety, quality, and productivity.

In order to obtain greater flexibility and efficiency, not only will the robots change, but the autonomous guided vehicle (AGV) also introduces new updates. Nowadays, these AGVs only move using magnetic fields, so they do not have any intelligence, which makes it too difficult to change logistics areas [5]. However, to introduce a free navigation intelligence, mapping, localization, path planning, and path following aspects have to be implemented.

Mapping and localization are solved using simultaneous localization and mapping (SLAM) [6,7]. Considering the mobile robot has no idea about where it is and where it is going to be, with SLAM it is possible to learn the environment characteristics in order to locate the vehicle. In the case that it was necessary to move it, another algorithm would be required. Here is where path planning and path following take part. Elsheikh et al. [8] confirmed, the differences between these, lie in how much the robot can do.

In path planning, the start and end points are given and then the vehicle must plan the best path. The robot navigation problem can be described as the process of determining a suitable and safe path, avoiding obstacles and collisions [9], therefore, the AGV navigates without any path following restrictions and moves wherever it wants to arrive at its goal position. There are different alternatives which can be implemented, such as swarm optimization (SO), which is based on particle swarm optimization as Ever [10] analysed, Hu et al. [11] studied dynamic path planning, and dynamic movements primitives is explained by Mei et al. [12].

Apart from these, there are other well-known path planning algorithms, such as the dynamic window approach developed by Fox et al. [13], vector field histogram related by Borenstein et al. [14], TangentBug and PointsBug, as compared by Buniyamin et al. [15], and the rapidly exploring random tree, as Adiyatov et al. [16] explained. Most of these well-known algorithms are used in actual applications.

Otherwise, in path following problems, the user gives the robot the path and it should only follow it. According to Wang [17], these controllers are designed to obtain the minimal lateral distance, as well as the heading between the vehicle and the command path to control the vehicle speed and steering to follow a specified path. Algorithms like pure pursuit [18] mentioned, moving to a point [19], and proportional navigation [20].

Sebi et al. [21] advised that apart from using SLAM, path following, and path planning, another algorithm is required. Without any obstacle avoidance intelligence, the AGV makes a safety stop to prevent collision with an object. This is the reason for the proposal of different obstacle avoidance algorithms [22–24]. In most cases the obstacles are detected using cameras, LIDAR, or ultrasonic sensors as Amin et al. [25] and Martinez et al. [26] explained. In the end, all of these sensors are trying to avoid making contact with the obstacle. However, there are some cases, that when the obstacle is detected as making a collision, the force is analyzed with piezoelectric sensors like Wooten et al. [27] studied.

The main goal of the present study is to improve a simple path following algorithm introducing an obstacle avoidance technique. This technique is implemented using simple equations, in order to avoid large calculations in the controller. After developing the algorithm, it will be tested with a traditional motion planning algorithm to analyze the differences between both.

2. Problem Formulation

The aim is to take a commercial AGV from industrial application and use its technology to introduce an obstacle avoidance algorithm. This commercial AGV uses a LIDAR to take information from the surroundings, a programmable logic controller (PLC) to control AGV periphery (motors, displays, lights, etc.) and an industrial personal computer (IPC) for the SLAM and path following algorithm. It is remarkable that the only programmable hardware is the PLC, and it does not have enough capability to process huge calculations like optimization.

All hardware is connected via Ethernet to synchronize the information. The IPC sends speed ($\|\vec{V}_{agv}\|$) and steering (γ) commands to the PLC, the PLC sends wheel-odometry information to the IPC, and LIDAR sends information to both the PLC and IPC.

In the Industry, it is better to use known trajectories, which is why these vehicles use only path following algorithms. In general, these algorithms are very simple and cannot avoid an obstacle. Hence, it was decided to implement an obstacle avoidance term into the path following.

Furthermore, the AGV has to arrive at a conveyor transport and synchronize its speeds with the moving target. The collaborative robot that was assembled in the AGV then takes an object from conveyor transport and moves it to another point of the factory. The objective is not to stop the conveyor transport to take that product, in order to reduce production time.

The idea of this paper is to take moving to a point mathematical equations and adapt them to implement avoidance technique. Corke [19] studied a moving to a point path following algorithm, in which the problem of moving towards a goal point (x^*, y^*) in the plane was analyzed. For this solution, the velocity of the robot considered in Equation (1) is controlled proportionally to the distance from the goal, as analysed in Equation (2).

$$\|\vec{V}_{agv}\| = error \cdot K_v \tag{1}$$

$$error = \sqrt{(x^* - x)^2 + (y^* - y)^2} \tag{2}$$

To steer towards the goal, which is represented in Equation (3), it is considered that K_h is the proportional controller, and the vehicle-relative angle is analyzed in Equation (4).

$$\gamma = K_h \cdot (\theta^* - \theta), K_h > 0 \tag{3}$$

$$\theta^* = atan\left(\frac{y^* - y}{x^* - x}\right) \tag{4}$$

Therefore, in the current study, a new obstacle avoidance strategy is proposed to be implemented in the moving to a point technique. It is based on measurement between the vehicle and the obstacle. Table 1 summarizes all terms and parameters used in this work.

Table 1. Terms and Parameters.

Name	Description
$error$ (m)	Gap between AGV and target
$\ \vec{V}_{agv}\ $ (m/s)	Speed setpoint
K_v (s^{-1})	Proportional constant for speed
K_h (rad/m)	Proportional constant for steering
γ (rad)	Steering setpoint
θ (rad)	AGV Rotation
(x^*, y^*, θ^*) (m, m, rad)	Target POSE
D_{obs} (m)	Distance to the obstacle
D_{target} (m)	Distance between the AGV and target
\vec{P}_{obs} (m)	Obstacle position
\vec{P}_{agv} (m)	AGV position
V_l (m/s)	Left wheel speed
V_D (m/s)	Right wheel speed
V_c (m/s)	Speed in the middle of the motorized axel
$G(V_{agv}, \dot{\gamma})$	DWA optimization function
$heading(V_{agv}, \dot{\gamma})$	Gap between AGV and goal for each trajectory
$dist(V_{agv}, \dot{\gamma})$	Distance to the obstacle for each trajectory
$vel(V_{agv}, \dot{\gamma})$	The speed of each trajectory
POSE	AGV position and orientation
AGV	Autonomous guided vehicle
POA	Proposed obstacle avoidance
DWA	Dynamic window approach

3. Implementation of Obstacle Avoidance

A new formulation is presented in this section, so the vehicle can autonomously correct the path avoiding the obstacle. Considering the measurement of the LIDAR, it is possible to obtain the distance between the AGV and the obstacle. According to Peng et al. [22], that distance is expressed in Equation (5).

$$D_{obs} = \sqrt{x_{obs}^2 + y_{obs}^2} \tag{5}$$

$$\vec{P}_{obs} = (x_{obs} \quad y_{obs}) \tag{6}$$

$$\vec{P}_{agv} = (x \quad y) \tag{7}$$

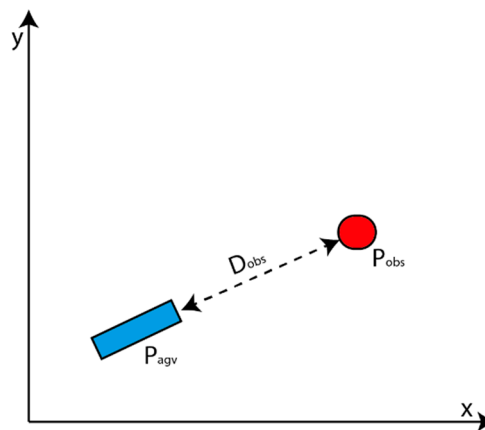


Figure 1. Autonomous guided vehicle (AGV) and obstacle on a plane.

x_{obs} and y_{obs} represent the relative coordinates of the distance between the AGV and the nearest obstacle defined by D_{obs} in Figure 1. The output of the Equation (5) can be represented in a graph. Figure 2 shows how the LIDAR measurement changes when some obstacle appears in the AGV’s way.

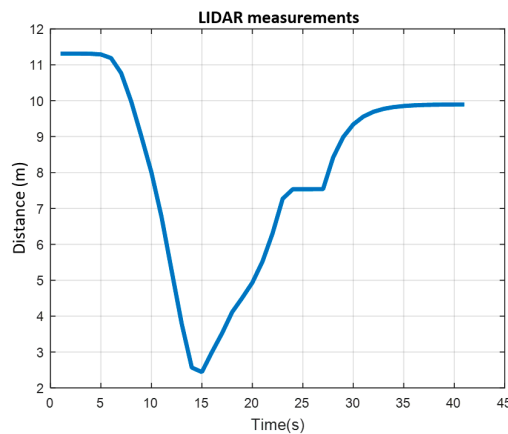


Figure 2. Distance between the object and AGV.

To use that measurement information as a correction for the steering command, the graph must start at 0 value. By subtracting the LIDAR maximum measurement range to D_{obs} , the answer will change. This way, when the AGV is approaching the obstacle, the value becomes more negative. This expression is analysed in equation (8) and the solution is represented in Figure 3.

$$C = D_{obs} - Lidar_{range}^{max} \tag{8}$$

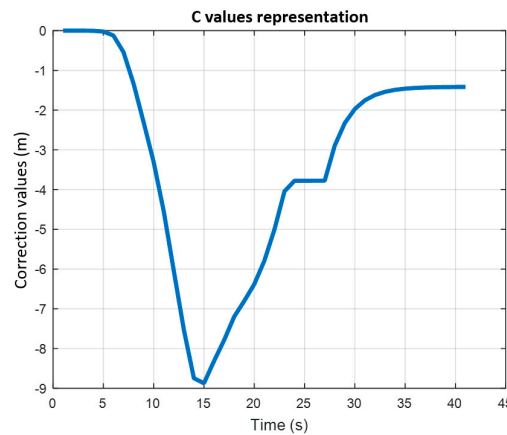


Figure 3. The correction of Figure 2.

In Equation (9), a safety distance term is introduced to prevent premature correction. This safety distance maintains the AGV direction until the object is between the AGV and that distance. In case of the object being far away, the obstacle avoidance stays in standby mode. In other words, it does not interfere with the moving to a point algorithm.

$$Avoid = \beta \cdot C \cdot e^{\alpha - D_{obs}} \tag{9}$$

β is a constant which adapts the intensity of the avoid function and $e^{\alpha - D_{obs}}$ is the safety implementation with a α parameter, which changes the safety distance. This avoid function adapts the AGV direction to avoid the obstacle. Therefore, this term must interact with the γ value. Instead of implementing directly the avoidance correction, it has been considered to introduce some logic between the γ value and avoid value, using a sigmoid function as shown in Equation (10).

$$Com = \frac{1}{1 + e^{\alpha - D_{obs}}} \tag{10}$$

Regrouping all the equations, the Moving to a Point algorithm with obstacle avoidance can be summarized with Equation (11).

$$\|\gamma\| = K_h \cdot [\delta \cdot Com \cdot Avoid + (1 - \varphi \cdot Com) \cdot (\theta^* - \theta)] \tag{11}$$

where δ , y , and φ are constants, which modify the interaction with the Com function. K_h is a proportional constant, which changes the measurement value to steering value. Until this point, the AGV can avoid an obstacle and it always turns to the right. In some cases, however, it is better to turn to the left so as not to lose too much time avoiding the obstacle. It can be done by analyzing the sign of the angles as represented in Equation (12).

$$\gamma = Sign(Arg(\vec{P}_{agv}) - Arg(\vec{P}_{obs})) \cdot \|\gamma\| \tag{12}$$

The vehicle velocity can be modified considering the same logic. When the obstacle is close, the AGV must reduce its speed. Taking Equation (9) as reference, Equation (13) can be made, which considers a variation of speed changing β by μ .

$$Avoid_{vel} = \mu \cdot C \cdot e^{\alpha - D_{obs}} \tag{13}$$

In addition, this function has to interact with the moving to a point velocity command, which is the reason why Equations (10) and (13) are combined to obtain Equation (14).

$$\|\vec{V}_{agv}\| = K_v \cdot [\delta' \cdot Com \cdot Avoid_{vel} + (1 - \phi' \cdot Com) \cdot error] \tag{14}$$

where δ' and ϕ' are constants, which modify the interaction with Com function. Finally, K_v is the proportional value that changes the measurement value to velocity value.

4. AGV Kinematics Equations

In this case, a specific AGV was used, so it is necessary to specify the kinematics of that vehicle. In some cases, it is possible to use defined kinematics like Corke [19] proposed in his study. This special AGV has two motors in the front axle. The autonomous vehicle uses different speed values in each motor to steer. Both axles are connected mechanically by means of a pulley mechanism to reduce the steering radio.

The inputs of the kinematics block are the AGV velocity and direction. The output, however, is the vehicle POSE. The velocity reference (V_{agv}) is in the middle of the vehicle. When the AGV goes in a straight line, both motors (V_i and V_d) have the same velocity. However, if the AGV takes a curve, the motors will have different velocities.

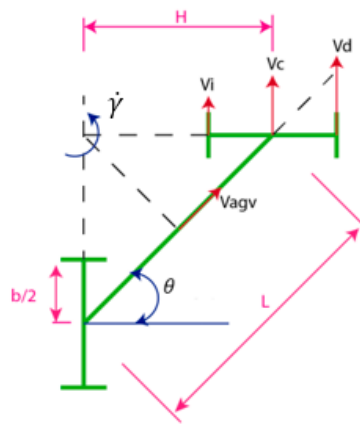


Figure 4. AGV dynamical behavior.

Hence, taking Figure 4 as a reference, wheel speed can be represented in Equations (15) and (16):

$$V_I = \dot{\gamma} \cdot (H - b/2) \tag{15}$$

$$V_D = \dot{\gamma} \cdot (H + b/2) \tag{16}$$

where H is the distance between rotation point and V_c , and b is the size of the wheel axle. Taking into account Equations (15) and (16), $\dot{\gamma}$ can be defined by motors velocities presented in Equation (17):

$$\dot{\gamma} = \frac{V_D - V_I}{b} \tag{17}$$

The V_{AGV} is split in \dot{x} and \dot{y} absolute coordinates.

$$V_x = \|\vec{V}_{agv}\| \cdot \cos\theta \tag{18}$$

$$V_y = \|\vec{V}_{agv}\| \cdot \sin\theta \tag{19}$$

V_C and V_{agv} are related each other, since the vehicle is considered as a solid element and the velocity propagates along all the structure, as illustrated in Figure 5.

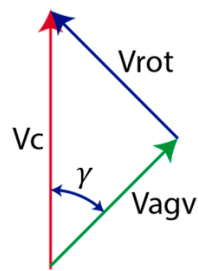


Figure 5. Velocities in the front axle.

$$\|\vec{V}_{agv}\| = V_c \cdot \cos\gamma \tag{20}$$

The velocity V_{rot} represented in Figure 5 turns the vehicle and generates the rotation in the middle of the AGV. The rotation of the vehicle can be defined with Equations (21) and (22):

$$\frac{\dot{\theta}L}{2} = V_{rot} \tag{21}$$

$$V_{rot} = V_c \cdot \sin\gamma \tag{22}$$

Combining (21) and (22), Equation (23) is obtained.

$$\dot{\theta} = \frac{2}{L} \cdot V_c \cdot \sin\gamma \tag{23}$$

Considering Equations (18–20) and (23), it is possible to define the vehicle dynamics in Equation (24):

$$\begin{vmatrix} \dot{V}_x \\ \dot{V}_y \\ \dot{\theta} \end{vmatrix} = \begin{vmatrix} \cos\gamma \cdot \cos\theta \\ \cos\gamma \cdot \sin\theta \\ \frac{2}{L} \cdot \sin\gamma \end{vmatrix} * V_c \tag{24}$$

where L is the wheelbase, θ is the AGV angle position in absolute coordinates, and γ is the value of the direction, which comes from the algorithm.

5. Implementation of Model

Once the dynamic block is ready, it is time to analyze this algorithm. Simulation software predicts what would happen in a “real” situation. The example scenario that comes with it has been modified and adapted to a more realistic surrounding.

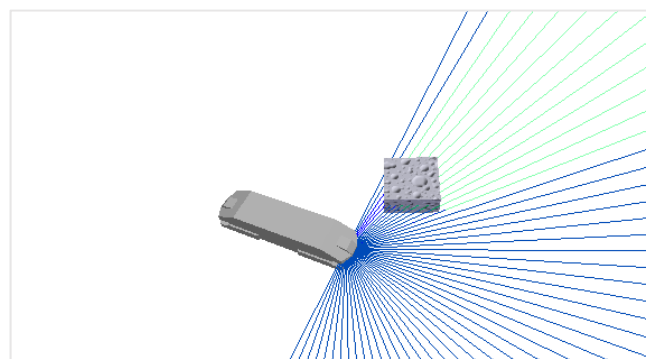


Figure 6. Scenario representation.

It is possible to add sensors to analyze the area in this program. In this case, a LIDAR sensor has been implemented to take measurements from the surroundings. This sensor has 51 lines distributed

from $-\pi/2$ rad to $\pi/2$ rad and there is a measurement for each line. Hence, the centre line of the LIDAR is located at 0 rad and it is the line 26. Figure 6 shows how the lines change the color from blue to green, when the LIDAR detects some object.

There are two important blocks to interact with that library in Figure 7. One is to represent the vehicle movement and the other one is to take measurements that form the area.

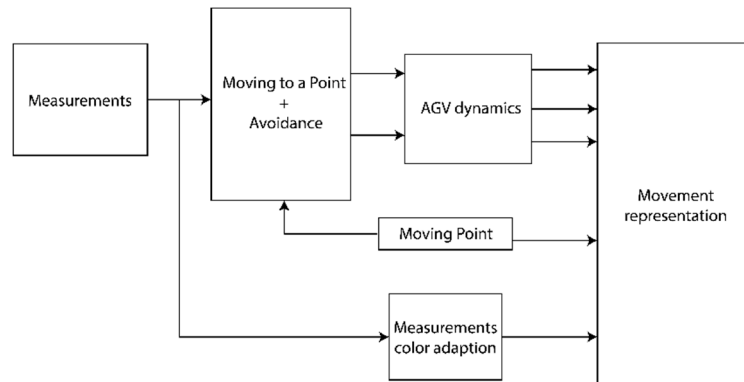


Figure 7. Block Diagram.

6. Results

Figure 8 shows the test area, which has three walls, two obstacles, and a moving point. This moving point is the target for the AGV and it is a non-static point.

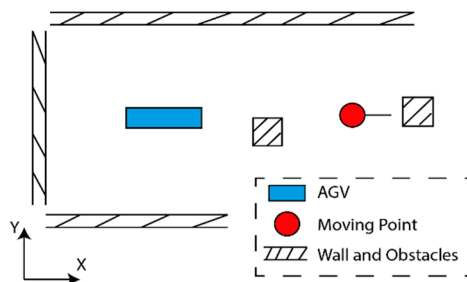


Figure 8. AGV Scenario.

Figure 9 shows the AGV trajectory after executing the simulation. It can be appreciated how the vehicle avoids the objects and selects where to turn depending on the obstacle localization.

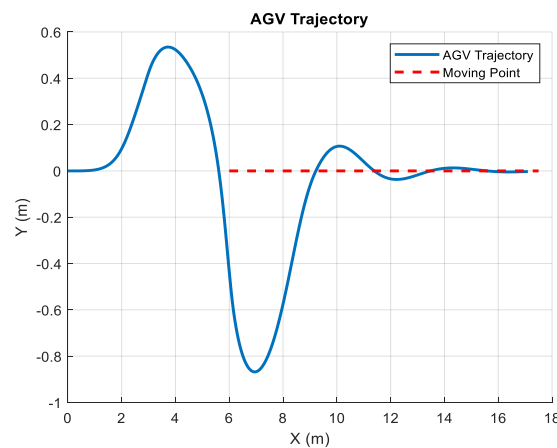


Figure 9. AGV Trajectory.

In addition, the AGV can follow a moving goal using the moving to a point algorithm. As expected, there is some error because this path following algorithm uses a proportional function. Therefore, some error occurs when it has a non-static goal.

Figure 10 shows the error between the moving point and the AGV. The error has been calculated as 0.4 m. This gap can be modified by adapting the K_v parameter.

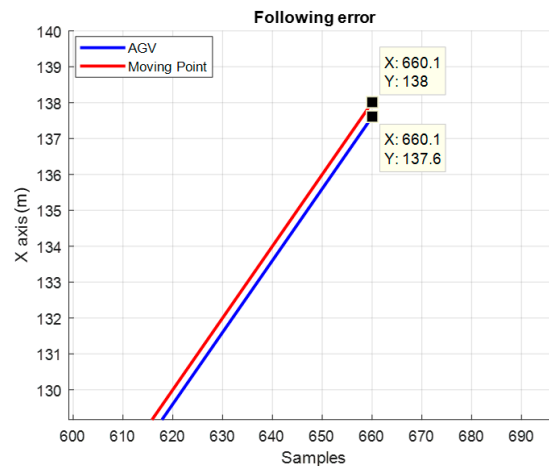


Figure 10. Following error.

After avoiding the obstacles, the system has some damping until it is stabilized. This damping can be modified adapting the K_h value. If the value of K_h increases, the AGV moves faster and can steer quicker. Figure 11 illustrates the trajectory of the AGV with different K_h values.

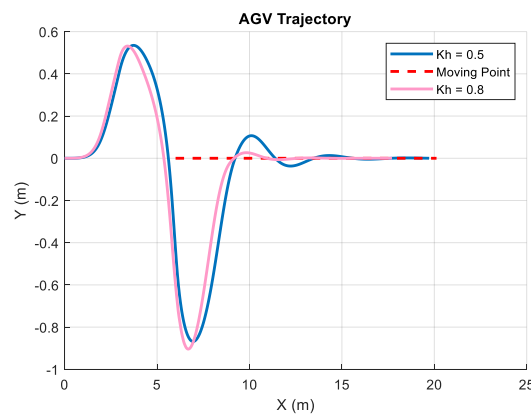


Figure 11. AGV trajectory with different K_h values.

Apart from modifying the direction of the vehicle, the velocity is reduced to be more precise in the avoidance maneuver. This speed variation is represented in Figure 12. The avoiding algorithm calculates a velocity correction (red line) and this correction is implemented on the moving to a point velocity term (blue Line). After avoiding the obstacle, the safety term stops actuating. In 20 sample times, the avoidance stops correcting the speed.

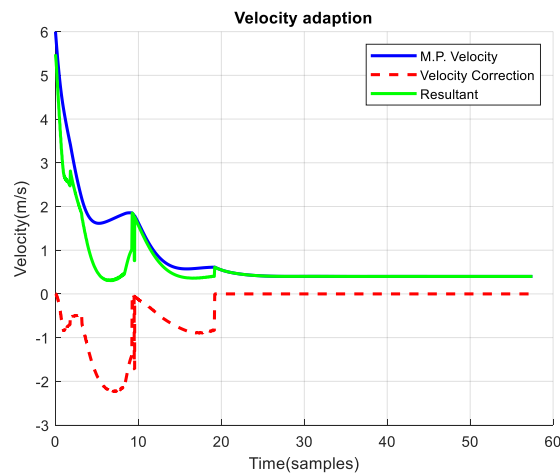


Figure 12. Velocity Adaption.

7. Dynamic Window Approach Comparison

The next step after testing the avoidance algorithm consists of comparing it with a classic motion planning solution. A well-known dynamic window approach (DWA) [13] algorithm is used. DWA is divided into two parts: The first part is search space, in which the different possible V_{agv} and $\dot{\gamma}$ are considered. This section is subdivided into three steps: circular trajectories, where it is calculated a two-dimensional velocity ($V_{agv}, \dot{\gamma}$) space, admissible velocities to restrict the admissible velocities ensuring safe trajectories, and dynamic window, which analyses only those velocities that can be reached on a short time interval considering robot acceleration limitations.

The second part is an optimization. The optimization is represented by Equation (25):

$$G(V_{agv}, \dot{\gamma}) = \sigma_D(\alpha_D \cdot heading(V_{agv}, \dot{\gamma}) + \beta_D \cdot dist(V_{agv}, \dot{\gamma}) + \gamma_D \cdot vel(V_{agv}, \dot{\gamma})) \quad (25)$$

Moreover, this second part is also subdivided into three steps: target heading, which is a measure of progress toward the goal, clearance, which is the distance to the closest obstacle, and velocity, which is considered the forward velocity of the robot.

Before starting the simulation of both algorithms, there are some aspects that can be appreciated without any execution. On the one hand, DWA uses an optimization term and normally these equations require a high computational capability. For the AGV that is used in our application, it is impossible to execute and process this optimization over a short period of time. That is the reason for the development of POA. Otherwise, with DWA it is possible to generate a proper trajectory, as it always selects the best path according to the space conditions. Furthermore, POA does not consider the vehicle kinematics, hence it makes a simpler calculation to avoid the obstacle. In other words, POA does not calculate a trajectory, it just modifies the path following command to change the direction and velocity when some obstacle appears in front of the vehicle. As a result, when some object is detected, the speed will be reduced to avoid it safely. DWA, however, always selects a trajectory considering the top speed value and minor acceleration loss.

On the other hand, in both cases adjustable gains are used to optimize the algorithms, making it more flexible to adapt to different surroundings or kinematics conditions. In addition, DWA can select which side turns the vehicle, in order to avoid the obstacle using the trajectory optimization. POA, such as the other algorithm, makes the same action. Otherwise, it uses Equation (12) to determinate which side turns the AGV.

In order to perform an easy comparison between both algorithms, a new simple scenario has been designed (see Figure 13). There are three elements in the scenario: an object, a moving obstacle, and the static goal position. With the moving obstacle, it is possible to analyze human–robot collaboration. This moving obstacle is considered to be a human and the AGV must avoid the collision.

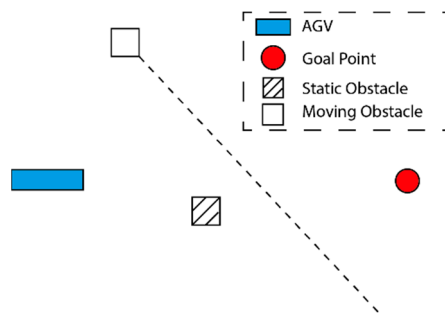


Figure 13. Comparison scenario.

Using Table 2 variables, the result of both algorithm executions is shown in Figure 14. Where the black * represents the static obstacle, the red * draws goal point, the pink * represents the moving obstacle, the pink dashed line shows moving obstacle trajectory, green lines represent DWA possible trajectories, and blue lines represent the algorithm trajectory.

Table 2. Variables and Constant.

Name	Description	Constant Value
K_μ (rad/m)	Proportional constant for steering	0.9
K_v (s ⁻¹)	Proportional constant for speed	0.9
$Lidar_{max\ range}$ (m)	Lidar maximum range	10
α (m)	Security distance	0.5
β	Intensity of the avoid function	0.5
μ	Intensity of the speed function	0.5
δ, φ, δ' and φ'	Constants for Sigmoid functions	$\delta = \delta' = 1;$ $\varphi = \varphi' = 0.5$
L (m)	Distance between axles	1.8
b (m)	Axle distance	0.2
$\sigma_D, \alpha_D, \beta_D$ and γ_D	DWA configurable constants	$\sigma_D = 0.1; \alpha_D = 0.2;$ $\beta_D = 0.1; \gamma_D = 3.0$

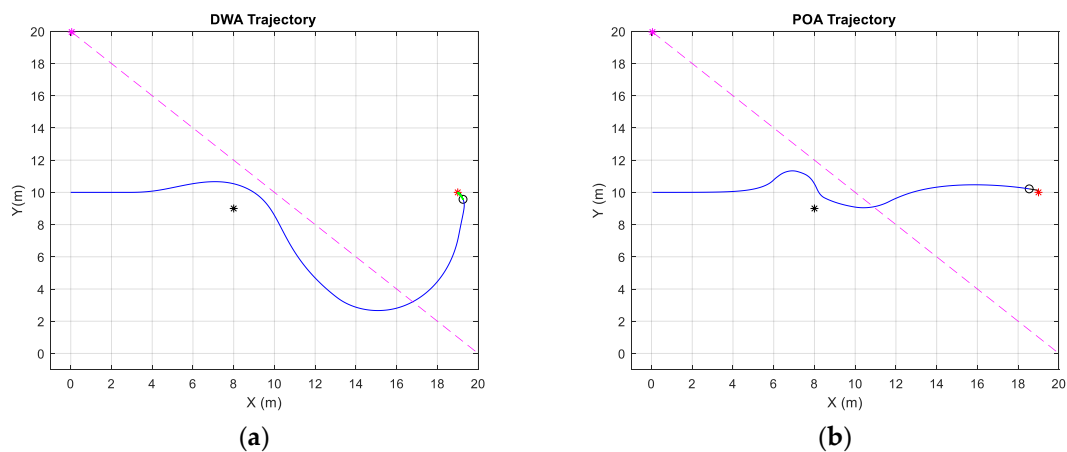


Figure 14. Both algorithm executions with the same conditions. (a) Dynamic window approach (DWA) trajectory execution and (b) proposed obstacle avoidance (POA) trajectory execution.

For this particular example, the DWA algorithm had more problems to avoid the moving obstacle than POA. DWA, with Table 3 optimization constant values, cannot fit an optimal trajectory to avoid the moving obstacle. To have a better idea of how the algorithms act with the mobile object, Figure 15 represents different time instants of the AGV trajectory. In the end, the algorithm does not

avoid the moving obstacle, until the obstacle is a particular distance away from the AGV. Hence, it would be a problem if the vehicle did not have enough D_{obs} .

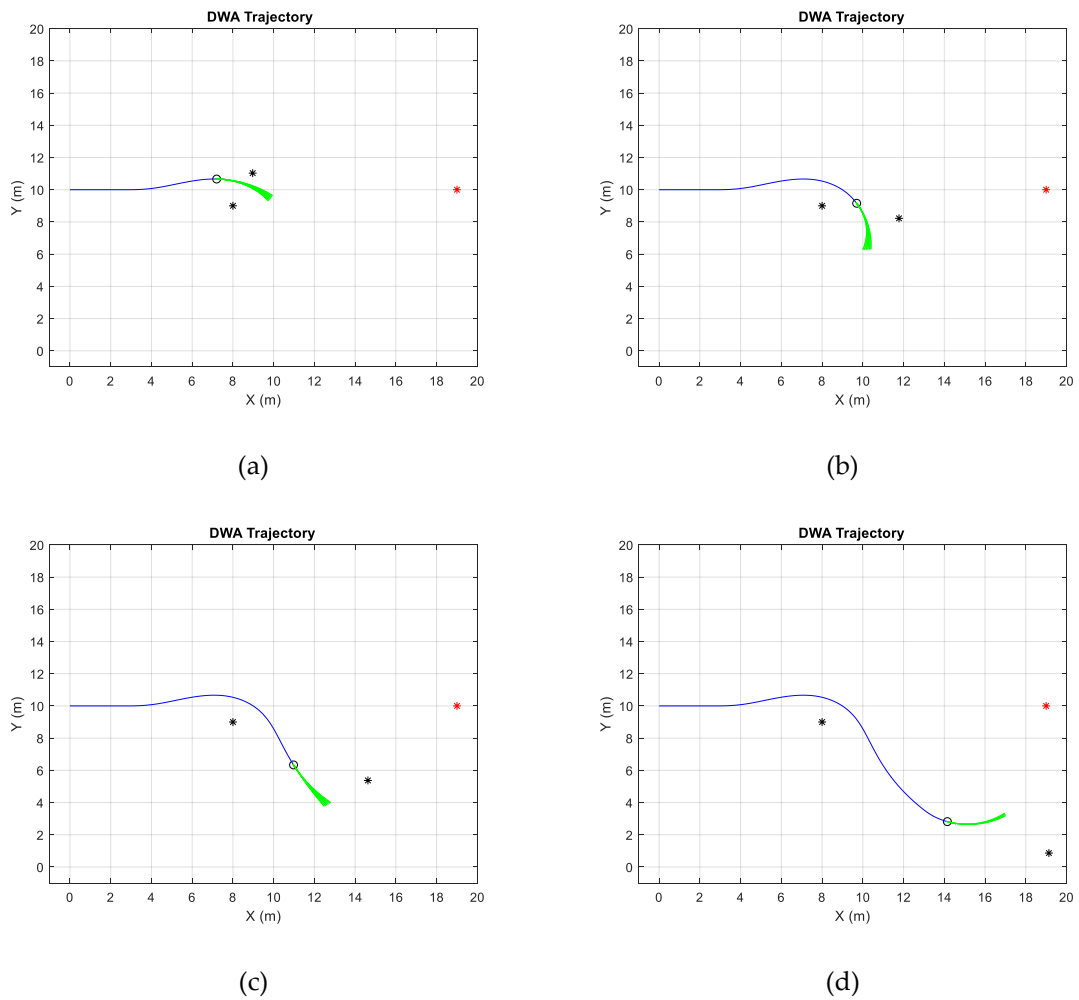


Figure 15. DWA different instant captured figures from the trajectory. (a) First time, (b) second time, (c) third time, and (d) fourth time.

Figure 16 shows the avoidance progress for POA. The POA algorithm tries to avoid the moving obstacle until the sign of Equation (12) changes. When that sign changes, the AGV continues to the goal without any problem.

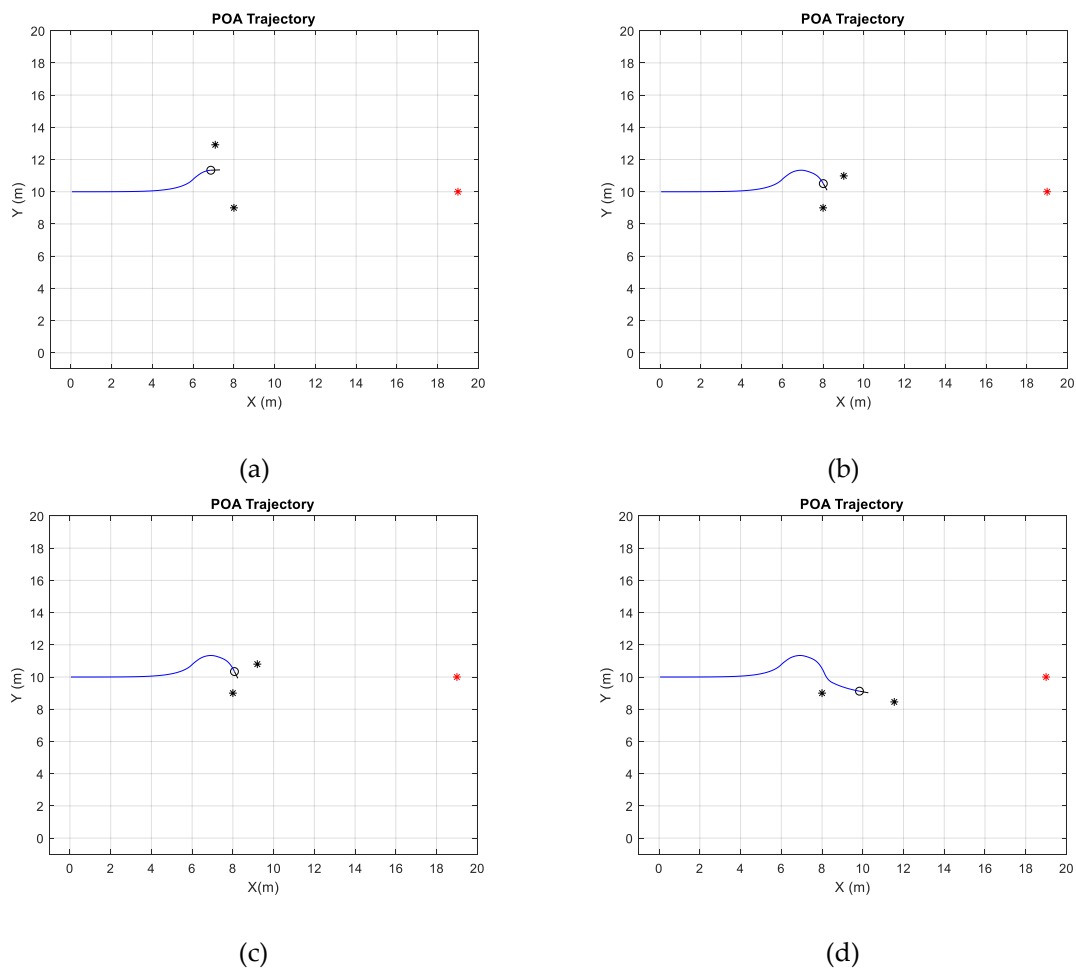


Figure 16. POA different instant captured figures from the trajectory. (a) first time, (b) second time, (c) third time, and (d) fourth time.

In order to stop simulating, new different starting points for the AGV are implemented. This makes it possible to analyze and compare what happens with the vehicle on different cases. Apart from this, DWA configurable constants are changed to reduce the speed window size. $\gamma_D = 2.0$ is used, in order to reduce the effect that is shown in Figure 15, in which the AGV needed more D_{obs} to avoid the obstacle.

Table 3. Analytics Results.

Test	Start Pose (m)	Algorithm	Odometer (m)	min D_{obs} (m)	Toal Time (s)	Mean Speed (m/s)	Speed Arriving at Goal (m/s)
1	(0,0,0)	POA	43.85	0.80	47.55	0.96	0.36
		DWA	36.42	2.00	35.45	0.92	0.81
2	(0,1,0)	POA	41.45	0.58	45.65	0.94	0.36
		DWA	36.43	2.00	35.5	0.91	0.8
3	(0,2,0)	POA	25.29	0.34	22.95	0.93	0.36
		DWA	36.34	2.00	35.4	0.91	0.8
4	(0,3,0)	POA	24.85	0.52	22.45	0.93	0.36
		DWA	36.21	2.00	35.25	0.91	0.81
5	(0,4,0)	POA	24.50	0.67	22.15	0.93	0.36
		DWA	36.08	1.94	35.1	0.91	0.81
6	(0,5,0)	POA	24.30	0.77	21.9	0.93	0.36
		DWA	36.41	1.84	35.45	0.91	0.81

Table 3. Cont.

Test	Start Pose (m)	Algorithm	Odometer (m)	min D_{obs} (m)	Total Time (s)	Mean Speed (m/s)	Speed Arriving at Goal (m/s)
7	(0,6,0)	POA	24.18	0.79	21.8	0.92	0.36
		DWA	36.88	1.78	35.9	0.91	0.82
8	(0,7,0)	POA	24.16	0.71	21.85	0.93	0.36
		DWA	36.94	1.86	35.95	0.91	0.83
9	(0,8,0)	POA	24.29	0.50	22.05	0.93	0.36
		DWA	24.21	0.54	39.65	0.51	0.82
10	(0,9,0)	POA	24.65	0.50	22.6	0.93	0.35
		DWA	24.05	0.71	32.45	0.62	0.81
11	(0,10,0)	POA	24.91	0.96	24.5	0.93	0.33
		DWA	28.67	1.2	27.6	0.89	0.81
12	(0,11,0)	POA	23.74	0.5	21.4	0.93	0.35
		DWA	28.60	1.14	27.55	0.9	0.8
13	(0,12,0)	POA	23.62	0.67	21.2	0.93	0.36
		DWA	28.37	1.04	27.35	0.89	0.79
14	(0,13,0)	POA	23.74	0.57	21.30	0.93	0.36
		DWA	28.17	0.90	27.1	0.89	0.8
15	(0,14,0)	POA	24.56	0.5	24.15	0.85	0.23
		DWA	24.63	1.6	24.15	0.85	0.81
16	(0,15,0)	POA	25.35	0.5	24.20	0.8	0.16
		DWA	25.17	1.82	24.6	0.86	0.81
17	(0,16,0)	POA	25.94	0.5	24.4	0.93	0.33
		DWA	25.66	1.97	24.85	0.87	0.82
18	(0,17,0)	POA	26.54	0.5	25.55	0.93	0.36
		DWA	26.18	1.9	25.05	0.88	0.82
19	(0,18,0)	POA	27.75	0.5	27.25	0.93	0.36
		DWA	26.38	1.33	25.25	0.88	0.82
20	(0,19,0)	POA	27.68	0.75	26.7	0.93	0.36
		DWA	25.51	0.69	25.3	0.85	0.82

In Table 3 different simulation results are shown, and it is necessary to consider that speed arriving at goal is the speed value at 1 s before arriving to the goal. Additionally, a static object is implemented, which is an $x = 8$ and $y = 9$ position, and a moving point, which starts in $x = 0$ and $y = 20$ position and ends in $x = 40$ and $y = -20$ position with 0.92 m/s constant speed. In addition, the goal is located in an $x = 20$ $y = 20$ position. All of these parameters also are used in the Figure 14 simulation.

Looking at Table 3, in general, DWA needs more space to avoid the obstacles with this configuration. That is why the value of D_{obs} is larger than POA. Hence, the odometry in DWA is going to be higher, because both parameters are proportional.

If total time is compared, however, the situation changes. POA uses more time to arrive at the goal position. When AGV is approximating the goal, it will reduce the speed more dramatically, as represented by the speed arriving at goal column. In the end, the difference between both approximation speeds is around 0.46 m/s. Otherwise, POA has the best mean speed and the difference between both mean speeds is about 0.07 m/s, which is negligible.

To give a general idea about algorithms failure, Figure 17, which represents the worst trajectory of POA in comparison with DWA, and Figure 18, which analyzes the worst trajectory of DWA in comparison with POA, are shown. To consider which is the worst trajectory, the largest odometry was analyzed, due to AGV diverting more from the ideal path, which is a straight line from the starting point to the goal.

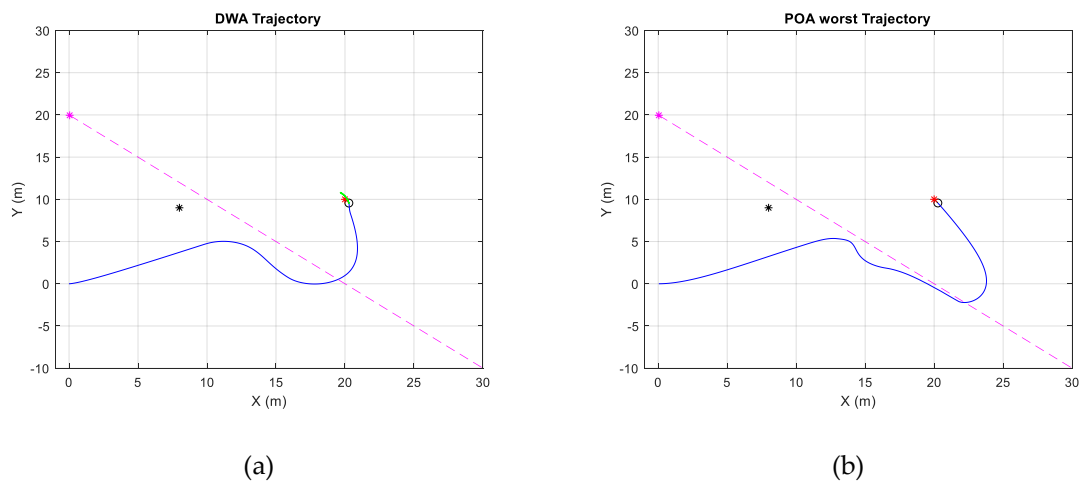


Figure 17. POA worst case: (a) DWA trajectory for POA worst case, and (b) POA worst trajectory.

In Figure 17, POA had a greater difficulty deciding where to avoid the moving obstacle. AGV tried to follow it until the obstacle was so far away that the vehicle stopped detecting it. A similar occurrence is observed in Figure 18. Instead of POA failing to avoid the obstacle, this time DWA failed. The dynamic window did not have enough space to pass the obstacle and it waited until having enough distance to avoid it. In both cases, the algorithms have a bad reaction with the moving obstacle, hence for future work it could be possible to analyze the space and detect moving obstacles. In case that some moving obstacle appears, the AGV stops and wait until the obstacle disappears.

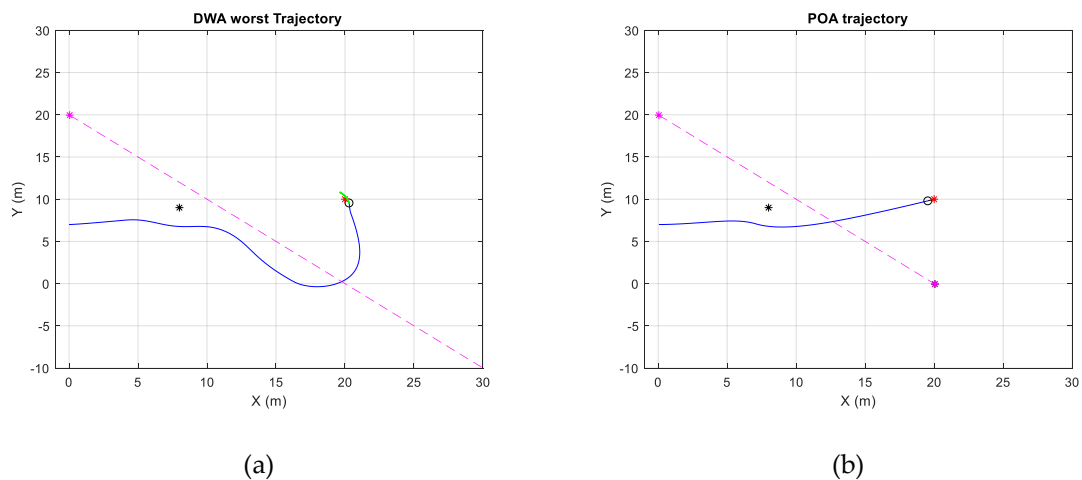


Figure 18. DWA worst case: (a) DWA worst trajectory and (b) POA trajectory for DWA worst case.

Overall, both algorithms work in the same conditions, though depending on the algorithm configuration the answer of the simulation can change. There are some cases in which the DWA trajectories are worst, and other times the POA's paths are not good enough. Generally, DWA approximates the goal with a higher speed and, depending on the application, this speed must be controlled to safely arrive. POA, however, gives less space between the AGV and object. In the end, this could be a problem if the AGV moves around humans.

8. Conclusions

The solution proposed in this paper can highly improve a simple path following algorithm behavior. It gives more intelligence to the AGV, a result of the obstacle avoidance parameters. Moreover, it introduces more flexibility to the algorithm, because there are some variables which change the

performance. Additionally, POA uses a simple equation to implement on a PLC, which is the only hardware available on the selected AGV. This hardware limitation is an important feature, because not all equipment supports large calculation capabilities, such as DWA, which uses an optimization function to select the best trajectory.

The optimization function gives the opportunity to analyze some trajectories and then select the best one. However, POA just modifies the path following trajectory depending on the gap between the AGV and obstacle. In other words, the algorithm does not optimize the trajectory.

In addition, POA has a good performance compared with traditional DWA algorithms. Both have similar behavior in avoiding the static obstacle. Furthermore, both can select which side turns the vehicle, in order to avoid the obstacle. DWA, however, tries to divert more from the obstacle, hence the AGV needs more space to avoid the obstacle. To solve this divergent, the algorithm always tries to go at maximum speed and it arrives faster than POA.

Apart from that, there is a considerable arriving speed value difference between both algorithms. For this application, the AGV must be able to cooperate with humans. Hence it would be safer for the AGV to arrive to the goal with less speed. Otherwise humans may interpret the fast approach as harmful. There is another similar situation, in which the vehicle is avoiding the human and it maintains a gap during the maneuver as an avoidance strategy.

It is remarkable that both algorithms can avoid a moving obstacle. Otherwise, they would need to be improved to avoid these kinds of obstacles. In future work, they could be improved using dynamic parameter values to change algorithm behavior depending on the situation. In some cases, it would be better to reduce AGV speed and wait until the obstacle disappears, while in other ones it would be possible to increase the speed to reduce commute time. In other situations, it could be possible to maintain a gap with the obstacle, for example, when the AGV has to avoid a box which is in its way. Moreover, it is important to control how much the AGV diverts from the obstacle, depending on the scenario this feature must be controlled. In industrial areas, for example, there is not much space to avoid the obstacle.

Author Contributions: D.T.-F.-B., E.Z. and U.F.-G. developed and programmed the simulation set up. They also wrote the manuscript. I.A. and I.U. made constructive contributions in the process of preparing the paper.

Acknowledgments: The funding from the Government of the Basque Country and the University of the Basque Country UPV/EHU through the SAIOTEK (S-PE11UN112) and EHU12/26 research programs, respectively, is gratefully acknowledged.

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

References

1. Pereira, A.C.; Romero, F. A review of the meanings and the implications of the Industry 4.0 concept. *Procedia Manuf.* **2017**, *13*, 1206–1214. [[CrossRef](#)]
2. Zhong, R.Y.; Xu, X.; Klotz, E.; Newman, S.T. Intelligent Manufacturing in the Context of Industry 4.0: A Review. *Engineering* **2017**, *3*, 616–630. [[CrossRef](#)]
3. Djuric, A.M.; Urbanic, R.J.; Rickli, J.L. A Framework for Collaborative Robot (CoBot) Integration in Advanced Manufacturing Systems. *SAE Int. J. Mater. Manuf.* **2016**, *9*, 457–464. [[CrossRef](#)]
4. Cherubini, A.; Passama, R.; Crosnier, A.; Lasnier, A.; Fraisse, P. Collaborative manufacturing with physical human–robot interaction. *Robot. Comput.-Integr. Manuf.* **2016**, *40*, 1–13. [[CrossRef](#)]
5. Cardarelli, E.; Digani, V.; Sabattini, L.; Secchi, C.; Fantuzzi, C. Cooperative cloud robotics architecture for the coordination of multi-AGV systems in industrial warehouses. *Mechatronics* **2017**, *45*, 1–13. [[CrossRef](#)]
6. Mustafa, M.; Stancu, A.; Delanoue, N.; Codres, E. Guaranteed SLAM—An interval approach. *Robot. Auton. Syst.* **2018**, *100*, 160–170. [[CrossRef](#)]
7. Nakajima, K.; Premachandra, C.; Kato, K. 3D environment mapping and self-position estimation by a small flying robot mounted with a movable ultrasonic range sensor. *J. Electr. Syst. Inf. Technol.* **2017**, *4*, 289–298. [[CrossRef](#)]
8. Elsheikh, E.A.; El-Bardini, M.A.; Fkirin, M.A. Practical path planning and path following for a non-holonomic mobile robot based on visual servoing. In Proceedings of the 2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference, Chongqing, China, 20–22 May 2016; pp. 401–406.

9. Bonin-Font, F.; Ortiz, A.; Oliver, G. Visual Navigation for Mobile Robots: A Survey. *J. Intell. Robot. Syst.* **2008**, *53*, 263–296. [[CrossRef](#)]
10. Ever, Y.K. Using simplified swarm optimization on path planning for intelligent mobile robot. *Procedia Comput. Sci.* **2017**, *120*, 83–90. [[CrossRef](#)]
11. Hu, X.; Chen, L.; Tang, B.; Cao, D.; He, H. Dynamic path planning for autonomous driving on various roads with avoidance of static and moving obstacles. *Mech. Syst. Signal Process.* **2018**, *100*, 482–500. [[CrossRef](#)]
12. Mei, Z.; Chen, Y.; Jiang, M.; Wu, H.; Cheng, L. Mobile Robots Path Planning Based on Dynamic Movement Primitives Library. In Proceedings of the 2017 36th Chinese Control Conference (CCC), Dalian, China, 26–28 July 2017. [[CrossRef](#)]
13. Fox, D.; Burgard, W.; Thrun, S. The dynamic window approach to collision avoidance. *IEEE Robot. Autom. Mag.* **1997**, *4*, 23–33. [[CrossRef](#)]
14. Borenstein, J.; Koren, Y. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Trans. Robot. Autom.* **1991**, *7*, 278–288. [[CrossRef](#)]
15. Buniyamin, N.; Ngah, W.A.J.W.; Mohamad, Z. PointsBug versus TangentBug algorithm, a performance comparison in unknown static environment. In Proceedings of the 2014 IEEE Sensors Applications Symposium (SAS), Queenstown, New Zealand, 17 April 2014; pp. 278–282.
16. Adiyatov, O.; Varol, H.A. Rapidly-exploring random tree based memory efficient motion planning. In Proceedings of the 2013 IEEE International Conference on Mechatronics and Automation, Takamatsu, Japan, 4–7 August 2013; pp. 354–359.
17. Aravindan, A.; Zaheer, S.; Gulrez, T. An integrated approach for path planning and control for autonomous mobile robots. In Proceedings of the 2016 International Conference on Next Generation Intelligent Systems (ICNGIS), Kottayam, India, 1–3 September 2016; pp. 1–6.
18. Wang, W.J.; Hsu, T.M.; Wu, T.S. The improved pure pursuit algorithm for autonomous driving advanced system. In Proceedings of the 2017 IEEE 10th International Workshop on Computational Intelligence and Applications (IWCIA), Hiroshima, Japan, 11–12 November 2017; pp. 33–38.
19. Corke, P. *Robotics, Vision and Control: Fundamental Algorithms In MATLAB® Second, Completely Revised, Extended and Updated Edition*, 2nd ed.; Springer International Publishing AG: Gewerbestrasse, Switzerland, 2011.
20. Heller, C.; Yaesh, I. Proportional Navigation with integral action. In Proceedings of the Melecon 2010—2010 15th IEEE Mediterranean Electrotechnical Conference, Valletta, Malta, 26–28 April 2010; pp. 1546–1550.
21. Sebi, S.A.; Sunny, D. Obstacle Avoidance in Mobile Robotic Sensors and Establishing Connection. *Procedia Technol.* **2016**, *25*, 364–371. [[CrossRef](#)]
22. Peng, Y.; Qu, D.; Zhong, Y.; Xie, S.; Luo, J.; Gu, J. The obstacle detection and obstacle avoidance algorithm based on 2-D lidar. In Proceedings of the 2015 IEEE International Conference on Information and Automation, Lijiang, China, 8–10 August 2015; pp. 1648–1653.
23. Catapang, A.N.; Ramos, M. Obstacle detection using a 2D LIDAR system for an Autonomous Vehicle. In Proceedings of the 2016 6th IEEE International Conference on Control System, Computing and Engineering (ICCSCE), Batu Ferringhi, Malaysia, 25–27 November 2016; pp. 441–445.
24. Lee, D.; Lu, Y.; Kang, T.; Choi, I.; Lim, M. 3D vision based local obstacle avoidance method for humanoid robot. In Proceedings of the 2012 12th International Conference on Control, Automation and Systems, Jeju Island, Korea, 17–21 October 2012; pp. 473–475.
25. Amin, N.; Borschbach, M. Quality of obstacle distance measurement using Ultrasonic sensor and precision of two Computer Vision-based obstacle detection approaches. In Proceedings of the 2015 International Conference on Smart Sensors and Systems (IC-SSS), Bangalore, India, 21–23 December 2015; pp. 1–6.
26. Martínez, M.; Martínez, J.; Morales, J. Motion Detection from Mobile Robots with Fuzzy Threshold Selection in Consecutive 2D Laser Scans. *Electronics* **2015**, *4*, 82–93. [[CrossRef](#)]
27. Wooten, J.; Bevely, D.; Hung, J. Piezoelectric Polymer-Based Collision Detection Sensor for Robotic Applications. *Electronics* **2015**, *4*, 204–220. [[CrossRef](#)]

