# Path-Following Control for Autonomous Trucks: A Pure Pursuit Approach for Autonomous Maneuvering

Adwaith Gopichand

***Abstract*–This paper presents an in-depth exploration of a novel pure pursuit path-following control strategy for autonomous trucks. The primary aim is to enhance autonomous manoeuvring in confined environments, such as distribution centres. In particular, this study has two main contributions. Firstly, a detailed examination of the pure pursuit controller is presented, emphasizing its working principles and a comprehensive geometric interpretation of the algorithm. We also establish a kinematic model that provides a more comprehensive understanding of the vehicle's dynamics. Lastly, we conduct simulations and real-world experiments to validate the efficacy of our proposed control strategy. The results show a significant improvement in path-following precision and stability, suggesting that our pure pursuit control algorithm has significant potential in real-world autonomous truck applications. Thus, this study underpins the potential of the pure pursuit approach in enhancing the reliability and safety of autonomous truck navigation, paving the way for further research in this area.**

***Index Terms*–Autonomous vehicles, Path-tracking, Lateral controller, Pure Pursuit algorithm, Distribution centre**

## I. Introduction

As the dynamics of the automotive and transportation industries continue to evolve, a paradigm shift is taking place. What was once an industry driven solely by mechanical systems has now been merged into electrical systems, software, and mechatronics, paving the way for sophisticated and highly integrated solutions. The ongoing multidisciplinary evolution is driving a significant transition, leading us into an era dominated by autonomous systems. Autonomous systems redefine the driving experience and greatly enhance safety, efficiency, and sustainability. As such, they have garnered substantial interest from leading companies such as Waymo, Amazon, and Uber, who are pioneering the development and public testing of autonomous vehicle technologies [3]. However, the application of autonomous systems is not only confined to public roads. The increasing necessity for sustainable and efficient logistics and freight transport systems has triggered significant research interest in Autonomous Driving Trucks (ADTs), especially in controlled environments like distribution centres.

In contrast to public roads riddled with intricate technical and legal challenges, logistic warehouses present a more suitable environment for adopting autonomous systems [9]. They provide an optimal setting, free from the unpredictable elements typically encountered on public roads. The controlled, low-speed operations within the distribution centre, coupled with a well-defined layout and short stopping distances, make it an optimal setting for the safe and efficient deployment of autonomous vehicles.


Fig. 1: TruckLab Environment in the TU/e Automotive Lab.

The main challenge with autonomous vehicles is guaranteeing that the vehicle can accurately track and follow a preset course, executing the movement strategy devised in the planning module. This is essentially the objective of lateral control, which must choose the necessary steering angle to correct any mounting errors and adapt to any changes in the direction of the path as they emerge [5]. In order to construct the lateral controller, the discrepancy between the actual vehicle location and the optimal path coordinates needs to be identified. Subsequently, choosing a control design strategy that mitigates errors to the slightest possible degree and duly adheres to the steering angle limitations becomes essential.

There are two categories of lateral control design. The initial category comprises geometric controllers. These controllers utilize the principles of geometry, working with the specified path coordinates and the vehicle's kinematic models. Notable examples of geometric controllers include the Pure Pursuit and Stanley controllers, which leverage these principles to establish vehicle control [1]. The second category, dynamic controllers, adopts a more advanced approach. The model predictive controller (MPC), an example in this category, employs finite horizon optimization to ascertain the most effective control command, considering the prevailing errors and potential future disturbances, offering a compre-

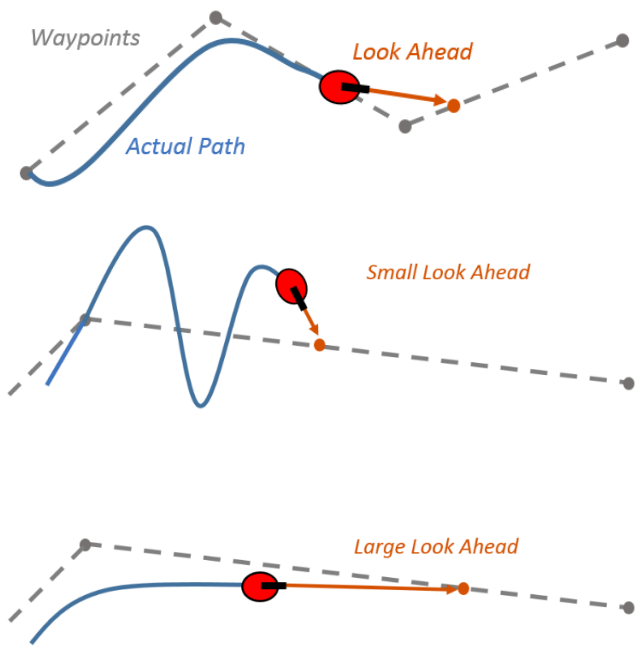


Fig. 2: Effect of varying look ahead distance on pure pursuit path tracking.

hensive and anticipatory control strategy [6].

This paper will focus on implementing the pure pursuit controller as an effective method of lateral control for Autonomous Driving Trucks (ADTs) in a distribution centre. Furthermore, it is essential to highlight that the development of this controller was done within TruckLab, which is a part of the TU/e Automotive lab, and is shown in Fig. 1.

*Related Literature:* The foundation of pure pursuit controller is grounded in the study of geometry and vehicle kinematics. An early example is the work by Coulter (1992), who introduced the Pure Pursuit path-tracking algorithm for autonomous vehicles [4]. The author proposed a simple, geometrically intuitive approach to path tracking, which has since become the foundation for many subsequent studies. Another noteworthy contribution is by Snider (2009), who extended Coulter's model by incorporating the dynamics and kinematics of different types of vehicles, thus enhancing the adaptability of the pure pursuit controller [8]. In terms of practical applications, pure pursuit controller has been implemented in various settings. For instance, Varundev Sukhil et al. applied the controller to an autonomous vehicle operating in a controlled environment, demonstrating its effectiveness in achieving accurate path tracking [10]. Similarly, Hans Andersen et al. successfully used the pure pursuit controller for an autonomous vehicle navigating in an autonomous golf cart through complex pedestrian environments, underlining its robustness [2]. In the context of Autonomous Driving Trucks (ADTs), studies have begun to explore the application of the pure pursuit controller. For instance, Kresimir Petrinec et al. implemented this controller in an autonomous forklift in a warehouse environment, providing empirical evidence of its viability for ADTs [7].

*Organization:* The remainder of this paper is structured as follows: Section II discusses the fundamental principles and operational mechanics of the pure pursuit controller. This section further elaborates on implementing the pure pursuit controller, detailing the step-by-step process, including controller design and simulation setup. Section III presents the experimental results, focusing on the performance of the pure pursuit controller in terms of accuracy and adaptability. Conclusions are drawn in Section IV, together with an outlook on future research.

## II. Methodology

This section will focus on four crucial aspects: The working principle of the pure pursuit controller, the kinematic vehicle modelling, the geometric interpretation of the pure pursuit control algorithm, followed by the design of the Controller, and the configuration of the simulation setup. These topics will offer a deeper understanding of the controller's operation, underlying principles, and integration with the vehicle's physical attributes.

### A. *Pure pursuit controller*

The primary functionality of this controller revolves around its ability to follow a reference path, relying solely on the vehicle's kinematic geometry and the characteristics of the desired path. In doing so, it simplifies the complexity of real-world vehicle dynamics by disregarding dynamic forces exerted on the vehicle. This assumption is primarily based on the premise that wheels maintain a no-slip condition, effectively adhering to the contact surface without skidding. The controller operates by continuously adjusting the vehicle's steering angle to ensure it is always directed towards a "goal point" on the reference path. This point is also known as the "Look Ahead" point. The algorithm continually shifts the look-ahead point along the path based on the vehicle's current position and continues until the vehicle reaches the final point on the path. The positioning of this Look Ahead point is determined by the "Look Ahead Distance" parameter, which dictates the distance between the vehicle and the look-ahead point and this is the main tuning parameter of the controller [11]. As shown in Fig. 2., the path is oscillatory and accurate for a smaller look-ahead distance and, for a larger look-ahead distance, the path is less oscillatory, but the tracking is poor.

### B. *Kinematic Bicycle model*

The proposed analysis uses a simplified bicycle model, shown in Fig. 3., to represent the vehicle, where the vehicle's front and rear wheels are each represented by a single wheel. This model provides a useful tool for exploring the vehicle's kinematics. Within this model, we establish a reference point, denoted as X, Y, which can be variably positioned on the vehicle. This point can be positioned at various locations,

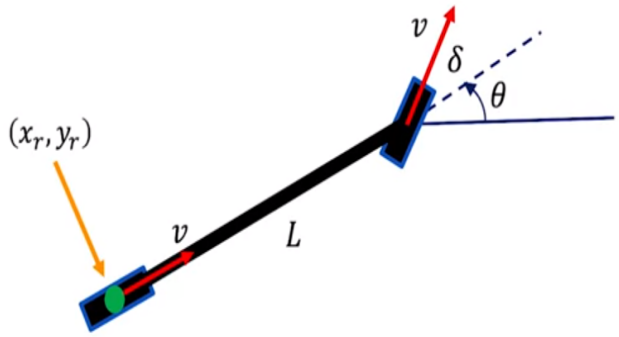


Fig. 3: Rear axle kinematic bicycle model.

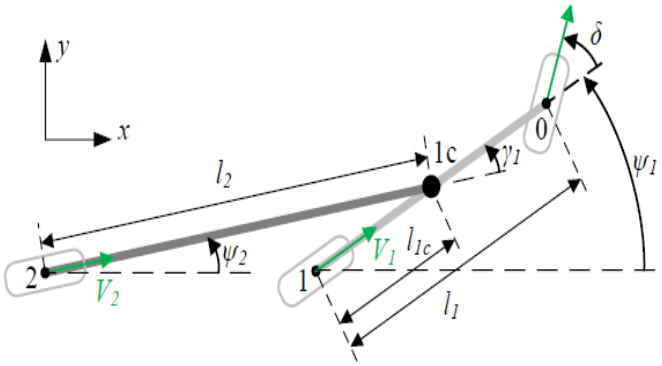


Fig. 4: Tractor semi-trailer kinematic model.

such as at the centre of the rear axle, the centre of the front axle, or the centre of gravity, often referred to as 'cg'. For the pure pursuit controller, the centre of the rear axle is selected as the reference point, denoted by '$x_r$' and '$y_r$'. '$\theta$' refers to the heading angle. Another key parameter is the wheelbase, denoted as '$L$', which is the distance between the front and rear axles. The steering angle, denoted as '$\delta$', is measured in relation to the forward direction of the bicycle model. The velocity '$v$' corresponds with the direction of each wheel, adhering to the 'no slip' condition. From the no-slip condition,

$$\dot{\theta} = \omega = \frac{v}{R}, \tag{1}$$

where $\omega$ is the bicycle's rate of rotation. From the similarity of triangles formed by R and L, and $\delta$ and $v$,

$$tan(\delta) = \frac{L}{R}, \tag{2}$$

From (1) and (2),

$$\dot{\theta} = \omega = \frac{v}{R} = \frac{v \cdot tan(\delta)}{L}, \tag{3}$$

From this configuration, the components of velocity for the reference point in the x and y directions are,

$$\dot{x_r} = v \cdot cos(\theta), \tag{4}$$

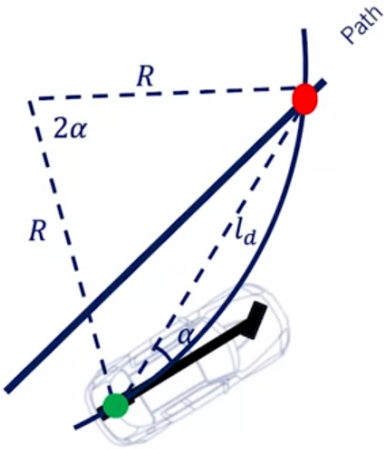


Fig. 5: Pure Pursuit geometry indicating the look ahead point and look ahead distance.

$$\dot{y_r} = v \cdot sin(\theta), \tag{5}$$

These equations, along with the equation for the bicycle's rate of rotation derived earlier, collectively constitute the rear axle bicycle model and is shown below,

$$\dot{\theta} = \frac{v \cdot tan(\delta)}{L}, \tag{6}$$

This model provides a concise mathematical representation of the vehicle's motion, which is crucial for designing the path-following controller.

For implementing the pure pursuit controller for ADTs, this research considers the dynamics of a Tractor Semi-Trailer model shown in Fig. 4. However, the existence of three non-steered trailer axles adds complexity. During a turning manoeuvre, side slip angles inevitably manifest on multiple trailer axles, irrespective of the speed. The consequential forces may induce side slip angles on the tractor as well. In the context of low-speed manoeuvring, it is reasonably assumed that no side slip angle occurs on the tractor and the second axle of the semi-trailer. Under these circumstances, the first and third trailer axles are disregarded for simplicity. One significant challenge with the tractor-semi-trailer configuration is the occurrence of jackknifing. In this situation, the articulation angle between the tractor and the semi-trailer becomes excessively large, potentially leading to a collision between the tractor cabin and the semi-trailer. Constraints are set on the maximum steering angle to avoid this scenario, especially during forward motion.

### C. *Geometric interpretation of pure pursuit algorithm*

For the Pure Pursuit control strategy, the key reference point on the vehicle is the centre of the rear axle. The line connecting this reference point to the target point on the planned path is defined as $l_d$, commonly referred to as the look-ahead distance, represented by the blue dashed line

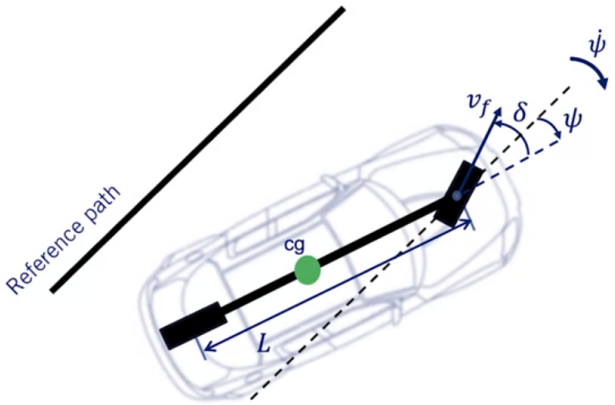


Fig. 6: Bicycle model showing the vehicle heading angle '$\psi$'.

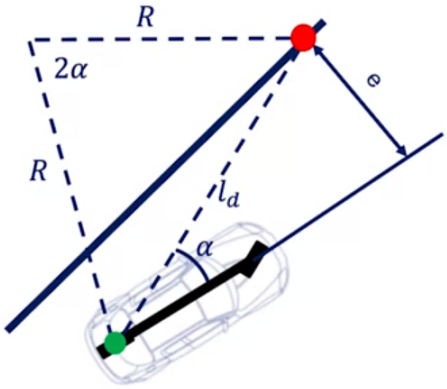


Fig. 7: Bicycle model showing the vehicle cross track error '$e$'.

in Fig. 5. The angle '$\alpha$' represents the angle between the heading of the vehicle and the look-ahead line. The interpretation of Pure Pursuit control from a geometric perspective relies on applying the instantaneous centre of rotation (ICR) principle. The ICR, the centre of the rear axle, and the target point on the path together form a triangle with two sides of length R (radius of the ICR circle) and one side of length '$l_d$' (the look-ahead distance). The arc the vehicle needs to follow to reach the target point from its current reference point can be defined within this triangle. This arc lies on the circumference of the ICR circle and spans an angle of $2\alpha$. By utilizing fundamental trigonometric identities and applying the law of sines, the angle $2\alpha$ can be obtained as shown below,

$$\frac{l_d}{sin(2\alpha)} = \frac{R}{sin(\frac{\pi}{2} - \alpha)}, \tag{7}$$

$$\frac{l_d}{2 \cdot sin(\alpha) \cdot cos(\alpha)} = \frac{R}{cos(\alpha)}, \tag{8}$$

$$\frac{l_d}{sin(\alpha)} = 2 \cdot R, \tag{9}$$

$$\kappa = \frac{1}{R} = \frac{2 \cdot sin(\alpha)}{l_d}, \tag{10}$$

where $\kappa$ is the curvature of the circular arc, which is the inverse of the radius of the ICR circle (R). Combining (2) and (10), the steering angle needed to track the arc is given by,

$$\delta = \tan^{-1}\left(\frac{2 \cdot L \cdot sin(\alpha)}{l_d}\right). \tag{11}$$

### D. *Pure pursuit algorithm- error dynamics*

The controller ensures that the autonomous truck aligns precisely with the intended route, minimizing deviations. The variable '$\psi$', shown in Fig. 6. represents the relative heading angle of the truck in relation to the desired path. Two critical types of errors come into play in this scenario: heading and cross-track errors. Heading error, the discrepancy between the intended path direction and the truck's actual heading, is a primary indicator of how well the vehicle aligns and moves in the direction of the desired path. The rate of change of the heading error, denoted as '$\dot{\psi}$', provides insight into the evolution of the heading error over time. The equation is shown below,

$$\text{Rate of heading error} = \dot{\psi}_{des}(t) - \dot{\psi}(t), \tag{12}$$

where $\dot{\psi}_{des}(t)$ is the desired rate of change of heading angle. The second form of error, the cross-track error, represents the displacement between the vehicle's reference point and the nearest point on the planned path. This is shown in Fig. 7. A flawless path-tracking mechanism would require the heading and cross-track errors to be reduced to zero. In the context of the pure pursuit controller for an autonomous truck, the cross-track error is the gap between the truck's directional vector and the target point on its path, represented with the symbol '$e$'. The following equation is obtained,

$$sin(\alpha) = \frac{e}{l_d}. \tag{13}$$

Integrating this with the previously derived expression for curvature, from (10) and (13),

$$\kappa = \frac{2}{l_d^2} \cdot e. \tag{14}$$

Equation (14) indicates that the path's curvature dictated by the pure pursuit controller is directly proportional to the cross-track error at the look-ahead point on the path.

### E. *Pure pursuit controller design and simulation setup*

A kinematic vehicle model of a tractor-semitrailer truck was provided in MATLAB/Simulink, enabling the simulation of low-speed manoeuvring. The model allows for the prescription of velocity and steering as a function of distance, facilitating the computation of the resulting vehicle motion. In the pure pursuit controller, critical parameters

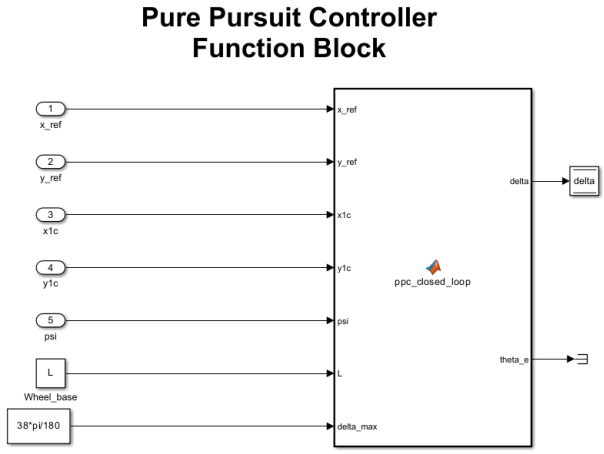


Fig. 8: Pure pursuit controller function block implemented in Simulink.

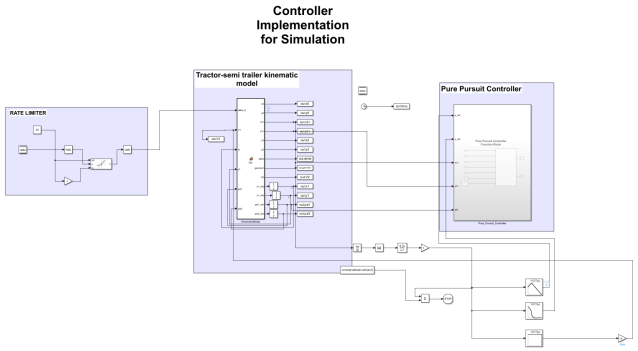


Fig. 9: An overview of the kinematic tractor-semitrailer Simulink block paired with its controlling entity. The system also features the integration of a rate limiter block to prevent jackknifing.

derived from the kinematic model, including the rear axle coordinates (x1, y1), the truck's current position (x1c, y1c), and the heading angle of the truck '$\psi$', serve as the input variables. The controller's function block shown in Fig. 8., processes these inputs and outputs the required steering angle accurately guide the truck along the planned path. The system diagram can be visualized in Fig. 9. A series of 1-D lookup tables implemented in Simulink handle the dynamic update of waypoints as a function of the distance travelled. The simulation is designed to halt once the vehicle reaches or is within a set proximity threshold of the last point in the waypoint.

Upon successfully validating the pure pursuit controller using the provided kinematic model, the next stage involves its integration and validation within a 3D virtual environment called the Virtual Entity. This environment has been designed utilizing the Unity Game Engine. The practical application of the controller, translated into MATLAB/Simulink blocks for driving the tractor within the Virtual Entity, is detailed in Fig. 10. For a more comprehensive understanding of the controller's operational code used in the virtual simulation,

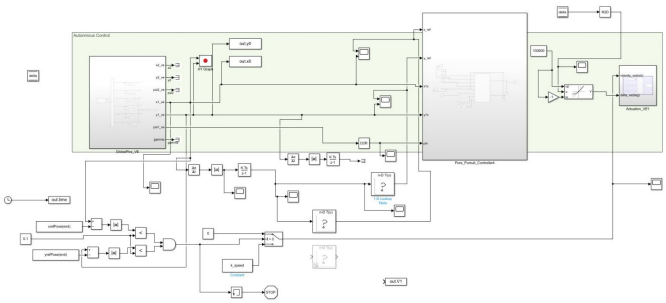

Fig. 10: Depicts the integration of the pure pursuit controller with the Virtual Entity blocks.

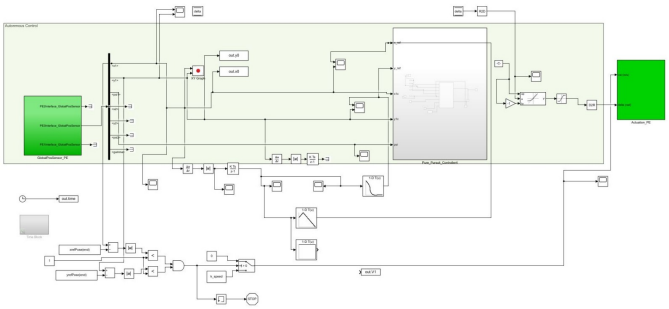

Fig. 11: Depicts the integration of the pure pursuit controller with the Physical Entity blocks.

readers are referred to Appendix B. This approach ensures a thorough validation process, extending from abstract model simulations to an immersive 3D environment, thus underlining the potential for practical deployment of the pure pursuit controller in real-world applications.

In the final stage of methodology, the pure pursuit controller is subjected to testing and validation using the AES Lab's physical truck. To facilitate this physical testing, the Virtual Entity blocks within the simulation setup are replaced with Physical Entity blocks. These new blocks interface with the global position sensor and actuator of the physical truck, ensuring a direct link between the controller and the hardware of the truck. This configuration is outlined in Fig. 11.

Through this multi-tiered approach, progressing from kinematic modelling, through a virtual environment, to actual physical testing, our methodology ensures a comprehensive validation of the pure pursuit controller, reinforcing its suitability for real-world ADT applications.

## III. Experimental Results

In this section, the key findings from the three-tiered simulation process are showcased, highlighting the performance of the pure pursuit controller under various simulation conditions. In the first stage, the kinematic model simulation, the plot shown in Fig. 12. demonstrates the truck's ability to almost precisely track the given path, which highlights the accuracy and reliability of the pure pursuit controller. An interesting observation from the plot pertains to a series of oscillations in the path. These oscillations can be attributed to

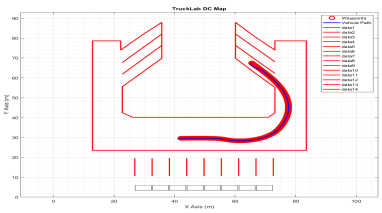


Fig. 12: Illustration of the TruckLab map.

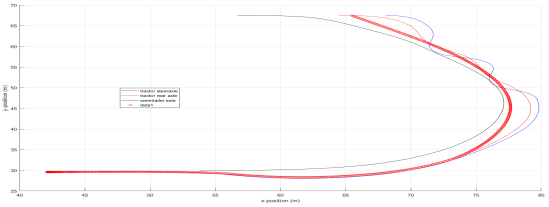

Fig. 13: Truck navigating through a set of waypoints.

the truck's starting position being proximate to the beginning of the waypoint. This proximity leads to the computation of a relatively small look-ahead distance, which subsequently introduces oscillations into the truck's trajectory. The TruckLab map is also provided in Fig. 13, illustrating the truck's path along the designated waypoints. This graphic representation affirms the practicality and feasibility of our controller when deployed in real-world distribution centre scenarios.

In the second stage, the findings from the virtual simulations are shown in Fig. 15 and Fig. 17. For these simulations, two calibrated path-following tasks were conducted involving the truck operating within a virtual representation of the distribution center. The first task involved navigating from the entrance of the distribution center to the docking area. The second task, involved navigating the truck as it made its way from the docking area back towards the exit of the distribution center. In both scenarios, shown in Fig. 14 and Fig.16, the simulation plots reveal that the truck maintained a high degree of accuracy in following the pre-defined path.

## IV. Conclusion

This study has significantly advanced our understanding of the potential and limitations of the pure pursuit controller in the context of Autonomous Driving Trucks (ADTs). By operating at a constant velocity, the controller can accurately follow a predetermined course, executing a planned movement strategy precisely. The controller's performance was evaluated across three scenarios: tractor semi-kinematic model simulation, virtual environment realized through Unity Game Engine, and, a real-world trial using AES Lab's physical truck. These tests underscore the controller's adaptability across multiple settings, highlighting its robustness and reliability. The pure pursuit controller has shown to be particularly effective in controlled environments like distribution centres, where its advantages can be fully realized.

This work calls for the following possible extensions: First, the static velocity model could limit the controller's adaptability to dynamic environments. Thus, a significant implication of our findings for practitioners and researchers alike is the potential performance enhancement that might be achievable through a variable velocity model. Second, an intriguing prospect would be to expand the controller's capability to include reverse motion. This development could significantly increase the versatility of ADTs, especially in scenarios where reverse movements are essential, such as rectifying path errors, and loading and unloading goods. However, implementing a reverse controller brings challenges, such as the risk of jackknifing. Therefore, a deeper investigation into the dynamics of the trailer would provide a comprehensive understanding of the factors influencing jackknifing, thereby informing the development of more sophisticated control strategies.

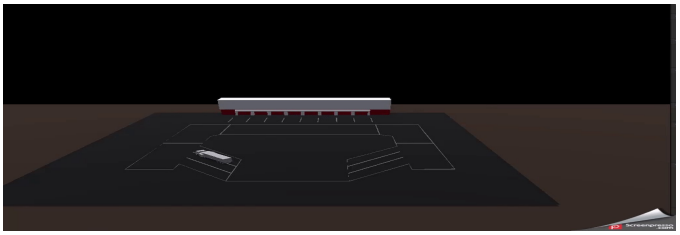

Fig. 14: Scenario 1: Truck located at the entrance.

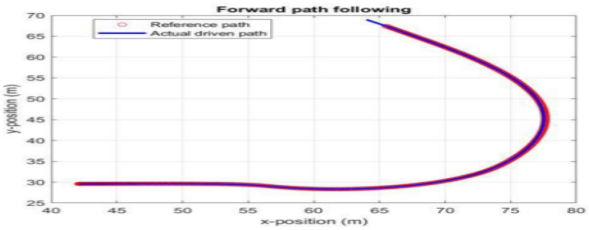


Fig. 15: Truck precisely following calibrated path 1.

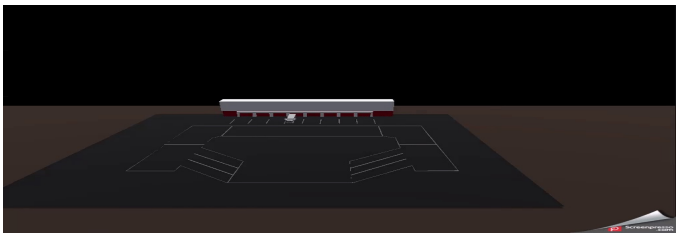

Fig. 16: Scenario 2: Truck located at the docking area.

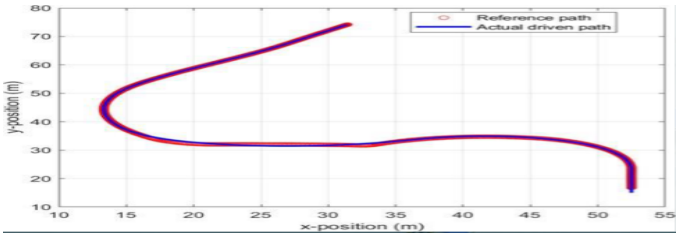


Fig. 17: Truck precisely following calibrated path 2.

### ACKNOWLEDGMENT

I am very grateful to Dr. Ion Barosan for proofreading this paper and countless papers and documents before this, teaching the authors something new every time. Moreover, this paper draws inspiration from "Guidelines for Writing Papers and Reports" by Dr. Mauro Salazar and Olaf Borsboom, which has taught me much about writing scientific papers.


## APPENDIX A
### MATLAB CODE FOR THE CONTROLLER FUNCTION

```
function [delta, theta_e]  =
    ppc_closed_loop(x_ref, y_ref, x1c,
    y1c, psi, L, delta_max)

%Obtaining the waypoint co-ordinates and
     the current position of the truck
waypoint = [x_ref, y_ref];
current_position = [x1c, y1c];

% Compute the desired heading angle
theta_d = atan2(waypoint(2) -
    current_position(2), waypoint(1) -
    current_position(1));

% Compute the heading error
theta_e = (theta_d - psi);

% Calculate the look ahead distance
ld = vecnorm(waypoint - repmat(
    current_position, size(waypoint, 1),
    1), 2, 2);

% Compute the steering angle based on
    the heading error and lookahead
    distance
delta = atan((2*L*sin(theta_e))/ld);
delta = sign(delta) * min(abs(delta),
    delta_max);

end
```

## APPENDIX B
### MATLAB CODE FOR RUNNING THE SIMULATION

```
clear all; close all;

TractorSemitrailer_parameters;   % load
                    % vehicle parameters

tmax= 100;      % simulation time [s]
fs  = 10;       % sample frequency for
                % data storage [Hz]
freq = fs;

%% Load Trajectory
path = load('SampleTrajTemp.mat') ;

%%
L = 3.6;        %Wheel base

%% Determine speed profile
xrefPose = path.m.fwd_x1c;
yrefPose = path.m.fwd_y1c;

dx = diff(xrefPose);
dy = diff(yrefPose);

cumulativeDistanceX = [0; cumsum(abs(dx)
    )];
cumulativeDistanceY = [0; cumsum(abs(dy)
    )];

k_speed = 0.1; % Gain factor for speed
v = k_speed*ones(size(
    cumulativeDistanceX,1),1);
v = [v(1:end-1); 0];

figure(99)
plot(cumulativeDistanceX,v,'black','
    LineWidth',1.5)
grid on
xlabel('Distance')
ylabel('Velocity')

%% For running the Simulink Simulation
tic
s=sim('control_temp_kin.slx');
toc

%% Plot
waypoints = [xrefPose, yrefPose];
relative_waypoints = waypoints;
figure
hold on
plot(s.x0,s.y0,'b') % Steer axle
    position
plot(s.x1,s.y1,'r') % Drive axle
    position
plot(s.x2,s.y2,'k') % Trailer axle
    position
legend('tractor steeraxle','tractor rear
     axle','semitrailer axle','Location',
    'best')
xlabel('x-position (m)')
ylabel('y-position (m)')
plot(relative_waypoints(:, 1),
    relative_waypoints(:, 2), 'ro', '
    MarkerSize', 5, 'LineWidth', 0.1);
grid on;
```

## REFERENCES


[1] Omead Amidi and Chuck E Thorpe. Integrated mobile robot control. In *Mobile Robots V*, volume 1388, pages 504–523. SPIE, 1991.

[2] Hans Andersen, Zhuang Jie Chong, You Hong Eng, Scott Pendleton, and Marcelo H Ang. Geometric path tracking algorithm for autonomous driving in pedestrian environment. In *2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1669–1674. IEEE, 2016.

[3] Ion Barosan, Arash Arjmandi Basmenj, Sudhanshu GR Chouhan, and David Manrique. Development of a virtual simulation environment and a digital twin of an autonomous driving truck for a distribution center. In *Software Architecture: 14th European Conference, ECSA 2020 Tracks and Workshops, L'Aquila, Italy, September 14–18, 2020, Proceedings 14*, pages 542–557. Springer, 2020.

[4] R Craig Conlter. Implementation of the pure pursuit path'hcking algorithm. *Camegie Mellon University*, 1992.

[5] Jingjing Jiang and Alessandro Astolfi. Lateral control of an autonomous vehicle. *IEEE Transactions on Intelligent Vehicles*, 3(2):228–237, 2018.

[6] A Ollero and O Amidi. Predictive path tracking of mobile robots. application to the cmu navlab. In *Proceedings of 5th International Conference on Advanced Robotics, Robots in Unstructured Environments, ICAR*, volume 91, pages 1081–1086, 1991.

[7] Krešimir Petrinec, Zdenko Kovacic, and Alessandro Marozin. Simulator of multi-agv robotic industrial environments. In *IEEE International Conference on Industrial Technology, 2003*, volume 2, pages 979–983. IEEE, 2003.

[8] Jarrod M Snider et al. Automatic steering methods for autonomous automobile path tracking. *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RITR-09-08*, 2009.

[9] Qunying Song, Emelie Engström, and Per Runeson. Concepts in testing of autonomous systems: Academic literature and industry practice. In *2021 IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN)*, pages 74–81. IEEE, 2021.

[10] Varundev Sukhil and Madhur Behl. Adaptive lookahead pure-pursuit for autonomous racing. *arXiv preprint arXiv:2111.08873*, 2021.

[11] SHIFA SULAIMAN and AP SUDHEER. Implementation of pure pursuit algorithm based controller for a mobile platform.