DOI: 10.21062/mft.2021.027 © 2021 Manufacturing Technology. All rights reserved. http://www.journalmt.com

Path and Trajectory Planning for an Automated Carrier Vehicle Equipped with Two Conveyor Belts used in Manufacturing Supply

Ákos Cservenák
Faculty of Mechanical Engineering and Informatics, University of Miskolc in 3515 Miskolc-Egyetemváros, Hungary, E-mail: cservenak.akos@uni-miskolc.hu

The AGVs or mobile robots are well used in today's manufacturing supply technologies and also can be used in engineering's education. The motion controlling and simulation of such vehicles are a crucial question. This paper introduces the steps of motion planning for a driverless carrier vehicle from the positions initially available to the speed of the wheels. The vehicle is located in the High-Tech Logistics Systems Laboratory of the Logistics Institute of the University of Miskolc. For motion controlling and simulation between two points the further modules are necessary: 1. path planner, 2. trajectory planner, 3. velocity-voltage converter using velocities gained from trajectory planner, 4. motion controlling and simulation of a motor dynamical model using voltages from the converter, 5. simulation of the path and 6. data processing. In this paper the first two modules are detailed, i.e. the path planning and then the trajectory planning. Path planning is based on a new approach, using Bezier-curves and Hermite curves. The trajectory planning tends to the mininum energy, which can be carried out by the examining the current consumption created in the other modules. The smaller consumption originated from the two curves determines the final path and trajectory.

Keywords: AGV, Manufacturing supply, Path planning, Trajectory planning, Mechatronics

1 Introduction

Today, the so-called Industry 4.0 has accelerated the spread of automation and mobile devices [1] and this phanomena is also true for the field of manufacturing technology and supply [2]. The design of a logistic process should always aim to modernise itself [3]-[4]. Robotics increases its role in the automation [5], both in the field of industrial robots and mobile robots. There is also a need for continuous improvements in the field of industrial robots, for instance designing a force feedback-based installation process [6] In the case of an industrial robot, its transport may also be a challenge [7].

A prototype AGV can be found in the High-Tech Logistics Systems Laboratory of the Logistics Institute of the University of Miskolc [8]. This AGV was no longer able to function automatically after the laboratory moved, as motion control was established at its previous location [51]. Therefore, the need for renewing the motion controlling created the topic of this paper, which introduces a new path and trajectory planning solution that is not room-bound but can be reused after transporting the AGV.

For motion controlling and simulation between two points the further modules are necessary: 1. path planner, 2. trajectory planner, 3. velocity-voltage converter using velocities gained from trajectory planner, 4. motion controlling and simulation of a motor dynamical model using voltages from the converter, 5. simulation of the path and 6. data processing. The module 3., 5. and 6. was described in [9], while implementation of the module 4. was written in [10].

This paper describes the steps of an algorithm that can use initially defined positions to produce the velocities of the wheels, i.e. implement module 1. and 2. The velocities gained on the wheels are the input for module 3., so control can already be achieved based on the modules that have already been completed.

Section 2. starts with the literature background on remaining path and trajectory planning techniques. Section 3. summarises the interpolation and approximation solutions, highlighting the Bezier and Hermite curves. The solution of path planning is shown in Section 4., and Section 5. supplements it with the necessary conditions in order to make the control and simulation work correctly. Then, Section 6. details the production of data on the trajectory (module 2.) generated on the basis of path data (module 1.). Finally, Section 7. writes concluding remarks.

2 Literature review in the topic of path and trajectory planning

Extensive research work has been carried out on the topic of robot path and trajectory planning which also appears in the literature.

Firstly, the [11] literature can be mentioned, in which the chapter "Path Planning and Trajectory Planning Algorithms: General Overview" is relevant to the topic. The mentioned chapter deals with the algorithms of a path and trajectory planning. A book chapter was also born at the University of Miskolc from the topic [12].

The [11] literature divides the field of path planning and trajectory planning into two separate groups. In terms of algorithm creating, not only the above category but also offline or online planning is possible [13].

The path planning can be divided into 3 techniques as follows [11]:

- roadmap techniques,
- cell decomposition techniques,
- artifical potential techniques.

The trajectory planning problem generates reference inputs for the robot controller, ensuring that the desired motion is achieved. Typically, the inputs are the path generated by the path planner and the dynamic and kinematic constraints of the robot. The most important optimization criterias for trajectory planning are as follows [11], [22]:

- minimum execution time,
- minimum energy or actuator effort,
- minimum jerk.

One of the above parameters can be considered, or more in the case of hybrid optimization criteria, such as time-energy optimal trajectory planning.

The subsections 2.1, 2.2 and 2.3 and the subsections 2.4, 2.5 and 2.6 present techniques and aspects used in path and trajectory planning, respectively. The subsection 2.7 presents the trajectory planning solution previously implemented on the examined AGV.

2.1 Path Planning: Roadmap Techniques

This technique reduces an n-dimensional configuration space to a one-dimensional path for the search, preferably in a graph. In this space, the [11] literature divides one "C-free" into free and one "C-obs" into non-free or collision spaces, the two spaces together forming a "C-space" configuration space.

Such a technique can be found in the literature [14], where a so-called visibility graph considers the endpoints of each collision space as nodes and connects them to a line. For example, a search for the shortest Euclidean route in the configuration space is already possible on these lines.

Another approach is to use Voronoi diagrams. Here, the obstacles are displayed in the form of a square, and the planned paths are created to be equidistant from at least two obstacles. An example of this approach is shown in [15].

2.2 Path Planning: cell decomposition techniques

This second known method divides the "C-free" free space into several regions, namely cells, and any two connections in the path are located between two adjacent cells. The so-called connection graph forms connections between adjacent cells. Then the path search problem turns into a graph search problem.

The dividing up into cells can be further divided into two groups:

- dividing into exact cells
- dividing into approximate cells

In the first case, it divides the free space into polygons [11], [16]. The set of polygons fills the free space exactly. In the second case, it divides the space using triangles [17], squares, or rectangles [11]. Since the totality of these plane shapes does not fill the space perfectly, the accuracy of the dividing is only approximate.

2.3 Path planning: artifical potential techniques

This technique takes a different approach to the problem of route search. The basic idea assumes that the robot is an object moving in configuration space in a potential area generated by obstacles in the target configuration and the entire "C-space", namely, the target configuration creates an attractive (attractive, favorable, seductive) potential, while barriers to a repulsive (repulsive, disgusting) potential. The combination of the two is the full potential that is visible to the robot by an artificial force that leads to the target avoiding obstacles. This technique can also be applied to a mobile robot [18].

However, this technique has a major problem: the occurrence of local minima where the robot finds itself trapped. Several solutions have been developed for this, such as the use of potential functions that do not have a local minimum. These functions are the navigation functions.

Another method for this technique is to use the RPP (Random Path Planners) [11], [20], which avoids local minima by combining the concept of artificial potential areas with random search techniques. Similar method that can achieve significant results for such problems is the PRM (Probabilistic Roadmap Planners) planner [11], [21], which uses probability-based algorithms such as random sampling.

2.4 Trajectory planning: Minimum execution time

If optimization is required, the first option that arises is to reduce the execution or cycle time. In today's automated production systems, lead time is key to high productivity. For this reason, a number of literature deals with trajectory planning based on execution time.

One possibility is to define an algorithm in a position-velocity phase plane [23], [24]. Here, the basic idea is to write the dynamic equation of the manipulator in parametric form, the abscissa of the curve, where is the trajectory as an independent parameters.

Another approach involves the use of dynamic programming techniques [25]. The basic idea is to discretize the state space on a grid of points, then it assigns velocity, acceleration, jerk limits to each point and defines the time required to move, taking into account the cost of each solution. Finally, an algorithm based on dynamic programming generates the minimum time trajectory.

The disadvantage of the above methods is that the algorithms do not result in continuous accelerations and joint moments, since the dynamic model takes into account a perfectly rigid body, while it does not take into account the actuator dynamics. In reality, only continuous accelerations and torques can occur, so this occurs with a delay and reduces the accuracy of trajectory tracking. For this reason, the track controller often has to intervene in the execution of the trajectory.

Several solutions can be found in the literature. One solution suggests using the phase plane method in conjunction with torque constraints[26], another solution uses not only the execution time but also the energy contribution in the objective function, such as integrating the square moments along the entire trajectory[27].

It is also possible to use spline interpolations, where the trajectory is divided into a finite number of sections and the joints of the sections are rounded or smoothed. A number of techniques are used in the literature, the differences are as follows:

- the constraints considered, either kinematic [28] or dynamic [29],
- the algorithm used to calculate the optimal trajectory [30],
- the possibility of extending the optimization problem, taking into account the optimization criteria [31].

2.5 Trajectory planning: Minimum energy

In some cases, instead of tending for the minimum execution time, the focus should be on the minimum energy consumption or actuator effort. Energy-based trajectory planning can be crucial in several cases. One case is that it generates smooth trajectories, thereby reducing the effort on actuators and mechanical structure. The other case is where it is not only necessary for economic reasons to reduce energy consumption, but there may also be many applications where energy use is limited for technical reasons, such as underwater research [32] or mobile robots, where the power supply is predominantly provided by batteries [33]. In this case, in the event of an extension of the operating time, the aim must be to use as little energy as possible.

Several literature deals with this optimization criterion, such as [34] and [35], where the trajectory is parameterized using cubic B-splines and the physical limits of the joints have been added to the torque and kinematic limits. Also, the objective function presented here primarily seeks to minimize energy regardless of execution time.

For a Three-Wheeled Omnidirectional Mobile Robot (TOMR), the Pontryagin minimum principle was applied to the trajectory planning algorithm, battery power consumption to calculate the cost function, and a dynamic model including actuator dynamics and Coriolis force [33],[36],[37].

Trajectory planning should also be regarded by a single-drive robot in which an electric motor drives the vehicle through a mechanism [38]. The Fuzzy and Newton-barrier method was used to plan a two-arm mobile robot trajectory developed for the study of a high-voltage transmission line [39].

A Wheeled Mobile Robot (WMR) was an indirect solution to the optimal control strategy for optimal trajectory planning, taking into account the nonlinear dynamic model of the system and the nonholonom constraints [40].

A [41] in the literature, cost-index-optimized motion control has been implemented for a Swedish-type mobile robot equipped with driven and steered wheels, taking into account the singularity of direct and indirect models, ignoring sliding motions, infinite estimation error and impossible control interventions.

A rapid cost-effective motion planning for a humanoid robot operating in a complex and realistic environment is presented in [42], where the RRT (Rapidly exploring Random Tree) sampling-based algorithm were used. The hexapod robot presented in [43] uses a combined index based on power consumption and workspace, applying unequal constraints to the mathematical model of nonlinear programming. For optimal route planning, the application of the Keymeulen / Decuyper fluid method is described in [44] for an nonholonomic vehicle.

A mobile robot designed to examine POIs (Points of Interest) takes into account not only the monitoring of energy consumption but also the security and cost of communication for operation. This problem can be raised as a Mixed Integer Linear Program (MILP) [45]. Selecting the correct gait for a two-legged robot can reduce energy consumption using a genetic algorithm based on a gait synthesis method [46].

The search for the velocity profile and route resulting from the trajectory planning is energy-optimized for a car-like robot using DC motors [47]. A new optimal motion planning for a mobile robot with a differential drive is presented in [48], in which the model can be used to formulate the energy consumption of kinetic energy conversion and to overcome slip resistances. The development and analytical studies of a prototype of a spherical rolling robot are described in

[49]. Here, different methods have also been developed for the minimum-time and minimum-energy-based trajectory of the robot.

2.6 Trajectory planning: Minimum jerk

This optimization criterion is mainly the characteristic of industrial robots, in the case of mobile robots the formerly presented criterions are used. This technique is used to avoid discontinuities in actuator torques. The algorithm described in [50] is based on interval search. The technique presented here looks for the minimum of the maximum absolute value of the jerk along a path where execution time gives a priority. The primitives of the trajectories are cubic splines, and the intervals between the intermediate points and the lowest maximum jerk value are also calculated. This literature presents a comparison with the method based on trigonometric splines, covering the highest values of jerk, torques, and torque variations.

2.7 Trajectory planning previously performed on the examined AGV

A Fuzzy Logic-based controlling has already been developed at the AGV in the High-Tech Laboratory of the Institute of Logistics at the University of Miskolc [51]. The AGV uses a Sick brand NAV350 LIDAR sensor. This sensor determines the position and orientation of the vehicle and creates a contour of the room. The contour of the room can be saved in a 2D CAD model and further modifications can be performed with an appropriate software. Thus, the permitted track, arrival directions and stations of the truck have been included in the model. The route consists of straight and curved sections, the curved sections have been replaced by a breakdown into straight sections.

To control automated movement, the authors took the following steps:

1 Route planning
  1.1 Scanning the contour of the room with the LIDAR sensor
  1.2 Design and export the allowed path in AutoCAD software
2 Processing the exported AutoCAD model and defining sections and points
3 Calculating the route to the destination
  3.1 Finding the section closest to the robot using vectors
  3.2 Dividing the nearest section into two parts
  3.3 Determining the shortest path using the A* graph search algorithm
4 Calculation of the robot displacement, since the low measurement frequency of the LIDAR sensor is not a sufficient source of information for the controller, the position is also determined based on the kinematics of the robot
5 Calculation of Fuzzy controller inputs (target distance and target angle)
  5.1 Defining membership functions for the Fuzzy controller using the rule for two inputs and two outputs (average angular velocity and angular velocity difference)
  5.2 Create a set of fuzzy rules with 6 rules to control the direction and velocity of the robot

3 Interpolation, approximation, and path planning solutions

The subsection 3.1 describes the general interpolation and approximation solutions found in the literature, the subsections 3.2 and 3.3 highlight the solutions in which interpolate the start and end points, but only approximate the other points.

3.1 General interpolation and approximation solutions

The Table 1 lists interpolation and approximation solutions found in the [52]-[65] literatures. Each solution is typically used for edge smoothing in 3D CAD modeling.

Four points ("P" _"0" "," "P" _"1" "," "P" _"2" " and " "P" _"3") are necessary for a cubic approximation. Not all listed solutions are good for path planning, only where it interpolates the start and end points and approximates the other points, as follows:

$$P'_0=P_0,\ P'_1{\sim}P1,\ P'_2{\sim}P_2 \text{ and } P'_3=P_3 \qquad (1)$$

where P'_0, P'_1, P'_2 and P'_3 are the generated points from the interpolation technique.

Reviewing the mentioned literature, it can be concluded that the above condition will be true only for the Bezier curve. However, in the case of the Hermite interpolation curve, the control vectors starting from the starting point and arriving at the end point will also make this solution suitable for path control, since the control vectors can be calculated from the rotation angles at each point. In the other solutions, either the interpolated curve passes through all the points, which generates an unnecessary extra path, or it just approaches each point, which is not allowed in the path planning.

Tab. 1 *Relationship between interpolation solutions occurring in the literature*

	[52]	[53]	[54]	[55]	[56]	[57]	[58]	[59]	[60]	[61]	[62]	[63]	[64]	[65]
Lagrange polynomial interpolation	X		X	X	X	X		X	X	X	X	X	X	X
Newton Divided-Difference formula	X		X	X	X	X	X	X	X		X			
Hermit polynomial interpolation	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Bezier curves - Ber-stein polynomial	X	X	X	X		X	X		X	X		X	X	X
Natural cubic spline interpolation	X	X	X	X	X	X		X	X		X	X		X
Catmull-Rom cubic spline interpolation				X					X			X	X	X
Cubic B-spline	X	X	X	X	X	X			X	X		X	X	X

3.2 Path planning solution using cubic curves: Bezier curves with Berstein polynomial

Bezier curves can be written in two ways [55]:

- Pascal triangle and binomial principle
- Berstein form

As the most other literatures [53]-[54],[57]-[58],[60]-[61],[63]-[65] use the latter, therefore this method is described briefly in this subsection.

Based on 4 points and the literature [55], the Bezier curve can be written for the interval "u ∈ [0,1]" used in [48] and [66]:

$$\boldsymbol{P}_B(u) = (u^3, u^2, u, 1)\begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} \boldsymbol{P}_0 \\ \boldsymbol{P}_1 \\ \boldsymbol{P}_2 \\ \boldsymbol{P}_3 \end{pmatrix} \tag{2}$$

Where:

$\boldsymbol{P}_B(u)$: Bezier curve

$\boldsymbol{P}_0 = \left(P_{0_x}, P_{0_y}\right)$: start point

$\boldsymbol{P}_1 = \left(P_{1_x}, P_{1_y}\right)$: control point

$\boldsymbol{P}_2 = \left(P_{2_x}, P_{2_y}\right)$: control point

$\boldsymbol{P}_3 = \left(P_{3_x}, P_{3_y}\right)$: end point

Written in a different way, which can already be programmed in the Scilab software:

$$\boldsymbol{P}_B(u) = (1-u)^3\boldsymbol{P}_0 + 3u(1-u)^2\boldsymbol{P}_1 + 3u^2(1-u)\boldsymbol{P}_2 + u^3\boldsymbol{P}_3 \tag{3}$$

The curve $P_x(u)$ and $P_y(u)$ is from coordinates along the *x* and *y* axes, respectively as follows:

$$P_{B_X}(u) = (1-u)^3P_{0_x} + 3u(1-u)^2P_{1_x} + 3t^2(1-u)P_{2_x} + u^3P_{3_x} \tag{4}$$

$$P_{B_Y}(u) = (1-u)^3P_{0_y} + 3u(1-u)^2P_{1_y} + 3t^2(1-u)P_{2_y} + u^3P_{3_y} \tag{5}$$

3.3 Path planning solution using cubic curves: Hermite interpolation curve

The equation of the Hermite curve can be written similarly to the Bezier curve with the help of [55], using 2 points and 2 control vectors, according to the interval u∈[0,1]:

$$\boldsymbol{P}_H(u) = (u^3, u^2, u, 1)\begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} \boldsymbol{P}_0 \\ \boldsymbol{P}_3 \\ \boldsymbol{P}_{S1} \\ \boldsymbol{P}_{2E} \end{pmatrix} \tag{6}$$

Where:

$\boldsymbol{P}_H(u)$: Hermite curve

$\boldsymbol{P}_0 = \left(P_{0_x}, P_{0_y}\right)$: start point, as by Bezier-curve

$\boldsymbol{P}_{S1} = \left(P_{1_x}, P_{1_y}\right)$: control vector from start point

$\boldsymbol{P}_{2E} = \left(P_{2_x}, P_{2_y}\right)$: control vector to end point

$\boldsymbol{P}_3 = \left(P_{3_x}, P_{3_y}\right)$: end point, as by Bezier-curve

Writing the curve differently, which can already be used in Scilab:

$$\boldsymbol{P_H}(u) = (2u^3 - 3u^2 + 1)\boldsymbol{P}_0 + (-2u^3 + 3u^2)\boldsymbol{P}_3 + (u^3 - 2u^2 + t)\boldsymbol{P}_{S1} + (u^3 - u^2)\boldsymbol{P}_{2E} \quad (7)$$

The coordinates along the *x* and *y* axes can be written in two dimensions:

$$P_{H_X}(u) = (2u^3 - 3u^2 + 1)P_{0_x} + (-2u^3 + 3u^2)P_{3_x} + (u^3 - 2u^2 + t)P_{S1_x} + (u^3 - u^2)P_{2E_x} \quad (8)$$
$$P_{H_Y}(u) = (2u^3 - 3u^2 + 1)P_{0_y} + (-2u^3 + 3u^2)P_{3_y} + (u^3 - 2u^2 + t)P_{S1_y} + (u^3 - u^2)P_{2E_y} \quad (9)$$

As will be seen in the subsection 4.1, these data are already available for calculating the points of the curve.

4 Implementation of career planning on the examined AGV

This section describes the initial data, the path planning using the Bezier- and Hermite-curve.

4.1 Initial data for path planning

The driverless vehicle can use 3 data using the Sick laser navigation sensor:

- P_{start}=(X$_{start}$,Y$_{start}$) [m] initial position,
- φ_{start} [°] initial orientation.

The vehicle's target position data is as follows similarly to the initial position and orientation:

- P_{end}=(X$_{end}$,Y$_{end}$) [m] target position,
- φ_{end} [°] target orientation.

The vehicle's LIDAR sensor returns the angle value in an anti-clockwise direction as shown in Fig. 1 on the *Y* axis. In order to achieve the movement of the vehicle, a curve must be drawn between these two points, at which the start and end points are interpolated, and the further control points are approximated.

In the next step, the control points P_1 and P_2 should be defined, which are required to generate the cubic curves, derived from the initial and end positions and orientations. The control vector $\boldsymbol{P_{S1}}$ is derived from the vector between the start point P_{start} and the first control point P_1, the length of this vector is $|\boldsymbol{P_{S1}}|$=1m due to the simplicity of the calculations. The $\boldsymbol{P_{2E}}$ control vector is derived from the vector between the second control point P_2 and the target point P_{end}, also with vector‘s length $|\boldsymbol{P_{2E}}|$=1m. From the orientation, the values of *X* and *Y* of point P_1 can be calculated by simple trigonometric relations using the unit vector length:

$$P_{1_X} = P_{start_X} + sin\,\varphi_{start} \quad (10)$$

$$P_{1_y} = P_{start_Y} + cos\varphi_{start} \quad (11)$$

The values of *X* and *Y* of point P$_2$ can also be calculated from the orientation:

$$P_{2_X} = P_{end_X} - sin\,\varphi_{end} \quad (12)$$
$$P_{2_y} = P_{end_Y} - cos\varphi_{end} \quad (13)$$

The two control point control vectors can be written:

$$P_{S1} = P_1 - P_{start} \quad (14)$$
$$P_{2E} = P_{end} - P_2 \quad (15)$$

where the vectors $\boldsymbol{P_{start}}$, $\boldsymbol{P_{S1}}$, $\boldsymbol{P_{2E}}$ and $\boldsymbol{P_{end}}$ are the vectors between the origin and the point P_{start}, P_1, P_2 and P_{end} respectively. The placement of each vector and point is shown in Fig. 1.

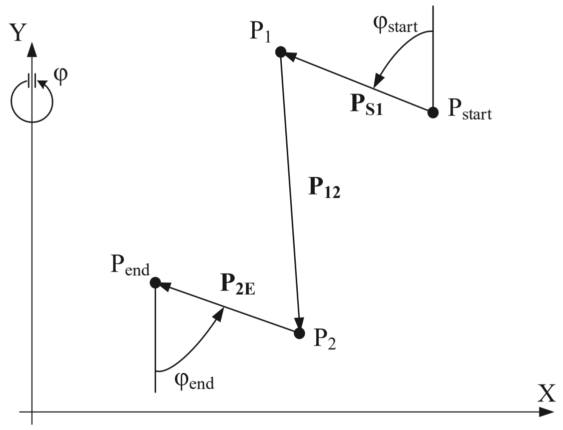


Fig. 1 *Points and vectors for path planning*

4.2 Path resulting from the Bezier curve

To draw, the following coordinates and angle values were entered into the Scilab program:

$$P_{start} = (0m, 1m), \varphi_{start} = 255° \quad (16)$$
$$P_{end} = (-0.5m, -1m), \varphi_{end} = 255° \quad (17)$$

The currently used form of the Bezier curve previously described in subsection 3.2:

$$\boldsymbol{P}_B(u) = (1-u)^3\boldsymbol{P}_0 + 3u(1-u)^2\boldsymbol{P}_1 + 3u^2(1-u)\boldsymbol{P}_2 + u^3\boldsymbol{P}_3 \quad (18)$$

In this case, the vectors drawn from the origin to the start and end points and to the two control points can be written as follows:

$$\boldsymbol{P}_0 = \boldsymbol{P_{start}} \quad (19)$$

$$\boldsymbol{P_1} = (P_{1X}, P_{1Y}) = (P_{start_X} + sin\,\varphi_{start}, P_{start_Y} + cos\,\varphi_{start}) \quad (20)$$
$$\boldsymbol{P_2} = (P_{2X}, P_{2Y}) = (P_{end_X} - sin\,\varphi_{end}, P_{end_Y} - cos\,\varphi_{end}) \quad (21)$$

$$\boldsymbol{P}_3 = \boldsymbol{P_{end}} \tag{23}$$

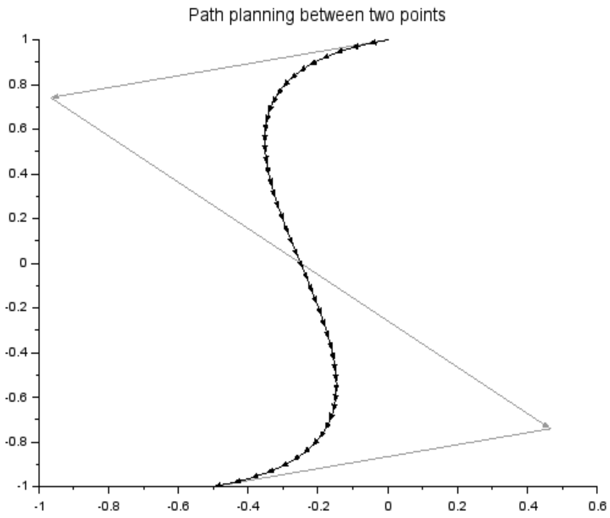


Fig. 2 *Path planning with Bezier curve*

The u interval also moves from 0 to 1 with 0.02 increments, so the curve will consist of 50 points.

The simulated curve is shown in Fig 2. The black curve indicates the Bezier curve, while the vectors marked in green show the vectors $\boldsymbol{P_{S1}}$ between the start and first control points, $\boldsymbol{P_{2E}}$ between the second control and end points, and the $\boldsymbol{P_{SE}}$ vectors between the start and end points.

4.3 Path resulting from the Hermite curve

The following coordinates and angle values were used in the program to draw, as by the Bezier curve:

$$P_{start} = (0m, 1m), \varphi_{start} = 255° \tag{23}$$
$$P_{end} = (-0.5m, -1m), \varphi_{end} = 255° \tag{24}$$

The currently used form of the Hermite curve, which was previously described in subsection 3.3, is as follow:

$$\boldsymbol{P}_H(u) = (2u^3 - 3u^2 + 1)\boldsymbol{P}_0 + (-2u^3 + 3u^2)\boldsymbol{P}_3 + (u^3 - 2u^2 + u)\boldsymbol{P}_{S1} + (u^3 - u^2)\boldsymbol{P}_{2E} \tag{25}$$

In this case, the vectors drawn from the origin to the start and end points, as well as the control vectors, can be written as follows:

$$\boldsymbol{P}_0 = \boldsymbol{P_{start}} \tag{26}$$
$$\boldsymbol{P_{S1}} = (\sin\varphi_{start}, \cos\varphi_{start}) \tag{27}$$
$$\boldsymbol{P_{2E}} = (\sin\varphi_{end}, \cos\varphi_{end}) \tag{28}$$
$$\boldsymbol{P}_3 = \boldsymbol{P_{end}} \tag{29}$$

The u interval also moves from 0 to 1 with 0.02 increments, so the curve will consist of 50 points.

The simulated curve based on the program written in Scilab software is shown in Fig 3. The red curve indicates the Hermite curve, while the vectors marked in green show the vectors $\boldsymbol{P_{S1}}$ between the start and first control points, $\boldsymbol{P_{2E}}$ between the second control and end points, and the $\boldsymbol{P_{SE}}$ vectors between the start and end points.

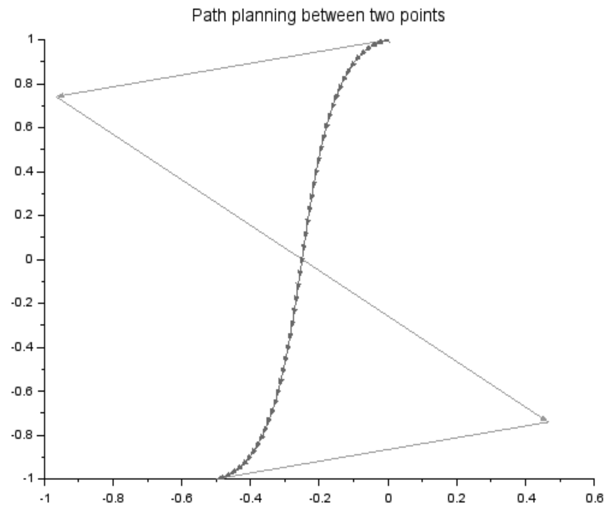


Fig. 3 *Path planning with Hermite curve*

4.4 Comparison of two curves

This subsection compares the curves described in subsections 4.2 and 4.3 (see Fig. 4). As in the previous sections, the red curve and the black curve represents the Hermite curve and the Bezier curve, respectively. As can be seen from the simulation figure, the Hermite curve results a significantly shorter path, while the Bezier curve performs a less sharp curve around the start and end points.

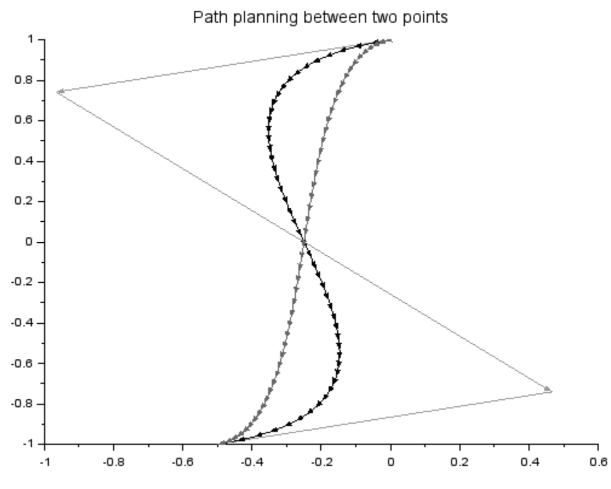


Fig. 4 *Comparison of curve types resulting from path planning methods*

5 Problems encountered during path planning and their solutions

5.1 Extra path due to the wrong approach

In the current example, at angular values, a vehicle starts and arrives rotated 180° compared to the previous examples, with unchanged start and end point coordinates:

$$P_{start} = (0m, 1m), \varphi_{start} = 255° \quad (30)$$

$$P_{end} = (-0.5m, -1m), \varphi_{end} = 255° \quad (31)$$

In this case, the curves shown in Fig. 5 are obtained. It can be stated that in this case an extra path occurs compared to the previous version due to the wrong approach to the initial and final direction.

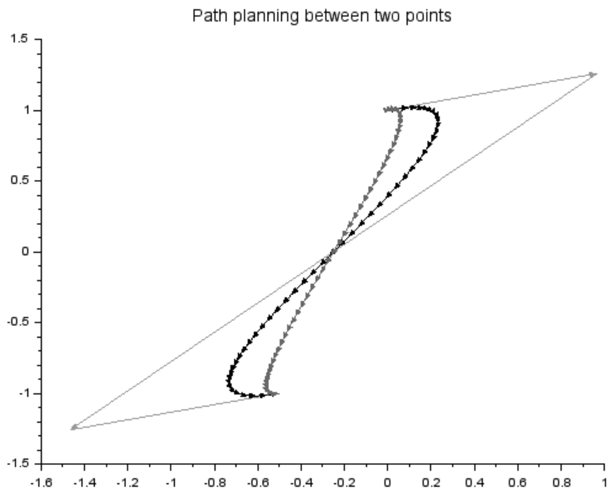


Fig. 5 *Comparison of path planning methods without change of direction for both curve types*

However, the symmetrical structure of the vehicle allows moving in the opposite starting direction between the same initial and target points, resulting a shorter distance. The program solves it with a condition examining.

The vectors $\boldsymbol{P_1}$ and $\boldsymbol{P_2}$ used for the calculation of Bezier curve are used and optionally changed with results of the vectors $\boldsymbol{P_1'}$ and $\boldsymbol{P_2'}$ as follows:

$$P_1' = \begin{cases} P_1' = P_{start} + P_{S1}, if\ \widehat{P_{S1}P_{SE}} < 90° \\ P_1' = P_{start} - P_{S1}, if\ \widehat{P_{2E}P_{SE}} > 90° \end{cases} \quad (32)$$

$$P_2' = \begin{cases} P_2' = P_{end} - P_{2E}, if\ \widehat{P_{2E}P_{SE}} < 90° \\ P_2' = P_{end} + P_{2E}, if\ \widehat{P_{2E}P_{SE}} > 90° \end{cases} \quad (3)$$

that is, it changes the direction of the vector $\boldsymbol{P_1'}$ or $\boldsymbol{P_2'}$ from the starting point and the end point, if the angle $\boldsymbol{\widehat{P_{S1}P_{SE}}}$ or $\boldsymbol{\widehat{P_{2E}P_{SE}}}$ is greater than 90 degrees, where the angle is enclosed by the vectors $\boldsymbol{P_{S1}}$ and $\boldsymbol{P_{2E}}$ or $\boldsymbol{P_{S1}}$ and $\boldsymbol{P_{SE}}$, respectively.

The vectors $\boldsymbol{P_{S1}}$ and $\boldsymbol{P_{2E}}$ used for the calculation of Hermite curve are used and optionally reversed with results of the vectors $\boldsymbol{P_{S1}'}$ and $\boldsymbol{P_{2E}'}$ as follows:

$$\boldsymbol{P_{S1}'} = \begin{cases} \boldsymbol{P_{S1}' = P_{S1}}, if\ \boldsymbol{\widehat{P_{S1}P_{SE}}} < 90° \\ \boldsymbol{P_{S1}' = -P_{S1}}, if\ \boldsymbol{\widehat{P_{S1}P_{SE}}} > 90° \end{cases} \quad (34)$$

$$\boldsymbol{P_{2E}'} = \begin{cases} \boldsymbol{P_{2E}' = P_{2E}}, if\ \boldsymbol{\widehat{P_{2E}P_{SE}}} < 90° \\ \boldsymbol{P_{2E}' = -P_{2E}}, if\ \boldsymbol{\widehat{P_{2E}P_{SE}}} > 90° \end{cases} \quad (35)$$

that is, it reverses the direction of the vectors $\boldsymbol{P_{S1}'}$ or $\boldsymbol{P_{2E}'}$, if the angle enclosed by the examined vectors $\boldsymbol{P_{S1}}$ or $\boldsymbol{P_{2E}}$ is greater than 90 degrees, respectively.

By reversing the direction, the curve shown in Fig. 5 changes to the path shown in Fig. 4, since in each condition tests the angle enclosed by the examined vectors is in each case larger than the tested value.

5.2 Path planning to the midway point between wheels

So far, for calculating a point-like body was assumed. However, in the reality the AGV has an extent, therefore the initial values in the program due to the extent of the vehicle are as follows:

- $P_{start,LIDAR}=(X_{start,LIDAR}, Y_{start,LIDAR})$ [m] initial position measured by LIDAR sensor,
- φ_{start} [°] initial orientation measured by LIDAR sensor,
- $P_{end,LIDAR}=(X_{end,LIDAR}, Y_{end,LIDAR})$ [m] position measured and adjusted by LIDAR sensor,
- φ_{end} [°] orientation measured and adjusted by LIDAR sensor.

The zero position and orientation in the coordinate system are shown in Fig. 6.

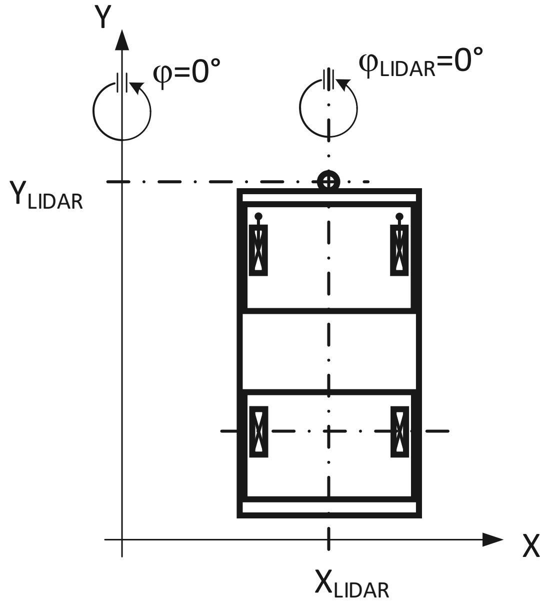


Fig. 6 *Position of LIDAR sensor compared to the vehicle and the coordinate system*

These values should be derived for the initial P_{start} and the target P_{end} position for the planning of the route, using the distance $s_{LIDAR\text{-}W}$=0.770m between the midway point of the wheels and the centre point of the LIDAR sensor as shown in Fig. 7:

$$P_{start} = \left(X_{start,LIDAR} - s_{LIDAR-W} \cdot sin(\varphi_{start}), Y_{start,LIDAR} + s_{LIDAR-W} \cdot cos(\varphi_{start})\right) \quad (36)$$

$$P_{end} = \left(X_{end,LIDAR} - s_{LIDAR-W} \cdot sin(\varphi_{end}), Y_{end,LIDAR} + s_{LIDAR-W} \cdot cos(\varphi_{end})\right) \tag{37}$$

The dimensions of the vehicle are shown in Fig. 7.

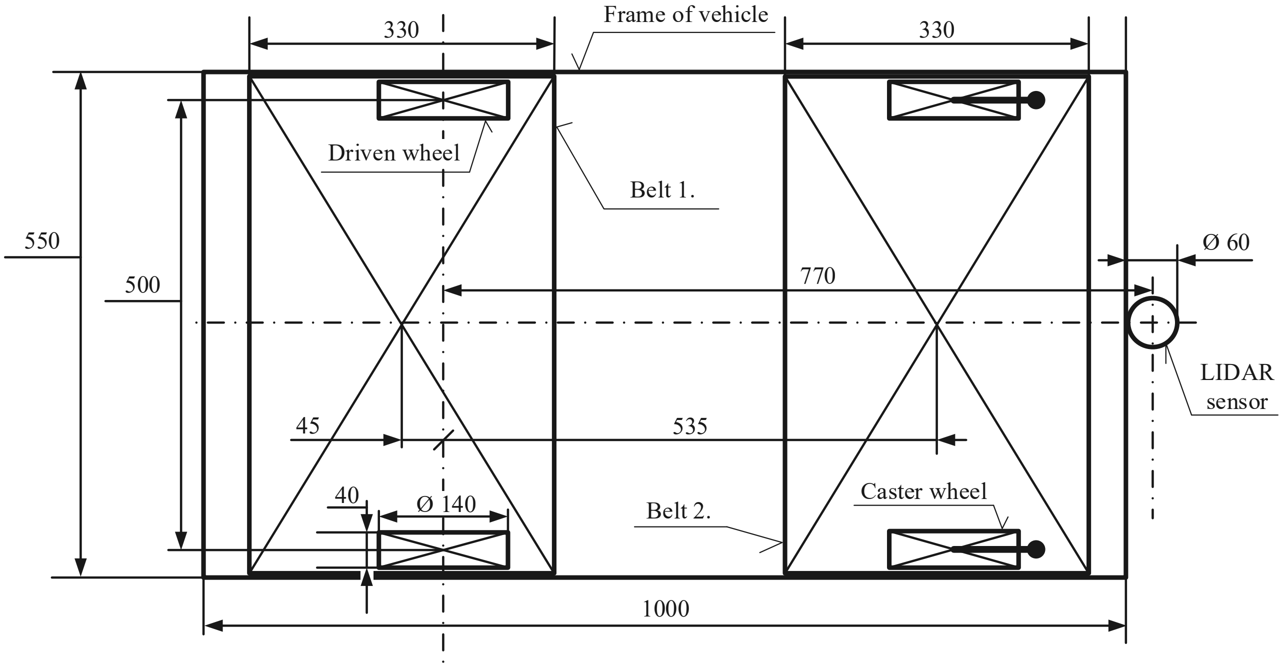


Fig. 7 *Dimensions and names of the important parts of the AGV*

5.3 Final positioning problem

If only one of the conditions was taken into account to in the subsection 5.1, the orientation of the AGV may not be in the correct direction, as shown in Fig. 8.

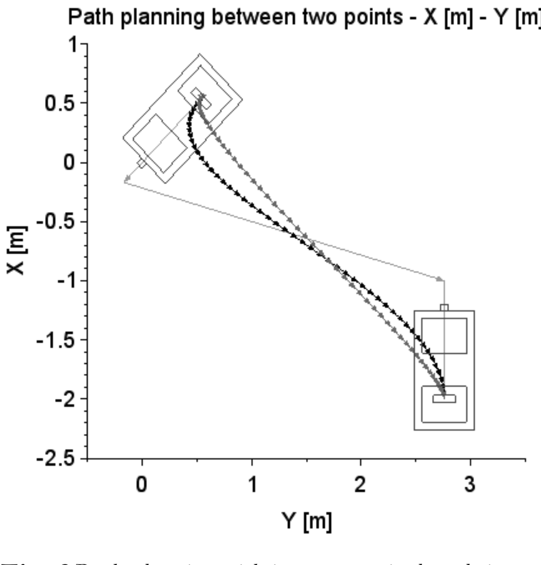


Fig. 8 *Path planning with incorrect arrival angle in case of reversed starting for both curve types*

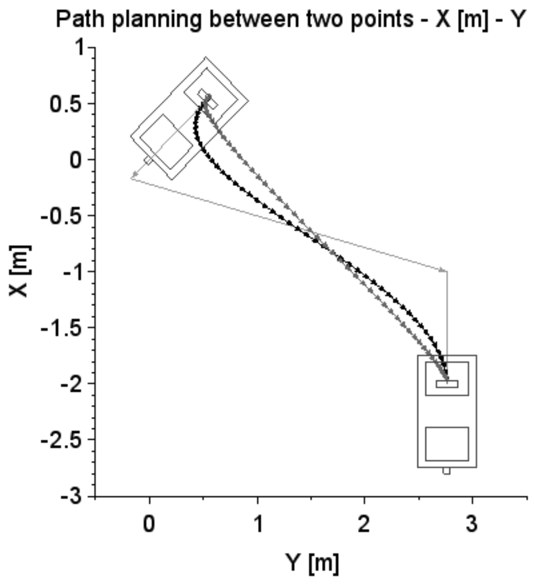


Fig. 9 *Path planning with correct arrival angle in case of reversed starting for both curve types*

The positions and orientations shown in Fig. 8.

The solution is to change the angle of the final position by rotating 180° and entering it into the simulation program.

$$P_{start}=(0m,0m),\ \varphi_{start}=315° \text{ and } P_{end}=(2m,-2m),\ \varphi_{end}=180° \tag{38}$$

5.4 Situation arising from the position on conveyor belts

In addition to the chaning orientation detailed in previous subsections, also changing target position of the AGV should be paid an attention, depending on which conveyor belt of the AGV is positioned. By binding the target position to conveyor belts, the AGV can pick or place a unit cargo from the correct belt to a material handling system. The position und ID of the belts can be seen in Fig. 7.

When determining the target position, the AGV must be moved to the connection point with belt 1., where the value from LIDAR sensor can be read and stored.

If it is necessary to shift from belt 1. to belt 2., a transforming in the target position P_{end} is required, using the longitudinal distance ($s_{belt_1,belt_2}$=0.580m) between the centre of the two belts (see Fig. 7):

$$X_{end} = X_{end,belt2} = X_{end,belt1} - s_{belt_1,belt_2} \cdot sin(\varphi_{end}) \tag{39}$$

$$Y_{end} = Y_{end,belt2} = Y_{end,belt1} + s_{belt_1,belt_2} \cdot cos(\varphi_{end}) \tag{40}$$

Another task to be solved in case of a change of direction is to determine the new target position for both belts, using the distance between the midway point of the wheel and each belt ($s_{belt_1,W}$=0.045m, $s_{belt_2,W}$=0.535m):

- belt 1.: if a change of direction is required:

$$X_{end} = X_{end,belt1} - 2 \cdot s_{belt_1,W} \cdot sin(\varphi_{end}) \tag{411}$$

$$Y_{end} = Y_{end,belt1} + 2 \cdot s_{belt_1,W} \cdot cos(\varphi_{end}) \tag{42}$$

- belt 2.: if a change of direction is required:

$$X_{end} = X_{end,belt2} - 2 \cdot s_{belt_2,W} \cdot sin(\varphi_{end}) \tag{43}$$

$$Y_{end} = Y_{end,belt2} + 2 \cdot s_{belt_2,W} \cdot cos(\varphi_{end}) \tag{44}$$

Fig. 10 and Fig. 11 summarize the equations in a 1-1 curve for the belt 1. and belt 2. in the case of without reversing and with reversing, respectively.

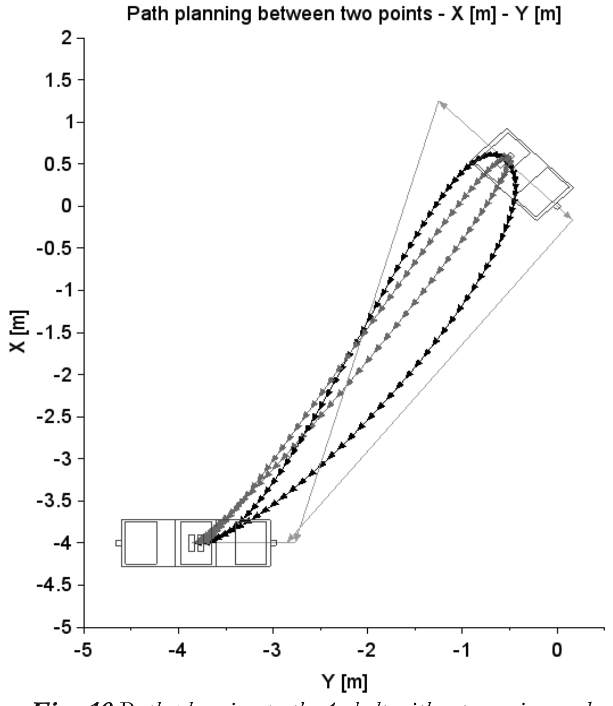


Fig. 10 *Path planning to the 1. belt without revesing and with reversing for both curve types*

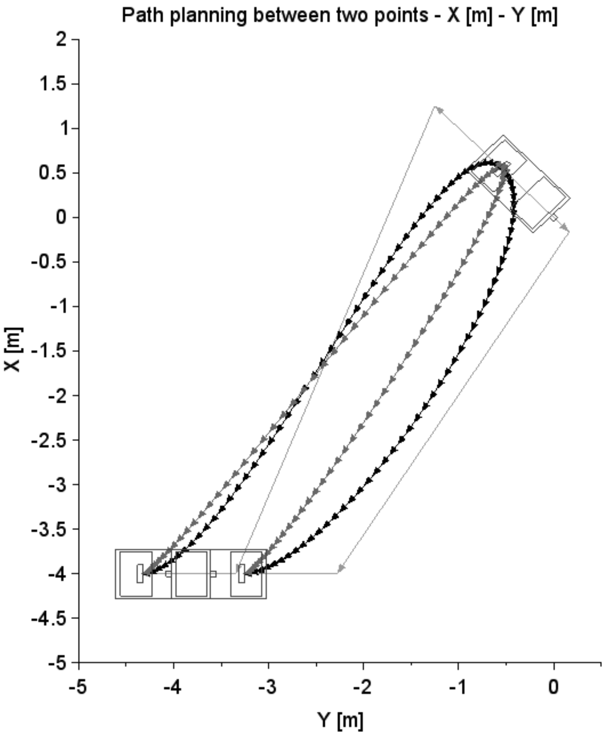


Fig. 11 *Path planning to the 2. belt without reversing and with reversing for both curve types*

5.5 Connect multiple segments

In the previous descprition the path planning has only been done with 1 segment, but there may be cases where the route needs to be assembled from multiple segments, typically avoiding pre-known obstacles. The literature [48] and [55] recommend connecting segments in different ways.

The [48] defines a series of waypoints $W_0, W_1,\dots,W_i,\dots,W_N$, where N is the number of waypoints. The generated route connects the adjacent waypoints W_{i-1} and W_i, where $i=1,\dots,N$. The waypoints W_{i-1} and W_i indicates the position and orientation of the and waypoints and their relationship as $q_{i-1}=[X_{i-1},Y_{i-1},\varphi_{i-1}]^T$ and $q_i=[X_i,Y_i,\varphi_i]^T$, respectively. With the Bezier curve, the path can be created as follows:

$$x(u_i) = (1-u_i)^3 X_{i-1} + 3u_i(1-u_i)^2 X_{ai} + 3u_i^2(1-u_i)X_{bi} + u_i^3 X_i \tag{45}$$

$$y(u_i) = (1-u_i)^3 Y_{i-1} + 3u_i(1-u_i)^2 Y_{ai} + 3u_i^2(1-u_i)Y_{bi} + u_i^3 Y_i \tag{46}$$

where $x(u_i)$ and $y(u_i)$ is the value x and y of the path and, respectively, where $u_i \in [0,1]$ is the interval; (X_{ai}, Y_{ai}) and (X_{bi}, Y_{bi}) parameters is to be defined. If value u_i changes from 0 to 1, functions $(x(u_i), y(u_i))$ range from (X_{i-1}, Y_{i-1}) waypoint to (X_i, Y_i) waypoint. For each $W_{i-1}W_i$ segment, the final W_i waypoint will be the starting point for the next W_iW_{i+1} segment.

The [55] describes, that a n-dimensional Bezier curve can be computation-intensive, therefore the path should be divided into multiple Bezier segments, each defined by 4-6 points. To connect two segments, the literature defines a set of points $P_0, P_1, \ldots, P_n$ and $Q_0, Q_1, \ldots, Q_n$, where $n=4 \div 6$. To connect P_n and Q_0, the point must be the same. In addition, the literature states that the P_{n-1}, P_n and Q_1 points must be aligned, thus ensuring a smooth connection between the segments.

In the used solution for the examined AGV, the program uses as its initial value the measured initial position $P_{start,LIDAR,1}=(X_{start,LIDAR,1}, Y_{start,LIDAR,1})$ and measured orientation $\varphi_{start,1}$ of the first segment. From the second segment, each point gets a measured final position $P_{end,LIDAR,segment}=(X_{end,LIDAR,segment}, Y_{end,LIDAR,segment})$ and measured orientation $\varphi_{end,segment}$, where the value segment represents the serial number of the current segment and can be maximum number of predefined segments. The position $P_{end,1}=(X_{end,1}, Y_{end,1})$ and orientation $\varphi_{end,1}$ of the final point of the first segment can be calculated as described in the previous subsections. From the second segment, the starting position and orientation of each segment will be given the position and orientation of the final point of the previous segment as follows:

$$P_{start,LIDAR,segment} = P_{end,segment-1}, \text{ if } i > 1 \tag{47}$$

$$\varphi_{start,segment} = \varphi_{end,segment-1}, \text{ if } i > 1 \tag{48}$$

Then the position $P_{end,segment}=(X_{end,segment}, Y_{end,segment})$ and orientation $\varphi_{end,segment}$ of the final point of each segment can be determined as already known.

An example can be seen in Fig. 12, where the initial position $P_{start,LIDAR,1}=(0m,0m)$ and initial orientation $\varphi_{start,1}=0°$, for the target of the 1. segment the point $P_{end,LIDAR,2}=(-2m,-3m)$ and orientation $\varphi_{end,2}=90°$ and for the target of the 2. segment point $P_{end,LIDAR,3}=(-4m,0m)$ and orientation $\varphi_{end,3}=45°$ are prescribed. In this case, the vehicle continues in the same direction from point 2.

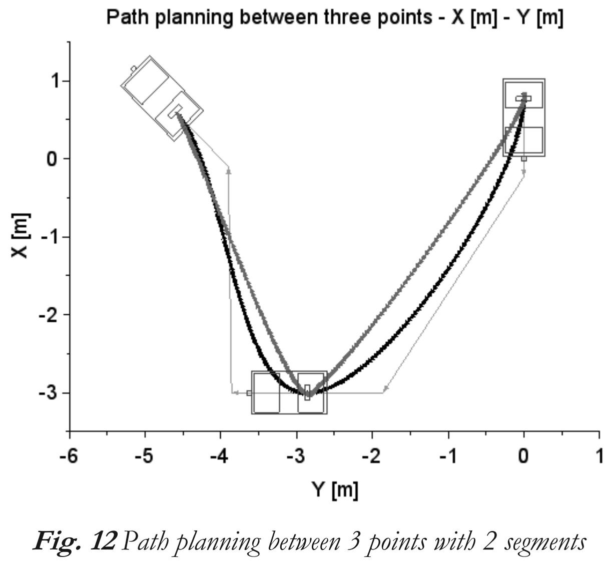


Fig. 12 *Path planning between 3 points with 2 segments without direction changing along the way*

However, there may be cases where the vehicle does not continue in the same direction from the intermediate point, but in the opposite direction to the previous one. An example for this direction changing along the way can be seen in Fig. 13, where the initial position $P_{start,LIDAR,1}=(0m,0m)$ and initial orientation $\varphi_{start,1}=0°$, for the target of the 1. segment the point $P_{end,LIDAR,2}=(-2m,-3m)$ and orientation $\varphi_{end,2}=180°$ and for the target of the 2. segment point $P_{end,LIDAR,3}=(-4m,0m)$ and orientation $\varphi_{end,3}=45°$ are prescribed.

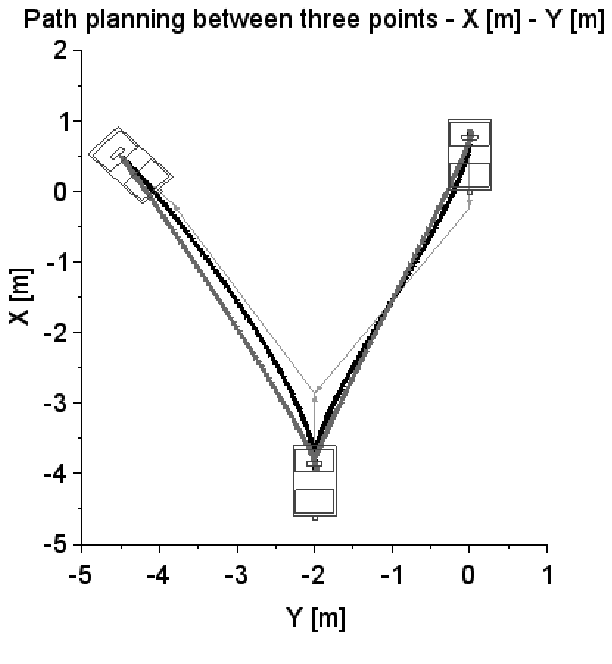


Fig. 13 *Path planning between 3 points with 2 segments with direction changing along the way*

6 Implementation of trajectory planning on the examined AGV

6.1 Data initially available for trajectory planning

The [48] literature uses a predefined velocity and anglangular velocity profile for trajectory planning. During the experiment presented in the literature, the robot accelerated to the desired velocity ($0,3\frac{m}{s}$) within 3 seconds (stage 1.) and then maintained this velocity (stage 2.), and at 6 seconds the angular velocity began to increase, reaching the value $\frac{\pi}{2}\frac{rad}{s}$ at 9 seconds. Finally, both linear and angular velocitys began to decrease at 10 seconds and drop to zero at 13 seconds (stage 3.). As it can be seen, the angular velocity was predefined here, but in the solution that I used, the angular velocity is arises from the curves produced by the path planning algorithm. However, before determining the angular velocity profile, the velocity profile must be produced from the waypoints of the curves.

As far as the velocity profile is concerned, the solution follows the tendency written in the literature, therefore the velocity increases in 0.5s (stage 1.), then holds the desired velocity between the two stages (stage 2), and then slows down to zero (stage 3) at the end. The intermediate time is predefined in the literature, but the developed trajectory algorithm calculates time from the path length.

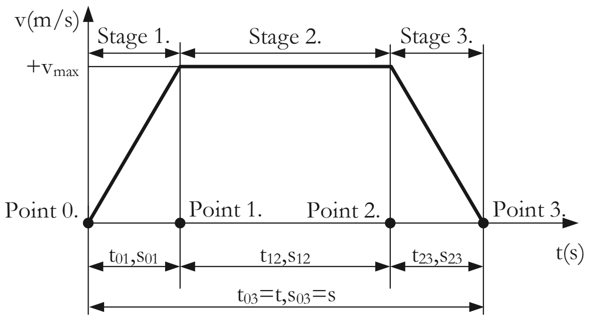


Fig. 14 *Planned velocity profile for one segment*

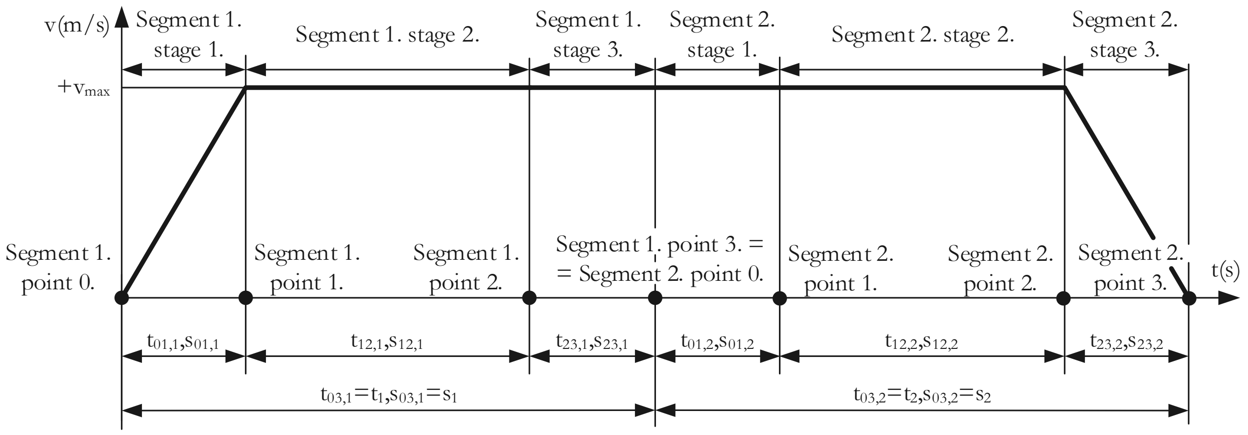


Fig. 15 *Planned velocity profile for two segments with midway direction changing*

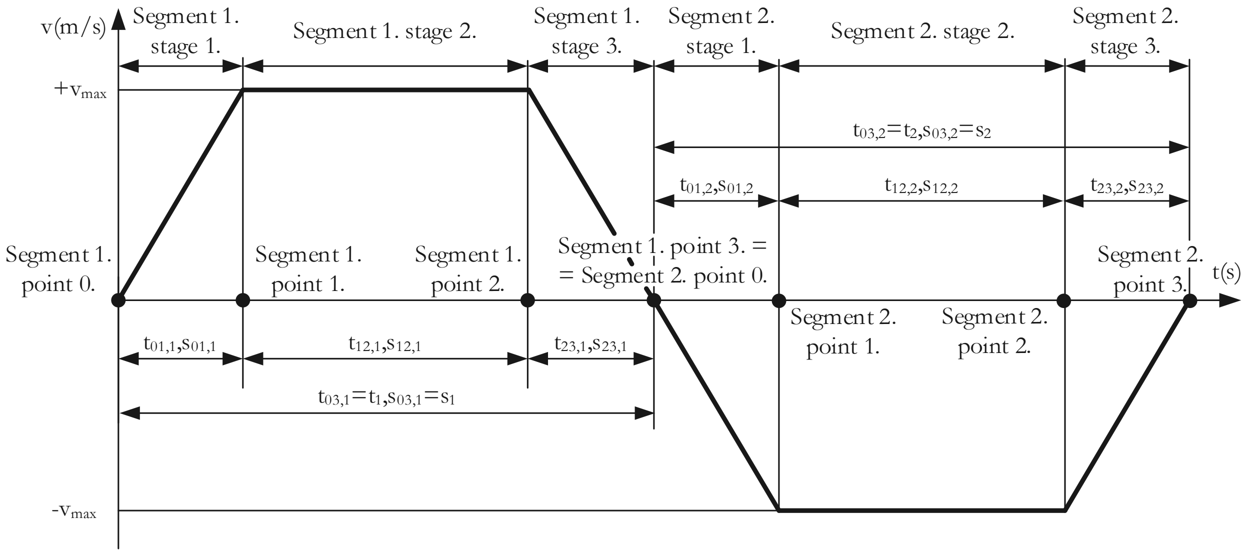


Fig. 16 *Planned velocity profile for two segments without midway direction changing*

Further resolvable task is implementation of the velocity profile for multiple segments. There are basically two possible cases between two segment changes:

- if the vehicle continues in the same direction from stage 1. to stage 2., as in Fig. 12, and do not have to slow down (see Fig. 15),
- if the vehicle has to change its direction into the opposite direction, as in Fig. 13, it must slow down to zero and then accelerate to the opposite velocity (see Fig. 16).

The velocity profiles shown in Fig 15 and Fig 16. may change to the opposite way.

Considering the above-mentioned facts, the lengths and time values of stages can vary. For stage data, firstly the total length of the segments must be determined:

$$s_{segment} = \sum_{i=1}^{u_{end}} s_{diff,i,segment} \qquad (49)$$

Where:

$s_{segment}$: Total length, where $segment=1, 2, \dots, segment_{end}$ [m],

u_{end}: Number of points in a segment of the curve [-],

$s_{diff,i,segment}$: Distance between the points of the curve at the point of that segment, where $i=1, 2, \dots, u_{end}$ [m].

6.2 Calculation of the time and length required for trajectory planning

The path length data and time data for segments are as follows, where the serial number of the segment $segment=1, 2, \dots, segment_{end}$:

1. t_{01}=0.5s, time value between point 0. and point 1. of the segment, true for all segments
   1.1 $s_{01}=v_{max}\cdot t_{01}$, for all segments, the distance between point 0. and point 1. of the segment if the directions of adjacent segments are the same
   1.2 $s_{01}=a_{max}\cdot\frac{t_{01}^2}{2}$, true for all segments, point 0. and point 1. of the segment, if the directions of adjacent segments do not match, e.g. in the case of a change of direction or first segment
2. $s_{12,segment}=s_{segment}-s_{01,segment}-s_{23,segment}$, the length of the path between point 1. and point 2. of the actual segment and therefore $t_{12,segment}=\frac{s_{12,segment}}{v_{max}}$ [s], the time value between point 1. and point 2. of the segment
3. t_{23}=0.5s, time value between point 2. and point 3. of the segment, true for all segments
   3.1 $s_{23}=v_{max}\cdot t_{23}$, is true for all segments, between point 2. and point 3. of the segment if the directions of adjacent segments are the same
   3.2 $s_{23}=v_{max}\cdot t_{23}-a_{max}\cdot\frac{t_{23}^2}{2}$, true for all segments, point 2. and point 3. of the segment, if the the directions of adjacent segments do not match, e.g. in the case of a change of direction or last segment

Finally, it is possible to determine the total time value of the examined segment between point 0. and point 3.:

$$t_{segment} = t_{03,segment} = t_{01} + t_{12,segment} + t_{23} \quad [s] \qquad (50)$$

6.3 Production of the velocity profile resulting from trajectory planning

The initial data, the time value and the path length value are so far available from the calculations in subsection 6.1 and in subsection 6.2. The current path length "s" _"i" assigned to the waypoints is calculated from the "X" and "Y" points of the curve produced at the Bezier-curve or Hermite-curve, where

$$s_{actual}(i) = s_{start,segment} + s_{diff}(i) \qquad (51)$$

$$s_{diff}(i) = \sqrt{(X_i - X_{i-1})^2 + (Y_i - Y_{i-1})^2}, \text{ if } i>1 \qquad (52)$$

Where:

$segment =1,2,\dots,segment_{end}$: Serial number of the actual segment, where $segment_{end}$ is the highest number [-],

$i=1,2,\dots,segment\cdot u_{end}$: Serial number of the actual waypoint, where $segment_{end}\cdot u_{end}$ is the highest number [-],

$s_{diff}(i)$: Distance between the two adjacent waypoints for the i. section of a segment [m].

In each segment, the maximum velocity may vary depending on the direction, therefore this phanomena should be determined by the direction coefficient $k_{start,segment}$.

The times, velocitys and accelerations assigned to waypoints taking into account a well-known kinematic formula of motion with constant acceleration and the initial time value and length of the path for each section, where and are given as follows, where the serial number of the segment $segment=1,2,\dots,segment_{end}$ and the number of the given waypoint $i=1,2,\ldots,u_{end},\ldots,segment_{end}\cdot u_{end}$:

1. current time, current velocity, and current acceleration between point 0. and point 1. of the actual segment

1.1 $t(i)=t_{0,segment}+\frac{s_i-s_{start,segment}}{v_{max}}$;

$v(i)=k_{start,segment}\cdot v_{max}$; $a(i)=0$, for the actual waypoint of the curve, if the the directions of adjacent segments are the same

1.2 $t(i)=t_{0,segment}+\sqrt{\frac{2\cdot(s_i-s_{start,segment})}{a_{max}}}$;

$v(i)=k_{start,segment}\cdot a_{max}\cdot\left(t(i)-t_{0,segment}\right)$;

$a(i)=k_{start,segment}\cdot a_{max}$, for the actual waypoint of the curve if the the directions of adjacent segments do not match, e.g. in the case of a change of direction or first segment

2. $t(i)=t_{0,segment}+t_{01}+\frac{s_i-s_{01}-s_{start,segment}}{v_{max}}$;

$v(i)=k_{start,segment}\cdot v_{max}$; $a(i)=0$, current time, current velocity, and current acceleration between point 1. and point 2. of the actual segment

3. current time, current velocity, and current acceleration between point 2. and point 3. of the actual segment

3.1 $t(i)=t_{0,segment}+t_{01}+t_{12,segment}+\frac{s_i-s_{12,segment}-s_{01}-s_{start,segment}}{v_{max}}$;

$v(i)=k_{start,segment}\cdot v_{max}$; $a(i)=0$, for the actual waypoint of the curve, if the the directions of adjacent segments are the same

3.2 $t(i)=t_{0,segment}+t_{01}+t_{12,segment}+$

$$+\frac{v_{max}-\sqrt{v_{max}^2-2\cdot(s_i-s_{12,segment}-s_{01}-s_{start,segment})\cdot a_{max}}}{a_{max}};$$

$v(i)=k_{start,segment}\cdot\left(v_{max}-a_{max}\cdot\left(t(i)-t_{01}-t_{12,segment}-t_{0,segment}\right)\right)$;

$a(i)=k_{start,segment}\cdot a_{max}$, for the actual waypoint of the curve if the the directions of adjacent segments do not match, e.g. in the case of a change of direction or last segment

From the example shown in Fig. 12 and Fig. 13, the path-time diagram can be seen in the left side of Fig. 17 and Fig. 18, while the velocity-time diagram can be seen in the right side of Fig. 17 and Fig. 18, respectively. In the first case, since the directions of both first segments are negative, the curves are opposite compared to Fig. 15 and Fig. 16.

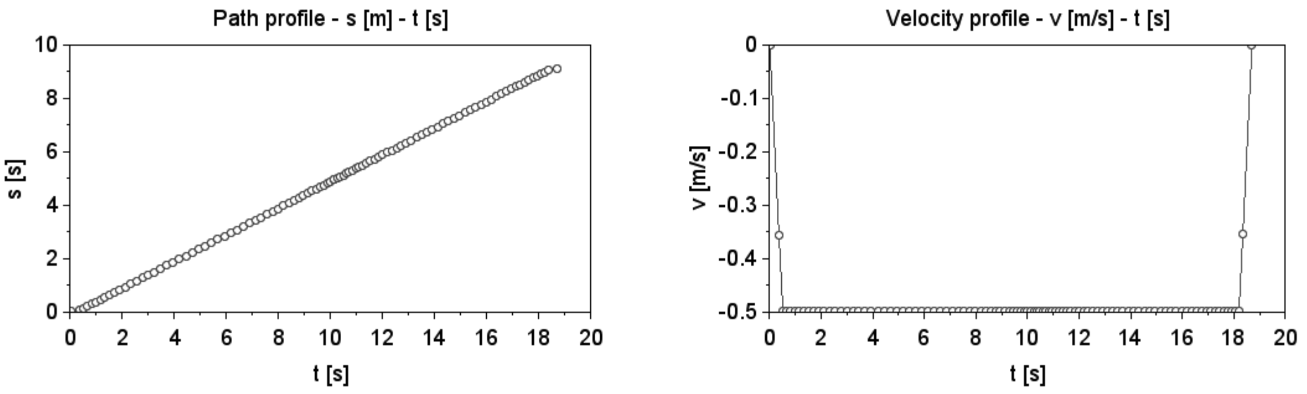


Fig. 17 *Result of trajectory planning: path length-time and velocity-time diagram with 2 segments without reversing*

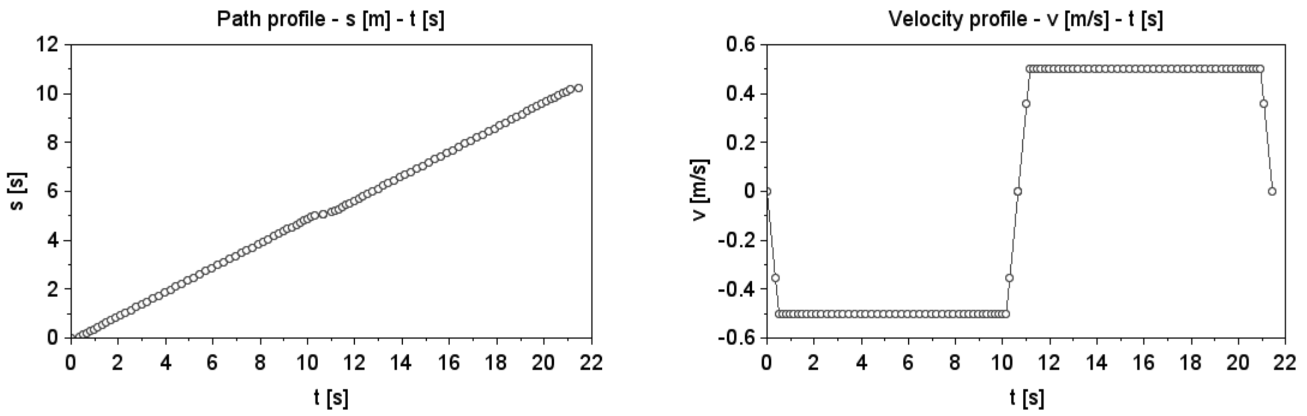


Fig. 18 *Result of trajectory planning: path length-time and velocity-time diagram with 2 segments with reversing*

6.4 Production of the anglular velocity profile resulting from trajectory planning

In addition to velocity, it is necessary to determine the angular velocity, as the vehicle moves predominantly not only along a straight line, but also turns. The angular velocity is independent from the velocity and it comes directly from the data on the curves. In this subsection, the serial number of the segment *segment=1,2,…,segment*$_{end}$ and the number of the given waypoint *i=1,2,…,u*$_{end}$*,…,segment*$_{end}\cdot u_{end}$ are used.

To determine the angle velocity profile, the angle values must be first determined for each waypoint from vectors. These vectors use data from two adjacent points for both path types:

$$\boldsymbol{P}_{actual}(i) = [X_{actual}, Y_{actual}] = \boldsymbol{P}_i - \boldsymbol{P}_{i-1}, \text{ if } i > 1 \tag{53}$$

The angle of these vectors, in relation to the vector $P_{vertical}=[X_{vertical}, Y_{vertical}]=[0,1]$ declared on the vertical axis, can be calculated from the well-known cosine similarity:

$$\varphi_{actual}(i) = \arccos\left(\frac{X_{actual}\cdot X_{vert} + Y_{actual}\cdot Y_{vert}}{|\boldsymbol{P}_{actual}|\cdot|\boldsymbol{P}_{vert}|}\right) \tag{54}$$

Since $X_{vert}=0$ and $|P_{vert}|=1$, therefore, the relationship is simplified as follows:

$$\varphi_{actual}(i) = \arccos\left(\frac{Y_{actual}\cdot Y_{vert}}{|\boldsymbol{P}_{actual}|}\right), \text{ where } \varphi_{actual}(i) \in [0°, 180°] \tag{55}$$

However, this only gives a value between 0° and 180°, but a value between 0° and 360° would be desirable. Therefore, one condition is introduced:

$$\varphi_{actual}(i) = \begin{Bmatrix} \varphi_{actual}(i) = \varphi_{actual}(i), \text{if } X_{actual} < 0 \\ \varphi_{actual}(i) = 360° - \varphi_{actual}(i), \text{if } X_{actual} > 0 \end{Bmatrix} \text{where } \varphi_{actual}(i) \in [0°, 360°] \tag{56}$$

In order to calculate the angular velocity at each point, an angular difference between each point is required:

$$\varphi_{diff}(i) = \varphi_{actual}(i) - \varphi_{actual}(i-1), \text{ where } i > 1 \text{ and } \varphi_{diff}(i) \in [-360°, 360°] \tag{57}$$

However, this may result in some incorrect values, for instance by $\varphi_{actual}(1)=355°$ and $\varphi_{actual}(2)=5°$ the calculation gives $\varphi_{diff}(2)=-350°$, but the correct value should be $\varphi_{diff}(2)=10°$. In other case $\varphi_{diff}(2)=350°$ is given by $\varphi_{actual}(1)=5°$ and $\varphi_{actual}(2)=355°$, but $\varphi_{diff}(2)=-10°$ should be the correct value. To resolve this phanomena, one more condition was added to the program:

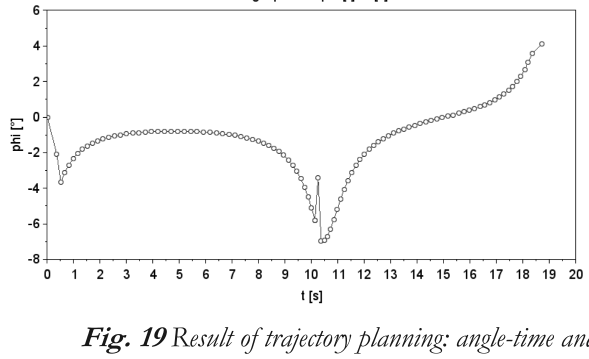
$$\varphi'_{diff}(i) = \begin{cases} \varphi'_{diff}(i) = \varphi_{diff}(i), \text{if } \varphi_{diff}(i) \in [-180°, 180°] \\ \varphi'_{diff}(i) = \varphi_{diff}(i) - 360°, \text{if } \varphi_{diff}(i) \in (180°, 360°] \\ \varphi'_{diff}(i) = \varphi_{diff}(i) + 360°, \text{if } \varphi_{diff}(i) \in [-360°, -180°) \end{cases}, \tag{58}$$

where finally $\varphi'_{diff}(i) \in [-180°, 180°]$.

Using the above mentioned, the angular velocity data for each waypoint now can be calculated as follow:

$$\omega(i) = \frac{\varphi_{diff}(i)}{t(i) - t(i-1)}, \text{ where } i > 1 \tag{59}$$

From the example shown in Fig. 12 and Fig. 13, the angle-time diagram can be seen in the left side of Fig. 19 and Fig. 20, while the angular velocity-time diagram can be seen in the right side of Fig. 19 and Fig. 20, respectively. As can be seen, there is no consistency between these figures and Fig. 17 and Fig. 18.



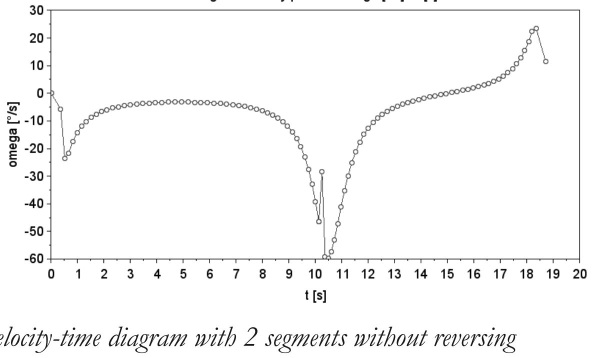
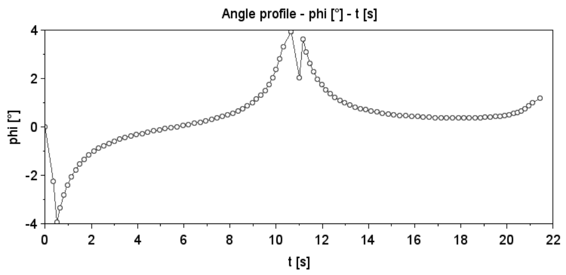


Fig. 19 *Result of trajectory planning: angle-time and angular velocity-time diagram with 2 segments without reversing*



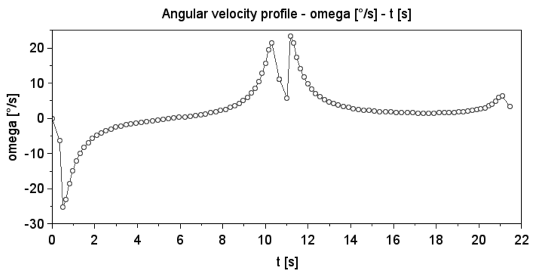


Fig. 20 *Result of trajectory planning: angle-time and angular velocity-time diagram with 2 segments with reversing*

6.5 Production of wheel velocitys during trajectory planning

In view of the velocity and angular velocity between the wheels of the forklift, it is already possible to determine the velocitys displayed on each wheel in the following way:

$$v_R(i) = v(i) - \frac{b}{2} \cdot \omega_{rad}(i) \tag{60}$$

$$v_L(i) = v(i) + \frac{b}{2} \cdot \omega_{rad}(i) \tag{61}$$

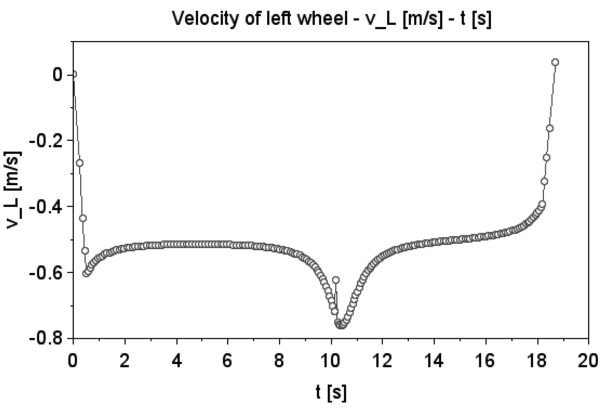
Where:

$v_R(i)$…Velocity of the centre of the right wheel,

$v_L(i)$…Velocity of the centre of the left wheel

b…Distance between the centre of each wheel and half of that radius for angular velocity

$\omega_{rad}(i)$…Angular velocity in radian.



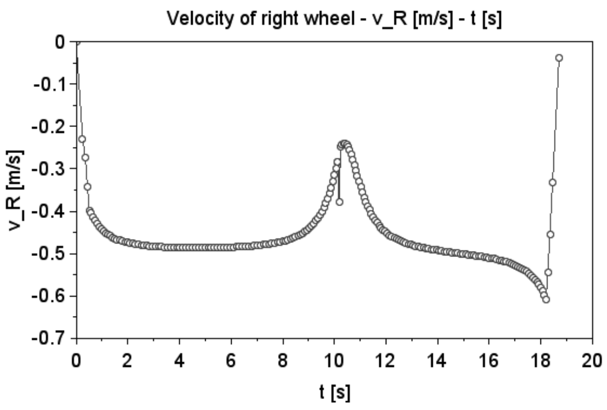
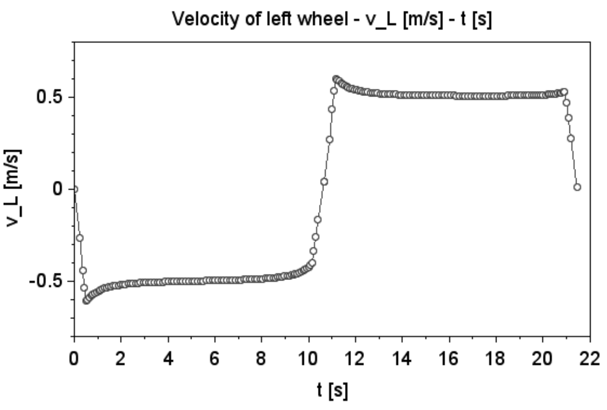


Fig. 21 *Result of trajectory planning: velocity-time diagrams for left and right wheel with 2 segments without reversing*

From the example shown in Fig. 12 and Fig. 13, the velocity-time diagram of the left and right wheel can be seen in the left and right side of Fig. 21 and Fig. 22, respectively.



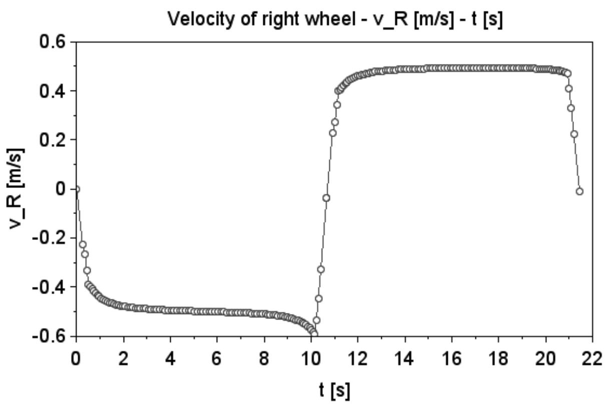


Fig. 22 *Result of trajectory planning: velocity-time diagrams for left and right wheel with 2 segments with reversing*

7 Conclusion

This paper presented the track and trajectory planning of a driverless carrier vehicle. The new algorithm, required for path planning, was implented after literature exploration using the Bezier-curve and Hermite-curve. The paper described the relations from these curve types, a general one from the lierature and a concrete one used for the vehicle. Due to the simulation and other problems, additional conditions have been included in the Scilab program. The optimisation basis for trajectory planning is the tending to minimum energy, which can be determined on the basis of electric current consumption. The definition of current consumption was included in previous papers, this paper covered the determination of the velocities on the wheels, from which the voltage to the DC motors and thus the current consumption can already be calculated. The paper presented two cases, taking into account approaching the targets, on the one hand, when the vehicle moves in the same direction and, on the other hand, when it moves by changing direction to the new target. The different curves obtained during the examination of the two cases are also illustrated in this paper.

Acknowledgement

The described article/presentation/study was carried out as part of the EFOP-3.6.1-16-2016-00011 "Younger and Renewing University – Innovative Knowledge City – institutional development of the University of Miskolc aiming at intelligent specialisation" project implemented in the framework of the Szechenyi 2020 program. The realization of this project is supported by the European Union, co-financed by the European Social Fund."

References

[1] MATYI, H., VERES, P., BANYAI, T., DEMIN, V., TAMAS, P. (2020). Digitalization in Industry 4.0: The Role of Mobile Devices, *Journal of Production Engineering*, Vol. 23, No. 1, pp. 75-78.

[2] SUJOVÁ, E., VYSLOUŽILOVÁ, D., ČIERNA, H., BAMBURA, R. (2020). Simulation Models of Production Plants as a Tool for Implementation of the Digital Twin Concept into Production, *Manufacturing Technology Journal*

[3] SUJOVÁ, E., STŘIHAVKOVÁ, E., ČIERNA, H. (2018). An Analysis of the Assembly Line Modernization by Using Simulation Software, *Manufacturing Technology*, Vol. 18, No. 5, pp. 839-845.

[4] TAMÁS, P. (2017). Examining the Possibilities for Efficiency Improvement of SMED Method Using Simulation Modelling, *Manufacturing Technology*, Vol. 17, No. 4, pp. 592-597.

[5] ULEWICZ, R., MAZUR, M. (2019). Economic Aspects of Robotization of Production Processes by Example of a Car Semi-trailers Manufacturer, *Manufacturing Technology*, Vol. 19, No. 6, pp. 1054-1059.

[6] RÓNAI, L., SZABÓ, T. (2020). Snap-fit Assembly Process with Industrial Robot Including Force Feedback, *Robotica*, Vol. 38, No. 2, pp. 317-336

[7] NOVAK-MARCINCIN J, JANAK M, TAKAC D. (2014). Computer Design of Robot ABB IRB 140 Transport System from Manufacturing Point of View, *Manufacturing Technology*, Vol. 14, No. 1, pp. 79-84.

[8] TAMÁS, P., BÁNYAI, T., ILLÉS, B., TOLLÁR, S., VERES, P., CSERVENÁK, Á. (2020). Hardai, I., Skapinyecz, R.: Development Possibilities of the High-tech Logistics Laboratory Established at the Institute of Logistics of the University of Miskolc, *Journal of Engineering Research and Reports*, Vol 13, No. 3, pp. 60-68.

[9] CSERVENÁK, Á. (March 2021). Simulation of a mobile robot's motion, *Academic Journal of Manufacturing Engineering*, accepted, under publication

[10] CSERVENÁK, Á. (December 2020). Simulation and modelling of a DC motor used in a mobile robot, *Academic Journal of Manufacturing Engineering*, accepted, under publication

[11] GASPARETTO, A., BOSCARIOL, P., LANZUTTI, A., & VIDONI, R. (2015). *Motion and Operation Planning of Robotic Systems*. Springer

[12] GHAFIL, H. M., JÁRMAI, K. (2020). *Optimization for Robot Modelling with MATLAB*, Springer Nature Switzerland AG

[13] RAJA, P., PUGAZHENTHI, S. (2012). Optimal path planning of mobile robots: A review, *International Journal of Physical Sciences* Vol. 7, No. 9, pp. 1314 – 1320.

[14] LOZANO-PEREZ, T. (1983). Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, Vol. 100, No. 2, pp. 108 – 120.

[15] TAKAHASHI, O., SCHILLING, R. J. (1989). Motion planning in a plane using generalized Voronoi diagrams. *IEEE Transactions on Robotics and Automation*, Vol. 5, No. 2, pp. 143 – 150.

[16] JANCHIV, A., BATSAIKHAN, D., KIM, B., LEE, W.G., LEE, S.-G (2013). Time-efficient and complete coverage path planning based on flow networks for multi-robots, *International Journal of Control, Automation and Systems*, Vol. 11, No. 2, pp. 369-376.

[17] ZHU, D., LATOMBE, J.-C. (1991). New Heuristic Algorithms for Efficient Hierarchical Path Planning, *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 1, pp. 9-20.

[18] MIN, H., LIN, Y., WANG, S., WU, F., SHEN, X. (2015). Path planning of mobile robot by mixing experience with modified artificial potential field method, *Advances in Mechanical Engineering*, Vol. 7, No. 11

[19] FEDELE, G., D'ALFONSO, L., CHIARAVALLOTI, F., D'AQUILA, G. (2018). Obstacles Avoidance Based on Switching Potential Functions, *Journal of Intelligent and Robotic Systems: Theory and Applications*, Vol. 90, No. 3-4, pp. 387 - 405.

[20] LI, C., WANG, F., ZHAO, L., LI, Y., SONG, Y. (2013). An improved chaotic motion path planner for autonomous mobile robots based on a logistic map regular paper, *International Journal of Advanced Robotic Systems*, Vol. 10

[21] BOHLIN, R., KAVRAKI, L.E. (2000). Path planning using Lazy PRM, *Proceedings-IEEE International Conference on Robotics and Automation*, Vol. 1, pp. 521 - 528.

[22] GASPARETTO, A., ZANOTTO, V. (2007). A new method for smooth trajectory planning of robot manipulators, *Mechanism and Machine Theory*, Vol. 42, pp. 455 – 471.

[23] BOBROW, J.E., DUBOWSKY, S., GIBSON, J.S. (1985). Time-optimal control of robotic manipulators along specified paths, *International Journal of Robotics Research*, Vol. 4, No. 3, pp. 554 – 561.

[24] SHIN, K.G., MCKAY, N.D. (1985). Minimum-time control of robotic manipulators with geometric path constraints, *IEEE Transactions on Automatic Control*, Vol. 30, No. 6, pp. 531–541.

[25] LEE, J. (1995). A Dynamic Programming Approach to Near Minimum-Time Trajectory Planning for Two Robots, *IEEE Transactions on Robotics and Automation*, Vol. 11, No. 1, pp. 160-164.

[26] CONSTANTINESCU, D. (1998). Smooth time optimal trajectory planning for industrial manipulators, *PhD disszertáció*, The University of British Columbia

[27] SHILLER, Z. (1996). Time-energy optimal control of articulated systems with geometric path constraints, *Journal of Dynamic Systems, Measurement, and Control*, Vol. 118, pp. 139 - 143.

[28] JOONYOUNG, K., SUNG-RAK, K., SOO-JONG, K., DONG-HYEOK, K. (2010). A practical approach for minimum-time trajectory planning for industrial robots, *Industrial Robots: An International Journal*, Vol. 37, No. 1, pp. 51 - 61.

[29] RUBIO, F., VALERO, F., SUNYER, J., CUADRADO, J. (2012). Optimal time trajectories for industrial robots with torque, power, jerk and energy consumed constraints, *Industrial Robot: An International Journal*, Vol. 39, No. 1, pp. 92 – 100.

[30] WANG, C.H., HORNG, J.G. (1990). Constrained minimum-time path planning for robot manipulators via virtual knots of the cubic B-Spline functions, *IEEE Transactions on Automatic Control*, Vol. 35, No. 5, pp. 573-577.

[31] LIN, C.S., CHANG, P.R., LUH, J.Y.S. (1983). Formulation and optimization of cubic polynomial joint trajectories for industrial robots, *IEEE Transactions on Automatic Control*, Vol. 28, No. 12, pp. 1066-1073.

[32] WEI, Y., KIM, D. (2014). Reliable and energy-efficient routing protocol for underwater acoustic sensor networks, *2014 International Conference on Information and Communication Technology Convergence (ICTC)*, Busan, pp. 738-743., doi: 10.1109/ICTC.2014.6983274.

[33] KIM, H., KIM, B.K. (2014). Online minimum-energy trajectory planning and control on a straight-line path for three-wheeled omnidirectional mobile robots, *IEEE Transactions on Industrial Electronics*, Vol. 61, No. 9, pp. 4771-4779.

[34] SARAMAGO S.F.P., STEFFEN JR. V. (1999). Dynamic Optimization for the Trajectory Planning of Robot Manipulators in the Presence of Obstacles, *Jorunal of the Brazilian Society of Mechanical Sciences*, Vol. 21, No. 3, pp. 1-12.

[35] SARAMAGO, S.F.P., JR. STEFFEN. V. (2000). Optimal trajectory planning of robot manipulators in the presence of moving obstacles, *Mechanism and Machine Theory*, Vol. 35, No. 8, pp. 1079-1094.

[36] KIM, H., KIM, B. K. (2012). Minimum-Energy Trajectory Planning and Control on a Straight Line with Rotation for Three-Wheeled Omni-Directional Mobile Robots, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura, Algarve, Portugal

[37] KIM, H., KIM, B. K. (2017). Minimum-energy Cornering Trajectory Planning with Self-rotation for Three-wheeled Omni-directional Mobile Robots, *International Journal of Control, Automation and Systems*, Vol. 15, pp. 1-10.

[38] PEIDRÓ, A., GALLEGO, J., PAYÁ, L., MARÍN, J. M., REINOSO, Ó. (2019). Trajectory Analysis for the MASAR: A New Modular and Single-Actuator Robot, *Robotics*, MDPI, Vol 8., No. 78

[39] SHRUTHI, C. M., SUDHEER, A. P., JOY, M. L. (2019). Optimal crossing and control of mobile dualarm robot through tension towers by using fuzzy and Newton barrier method, *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, Vol. 41, No. 245, pp. 1-25.

[40] NAZEMIZADEH, M., RAHIMI, H.N. AMINI KHOIY, K. (2012). Trajectory planning of mobile robots using indirect solution of optimal control method in generalized point-to-point task, *Frontiers of Mechanical Engineering*, Vol. 7, No. 1, pp. 23–28.

[41] GRACIA, L., TORNERO, J. (2008). Optimal Trajectory Planning for Wheeled Mobile Robots Based on Kinematics Singularity. *Journal of Intelligent Robot Systems*, Vol. 53, pp. 145–168.

[42] SUH, J., GONG, J., OH, S. (2017). Fast Sampling-Based Cost-Aware Path Planning With Nonmyopic Extensions Using Cross Entropy, *IEEE Transactions on Robotics*, Vol. 33, No. 6, pp. 1313-1326.

[43] ZHU, Y., JIN., B., LI, S (2014). Optimal design of hexapod walking robot leg structure based on energy consumption and workspace, *Transactions of the Canadian Society for Mechanical Engineering*, Vol. 38, No. 3, pp. 305-317.

[44] LOUSTE C., LIÉGEOIS, A. (2002). Path planning for non-holonomic vehicles: a potential viscous fluid field method, *Robotica*, Vol. 20, pp 291-298.

[45] YAN, Y., MOSTOFI, Y. (2014). To Go or Not to Go: On Energy-Aware and Communication-Aware Robotic Operation, *IEEE Transactions on Control of Network Systems*, Vol. 1, No. 3, pp. 218-231.

[46] CAPI, G., NASU, Y., BAROLLI, L., MITOBE, K., TAKEDA, K. (2001). Application of Genetic Algorithms for biped robot gait synthesis optimization during walking and going up-stairs, *Advanced Robotics*, Vol. 15, No. 6, pp. 675-694.

[47] TOKEKAR, P., KARNAD, N., ISLER, V. (2014). Energy-optimal trajectory planning for car-like robots, *Auton Robot*, Vol. 37, pp. 279–300.

[48] LIU, S., SUN, D. (2014). Minimizing Energy Consumption of Wheeled Mobile Robots via Optimal Motion Planning, *IEEE/ASME Transactions on Mechatronics*, Vol. 19, No. 2, pp. 401-411.

[49] BHATTACHARYA, S., AGRAWAL, S. K. (2000). Spherical rolling robot: a design and motion planning studies, *IEEE Transactions on Robotics and Automation*, Vol. 16, No. 6, pp. 835-839.

[50] PIAZZI, A., VISIOLI, A. (2000). Global minimum-jerk trajectory planning of robot manipulators, *IEEE Transactions on Industrial Electronics*, Vol. 47, No. 1, pp. 140-149.

[51] PAPP, Á., SZILASSY, L., & SÁROSI, J. (2016). Navigation of differential drive mobile robot on predefined, software designed path. *Recent Innovations in Mechatronics* (RIiM), Vol. 3, No. 1-2.

[52] CHEN, C. (2009). Numerical Methods, *6. Interpolation and Approximation, educational material*, Department of Computer Science, National Tsing Hua University

[53] VELICHOVÁ, D. (2012). Constructive Geometry B – Lectures, *6. Approximation and Interpolation Curves, educational material*, Slovak Technical University, Bratislava

[54] DEUFLHARD, P., HOHMANN, A. (2003). *Numerical Analysis in Modern Scientific Computing*, Springer-Verlag New York

[55] SALOMON, D. (2005). *Curves and Surfaces for Computer Graphics*, Springer-Verlag, Berlin, Heidelberg

[56] HEATH, M. T, MUNSON, E. M. (1996). *Scientific Computing: An Introductory Survey*, McGraw-Hill Higher Education

[57] LYCHE, T., MØRKEN, K. (2008). *Spline Methods*, Draft, Department of Informatics, Centre of Mathematics for Applications, University of Oslo

[58] JAKLIC, G., KOZAK, J., KRAJNC, M., VITRIH, V., ZAGAR, E. (2012). Hermite geometric interpolation by rational Bézier spatial curves, *Society for Industrial and Applied Mathematics Journal on Numerical Analysis*, Vol. 50, No. 5, pp. 2695–2715.

[59] MAHAFFY, J. M. (2018). *Math 541 - Numerical Analysis Interpolation and Polynomial Approximation — Piecewise Polynomial Approximation; Cubic Splines*, presentation, Department of Mathematics and Statistics Dynamical Systems Group Computational Sciences Research Center San Diego State University

[60] BARBIC, J. (2019). *CSCI 420 Computer Graphics Lecture 9*, Splines, presentation, University of Southern California

[61] HASHEMI-DEHKORDI, S., VALENTINI, P.P. (2014). Comparison between Bezier and Hermite cubic interpolants in elastic spline formulations. *Acta Mech*, Vol. 225, pp. 1809–1821.

[62] MANDZÁROS, J. (2011). *Numerical Methods, II. Approximations of functions*, educational material, Department of Applied Mathematics, University of Miskolc

[63] LI, H. (2014). CSCI 420 *Computer Graphics*, 4.2 Splines, presentation, University of Southern California

[64] LENSCH, H. (2007). *Computer Graphics*, Splines, presentation, University of Tübingen

[65] FORSYTH, D. A. (2013). *Interpolating Curves*, presentation, Grainger College of Engineering Computer Science, University of Illinois

[66] EREN, H., FUNG, C., EVANS, J. (1999). Implementation of the spline method for mobile robot path control, *Conference Record - IEEE Instrumentation and Measurement Technology Conference* 2., Vol. 2., pp. 739 – 744.