Tampere University

Ahmed Abdulsahib

PATH FOLLOWING AND MOTION CONTROL FOR ARTICULATED FRAME STEERING MOBILE WORKING MACHINE USING ROS2

Master of Science Thesis
Faculty of Engineering and Natural Sciences
Examiners: Associate Professor Reza Ghabcheloo
Jukka Yrjänäinen
December 2022

ABSTRACT

Ahmed Abdulsahib: PATH FOLLOWING AND MOTION CONTROL FOR ARTICULATED FRAME STEERING MOBILE WORKING MACHINE USING ROS2
Master of Science Thesis
Tampere University
Degree Programme in Automation Engineering, MSc. (Tech)
December 2022

Autonomous vehicles (AVs) have been studied and researched at least since the middle of 19s century, and the interest in these vehicles has grown in the last decade [1]. There are many vehicle types with different steering techniques. Each is designed and manufactured depending on the need to perform specific tasks (for example, transporting passengers, transporting goods, and doing heavy duties like cutting trees, digging earth, and harvesting crops). This thesis highlights the autonomous articulated frame steering (AFS) heavy-duty mobile working machines and aims to address the problems of autonomizing the AFS machine with basic autonomy requirements, which makes the machine move without the need for human direct and indirect control.

The working environment (like mines, forests, and construction sites), where heavy-duty machines are used to perform some tasks, requires an expert machine operator to drive it and control its manipulator, which increases the operator's workload. However, due to the working environment's limited area, the machine mostly has repetitive tasks that include following the same paths; therefore, we proposed implementing a path-following control system that could be used to help the operator by reducing the work amount.

The proposed path following [2] is based on controlling the vehicle's position and orientation to match the desired positions and orientation on a specified path where the position's lateral error and orientation error are minimized to zero while the vehicle follows the given path. The implemented control system is divided into many subsystems; each is responsible for a specific task, and to communicate between them we used the Robot Operating System ROS2.

In this thesis, we are focusing on two of these subsystems. The first subsystem, called path following that, generates linear and angular velocities needed to make the machine follow the path. The other subsystem, called motion control, is responsible for converting the linear and angular velocities to machine commands (gear, steering, gas) and controls the machine's acceleration and steering angle. The implemented path-following control system required understanding the machine's kinematics and modeling the steering system.

The implemented system is tested first using an AFS robot in a simulation environment, then tested on a real AFS heavy-duty machine owned by Tampere university. Moreover, the tests repeated for another path following based on the modified pure pursuit technique provided by ROS2 navigation for compression and evaluation purposes.

Keywords: Path Following, Path Tracking, Motion Control, Articulated Frame Steering, AFS, Heavy Duty Machines, Mobile Working Machine, ROS2, Autonomous Vehicle, Control System, Robot Operating System

PREFACE

I want to thank my supervisors, Prof. Reza Ghabcheloo and Mr. Jukka Yrjänäinen, for giving me the opportunity to work as a research assistant on this interesting topic under their supervision, as well as for their guidance and mentorship while working on and writing my thesis. They helped me discover my interest in working with Autonomous Mobile Working Machines. Furthermore, I'd like to thank Teemu Mökkönen for his assistance in developing the simulator used in testing, as well as his assistance and support with the experiment and data collection on the real machine, and Bharath Garigipati for configuring the localization system on the machine. Thanks to Ashish Sonawane and Hossein Abdi for their help and insights during the writing process.
Finally, I would like to thank my parents and sisters for their encouragement throughout my studies, and a special thanks to my wife, Iman and children (Maher and Lamar). They illuminated my life by walking beside me the entire way, providing love, support, and motivation.

Tampere, 31st December 2022

Ahmed Abdulsahib

CONTENTS

GLOSSARY OF ABBREVIATIONS

AFS	Articulated Frame Steering
AVs	Autonomous Vehicles
GPS	Global Positioning System
IMU	Inertial Measurement Unit
Lidar	Light Detection and Ranging
MPC	Model Predictive Control
MPR	Multi-Purpose Rover
MWM	Mobile Working Machines
Nav2	ROS2 Navigation
PPC	Pure Pursuit Control
Radar	Radio Detection and Ranging
ROS	Robot Operating System
RPP	Regulated Pure Pursuit
Rviz	ROS Visualization
SAE	Society of Automotive Engineers
STVL	Spatio-Temporal Voxel Layer
TEB	Timed Elastic Band

1. INTRODUCTION

Mobile Working Machines (MWMs) are vehicles designed and manufactured to perform specific tasks besides moving from one place to another. Due to their high power, steering systems, and unique attached tools (saws, drills, booms, grippers, and more), they have been widely used in many applications containing heavy-duty work, such as cutting trees, digging the earth, harvesting crops, and lifting heavy objects. Therefore, they are an essential part of the economy [3].

The working environments (mines, forests, and construction sites) where MWMs are used have many potential dangers due to the existence of working equipment, materials, and noises. MWMs increase the possibility of fatal accidents in such limited workspaces [4]. In addition, driving them and controlling their manipulators are stressful and exhausting tasks [5]. This causes an increase in the probability of human error, which is the main reason for many accidents [6][7][8][9]. So, providing autonomous driving control systems has been considered an appropriate solution that increases safety and offers many advantages like increasing productivity, efficiency, and reducing the workload [10][11][12].

One of the fundamental missions of autonomous vehicles is the concept of path following, which is the process of moving between two points on a specified path. A good understanding of the machine's kinematics and structure is required to perform an accurate path following task.[13][14]

The mainly used steering systems in MWMS are Ackerman, skid, and Articulated Frame Steering (AFS) [15]. From the steering types mentioned above, the vehicles with AFS are extensively used in the construction, forestry, and mining sectors due to their maneuverability, mobility, and enhanced traction performances [16][17][18][19][20]. Therefore, this research focuses on implementing path following and motion control for these machines. Usually, the control system is divided into many sub-systems [21][22][23][24]; and in that case, it is important to ensure they communicate with each other by using a capable communication system that manages the message exchange within the control system. One of the available framework that has many features including taking care of the communications is the Robot Operating Systems (ROS).

During the last few years there have been increase in developing applications with this platform by the research community and adding features to it [25]. Also, the interest in

that framework has been growing among industries. However, it is still not fully adopted by most of them, unlike Matlab and Simulink, which are widely used in many industries [26]. Therefore, the main question that will be answered in this thesis is that:-

"Is it possible for ROS2 to be an alternative platform to implement path following and motion control instead of the popular platforms in industries such as Matlab/Simulink?"

There are many path following methods provided by ROS2 for autonomous vehicles. However, we are eager to propose a new method that can be used for autonomous articulated machines. Therefore the other question is:-

"How beneficial is using our proposed path following in compression to an other developed and commonly used strategy in ROS2 such as Regulated Pure Pursuit (RPP)?"

To answer these questions, we address the problem of implementing path following and motion control for AFS vehicles using ROS2 . Also, in this thesis, we state the differences between the implemented path following and ready from the shelf path following controller provided in ROS2 Navigation (Nav2) [27] by comparing the two implementations using a simulation environment with gazebo and later using an actual AFS vehicle.

In general, the structure of this thesis starts with the needed background to understand the essential parts of the implemented system, which are introduced in the first four chapters. The current chapter introduces the main topic of this thesis and the importance of this work, as well as raises the scientific questions that are required to be answered. Chapter (2) illustrates what mobile working machines are and what makes them suitable for heavy duties. It also briefly explains the hydraulic system on these vehicles and discusses the different steering systems they use. Then, autonomous Vehicles (AVs) are introduced, and the Autonomous control system for these vehicles is explained. Furthermore, lately some sensors that are used on these vehicles are illustrated. In chapter (3), ROS2 is explained, starting from the main purpose of developing it to the different services it provides, and diving into the Nav2, which contains the ready-developed path that is compared to the work done in this thesis. Chapter (4) demonstrates the path following concept and strategies, then discusses some research papers in the literature on using articulated steering machines with the path following, and at the end of this chapter, the uncovered sides in these papers are mentioned, and the contribution of our work is illustrated. Chapter (5) introduces the algorithmic part of the control system that is proposed in this thesis, and it starts with the path following strategy, then the kinematics models of the vehicle and the steering actuator, which are needed for the motion control that is explained at the end of this chapter. Then in chapter(6), the articulated vehicle used in this thesis is demonstrated, and the implemented control system for that machine is explained. This chapter covers the entire control system from a high point of view.
Chapter (7) summarizes the test and results to evaluate the proposed path following the controller and compares it to another popular and tested controller.

Chapter (8) is the last chapter that contains the conclusion, which summarizes the findings and answers the questions issued in this thesis.

2. MOBILE WORKING MACHINES (MWM) AND AUTONOMOUS VEHICLES (AVS)

Mobile working machines have been used widely in many different working environments, as explained in chapter (1); however, they are unsuitable for carrying passengers [28]. These machines have three characteristics, which can be summarized as follows, they are mobile, designed and implemented to perform specific tasks in the working environment, and have a high energy share in their working functions [15][29].
The diversity of these machines comes from the requirements needs to be fulfilled by them, such as [15]:

- Need for functionality.
- Good productivity and high efficiency.
- Manufacturer and customer economics.
- Easy maintenance, Reliable availability.
- Safety and security.
- Compliance with standards and legislation.

This chapter briefly describes the general structure of mobile working machines and a few basics of the hydraulic systems that clarify why these systems are used with the mentioned machines. Then in the last section, autonomous vehicles are introduced, and some sensor technologies are explained briefly to provide some basic information that helps to automatize the MWM.

2.1 Typical Structure of Mobile Working Machine

There is a long list of mobile working machines designed to work in different environments, such as forestry, mining, construction, and logistics, to perform different tasks. One example of these machines is the loader which is used in construction to move materials such as soil, rock, and sand from one place to another or load them to other vehicle. Despite these machines' diversity, they have common standard structure which is identified by four parts, work function, operator's workplace, power train, and chassis. Fig 2.1 shows different MWMs types with a typical structure [15].

The work function describes the machine work principle and its construction. The operator's working place is the control cabin where the driver is sitting, and it is designed to allow the driver to perform the essential work functions, driving the machine and see the travel path. This working place contains monitoring and controlling devices for driving and working. Moreover, it protects the driver by its surrounding frame or machine structure. The chassis of these machines include a steering system, locomotion, and attachments, where the frame is the central load-bearing assembly of the chassis, which can consist of one solid part or several parts connected by joints. The means of locomotion, such as wheel axles or crawler tracks, are connected to the frame. The locomotion devices transfer the vertical and horizontal forces to the ground via fixed, hinged, or suspended connections. The power train needs a primary energy source. The commonly used energy source is internal combustion engines. [15]

MWMs are required to perform tasks that need high forces, therefore they use hydraulic power systems to generate the needed forces for work and steering. These systems are introduced in 2.1.1.

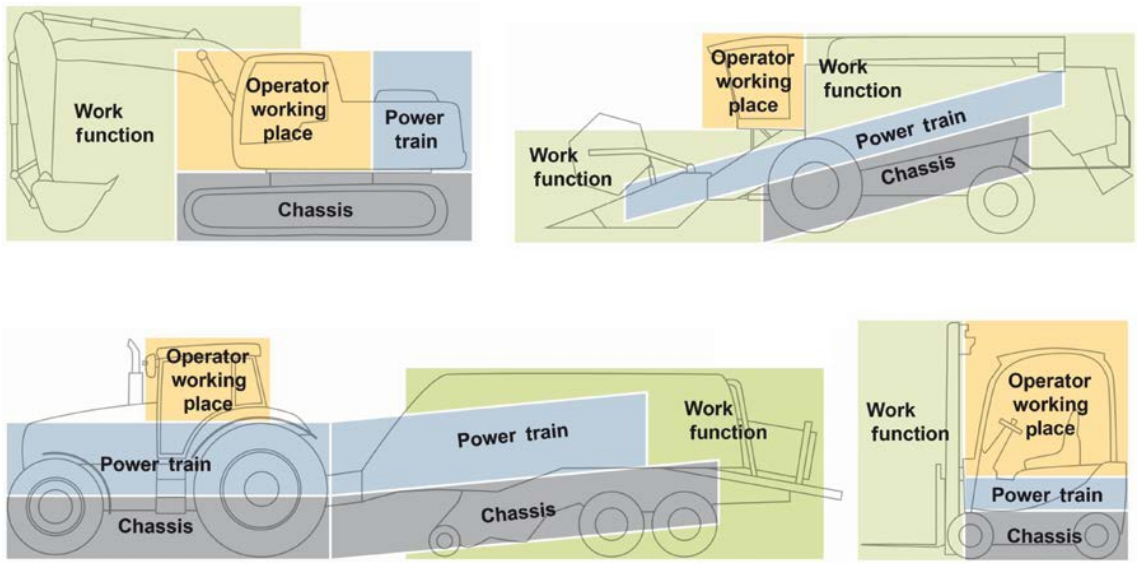


Figure 2.1. *Structure of four typical mobile working machines. adopted from [15].*

.

2.1.1 Power System on Mobile Working Machine

Backhoe loaders, tunnel drills, hydraulic excavators, agricultural tractors, bulldozers, and forestry forwarders are some examples of MWM. Each of these working machines is designed to do different tasks so that their appearance and structures are different from the others, and the hydraulic transmission system for each machine is designed and implemented to let it accomplish its task efficiently.

In general, the hydraulic power transmission system is designed to transmit the input en-

ergy from its source using pressurized hydraulic fluid to its point of deployment, where the transmission of energy in this system starts from the input kinetic energy that is transformed to pressure potential energy, then to work, and this work is converted at the end to kinetic energy again, as shown in Fig 2.2. [30]

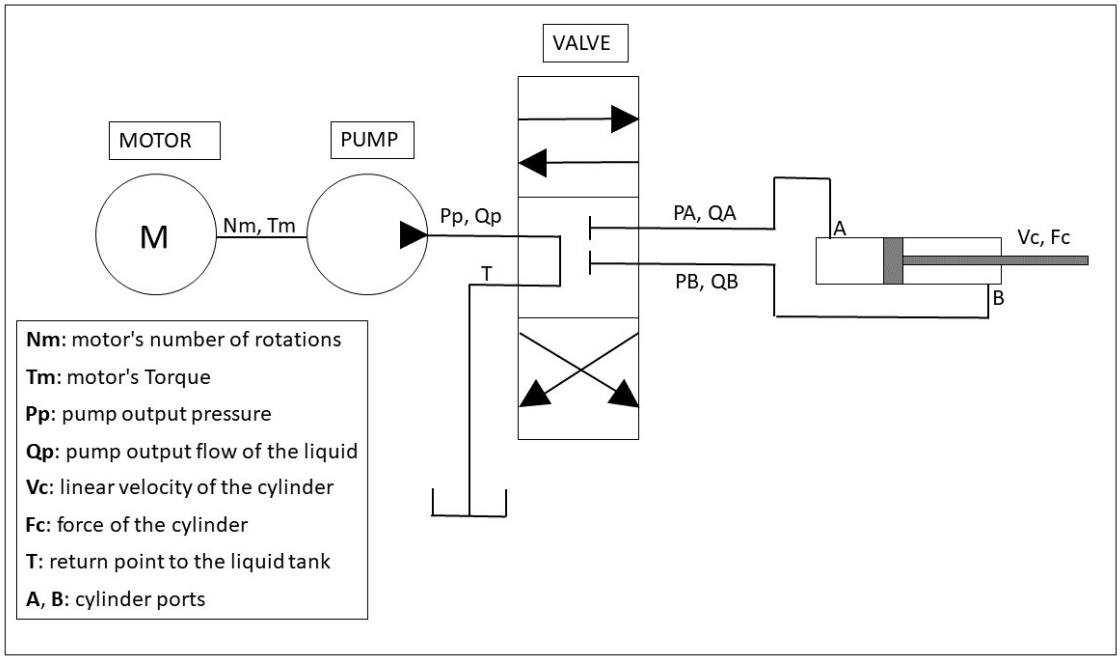


Figure 2.2. *Illustration of the principle of energy transmission process in a hydraulic power system [30].*

In Fig 2.2, the motor provides input mechanical power to the pump in the hydraulic transmission system, which produces a fluid flow to the cylinder. That action pressurizes the fluid in the cylinder, increasing the energy level from zero to the maximum value.
Hoses deliver the fluid to the control valve, and the valve regulates the pressurized fluid's amount and flow direction to deliver it to different cylinder ports to achieve the work. During the fluid flow from the pump to the actuator, the fluid friction and the leaking of the pressurized fluid, which is possibly noticeable, are causing some energy loss.[30]

Power generation, distribution, deployment, and regulation are the subsystems of the hydraulic power system, which are determined according to the system's functions. The supplied energy generated by the prime mover is converted from kinetic energy to potential hydraulic energy in the pressurized fluids using a pump, which mainly represents the power generation subsystem. On the other hand, the fluid flow directions, rate, and pressure are regulated and controlled by the power distribution subsystem, which contains various control valves to do its function. The power deployment subsystem is responsible for moving the loads by utilizing the potential energy carried by the pressurized fluid. Finally, the power regulation subsystem contains the fluid tank, hydraulic hoses, filters, and accumulators. [30]

2.1.2 Fundamentals of Hydraulic Power Transmission

There are three fundamental concepts of the hydraulic power transmission, that explain the power transmission process in hydraulic systems. These concepts are [30]:

- Multiplication of force:
  Pascal's law is one of the fundamental rules for fluid power transmission that serves as the foundation for static pressure transmission for the fluid. According to Pascal's rule, the pressure applied to a contained fluid operates equally in all directions at right angles to the confining surface, implying that fluids may convey energy when confined. Therefore, applying force on the arm of a small hydraulic cylinder that is transmitting the compressed fluid to a larger cylinder will increase the force on the larger cylinder's arm. Similar to using a car jack, a hand applies force on the jack handle, causing the lifting of the heavy car.
- Conservation of Energy:
  Energy conservation can be seen in the hydraulic transmission system, where the input kinetic mechanical power is converted to hydraulic potential energy first by a hydraulic pump and then to kinetic mechanical energy power by a hydraulic actuator at the output.
- Continuity of fluids:
  The fluid flows continuously within the power transmission system. This continuous flow principle is critical in hydraulic system analysis. The flow continuity expresses the state that the total inlet flow rate equals the total outlet flow rate for steady flow in a control volume.

2.2 Steering Systems for Mobile Working Machines

The commonly used steering principles in mobile working vehicles are Ackerman steering, skid steering, and articulated frame steering, which can be shown in Fig 2.3. Where Ackerman steering is the standard for many types of vehicles, including tractors, excavators, and telehandlers; these vehicles use steering actuators that steer either one axle or two axles. Skid steering depends on different tires' speeds to steer the vehicle by rotating each tire from the same side at the same speed, similar to differential drive robots or tanks. Articulated frame steering (AFS) vehicles are designed to be separated into two parts connected using a center joint and can be steered using attached hydraulic cylinders to the parts [15].

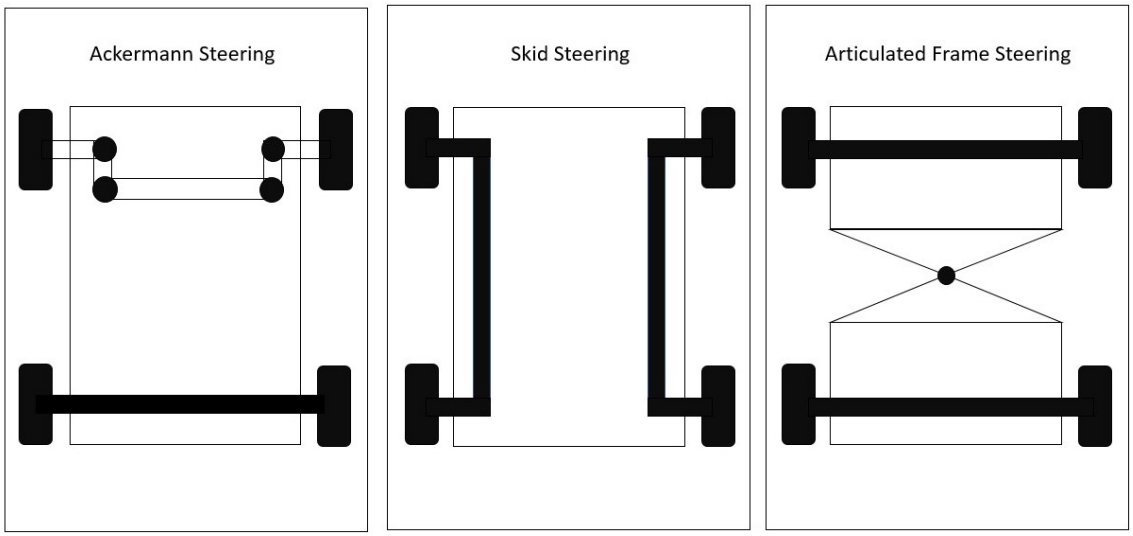


Figure 2.3. *Mobile working vehicles steering types.*

2.2.1 Ackerman steering

This steering type also called four-bar linkages, it is the generally used mechanism to steer four-wheel vehicles [31], [32]. To steer the machine, a steering box transmits the input motion from the steering wheel to the steering knuckles, using Ackerman steering linkages [33]. In Ackerman, the tire closer to the curved path center has a smaller turning radius than the other tire, so that it is steered with a greater angle, which helps to prevent tire rubbing.

2.2.2 Skid steering

It is accomplished by driving wheel pairs on one side of the vehicle with different speed than the other side wheels. Even though the skid steering system provides certain mechanical benefits, controlling it is a complex undertaking since following the curved path require the wheels to skid [34]. Skidding may result to change the position of the instantaneous center of rotation which affects the vehicle's motion stability [35].

2.2.3 Articulated Frame Steering (AFS)

Due to their superior maneuverability and high steering generated force compared to other steering types, mobile working machines with AFS are dominant in many working environments, such as mines, forestry industries, and construction sites [36][37]. In general, vehicles that use articulated frame steering are made up of two separate front and rear bodies connected by an articulated joint. That joint could have up to two degrees of freedom on roll (rotation around the x-axis, it is usually the vehicle heading axis) and yaw (rotation around the z-axis).

Haulers, have two degrees of freedom, and other vehicles, such as compact-size loaders, have only one degree of freedom around the z-axis. In Fig 2.4 the loader center joint has two degree of freedom on roll and yaw.

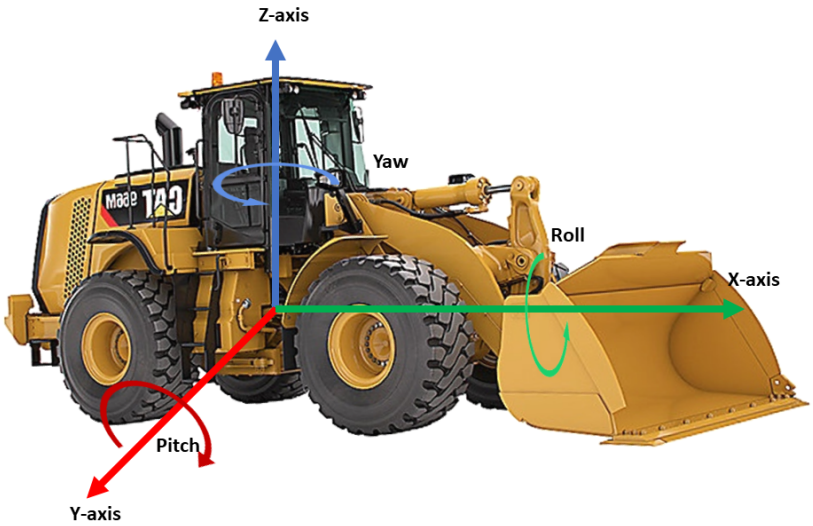


Figure 2.4. *Illustration of roll, pitch and yaw of the center joint.*

When the steering wheel is turned, a hydraulic cylinders changes the articulation joint angle. Most of the AFS machines are used for off-roading, but they have the ability to travel on roads and highways.[38]

Driving these vehicles with high speeds causes issues primarily related to the vehicle's stability while moving on a specific path. For example, at high speed, a slight change in the center joint angle causes the two bodies of the vehicle to oscillate relative to each other, which leads to oscillation in movement. [38][39]

2.3 Autonomous Vehicles (AVs)

Vehicles that can move and take action without any human direct or teleportation control are known as Autonomous Vehicles (AVs) [40]. These vehicles have been explored and investigated since at least the mid-nineteenth century. Carnegie Mellon university created the first autonomous car in 1984 [41][42], then in 1987, Munich university and Mercedes-Benz developed it [43]. AVs are known to be able to sense the environment and operate without human involvement. These vehicles are able to imitates the human drivers without the need of human presence [44][45].

The Society of Automotive Engineers (SAE) categorizes Automated Driving Systems (ADS) into six levels starting from 0 (no autonomy) to 5 (fully autonomous) according

to SAEJ3016 standard. The levels are [46][47][48]:

- **Level 0 (No Autonomy):** The vehicles are manually controlled by the human driver.
- **Level 1 (Driver Assistance):** It is considered the lower level of automation, where the vehicle contains one feature to assist the driver.
- **Level 2 (Partial Driving Automation):** A driving assistant can control the steering and the vehicle's speed, and the driver supervises the driving process and can take control at any time.
- **Level 3 (Conditional Driving Automation):** Despite that, the driver can override the vehicle's automated control as the driver is needed onboard; the vehicle can make decisions and have environmental detection.
- **Level 4 (High Driving Automation):** At this level, the vehicle can be driven by a human; still, it can fully drive itself and make decisions so that human presence is not required in most cases.
- **Level 5 (Full Driving Automation):** The vehicle is self driven, and it is not equipped with any manual control, such as steering or paddles, so the human can not take manual control at any time.

In the levels higher than 2, the AVs perform the driving task by dividing it into three stages, starting with sensing and perception, path planning, and path following (path tracking). This thesis focuses on the last stage (Path following) and the motion control for articulated frame steering.

2.3.1 Autonomous Vehicle System

Some researchers devide the autonomous vehicles system to four parts, sensing, perception, planning, and control, as shown in Fig 2.5 [49][50][51]. Other researchers divide it into five parts which are sensing, perception, modeling, planning, and control, where the modeling include constructing the environment and creating world map for navigation [52].

The vehicle senses the environment using number of sensors installed on the AV. The sensors are hardware components that can collect the information about their surroundings. The perception block can be defined as the process of collecting the data from the sensors and combining it into meaningful information, then passing it to the modeling for mapping and to the planner, which uses the received information after the perception presses and mapping to perform behavior planning and path planning for long and short paths. The output from the planning is used by the control block to generate control commands and sends them to the vehicle's actuator to ensure that the vehicle follows the behavior and the planned path determined by the planner.[53][54][55]

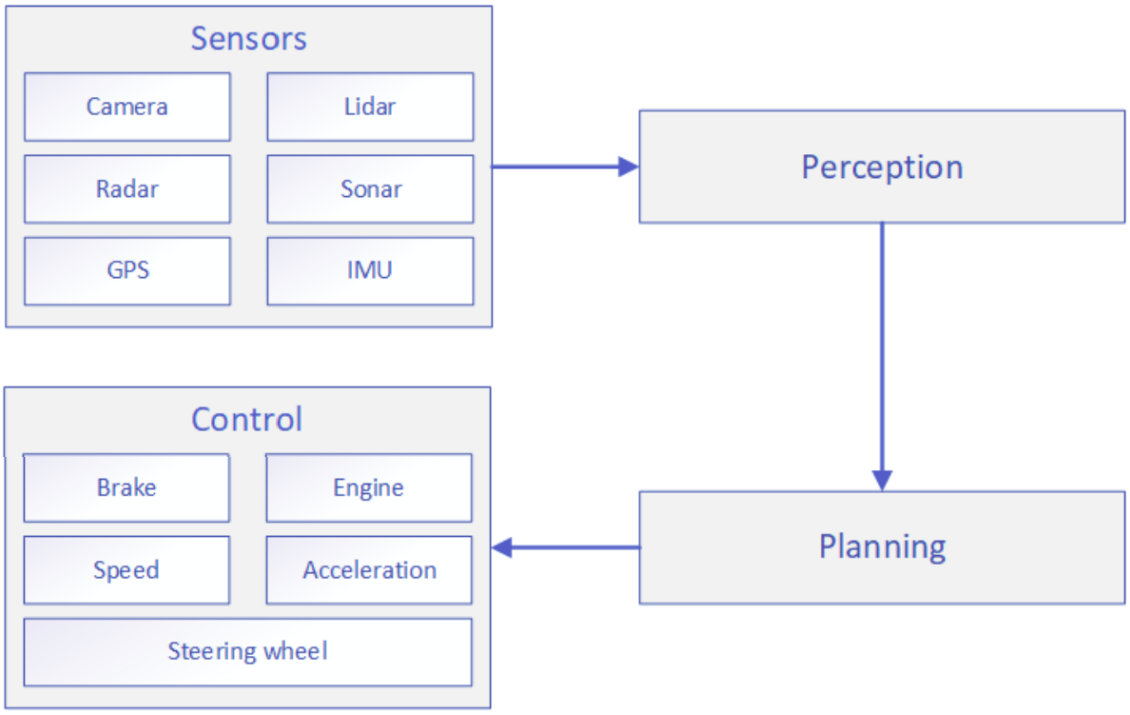


Figure 2.5. *Autonomous vehicle system block diagram [53].*

In the next section we are focusing on some of the common sensors used in many autonomous vehicles.

2.3.2 Sensors in Autonomous Vehicles

In AVs, sensors such as cameras, lidar, radar, sonar, Global Navigation Satellite Systems (GNSS), an Inertial Measurement Unit (IMU), rotary encoders, and resolvers are essential components. These sensors collect data to be processed by the computer in the AV and used to determine the control actions for the steering, braking, and speed. In addition to the sensor's data, another source is the environment information stored by other AVs in the cloud, which are used to make vehicle control decisions.[53]
Some of these sensors are demonstrated shortly below:

- IMU
  It used at first in aircraft navigation at 1930s [56]. It is mainly works as three combined sensors, gyroscope, accelerometers and magnetometer to measure the orientation, velocity and acceleration As shown in Fig 2.6 [57][58][59]. Therefore, it can be used in many devices, such as robotic arms, to perform the trajectory tracking of the arm's gripper [60].
- Camera
  In autonomous vehicles, one of the first sensors used is the camera. Nowadays, it is suggested to be a fundamental technology in autonomous vehicles by the man-

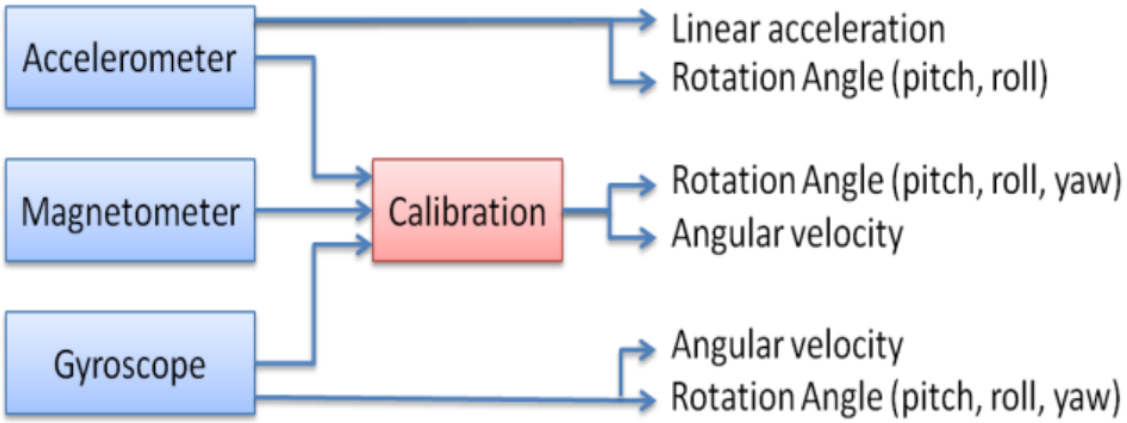


Figure 2.6. *Three sensors combined to form IMU [61].*

ufacturers [61][62]. The vehicle can visualize its surroundings using the camera [61].

Compared to lidar and radar, the camera is more affordable; also, the camera is efficient at classifying objects. Using a camera requires high processing abilities as it provides up to multi-megabytes of data in real time [53]. The camera utilization provides exceptional benefits in self driving cars [63][64][65] [66][67].

- Light Detection and Ranging (Lidar)
  This sensor determines the distance between itself and detected objects using an infrared laser beam. It has a focused laser beam, therefore, its resolution is higher than the radar, and it is not affected by lighting conditions as the camera [68]. However, The weather and dirt on the sensors can affect the its performance [69]. Some types of of this sensor are spinning lidar which covers 360 degrees of the surrounding environment , the line scanning lidar that sense the objects in one plane and flash lidar that sense objects in a single direction [70][71]. In spinning lidar, A spinning swivel scans the laser beam across the detection area using laser pulses and senses the reflected pulses from the objects. It calculates the distance to detected objects according to the time between the emitting and the detection of the reflected laser light, then obtains the point cloud that is useful for 3D mapping [72]. The lidar can be used to measure the detected object's speed [73].
- Radio Detection and Ranging (Radar)
  The radar sensor is added to the vehicle to perform several functions such as collision avoidance, collision warning and adaptive cruise control [74][75]. The radar locates the objects by transmitting microwaves and receiving the reflections at light's speed [76] and can calculate the objects speed using Doppler effect, so that it provides the velocity as independent parameter from the position, unlike the other sensors that uses the difference in position with time to calculate the speed [53].
- Rotary encoder

Because of their ease of integration into digital circuits and usefulness as position and speed detectors, these sensors are frequently used to measure the rotational speeds in the control systems that use digital controllers. Rotary encoders could be optical or magnetic encoders. Magnetic rotary encoders are smaller and respond faster than optical encoders.[77], [78]

3. ROBOT OPERATING SYSTEM (ROS)

ROS is an open-source middleware for implementing complex robotic systems and applications. It is not a real operating system; in fact, it is a framework, which needs to be installed on operating system (commonly Ubuntu) to work [79]. From another point of view, it can be considered as a communication system that allows different programs on the same hardware or even on different hardware to communicate with each other's using the ROS messages [80]. In addition, it provides many different communication features (For example, Messages, Topics, Services, and Actions) and the ability to customize these features according to the developer's needs.

The start of ROS was in 2006 when Eric Berger and Keenan Wyrobek created a program named Stanford Personal Robotics Program. They aimed to implement a framework that allowed programs to communicate with each other. They intended to use it with the First Personal Robot (PR1) [81].

Implementing the framework was driven by the need to solve two problems at that time. The first is the required long time to re-implementing complex programs and algorithms for the robots' infrastructures (basically, the drivers to the sensors and the communication between the different programs on the same robot) that are already implemented by others. The second problem was the short time dedicated to implementing the intelligent robotics programs, which are based on the mentioned infrastructure. Fig 3.1 gives an example of the time spent reinventing exited technology by many organizations.

Based on the Stanford Personal Robotics Program idea, ROS development started at Willow Garage in 2008, and the first distribution of it was released in 2009, called ROS Mango Tango, also called ROS 0.4. then, it evolved to ROS1, which was released in 2010 [83], and in 2011 the developers released the first robot called Turtlebot. The robot was a simple and cheap, and allowed anyone using it to learn the basics of robotics using the robot operating system. Therefore the community of that platform increased. Later in 2018, ROS2 version was released.[82]

Other than the primary goal of ROS (codes re-usability), there are multiple goals [84] such as:

1. Thin: ROS is designed to integrate easily with other frameworks such as OpenRAVE, Orocos, and Player.

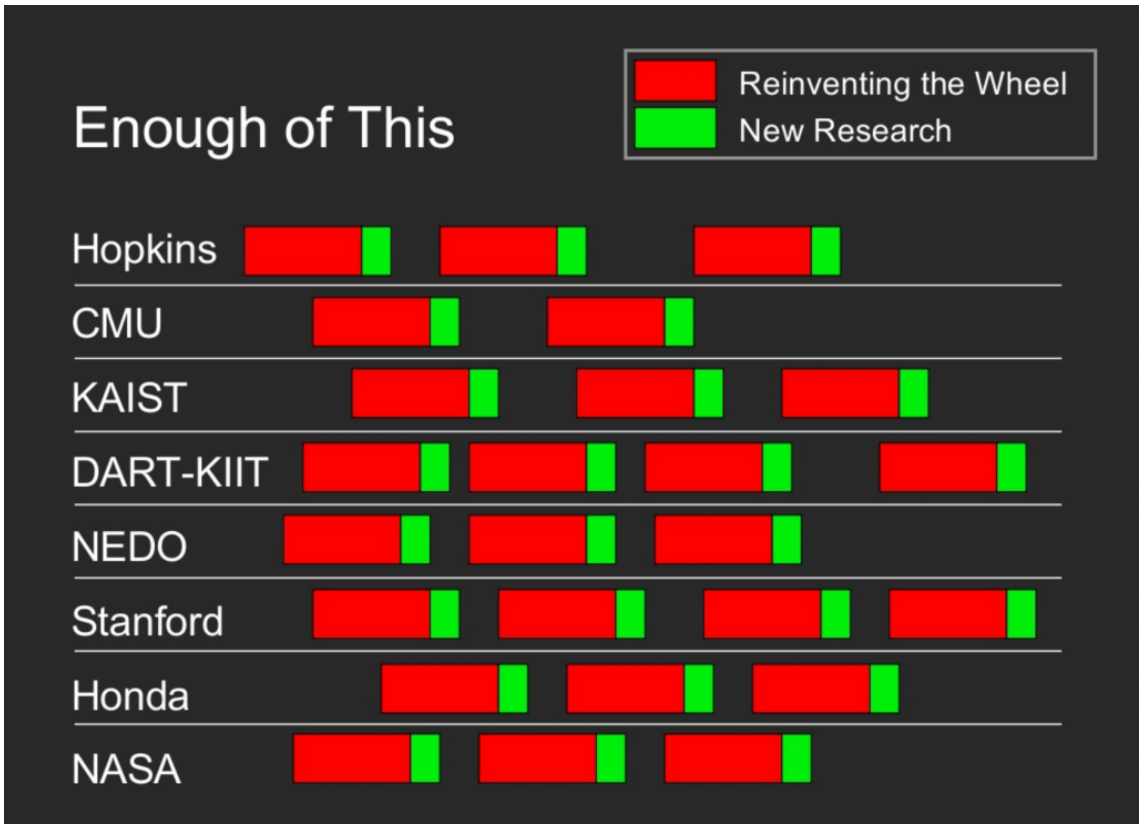


Figure 3.1. *Most of the time spent in robotics was reinventing the wheel.[82]*

2. Language independent: it can be implemented in any modern language, for example, python, c++, Java, Lua, and Lips.
3. Easy testing: because it has rostest, a built-in unit/integration test framework, it is simple to set up and take down test fixtures.

3.1 ROS1 and ROS2

ROS1 was developed to be a quality and performant system, and commercially it was raised due to the flagship projects that provided control, simulation, visualization, autonomous navigation, and other services [85][86][87]. It contains many useful libraries for making different types of robots [79]. This platform allows a small team to develop complex robotic software because it has a large ecosystem that covers a variety of sensors, controls, and algorithmic packages that community contributors share. Regardless, ROS1 developers did not prioritize the uptime, network topology, and security which are essential parts of the industry development [79]. Therefore, the platform sufferers from some limitations, such as data delivering instability when using lossy links (for example, wifi), it has a single point of failure, and it does not contain security mechanisms. The limitations had to be solved to integrate this platform into the industry, especially when many companies started to involve ROS1 for creating reliable applications [88].

ROS2 is the successor of ROS1, and it was redesigned from the ground to address its predecessor's limitations while benefiting from its community-provided capabilities [27]. ROS2 design was guided by some principles [88] such as:

- Distribution: In complex field such as robotics, it is favorable to reduce the complexity using distributed systems approached [89].
- Abstraction: The data messages must be specified to govern the communication and define the semantics of the data exchanged; also, balancing between exposing the details of a component and the over-fitting cost is favorable.
- Asynchrony: ROS2 messages are exchanged asynchronously between the nodes, which creates an event-based system [90]. This principle allows the applications to work across multiple time domains, which result from different frequencies of the hardware and software components in the system.
- Modularity: It is applied at different levels, such as library APIs, message definitions, and the platform ecosystem.

Based on the design principles, ROS2 aims to fulfill specific requirements and the needs of robotics developers, such as security, embedded systems support, multiple network connectivity, real-time computing capability, and safety. It is reliable and high-quality robotics framework that supports a diverse set of applications. The developers added many features and spent much effort to improve this platform, and the different features of ROS2 compared to ROS1 are demonstrated in Fig 3.2. [88]

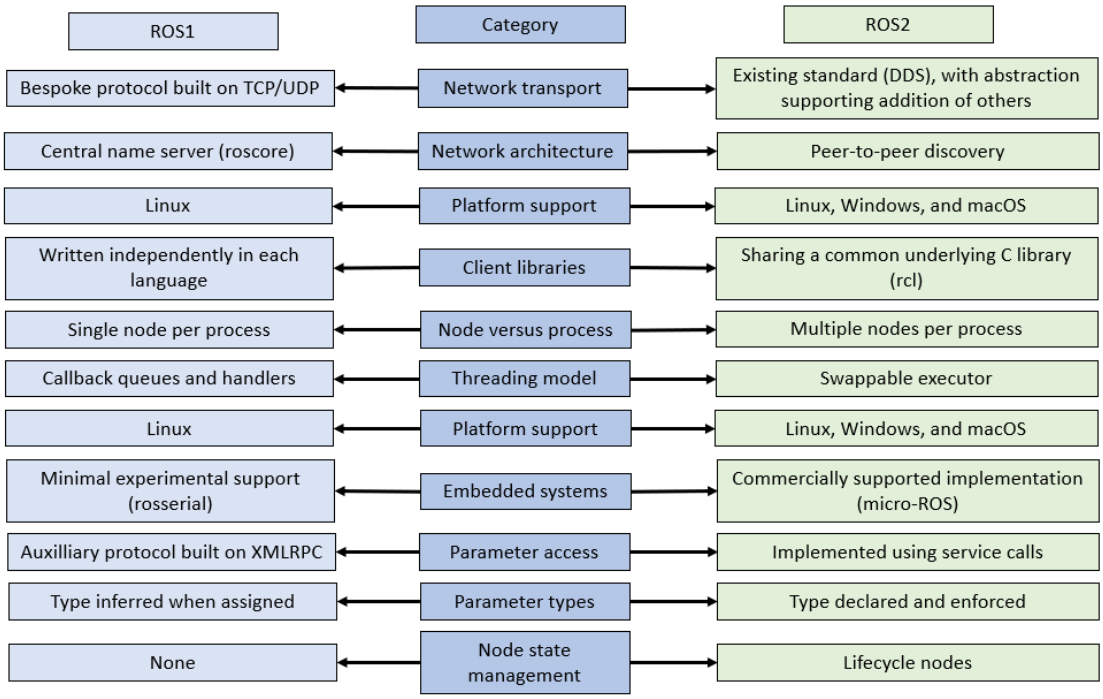


Figure 3.2. *ROS2 features compared with ROS1 [88].*

3.2 ROS Concepts Levels

Three concepts levels defines ROS [91]:

- Filesystem: This level concerns ROS resources and it define how the software is structured and saved such as packages, metapackages, package manifests, service, and messages.
- Community Level: This concept is a ROS asset that allow different networks to exchange codes and information. Some OF these assets are distribution, repositories, mailing lists ROS Wiki web site.
- Computation Graph Level: There are many concepts in this level such as nodes, topics, services, actions. This level represents ROS processes peer to peer network which processes the data. All the nodes have access to this network and they can communicate with each other using ROS messages.

3.3 ROS Node

A node is an independent running piece of code that processes computation and may subscribe to many topics to get inputs and publish to any number of topics to send outputs [92]. A robot control system usually has multiple nodes communicating with each other and collaborating to perform needed tasks and the nodes can not have the same names [93]. The nodes do not have information nor care about the existence of other nodes in the system [94]

There are libraries in the used languages in ROS1 and ROS2 to write the nodes, such as rospy (python library for ROS1), roscpp (C++ library for ROS1), rclpy (python library for ROS2), and rclcpp (C++ library for ROS2). Since Nodes can be written in different languages, the developer that uses, for example, python to implement nodes for their control system can re-use any implemented node in C++ to communicate with implemented node, which means it is not needed to implement nodes that are already existed even if they are written in different languages [95].

3.4 ROS Topics

They are the way of communication in ROS. The topics allow the nodes to communicate with each other using the publish and subscribe models. ROS provide a list of the available topics. The publisher is the node that sends the messages and the subscriber is the node that receives a message. ROS provide lists of the publishers and the subscribers. The messages are not sent directly from node to the other, instead, ROS topic is used as a bridge to deliverer the messages from the publisher to the subscriber. One publisher can publish to multiple subscribers and one subscriber can receive messages from

multiple publishers as shown in Fig 3.3.

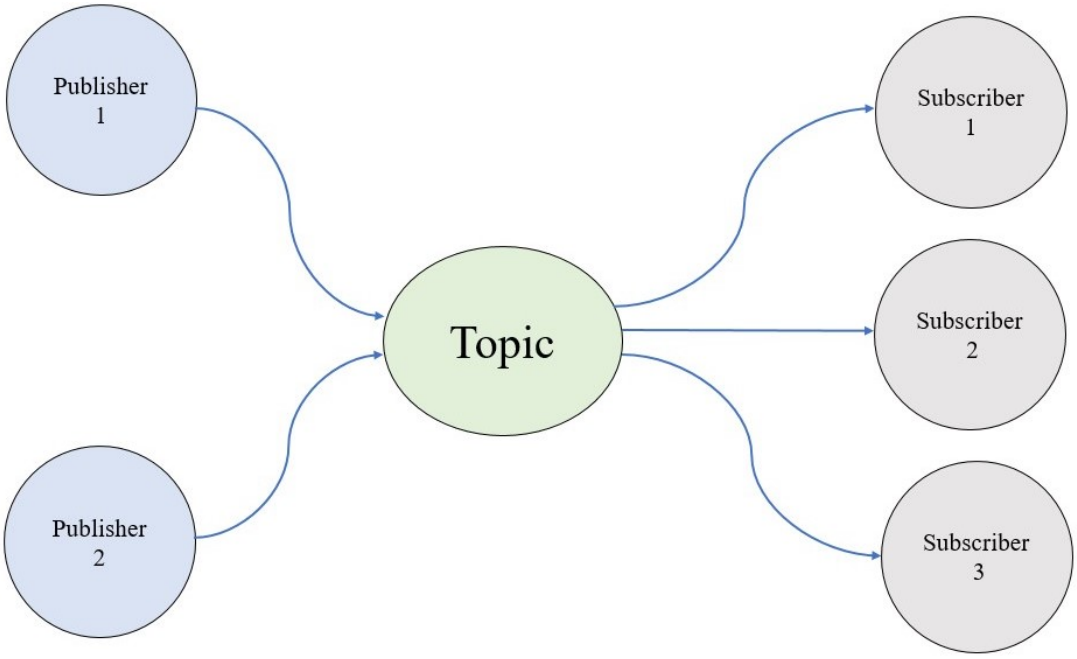


Figure 3.3. *Topic with multiple publishers and multiple subscribers. Adopted from [96].*

3.5 Services

Services are another communication method between nodes; similar to the publish-subscribe model; it uses topics to exchange messages between the nodes. It is working based on the request and response model, where a node (client) sends a request message (for performing a specific task) to the topic. The topic forwards that request to another node called the server. The server is responsible for performing the task, and when the request is received, an acceptance message is sent back from the server to the client on the same topic. Then, the server starts to perform the task.

This communication model does not provide any feedback or update other than accepting messages from the server. This model is used when the client requests service; all it needs to know is that the server accepted the task. In addition, one or more clients can ask for the service from the same server, but only one server is allowed to provide the specified service, as shown in Fig 3.4.

3.6 Actions

Actions are another type of communication in ROS, and it is suitable for long-running tasks where the client requests the server to do a task that takes a long time. This type is used when the client is required to be involved in the task from the start to the end, which requires receiving updates regarding the task's progress. Therefore it is designed to have

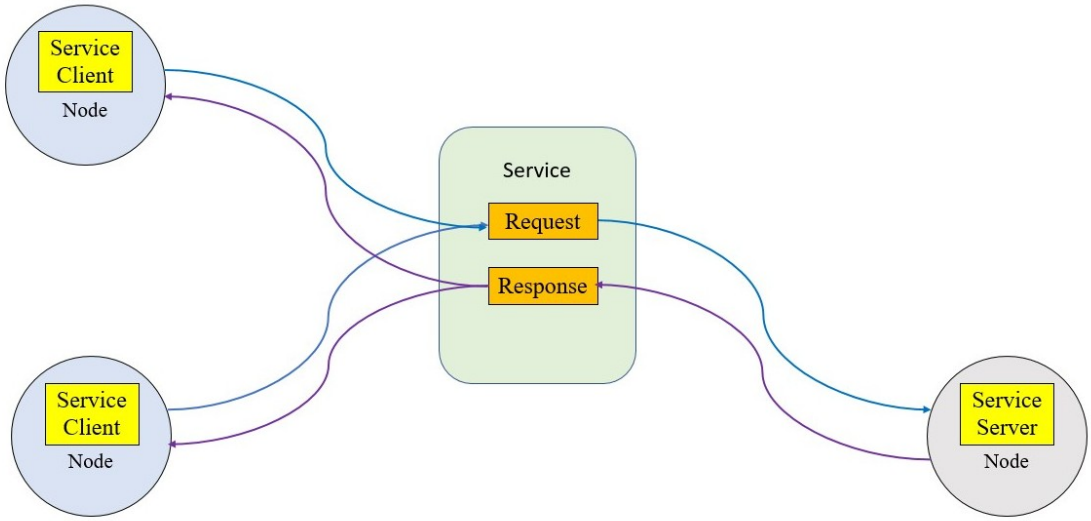


Figure 3.4. *Service diagram fore more than one clients. Adopted from [96].*

three parts: a goal, feedback, and results, as shown in Fig 3.5. The goal is the message, which contains the desired task inputs, and the feedback is the message that contains the desired information to illustrate the current state of the task, and the result is the last message that declares the success or failure of the task.

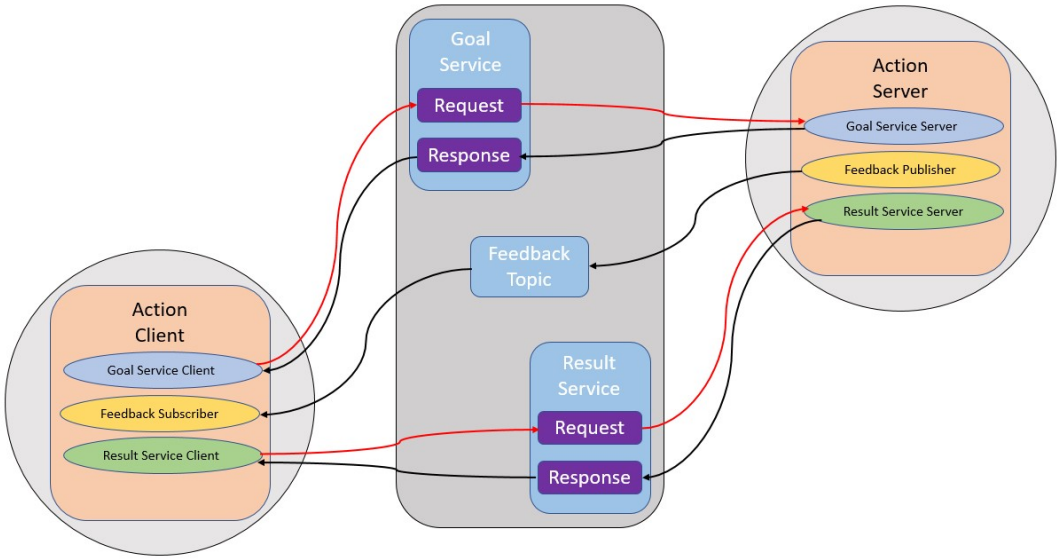


Figure 3.5. *ROS action diagram. Adopted from [96].*

The actions are built on top of the servers and have the same basic principles: request and acknowledgment. As shown in Fig 3.5, actions provide more features where a communication channel is opened between the client and the server, which allows the client to receive the feedback and the result, and request goal cancellation or a new goal.

A client-server model is used in actions where the client sends a goal request message

to the server that triggers a response to reject or accept it. When it is accepted, the server will start executing the wanted task, and during working on it, the server returns a stream of feedback to the client. The result is also returned when the task is completed. If the task cancellation is applied during the execution, the server abandons the task and notifies the client, furthermore, abounding a task and starting a new one is applicable by sending a new request during an ongoing task as shown in Fig 3.6.

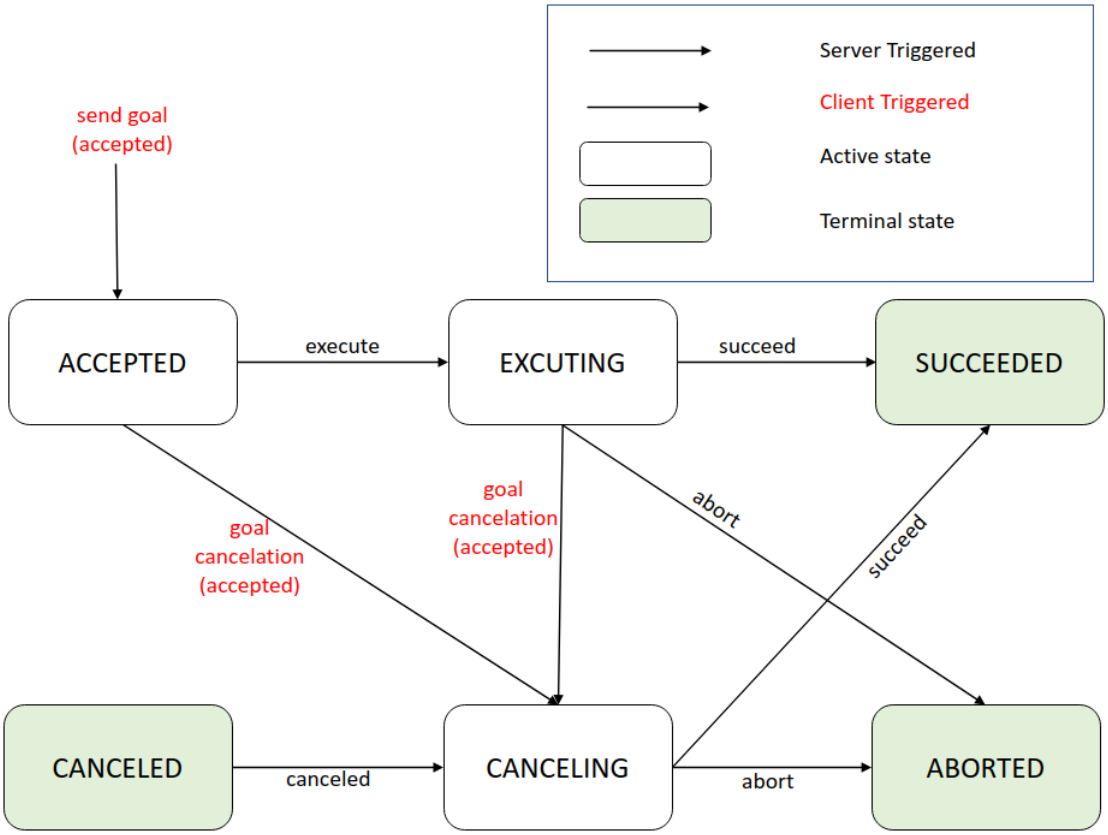


Figure 3.6. *ROS action goal state diagram. Adopted from [97].*

3.7 ROS Navigation stack and ROS2 Navigation

Navigation in robotics is the ability of the robot to localize itself in its environment according to the reference frame, plan the path to a goal point, and then follow the path. This concept is essential for service robots to work in the environments such as warehouses, stores, and factories. Therefore, many navigation systems have been introduced since the first development of the service robot [98][99].

ROS navigation stack is one of ROS's helpful tools that control the robot's speed by performing close loop control to move a robot to the desired point in a 2D world. It receives odometry information (which provides the position, speed, and orientation), sensor streams, and desired goal position, then generates control outputs for safe navigation [27]

ROS2 navigation (Nav2) is the enhanced version of the ROS navigation stack, which supports more robot types for more complex environments. It is a system developed for controlling the robots to enable them to reach a goal state autonomously. This control system generates a plan to reach the goal when it is fed with the current pose, the map, and the goal. The output commands to drive the robot autonomously are generated to ensure safe obstacle avoidance along the way. Unlike the first version, Nav2 is designed to support different robots such as holonomic, differential-drive, legged, and Ackermann robots; however, it is not supporting articulated steering robots, which are widely used in mobile working machines. [27]

3.7.1 ROS2 Navigation Design

Nav2 is designed to be reliable, modular, safe, and secure platform for supporting research, education, and industry. Furthermore, it is configurable and expansible for future developments. The Nav2 tasks are orchestrated by the Behavior Tree Navigator, which is the Nav2 top-layer that follows the progress of the planner, controller, and the recovery servers, as shown in Fig 3.7. The Behavior Tree (BT) allows the navigator node to run plugins (functions) to perform specified tasks; however, it can be modified to perform desired tasks by applying the changes to its XML file. These plugins are shown in Fig 3.8. [27]

The recovery behavior type in Fig 3.7 is responsible for reducing or eliminating the failure effects of the navigation system failure. There is multiple recovery behavior for different cases, such as clearing the cost map at the system perception failure or the waiting behavior to avoid collection when a dynamic obstacle enters the working environment. The perception behaviour is dependent on long range and high resolution sensor technologies that have been enhanced over its predecessor. Therefore a layered costmap approach is used to collect the data from the sensor, then build a 2D map, and when new information from the environment is received, it updates the map and then forwards it to planners and controllers. A Spatio-Temporal Voxel Layer (STVL) plugin is responsible for obstacle detection in a three-dimensional environment.[27]

The control layer plugins are implemented to perform smooth navigation in dynamic environments. There are four controller plugins, as shown in the Fig 3.7. The Regulated Pure Pursuit is one of the commonly used path-following controller plugins developed from a pure pursuit algorithm (explained in 5) with added features. This plugin has many variables, such as look ahead distance and the linear and angular wanted speeds, that can be modified to perform an accurate path-following task.

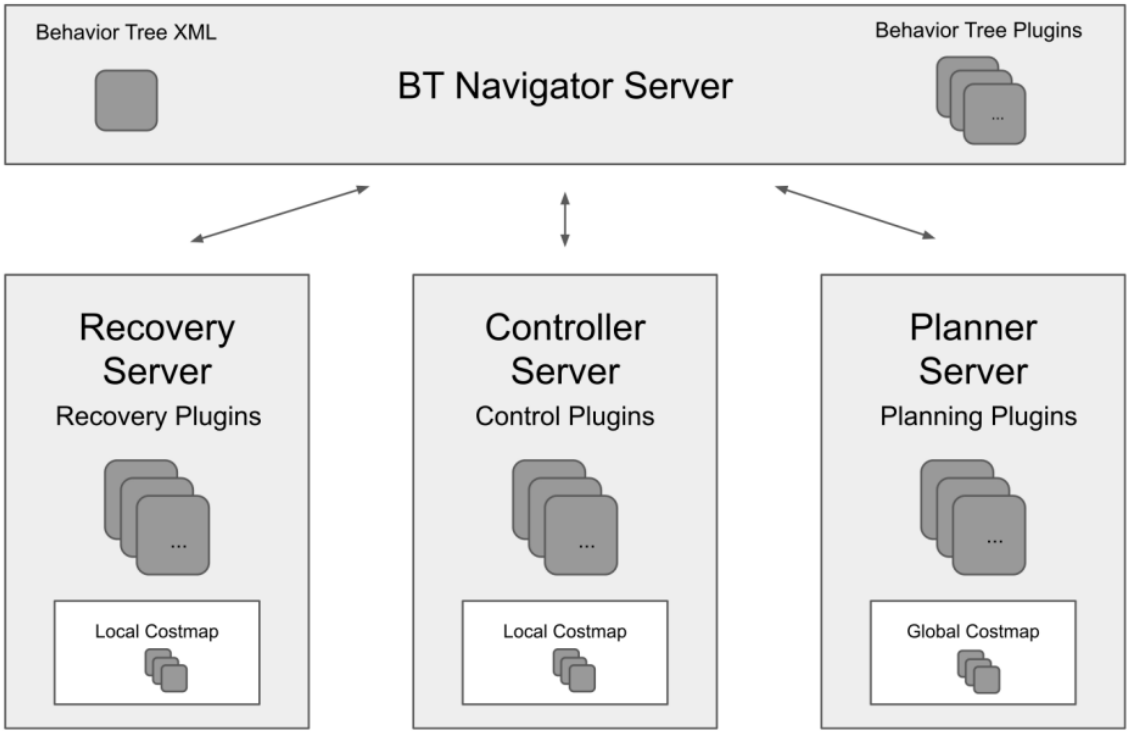


Figure 3.7. *Illustration of Nav2 design. [27].*

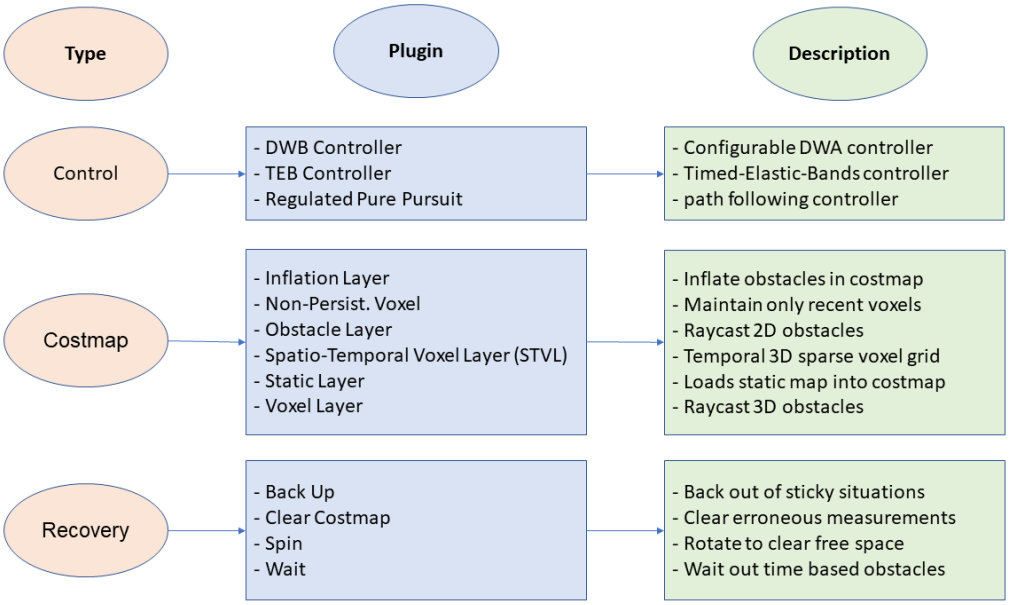


Figure 3.8. *Some of Nav2 plugins [27].*

3.8 Rviz and Gazebo

Rviz stands for ROS visualization; it is 3D visualization software created by ROS developers as a tool for robots. This tool enables the user to see the perception of the robot world and visualize the robot's states. Rviz accurately depicts the robot's world using sensor data. Rviz has some plugins to enable the user to monitor the robot's state and sensor

data.

Gazebo is a 3D robot simulator that simulates a robot and shows the user the realistic behavior of the robot similar, to a real-world environment. There is some confusion between Gazebo and Rviz because both show a 3d model of the robot, and both are used with ROS. Morgan Quigley, who is one of the Original ROS developers, stated that: "Rviz shows you what the robot thinks is happening, while Gazebo shows you what is really happening." [100].

4. PATH FOLLOWING STRATEGIES

Autonomous Vehicles (AVs) that are categorized in level three and above in the autonomy definition are known to be able to move on a specific path from one place to another. This process is a fundamental concept in AVs called path following or path tracking. Therefore, researchers tackled it for many years [101], and many control strategies were proposed to enable the AVs to follow a given path.

In this chapter we focuses on explaining the path following concept and some of its strategies. Also, some researchers' work related to this thesis topic are explained and at the end of the chapter, we state the difference between this thesis and the discussed research, then illustrate our contribution to this topic.

4.1 Path Following Strategies

In the path following, a control system is responsible for changing the vehicle's linear and angular velocities while moving to minimize the error in position and orientation between the vehicle and a desired point on the given path [102].

During the this process, the data is collected from the sensors such as speed encoders, IMUs and GNSS. Then it is used to form a closed-loop control scheme, when the control strategy is responsible for minimizing the errors.
In other words, given a reference path which is formed by n^{th} number of points each has position and orientation $\boldsymbol{x}_{ref}$, the path following aims to design control law such that $\lim_{t\to\infty} \|\boldsymbol{x} - \boldsymbol{x}_{ref}\| = 0$, where $\boldsymbol{x}$ is the vehicle state (position and orientation). When the vehicle model is represented by the differential equation $\dot{\boldsymbol{x}} = F(\boldsymbol{x}, u)$ and $\dot{\boldsymbol{x}}$ represents the vehicle state time derivative (linear and angular velocities).

To implement path following for AVs, There are many control strategies (for example, Pure Pursuit Control (PPC) [103], Stanley controller [104], Feedback Linearisation (FL) [105], Lyapunov's Direct Method (LDM), Linear Quadratic Regulator (LQR) [106], Model Predictive Control MPC [107], etc.). The mentioned strategies are popular due to the high performance they provided in many studies. In the path following, the complexity of the vehicle model, as well as the precision and robustness of the control strategy, are factors that affect the performance [102].

4.1.1 Pure Pursuit Control (PPC)

Pure pursuit is an algorithm determining the curve that brings a vehicle from its present location to the desired position. The algorithm aims to select a goal point at a specified distance ahead of the vehicle on the path. The name of this algorithm is derived from the intended behavior of its process.

PPC was originally developed for guiding missiles toward moving targets [108] and then used to ensure the vehicle follows a desired point on the path. While the point moves on the path, the vehicle keeps pursuing that moving point. The distance between the desired point and the vehicle along the vehicle's longitudinal axis is called look-ahead distance [109].

The method of achieving the pure pursuit is finding the curve that will drive the vehicle to a predetermined path point, known as the goal point. The pure pursuit algorithm has straightforward outlines as follows [110]:

- Determining the vehicle's position which is the rear axle center point.
- Finding the closest point on the path to the vehicle.
- Transforming the closest point's position to vehicle coordinates.
- Calculating the curve that connect the vehicle and the point.
- Changing the steering so that the vehicle movement matches the curve.

The pure pursuit control geometry of a bicycle model is shown in Fig 4.1. In that figure l_d is the look ahead distance and R is the curve radius. PPC aims to find the needed front tire steering angle δ to follow the curve and reach the goal point as shown in the following equation [110]:

$$\delta = \tan^{-1}(\frac{2L\ \sin(\alpha)}{l_d}) \tag{4.1}$$

Where L is the distance between the front and rear tire axle. α is the angle between the forward vector of the vehicle and the vector that connects the goal point and the rear axle of the vehicle and can be calculated as follows [110]: Where l_e is the lateral error that represent the distance between the goal point and the vehicle's rear axle.

4.1.2 Stanley Controller

Stanley controller is a geometry-based path following introduced in 2005 [111], also known as "Hoffman Controller." Unlike the PPC, this controller depends on the vehicle's front axle center as a reference point. The controller determines the errors in heading

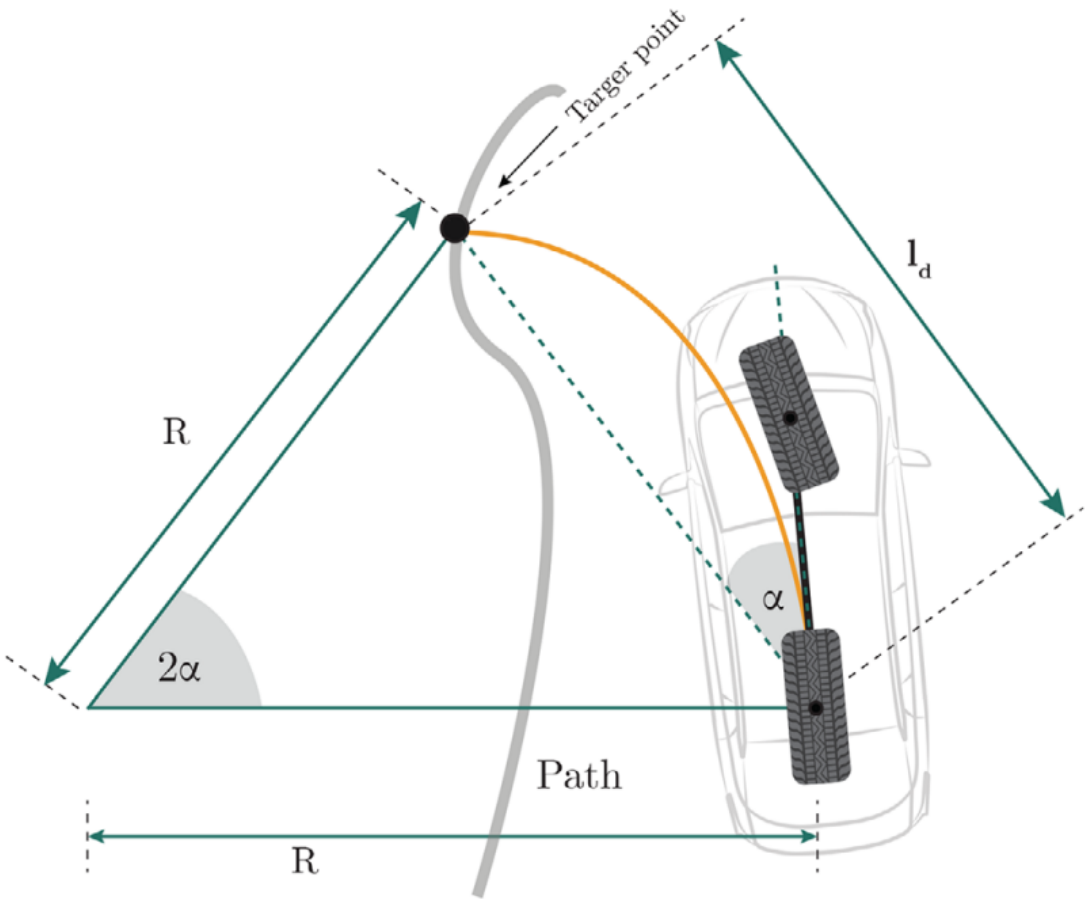


Figure 4.1. *The pure pursuit control geometry of a bicycle model [54]*

.

and orientation with the closest point on the path, then corrects the errors and saturates the steering signal according to the limits. The stanley controller geometry of a bicycle model is shown in Fig 4.2.

In that figure, e_l is the distance between the vehicle's front axle center p_c and the desired point on the path p_d and θ is the angle between the vehicle's heading and the tangent vector at the p_c. At the forward speed of the vehicle v, the required steering angle ψ to follow the path is calculated as [112]:

$$\psi = \theta(t) + tan^{-1}\left(\frac{k\, e_l(t)}{v(t)}\right) \tag{4.2}$$

where k is a tunable gain and the steering angle can not exceed the physical limitations of the steering system so that in the controller, if ψ exceeds the limits, it is set to the maximum value ψ_{max} to the turning direction as follows [112]:

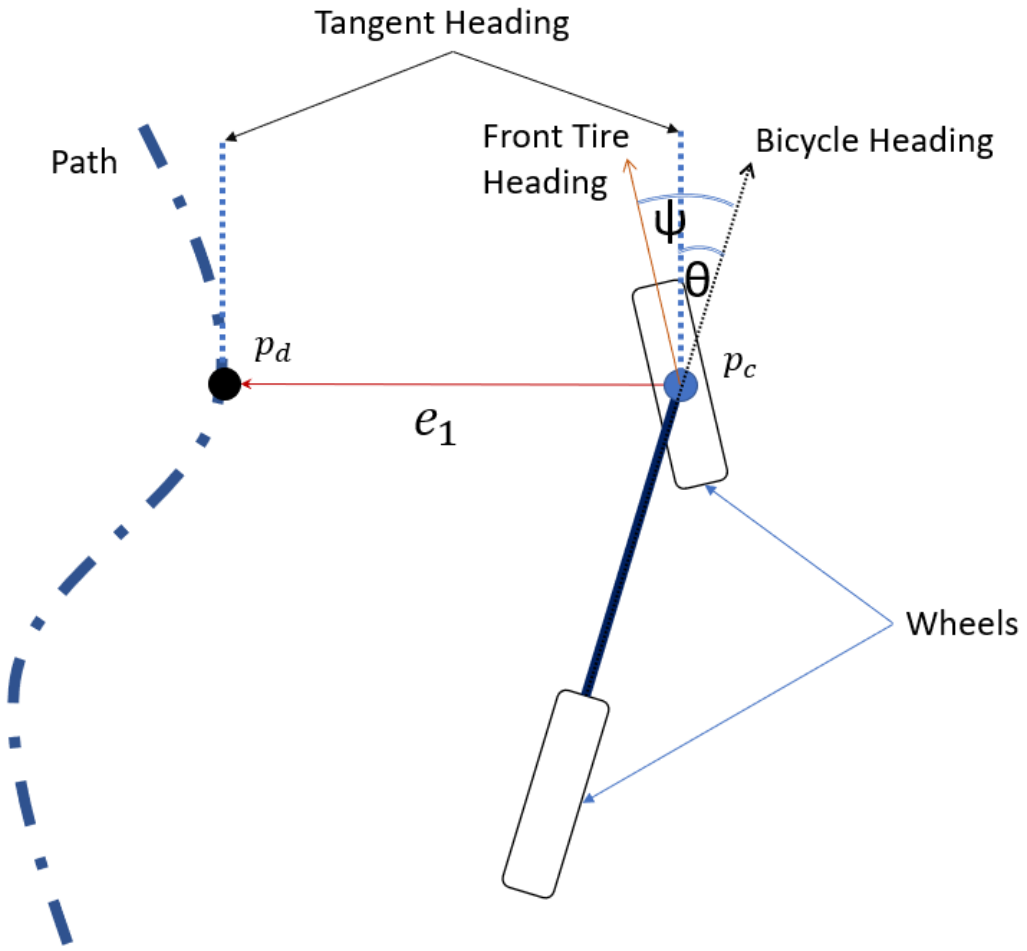


Figure 4.2. *The stanley controller geometry of a bicycle model. Adopted from [54]*

.

$$\psi(t) = \begin{cases} \theta(t) + tan^{-1}\left(\frac{k\, e_l(t)}{v(t)}\right) & if \left|\theta(t) + tan^{-1}\left(\frac{k\, e_l(t)}{v(t)}\right)\right| < \psi_{max} \\ \psi_{max} & if \left(\theta(t) + tan^{-1}\left(\frac{k\, e_l(t)}{v(t)}\right)\right) \geqslant \psi_{max} \\ -\psi_{max} & if \left(\theta(t) + tan^{-1}\left(\frac{k\, e_l(t)}{v(t)}\right)\right) \leqslant \psi_{max} \end{cases} \tag{4.3}$$

4.1.3 Model Predictive Control (MPC)

MPC predicts the state of the system in some finite horizon in the future at each time step using the dynamic of the system. Based on the predicted state, the aim of the controller is to determine the optimal sequence of the control inputs. According to the MPC a cost function 4.4 needs to be minimized when the constrains 4.5 are satisfied [113].

$$\min_{x_{i|k}\,,\,u_{i|k}} \; J(u_{i|k}\,,\,u_d(k)) \tag{4.4}$$

$$x_{0|k} = x(k) \tag{4.5a}$$

$$x_{i+1|k} = f(x_{i|k}\,,\, u_{i|k}) \tag{4.5b}$$

$$x_{i|k} \in x \tag{4.5c}$$

$$u_{i|k} \in u \tag{4.5d}$$

where $x_{i|k}$ is the state predicted i time steps ahead, at time k. The constrains are as follows:

- (4.11a) is the initial state.
- (4.11b) according to this dynamic the system is predicting along the horizon.
- (4.11c) and (4.11d) are the state and input constrains.

In general, linear and non-linear approaches were used to implement MPC for path following. The linear approach used with linear approximation of vehicle model and the non linear used with nonlinear vehicle model.[54]

4.2 Path Following for AFS AVs in Literature

In this section we will present shortly some of the research that focus on the path following for AFS AVs. Some of the researchers used ROS1 framework to implement the controller.

During the last few decades, numerous types of AVs with different degrees of autonomy and different steering types have been researched. The AVs with articulated frame steering have been approached from a variety of perspectives. The path following problem has been the focus of many research (for example, [114][115][116][117][118][119][120] [121][122][123][124], etc), as it fulfills one of the basic requirements for AVs.

Yongming Bian and Meng Yang et al. [125], proposed path following control based on the Lyapunov stability method for articulated drum roller where the control system was evaluated using Matlab/Simulink then experimented in an uneven construction site. The simulation and experimental results showed that the errors converged to 0 over time, and the vehicle followed the path.

A simple path following control was developed by Glen Rains, and Adam G. Faircloth et al. [126], for articulated steering vehicle using pure pursuit. The implementation was tested using five different paths with two speeds and three control pulse signals. The results showed successful path following at low speeds with full pulse signal on the straight and sinusoidal paths.

Some types of articulated steering mobile working machines are used in farming, and

autonomizing these vehicles is beneficial for reducing the workload and increasing safety in the working area. Kadeghe Fue and Wesley Porter et al. [127], were motivated to provide a solution for the cotton farm owners who have small lands and cannot afford large tractors and harvesting equipment to harvest their cotton. A Multi-Purpose Rover (MPR) with a manipulator and sensors attached, as shown in Fig 4.3, was developed as a solution to navigate along crop rows autonomously and harvest undefoliated cotton. A path following control based on a regulated pure pursuit strategy was implemented using ROS1. The verification tests were performed using the MPR in an actual cotton field. The results showed that the developed MPR and the path following controller are reliable and can be used to harvest the cotton and by following the cotton rows in the fields .[127]

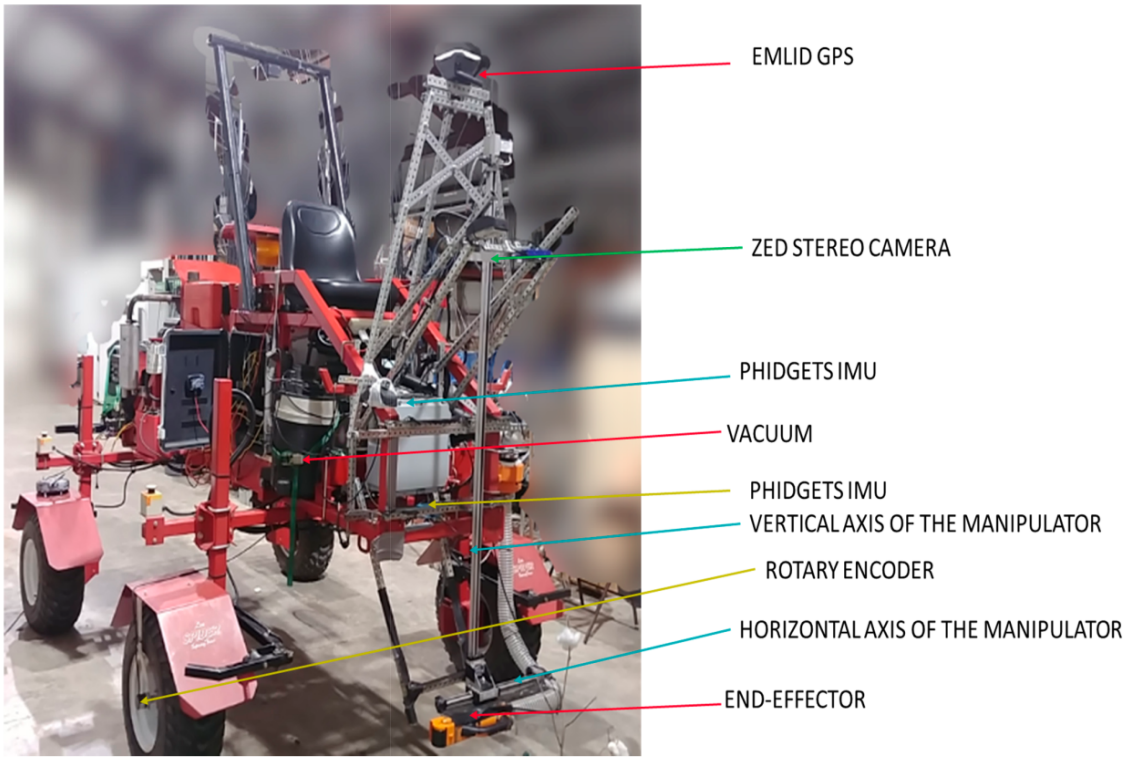


Figure 4.3. *The MPR with sensors attached and the manipulator [127].*

Julius K.Kolb, Gunter Nitzsche, and Sebastian Wagner [128] have developed a simple path following control using ROS1 for an Autonomous two-axle semi-trailer truck with a trailer. The Authors stated that, the trailer had only a minor effect on the controller and tests results showed sufficient tracking.

M. Murillo and G. Sánchez et al. [129], presented a path following controller for a hybrid steering tractor. The controller-targeted scenario where the tractor is toeing a trailer. The path following implementation is based on a predictive control strategy. The Authors archived the trailer's position control in addition to the tractor's position. ROS and Gazebo simulation environments were used to test the implemented control system. The simulation test showed an accurate path following.

To achieve safe autonomous movement for an articulated dump truck in mining environments, Bilal Hejase and Umit Ozguner et al. [130], developed testing simulation environment, path following, and vehicle module with taking into account the two frames of the articulated machine. The path following implementation is based on path tracking criterion from [131]. According to the experiments and testing results, the implemented system supported the vehicle model and showed successful path following.

4.3 Contribution

In some of the discussed research from the previous section, ROS1 was used but not ROS2 and the researchers did not focus on implementing a modular path following systems. Therefore, implementing modular path following and motion control for articulated mobile working machines using ROS2 is the main contribution of this thesis. The path following and the motion control are implemented as a separated ROS2 nodes without using Nav2. The path following node receives the sensor data and the desired path, then produce linear and angular velocities (v) and (w) as outputs that are needed to follow the path. The motion control node receives the generated speeds from the path following and the sensor data to generate the needed commands for steering the AFS vehicle. This architecture allows the path following to work with any steering vehicle by replacing the motion control node with other suitable controller that fits the steering type needed commands. The motion control is designed and implemented for any articulated steering vehicle. therefore, replacing the path following node by any other implementation as long as the outputs are v and w, is easily achievable.

5. MODELING AND CONTROL DESIGN

For designing and implementing a control system for an autonomous vehicle to perform a specific function, it is essential to model the vehicle, understand the kinematics, specify the required function, choose the control strategy to perform the task, and identify the hardware and software needed for the task.

The main focus of this chapter is implementing a path following and motion control for autonomous articulated vehicles. In the next sections, we will explain the path following, as a generic strategy that can be used for any type of ground vehicle. Then the motion control is illustrated by driving the kinematics of the AFS vehicle, modeling the steering actuator, which is essential to implement the motion control. Then we will illustrate the motion control mathematical equations and control signals.

5.1 Path Following Implementation

The implemented path following algorithms in this thesis is developed by Reza Ghabcheloo [2]. This method is based on the assumption that the vehicle is following an imaginary point P moving on a given path. The position of that point on the path is determined with respect to normalized value $s \in [0, 1]$. Where $(s = 0)$ at the beginning of the path and $(s = 1)$ at the end of the path. In order to calculate the linear and angular velocities (v, w) required by the vehicle to follow the path, the following procedure is proposed:

1. Calculating the errors in heading and position between the a point on the the vehicle and P at specific s, as follows:

$$\begin{bmatrix} x_e \\ y_e \\ \psi_e \end{bmatrix} = e = \begin{bmatrix} {}^{T}_{W}\mathrm{R} & 0 \\ 0 & 1 \end{bmatrix} (q - q_d) \tag{5.1}$$

   Where q is the vehicle's position and orientation. The position and orientation of P at at specific s is q_d. The error in the position for both axis (x,y) are (x_e, y_e). Since the errors are calculated with respect to the world coordinate frame (the frame of the given path and the vehicles position). They are translated using the rotation matrix $({}^{T}_{W}\mathrm{R})$ to q_d tangent frame, where the axis x_p represent the heading at the

point on the path at s as shown in Fig 5.1.
The difference between the vehicle's heading angle ψ_h and the heading of a point ψ_T on the path at s is ψ_e, which can be calculated as:

$$\psi_e = \psi_h - \psi_T \tag{5.2}$$

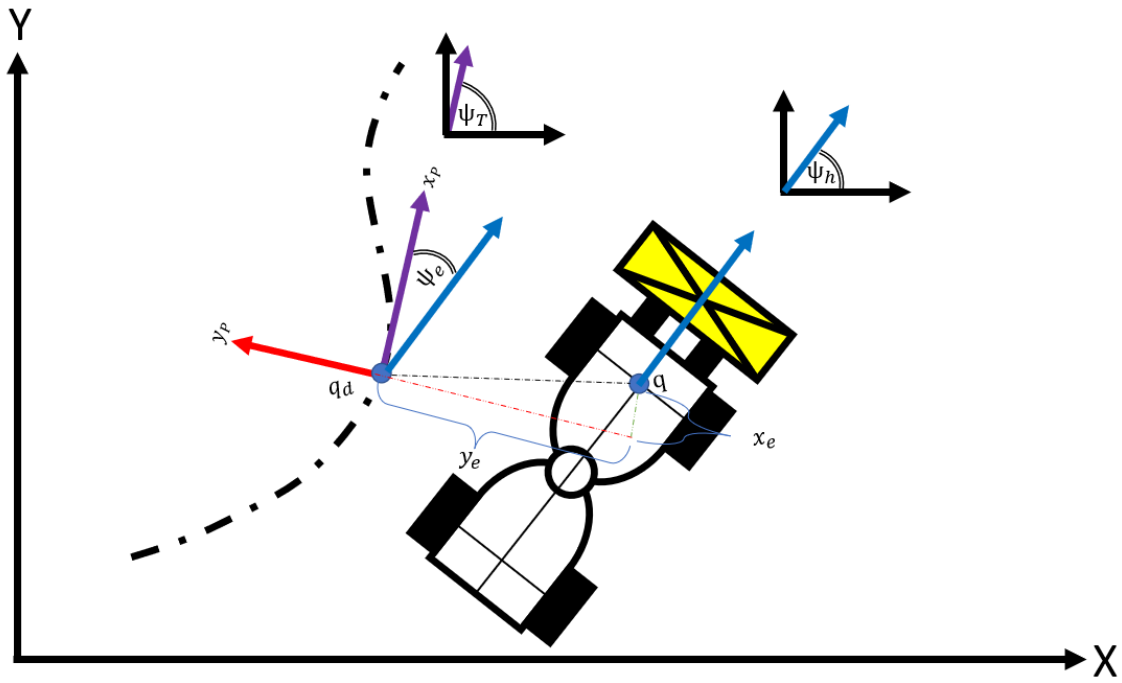


Figure 5.1. *Path following strategy*

2. The path following error dynamics are:

$$\begin{cases} \dot{x_e} = \dot{s}\,(y_e\,C_c(s) - 1) + v\,cos(\psi_e) \\ \dot{y_e} = v\,sin(\psi_e) - \dot{s}\,x_e\,C_c(s) \\ \dot{\psi_e} = w - \dot{s}\,C_c(s) \end{cases} \tag{5.3}$$

where $C_c(s)$ is the curvature at a specific value of s, and (w, v) are the measured linear and angular velocities of the vehicle

3. To determine q_d on the path, s have to follow the dynamic below:

$$\dot{s} = v_h\,cos(\psi_e) + kx_e \tag{5.4}$$

where $\dot{s}$ is the derivative of s.

4. The linear velocity command (v_c) is non-linear function of the heading error and the reference velocity (v_{ref}), as shown below:

$$v_c = F(v_{ref},\, \psi_e) \tag{5.5}$$

Where $F(v_{ref},\, \psi_e)$ is a function that insures the speed is reduced when the error is high.

5. The angular velocity command w_c is calculated as:

$$w_c = C_c\, \dot{s} - k_e(\psi_e - \psi_{los}) \tag{5.6}$$

where k_e is a gain related to the reference speed v_r, and ψ_{los} is angle depends on the line of sight los and it is calculated as follows:

$$\psi_{los} = \begin{cases} \frac{sign(y_e)\,\pi}{2} & |y_e| > \ los \\ -\sin^{-1}(\frac{y_e}{los}) & |y_e| \leq los \end{cases} \tag{5.7}$$

5.2 Motion Control

The path following controller explained earlier calculates the linear and angular speed commands ($v_c,\ w_c$). These commands allow the AFS machine to achieve the path-following task. Therefore, it is essential to control the machine's motion according to the speed commands. The motion control is implemented for that purpose, where the received commands and some sensor measurements are used to calculate the control signals that control the machine's steering, gas, and gear.

As explained earlier, AFS vehicles consist of a front part and a rear part connected using a free joint. The AFS vehicle has non-steerable wheels, and each vehicle part has a single axle. The kinematics model of the used vehicle in this thesis was driven under two assumptions. These assumptions are the vehicle is not skidding, and the vehicle's tires are not slipping.[132].

In this section, the kinematics models of the vehicle and the steering actuator are driven and explained to finally explain the motion control signals.

5.2.1 Kinematics Model

It is essential to model a machine for implementing a control system that controls it accurately. The modeling design is a process that includes multiple parts, such as:-

- Developing and analyzing the model to describe the behavior.
- Designing a suitable controller for the dynamic motion of the machine.
- Simulating the dynamic motion.

- Deploying the controller.

The kinematics model is driven based on the model from [132]. Each part of the vehicle is indexed by i, which refers to f (front) and r (rear). In Fig 5.2 the AFS vehicle has two parts, each has a length $\left|\overrightarrow{OQ_f}\right| = l_f$, $\left|\overrightarrow{OQ_r}\right| = lr$ that represent the distance between the center of the articulation joint (O) and the center of the related part wheel's axial (the centers of gravity of the part denoted as $Q_f = (x_f, y_f)$, and $Q_r = (x_r, y_r)$). The steering method is simply turning each part against the other in the yaw plane using the articulation joint [132]. Generally, the AFS vehicle is designed in such a manner ($l_f = l_r$), that both parts of the vehicle move on the same turning route, eliminating off-track error and making path following simpler. Turning the vehicle is achieved by changing the articulated angle β at the joint. [133]

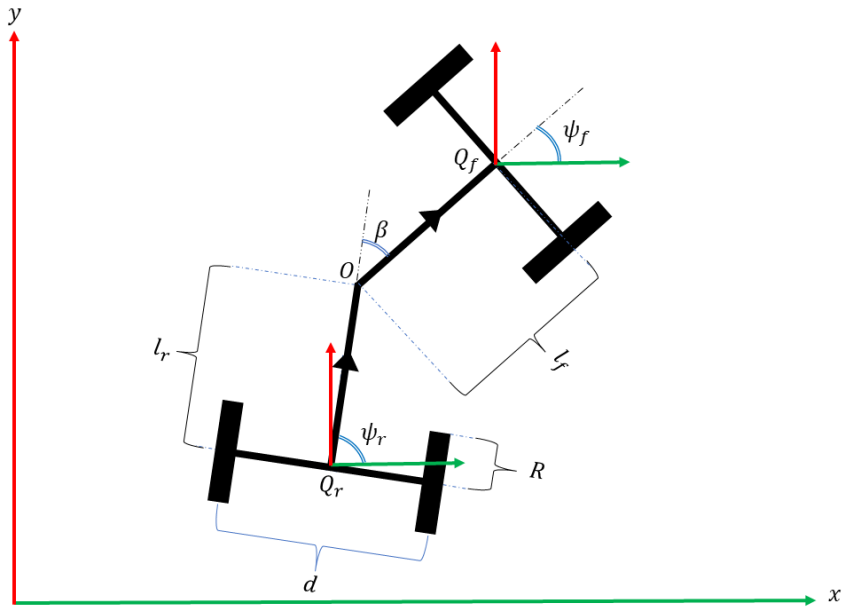


Figure 5.2. *Articulated vehicle kinematics*

By looking to Fig 5.2, the AFS vehicle's heading normally determined as the angle ψ_f between the reference world frame axis (in this figure $x - axis$) and the vector $\overrightarrow{OQ_f}$ of the front part. The joint angle can be calculated as

$$\beta = \psi_f - \psi_r \tag{5.8}$$

By denoting v_i as linear velocity and w_i as angular velocity, the kinematics equation of each part under the stated assumptions is determined by

$$\begin{aligned} \dot{x_i} &= v_i\, cos(\psi_i) \\ \dot{y_i} &= v_i\, sin(\psi_i) \\ \dot{\psi_i} &= w_i \end{aligned} \tag{5.9}$$

where $\dot{x_i}$ and $\dot{y_i}$ are the the linear velocities of each part of the vehicle along $x - axis$ and $y - axis$ respectively.

From Fig 5.2 the relationship between the front and rear coordinates is defined as follows:

$$\begin{aligned} x_f &= x_r + l_f\, cos(\psi_f) + l_r\, cos(\psi_r) \\ y_f &= y_r + l_f\, sin(\psi_f) + l_r\, sin(\psi_r) \end{aligned} \tag{5.10}$$

By derivate 5.10 and 5.9 and using 5.8 we get

$$\begin{aligned} v_f\, cos(\psi_f) &= v_r\, cos(\psi_r) - l_f w_f\, sin(\psi_f) - l_r w_r\, sin(\psi_r) \\ v_f\, sin(\psi_f) &= v_r\, sin(\psi_r) + l_f w_f\, cos(\psi_f) + l_r w_r\, cos(\psi_r) \\ \dot{\beta} &= w_f - w_r \end{aligned} \tag{5.11}$$

where $\dot{\beta}$ is the articulated angle rate of change.

When the vehicle is turning, the tires on each side of the vehicle will move on different curve. the outer tires moves on a wider circle than the inner tires, as shown in Fig 5.3.

From 5.11 and under the assumption that the vehicle has no slipping and no skidding the following kinematics are valid.

$$\begin{aligned} v_r &= v_f\, cos(\beta) + l_f\, w_f\, sin(\beta) \\ l_r\, w_r &= -l_f\, w_f\, cos(\beta) v_f\, sin(\beta) \end{aligned} \tag{5.12}$$

$$l_r\, w_r = -l_f w_f\, cos(\beta) + v_f\, sin(\beta) \tag{5.13}$$

$$\begin{aligned} w_i &= \frac{R}{2d}\,(W_{i,r} - W_{i,l}) \\ v_i &= \frac{R}{2}\,(W_{i,r} + W_{i,l}) \end{aligned} \tag{5.14}$$

where $W_{i,r}$ is the rotational speed of the right tire and $W_{i,l}$ is the rotational speed of the left tire.

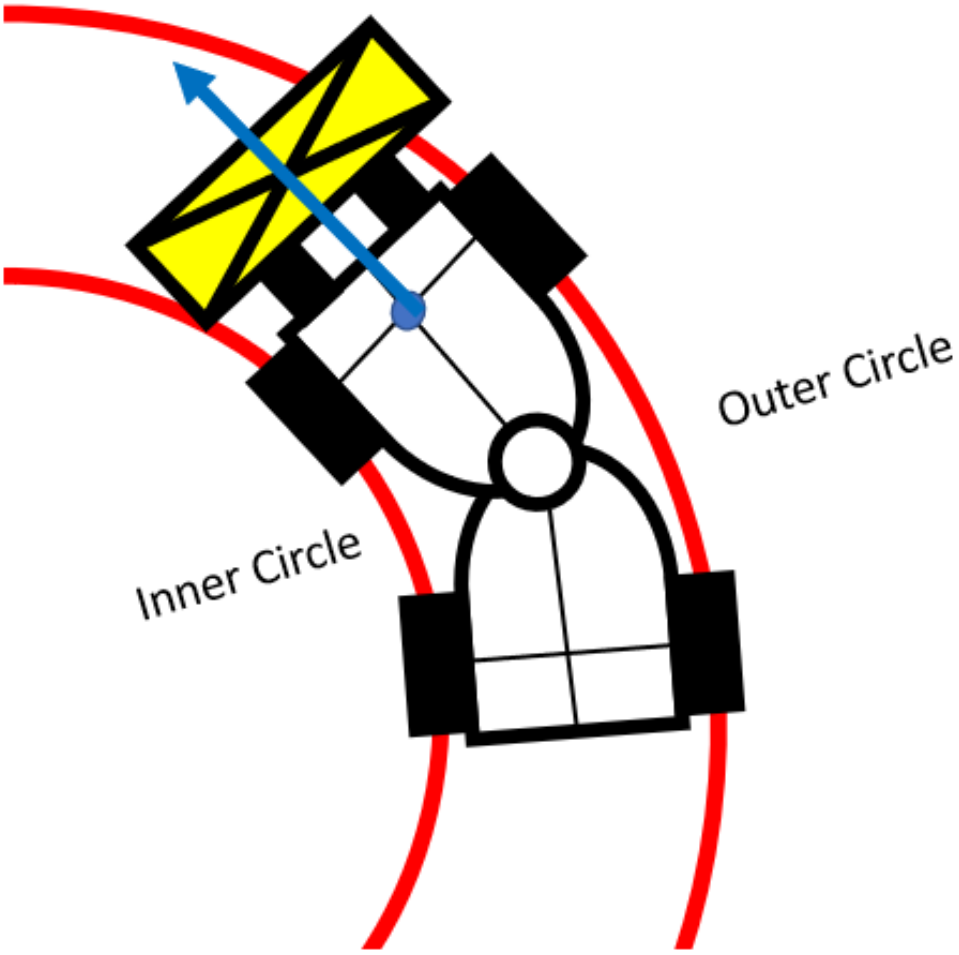


Figure 5.3. *AFS Machine on circular path. Both circles have the same center.*

5.2.2 Steering Actuator Model

In this part a simplified model is derived to provide guideline for designing a motion control strategies. Modeling of the steering actuator is adopted from [132]. The steering of the articulated vehicle is driven by a hydraulic actuator (cylinder), which is shown in Fig 5.4. In that figure, three joints, A, B, and O, are connected to form a triangle. Two sides of the triangle OA and OB has fixed distance a and b, but the third side AB has changing distance Γ. The actuator is responsible of changing Γ, and as a result changing the angle γ that is related to β.

The flow to the actuator in Fig 5.4 is controlled using a proportional compensated valve, where the valve's flow is proportional to the control signal as shown in Fig 5.5.[132]

Since the control signal u is proportional to the flow, then it can be written as

$$u = k\,\dot{\Gamma} \tag{5.15}$$

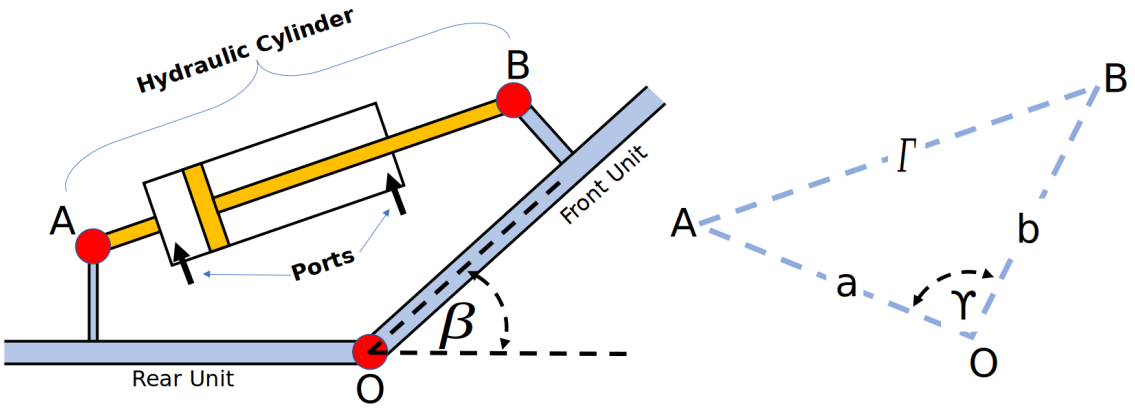


Figure 5.4. *AFS mechanism adopted from [132]*

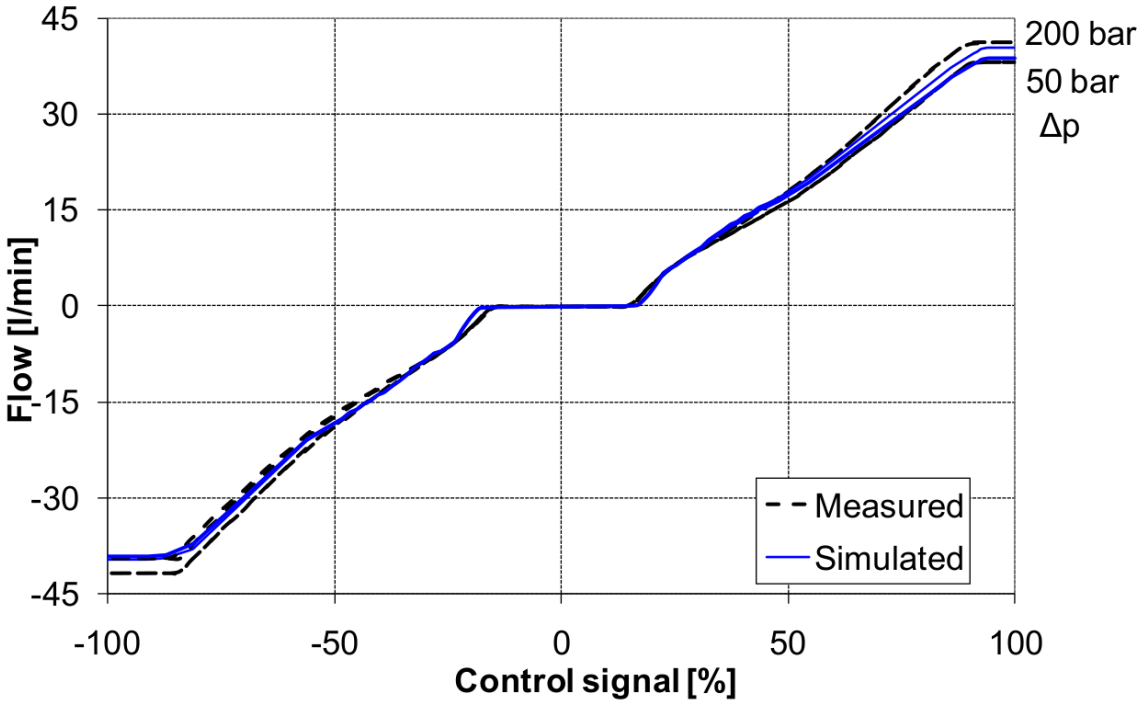


Figure 5.5. *The pressure compensated proportional valve characteristic curve [132]*

where $(k > 0)$ and $\dot{\Gamma}$ is the rate of change in the distance Γ, and $u \in [-1, 1]$, as the saturation effects [134] is noticeable from Fig 5.5.
Using simple trigonometry, the rate of change in the distance Γ can be written as

$$\dot{\Gamma} = \frac{a\, b\, sin(\gamma)}{\sqrt{a^2 + b^2 - 2\, a\, b\, cos(\gamma)}}\, \dot{\beta} \tag{5.16}$$

where γ is related to β such that $\gamma_o = \gamma + \beta$ according to that and from 5.17 and 5.18 it is noticeable that the control signal u to $\dot{\beta}$ is nonlinear function of β, then 5.17 can be written as:

$$u = K(\beta)\, \dot{\beta} \tag{5.17}$$

5.2.3 Motion Control Signals

The motion control signals in this part are proposed for ideal articulated frame steering machine, where 5.12,5.13, 5.14 and 5.17 are fulfilled[132].
The path following strategy in 5.1 is generic controller that relate specific point in the vehicle frame (q) to desired point on the path (q_d). In our implementation (q) is the vehicle's front axial center point that has position (x_f, y_f) in the global frame. Therefor, the angular velocity command calculated from the path following controller in 5.1, is (w_f^c), and using 5.13 it is translated to ($\dot{\beta}^c$), then using 5.17 translated to (u). Let

$$\dot{\beta}^c = -\frac{v_f}{l_r}\, sin(\beta) + ((\frac{l_f}{l_r}\, cos(\beta)) + 1)\, w_f^c \tag{5.18}$$

Then the control signal given by

$$u = K(\beta)\, \dot{\beta}^c \tag{5.19}$$

Beside the control signal (u) which is the steering command, there are two other commands for controlling the vehicle's gear and speed. The gear command represent the required movement direction of the vehicle and it is determined by v_c. The gas command is a saturated value $\in [0, 1]$ that obtained by interpolating the required speed to get the speed actuator equivalent signal. The speed command constrained using rate limiter, which limits the acceleration and deceleration of the machine.

6. CONTROL SYSTEM IMPLEMENTATION

The motion control and the path following implemented controllers in this thesis are intended for controlling an AFS vehicle (AVANT 653 shown in Fig 6.1). That vehicle was modified at Tampere university to be autonomous for research purposes. Therefore, a power transmission, multiple sensors, and controllers were integrated into the machine to control it autonomously. The only original parts that remain in the machine are the metal frame and the hydraulic actuators. In its current state, the vehicle can perform path following tasks autonomously and can be teleoperated using different controllers.

Figure 6.1. *Modified loader for research purposes. It is based on AVANT-635.*

The control system on this machine is divided into two levels, low and high as shown in Fig 6.2. In the low-level, there are hydraulic cylinders, pumps, valves, a diesel engine, sensors (IMUs, GNSS, Lidar, Cameras, resolver, pressure sensors, and speed encoders), and micro-controllers. The low-level controllers are connected to the sensors, actuators and the higher level using CANBus technology. These controllers collects the sensor data then sends them to the high-level controller and control the actuators depending on the received commands signals.

The high-level control system is implemented using ROS2 (chapter 3) on a computer with

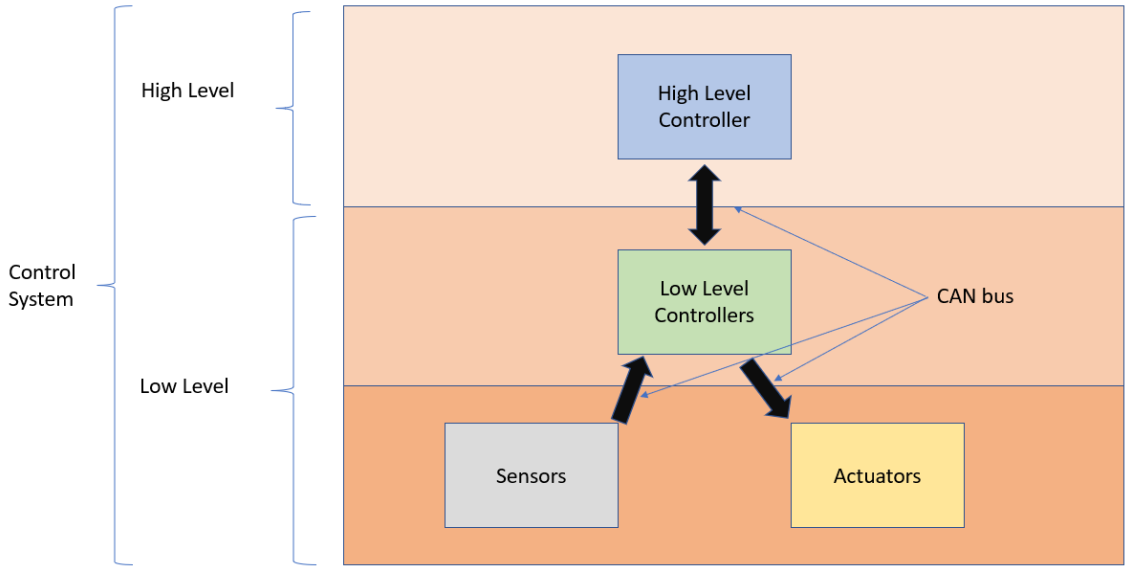


Figure 6.2. *Control levels in the autonomous articulated machine at Tampere university*

Linux operating system. That control system is designed and structured to have multiple ROS2 nodes, each representing a subsystem responsible for performing specific tasks. These nodes are named according to their functions as shown in Fig 6.3.

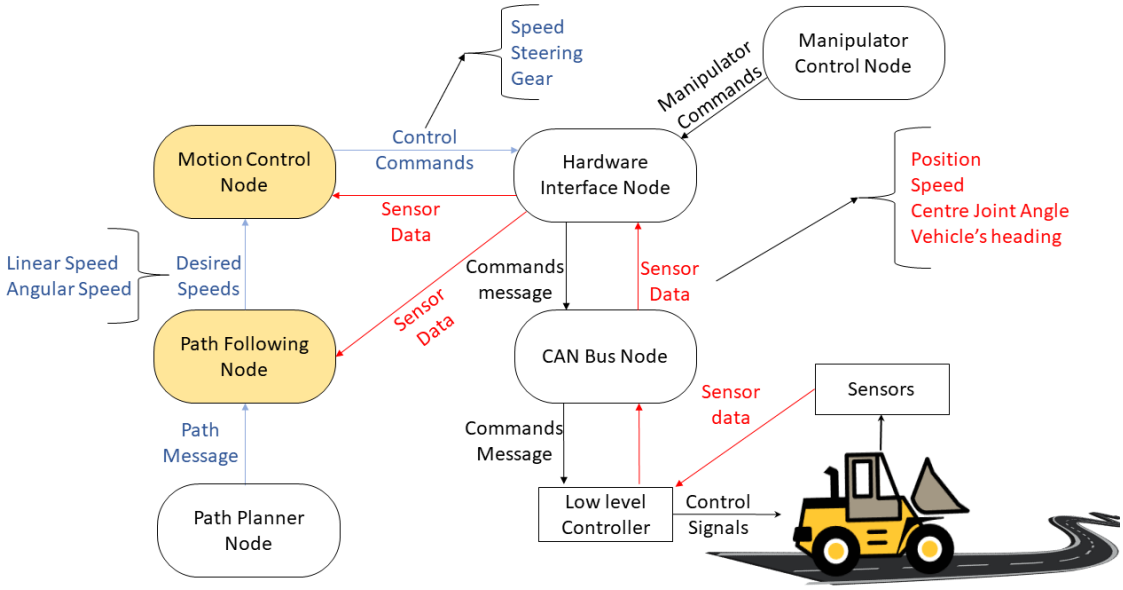


Figure 6.3. *Control system for AFS autonomous machine using ROS2.*

6.1 Structure of the High Level Control system

The high level control system is implemented using ROS2. This level is divided into subsystems and each is implemented as ROS2 node as shown in Fig 6.3. The path following and the motion control nodes are explained separately in other sections in detail

as they contains the implemented controllers of this thesis. In this section the other nodes in Fig 6.3 are demonstrated as follows:

- **CAN bus node**: The data such as speed, orientation, position, and center joint angle are delivered by the CAN bus to the CAN bus node. This node represents the bridge between the high-level control system and the CAN bus. It delivers the sensor data to the Hardware interface node and delivers the control commands (driving and manipulator commands) to the CAN bus.
- **Hardware interface node**: This node is responsible for parsing the messages that contain the sensor data and the control commands and packing them in suitable ROS messages, then sending them to the nodes that needs them.
- **Manipulator control node**: The used MWM is a loader that contains a tool (manipulator) for performing work tasks. The machine manipulator control commands are generated by the manipulator control node.
- **Path planner node**: This node is providing the required path as a Bezier curve which is represented by six control points. This node also provides movement direction and five profile speeds which are interpolated to determine the desired speed at every specific point on the path. The mentioned data are sent to the path following node to perform the path following task.

6.2 Path Following Node

The path following node is designed as ROS2 action server. The purpose of implementing this node as action server is to perform a path following task that requires relatively long time to be completed and during the task progressing the server provide continuous feedback messages. In addition, that node provides the possibility to cancel the task and start new one according to the client request.

This node starts its task by reserving request message contains the path defined by six Bezier control points, the reference profile speeds (which are interpolated later to determine the desired speed at every specific point on the path), and direction signal from the path planner. This message fed to the path following node as an input to perform the path following according to the specified movement direction. The sensor data messages are received as additional inputs during the path following presses. The outputs of this node are desired linear and angular velocities that are published in a single ROS2 message. When the command message is sent to the motion control node, a feedback message is also sent to the client, to inform about the task progress. At the end of the task the client will receive a results message.

This node consists of some functions that are implemented to do the calculations. These functions are shown in Fig 6.4. The direction master receives the path message, and

according to the desired movement direction, it specifies the position and the orientation of the leading part of the vehicle. Then, the errors of the position and orientation with respect to the reference values, which are explained previously in 5, are calculated by the error calculations. According to these errors and the measured speed of the vehicle, the required speed function will calculate the desired linear and angular velocity as outputs.

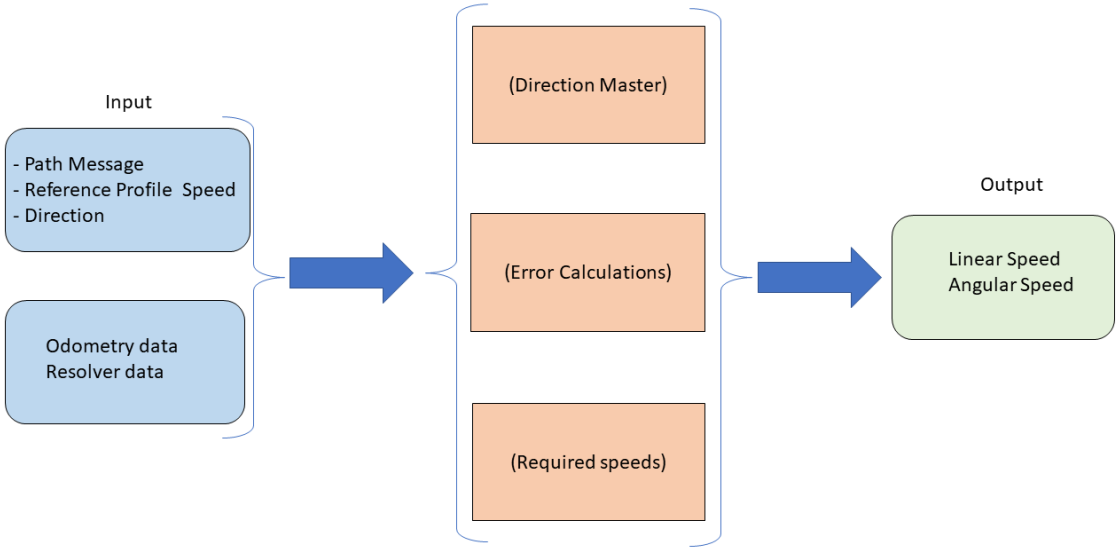


Figure 6.4. *The path following node structure.*

6.3 Motion Control Node

The motion control node is implemented to translate the wanted speeds generated by the path following nodes and produce control commands that can be used by the articulated vehicle to adjust the speed and steering angle and change the gear to achieve the desired movement. This node controls the vehicle acceleration and deceleration to provide smooth transition movement, also the node assures that the vehicle's steering angle is not exceeding the specified maximum steering angle. The needed inputs for this node are the desired speeds, measured joint angle, and heading speed of the vehicle. The output of the node is a control message that contains three commands, gear, steering, and gas.

Gear Command

The gear command $(u_{gear} = \pm 1)$ is controlling the direction of the movement, where u_{gear} is 1 for the forward movement and -1 for the backward movement. The node is implemented to ensure that the gear change is allowed only when the machine is stopped. Changing the direction of the movement when the machine is moving is taken care of by the node, where the node ensures reducing the speed to stop the vehicle, reverses the gear, and moves the vehicle in the required direction.

Gas Command

The gas command $(u_{gas} \in [0, 1])$ is calculated depending on the interpolation of the desired speed. The desired speed is effected by the acceleration limiter. The illustration of how u_{gas} response to the desired speed in this implementation shown in In Fig 6.5.

Steering Command

The steering command $(u_{steering} \in [-1, 1])$ calculated to control the valve that feeds the steering hydraulic actuator as explained previously in 3. The steering command value is proportional to the the difference between the desired joint angle and the actual measured angle. The command can be explained as follows:

1. $u_{steering} = 0$, when turning is not required and the current joint angle is at the desired value.
2. $u_{steering} > 0$, when a clockwise turn is required.
3. $u_{steering} > 0$, when a counterclockwise turn is required.

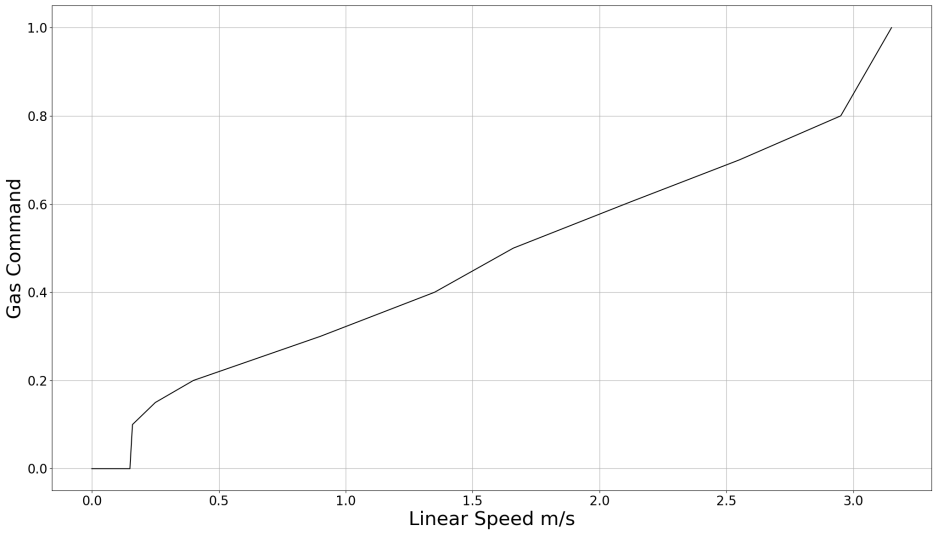


Figure 6.5. *The gas command at each speed.*

7. TEST AND RESULTS

The implemented path following and motion control are tested and validated using a simulation loader model developed at Tampere university and integrated into a real-world physics simulator (Gazebo) used in the Robot operating system. Another test was performed using the real articulated loader illustrated in the chapter 6. The evaluation is based on the performance of the implemented controller for different paths and the vehicle behavior during the process of following the path. A comparison was performed between the implemented path following and the modified pure pursuit path following from Nav2, to answer the second question raised in chapter 1 "How beneficial is using our proposed path following in comparison to another developed and commonly used strategy in ROS2 such as RPP?".

The tests were performed using the simulation model and real machine, where it is noticeable that their control systems have similarities and differences which will be explained and discussed in the next sections, besides diving into the testing scenarios and discussing the effects of some parameters on the performance of the implemented controllers.

7.1 Path Following Evaluation Method

We will test both controllers to evaluate the implemented controller and compare it to the Nav2 path. The controllers will receive the same paths, then when the vehicle moves on the path, its position and heading are recorded. The evaluation is performed by calculating the lateral maximum error (L_{maxe}), lateral root mean square error (L_{rmse}), the orientation maximum error (ψ_{maxe}), and the orientation root mean square error (ψ_{rmse}).

The lateral error represents the distance between the vehicle's position and the closest point position on the reference path, and it is used in many researches for path following evaluation (for example it is used in [135][136][137][138]). The orientation error is the angle difference between the vehicle's orientation and the tangent orientation of the closest point on the reference path. The root mean square error can be calculated as: $root\,mean\,square\,error = \sqrt{\sum_{i=1}^{n} \frac{1}{n}(x_i - \hat{x}_i)^2}$, where $(x_i$ is the reference state and $\hat{x}_i$ is the real measured state [139][140][141][142][135]. That means these errors are calculated as the differences between the reference given path and the actual path generated

by the vehicle movement. The performance of the controller depends on the mentioned errors. Where the one that has less error with respect to the reference path is considered to be better strategy for the articulated frame steering machines.

The tests in real world was performed before the simulator. Therefore, the parameters Los and v_r where tuned in the real wold test. This is the reason that in real world we performed tests with different values of the mentioned parameters. In the simulation only the tuned parameters used but with different paths.

7.2 Simulation Environment

In this part, we evaluate the performance of our implemented controller using the Gazebo simulator as mentioned earlier. The test is performed for two different paths to evaluate the performance of the controller in sharp and wide turns. Then, the test is repeated using RPP controller provided by Nav2. That path following controller is used for comparison, as it is one of the commonly used controllers in Nav2, which is a validated and tested.

The used vehicle model in simulation is designed and implemented based on AVANT-635 articulated loader. That model is integrated to be deployed and used in the Gazebo simulation environment and Fig 7.1 shows the model and the map in Gazebo.

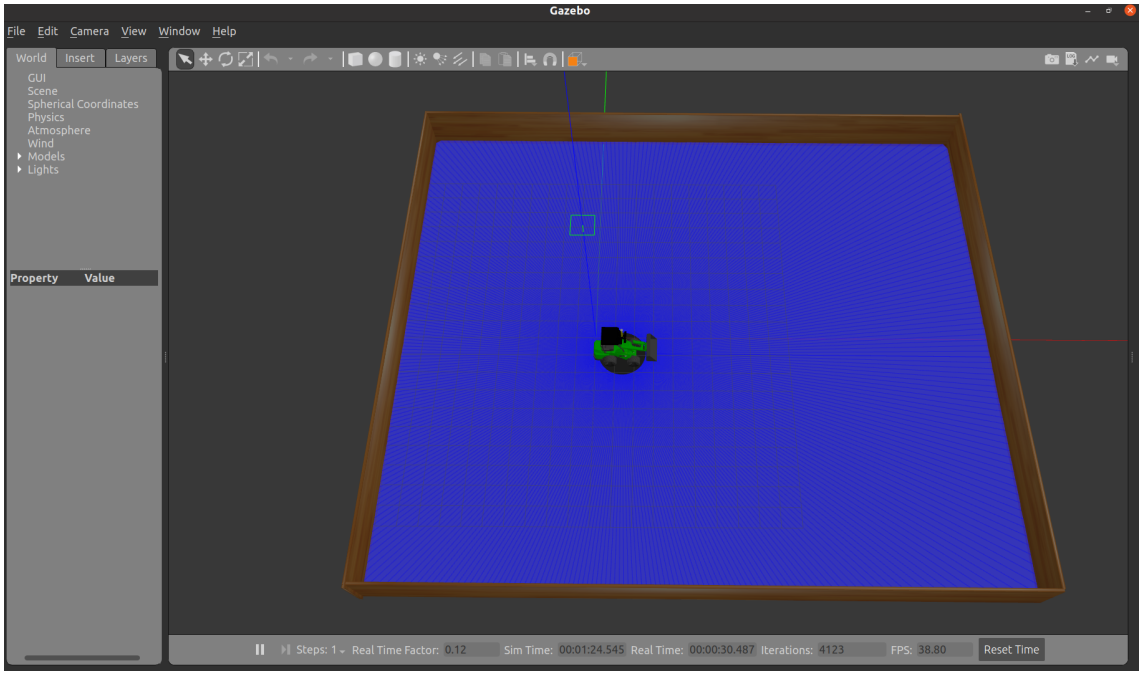


Figure 7.1. *Simulation environment in gazebo.*

In the simulation model, the steering is performed by changing the center joint angle that connects the vehicle's front and rear parts. Moreover, according to the kinematics of the articulated frame steering vehicles, the tires are controlled using a differential drive function. Therefore, the differential functionality is determined according to the center

joint angle and the speed to perform accurate turning movement. ROS Visualization (Rviz) tool and Gazebo simulator are used as user interfaces to track and monitor the vehicle behavior, but the data is recorded using ROS bags and analyzed separately.

Since the main object is focusing on the path following and the motion control nodes without diving into other subsystems, a simplified control system diagram is demonstrated in Fig 7.2. The gazebo simulator allows attaching various sensors to the vehicle simulation model that can be used for collecting data about the machine and its environment and helps to perform the path following task as the data is fed to the control nodes to achieve their tasks.

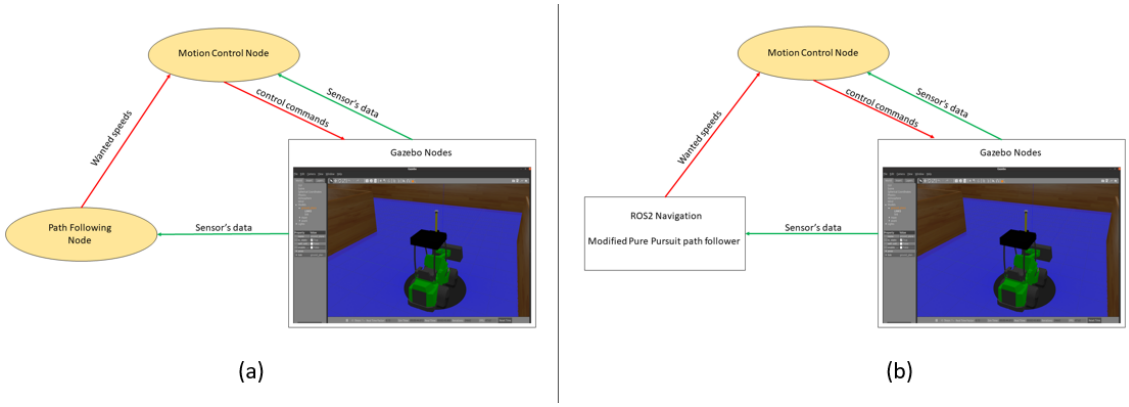


Figure 7.2. *Simplified diagram of the control systems for the simulation with: (a)the implemented path following and motion control, (b) the Nav2 and and motion control.*

7.3 Test Scenarios in Simulator

Using the simulator, two test scenarios were applied. In both scenarios a path defined by a fifth-order bezier curve is provided. The mentioned curve determined by six control points. The first path control points are shown in table 7.1. The path has a wide turns as shown in Fig 7.3 (a). The initial position of the machine model in the simulator is set to be the first control point P_0.

$ControlPoints$	P_0	P_1	P_2	P_3	P_4	P_5
x_{axis}	0	1	15	0	14	15
y_{axis}	-5	-5	-5	5	5	5

Table 7.1. *The control points of the first path for simulation test.*

For the second scenario, the path control points are shown in table 7.2.

The path defined by these control points has sharp turns as in Fig 7.3(b), that require a high steering angle that exceeds the joint angle limitation; this allows us to observe the vehicle's behavior when it gets out of the path due to some physical limitations. Moreover,

$ControlPoints$	P_0	P_1	P_2	P_3	P_4	P_5
x_{axis}	0	7.5	15	0	7.5	15
y_{axis}	-5	-5	-5	5	5	5

Table 7.2. *The control points of the second path for simulation test.*

we will be able to see if it gets back to the track and the distance needed for achieving this.

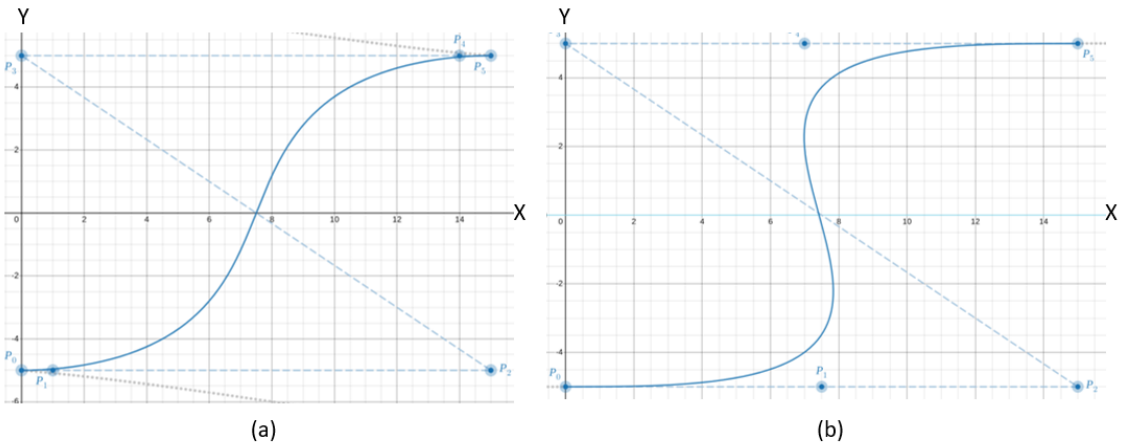


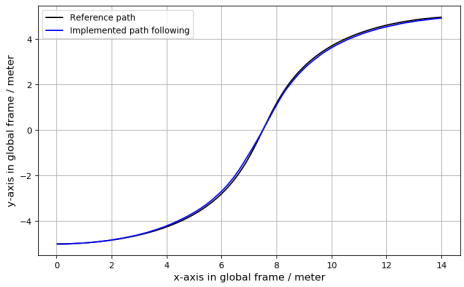
Figure 7.3. *The given paths defined by six control points and the distance measured in meters. a) Path with wide turns. b)Path with sharp turns.*

7.3.1 Simulation Test and Results

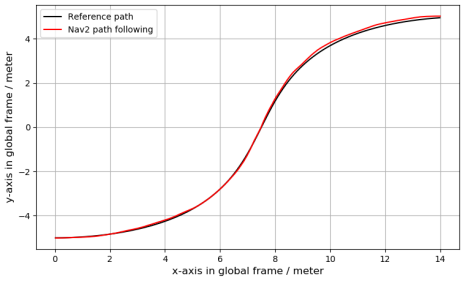
For both scenarios, our implemented path following node received command message to follow one of the Paths shown in Fig 7.3. The reference profile speed $(v_r = 0.5m/s)$ and the desired movement direction is forward, meaning the front body of the vehicle is the leading part. The path following sent the needed speeds to the motion control node, which translated them to control commands and fed them to the model in the simulator. During the path following process, we recorded the model's position, linear speed, and heading to analyze them. The data showed that the simulation model followed the path with high performance. The path formed from the recorded model's positions is plotted with the given input path to illustrate the model's behavior. The same test using MPP path following from Nav2 was repeated for both scenarios. The path following tasks performed by both controllers and can be seen in Fig 7.4 and Fig 7.5

Form the recorded data, the lateral and orientation errors is calculated from the beginning till the end as shown in Fig7.6.

The errors shown in Fig7.6 used to calculated (L_{maxe}),(L_{rmse}), (ψ_{maxe}), and (ψ_{rmse}). The results of the first scenario from the simulation can be seen from table 7.3 and Fig 7.7. The results of the second scenario are shown in table 7.4 and Fig 7.8.

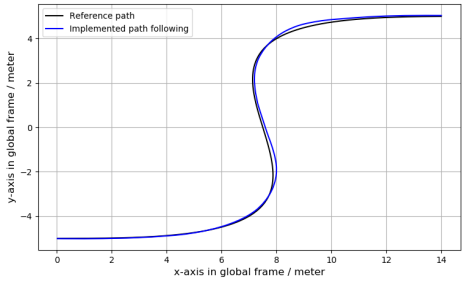


(a) *The implemented controller*

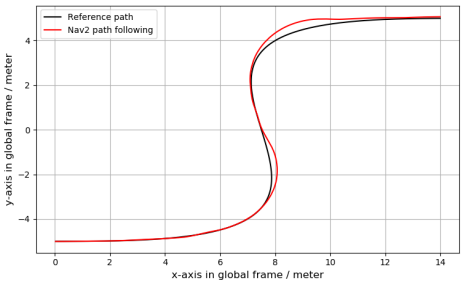


(b) *Nav2 controller*

Figure 7.4. *The first given path (Fig 7.3.(a)) as a reference vs the actual recorded path from the simulator.*



(a) *The implemented controller*



(b) *Nav2 controller*

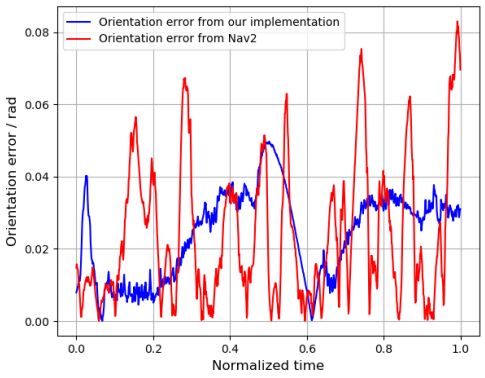
Figure 7.5. *The second given path (Fig 7.3.(b)) as a reference vs the actual recorded path from the simulator.*

$Calculated\,Errors$	$L_{maxe}\,(m)$	$L_{rmse}\,(m)$	$\psi_{maxe}\,(rad)$	$\psi_{rmse}\,(rad)$
$Implemented\,Controller$ $at\;Los = 2.5$	0.073	0.050	0.050	0.027
$Nav2\,Controller$	0.122	0.062	0.083	0.031

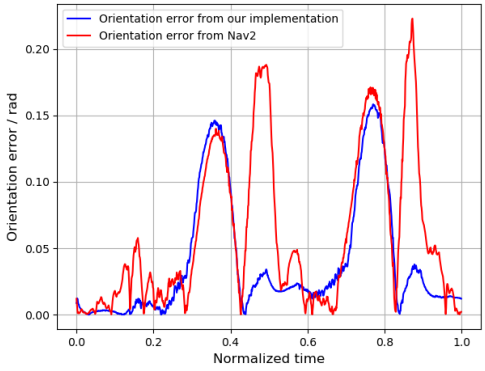
Table 7.3. *The calculated errors from the first simulation scenario test.*

$Calculated\,Errors$	$L_{maxe}\,(m)$	$L_{rmse}\,(m)$	$\psi_{maxe}\,(rad)$	$\psi_{rmse}\,(rad)$
$Implemented\,Controller$ $at\;Los = 2.5$	0.157	0.76	0.158	0.060
$Nav2\,Controller$	0.372	0.138	0.223	0.085

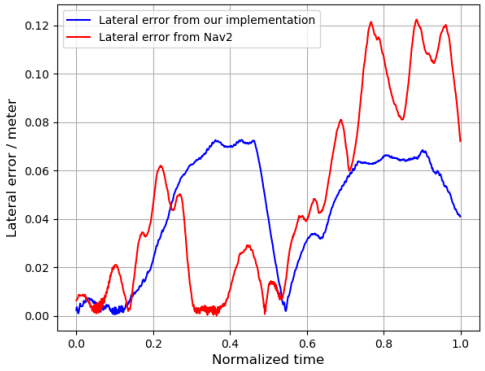
Table 7.4. *The calculated errors from the second simulation scenario test.*



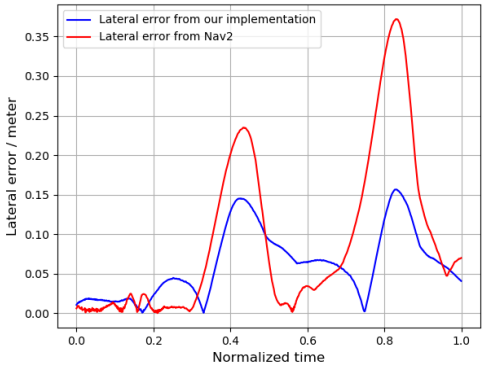
(a) *Orientation error of following path (a) from Fig7.3*



(b) *Orientation error of following path (b) from Fig7.3*



(c) *Lateral error of following path (a) from Fig7.3*



(d) *Orientation error of following path (b) from Fig7.3*

Figure 7.6. *The simulation tests calculated errors vs the normalized time s.*

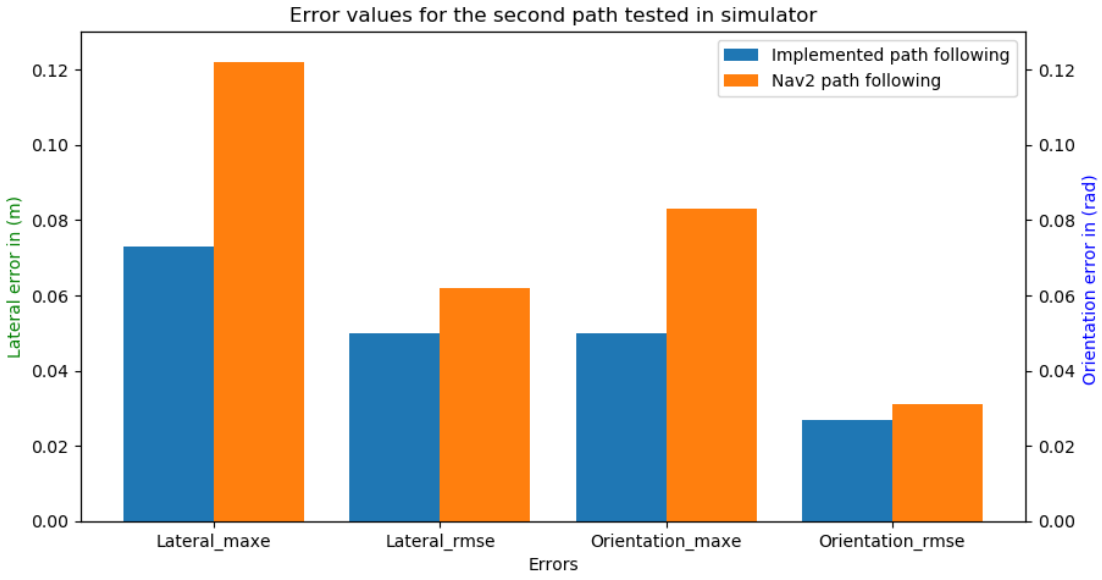


Figure 7.7. *Errors from the simulation test using the first path Fig 7.3(a).*

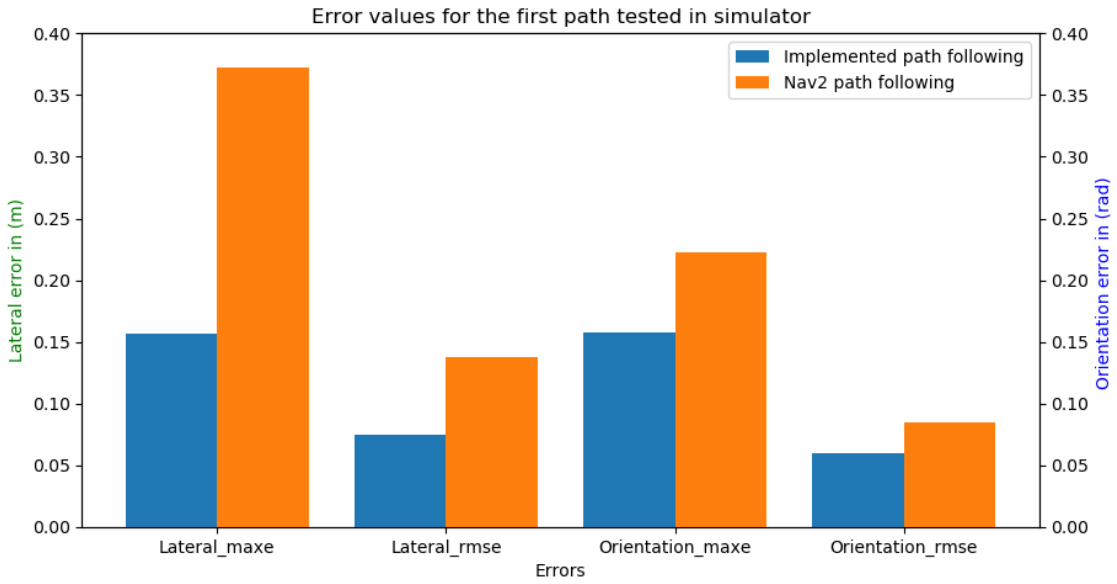


Figure 7.8. *Errors from the simulation test using the second path Fig 7.3(b).*

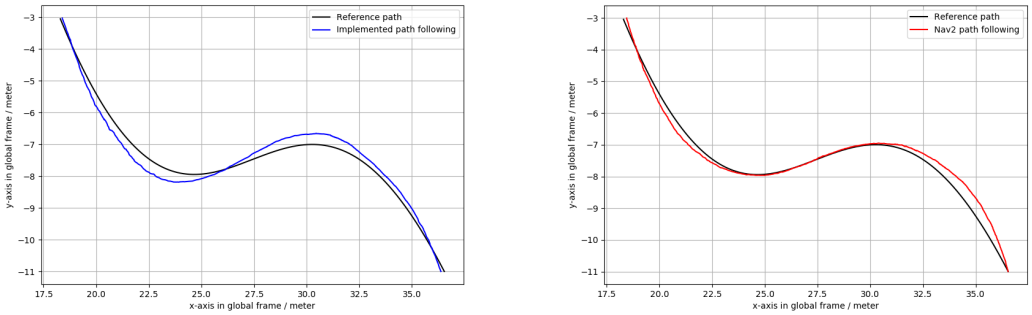
7.4 Test Using AFS Vehicle

In real world we tested the path following control system using the real machine explained earlier. A path, which is a fifth order Bezier curve, that is defined by six control points as shown in table 7.5.

$ControlPoints$	P_0	P_1	P_2	P_3	P_4	P_5
x_{axis}	38.36	35.5	32.82	28.2	25.5	18
y_{axis}	-13.58	-9.16	-6.3	-2.33	-15.5	-2.56

Table 7.5. *The control points of the first path for the experiment in real world.*

This path started from $(38.36, -13.58)$ in the testing cite. During the test, the sensor data was recorded and used to demonstrate the reference path vs the recorded paths for both controllers as shown in Fig 7.9.



(a) *The implemented controller* ***(b)*** *Nav2 controller*

Figure 7.9. *The given path as a reference vs the actual recorded path from the machine.*

In this test, the Los of our implemented controller was set to 4.5 meters with $v_r = 1m/s$, which affected the results as shown in table 7.6. Therefore, a separate test was performed for $Los = 2.5\,meter$ and $v_r = 0.5\,m/s$ and it showed high improvement in the path following task, where the resulted path vs the reference path can be seen in Fig 7.10 and the error calculations in table 7.6. The lateral and the orientation errors were recorded demonstrated vs normalized time in Fig 7.11

$Calculated\,Errors$	$L_{maxe}\,(m)$	$L_{rmse}\,(m)$	$\psi_{maxe}\,(rad)$	$\psi_{rmse}\,(rad)$
$Implemented\,Controller$ $at\,Los = 2.5$	0.16	0.068	0.238	0.0867
$Implemented\,Controller$ $at\,Los = 4.5$	0.452	0.232	0.9	0.46
$Nav2\,Controller$	0.35	0.183	0.49	0.3

Table 7.6. *The calculated errors from testing the real machine.*

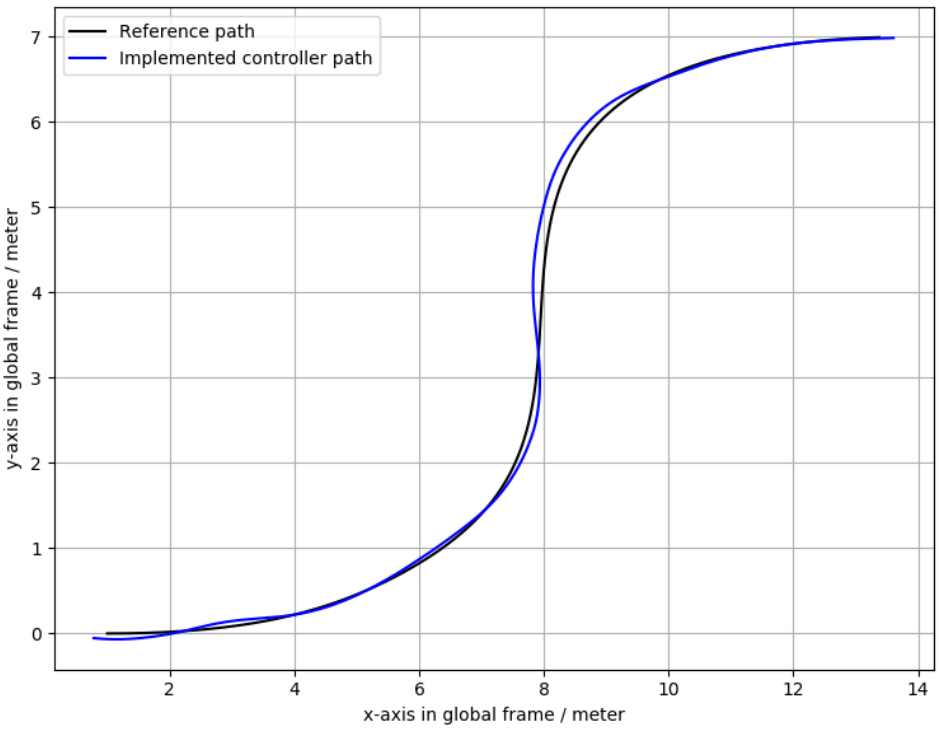
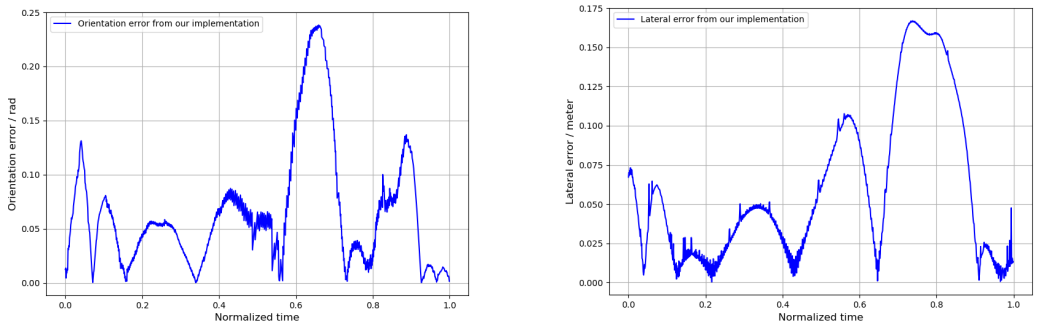


Figure 7.10. *Illustration of Separate test for implemented path following control using real machine when* $Los = 2.5\,meter$ *and* $v_r = 0.5\,m/s$.



(a) *Orientation error* ***(b)*** *Lateral error*

Figure 7.11. *The recorded errors from the test at* $Los = 2.5\,meter$ *and* $v_r = 0.5\,m/s$.

7.4.1 Comparison

From the simulator, according to the observations and the calculated errors in the first scenario, it is obvious that the implemented controller is slightly better than Nav2 controller where the difference in the maximum lateral error is around $5\,cm$ as in table 7.7. However, in the second scenario Fig 7.8 the difference is about $25\,cm$. The second scenario showed a higher error because the path had a sharp turn, which cause the machine to

leave the path and then try to return. The experiment (the test using the real machine) at first showed that Nav2 controller performed better than the implemented controller when the $Los = 4.5\,m$ and $v_r = 1\,m/s$ but in the second test at $Los = 2.5\,m$ and $v_r = 0.5\,m/s$ it showed that the implemented path following controller in better than the MPP Nav2 controller see table 7.6. The test results from the simulator and the experiment using the real vehicle showed that the implemented path following in this thesis is more suitable for the articulated frame steering machine to provide smooth movement and provided better path tracking.

8. CONCLUSION

The path following and the motion control that is implemented in this thesis for AFS MWM using ROS2 is a re-engineered version of Simulink implementation developed by Reza Ghabcheloo. The motion control is introduced in [132] and the path following strategy adopted from [2]. The ROS2 implemented version is tested using a simulation environment and a real articulated machine. According to the test results from chapter 7, both the simulation model and the real machine followed the given paths accurately, this leads to the conclusion that answers our first question:

"Is it possible for ROS2 to be an efficient alternative for powerful platforms like Matlab/Simulink?"

The results showed that ROS2 is efficient framework with many useful tools that supports developing autonomous vehicle control systems. This framework is used to implement accurate path following and motion control successfully. According to that, ROS2 can be used as an alternative to Matlab/Simulink for implementing simple controllers such as the path following and the motion control. In addition, ROS2 can be used as a communication system to integrate different software.

According to the test results which were explained in the previous chapter, the implemented path following showed high-performance results that are better than the performance of the modified pure pursuit controller provided by ROS2 navigation. Therefore, the answer to the second research question:

"How beneficial is using our proposed method in comparison to other developed and commonly used strategy in ROS2?"

The answer is that the implemented method provides a more accurate, path following task, than the RPP from Nav2. According to that using the implemented controller is more suitable for the articulated frame steering. The parameters Los and v_r mentioned in chapter 7 of the implemented controller can be tuned more to get better results.

Besides implementing accurate and robust path following and motion control for AFS autonomous vehicle, the implementation is structured to be modular. That means, the path following controller can be used separately in any autonomous vehicle regardless of the steering system type. Moreover, the motion control is implemented to integrate any

controller that provides linear and angular velocities to control any articulated steering vehicle.

8.1 Limitations and Future Work

One of the limitations of this work is the safety part. The implementation did not include any collision or obstacle avoidance which is an essential part of AVs. Therefore, a future contribution could be adding another controller for obstacle avoidance using ROS2. Another limitation is that the implemented path following is designed to follow a path represented by Bezier curve control points. It can be modified to work with different path formats like b-spline points.

This control system is implemented for self driving AFS MWM which does not require the presence of a human driver. The controller can be modified to provide shared control between a human driver and the machine where the driver could take control of the speed and the manipulator, while the controller is responsible for the steering.

REFERENCES

[1] H. Liikanen, M. M. Aref, R. Oftadeh, and J. Mattila, "Path-following controller for 4wds hydraulic heavy-duty field robots with nonlinear internal dynamics," *IFAC-PapersOnLine*, vol. 52, no. 8, pp. 375–380, 2019.

[2] R. Ghabcheloo, "Coordinated path following," Ph.D. dissertation, Technical University of Lisbon, 2007.

[3] A. Prashar, "Heavy lifters, india: Towards a circular economy era," *Vision*, 2021.

[4] M. Bui, V. Frémont, D. Boukerroui, and P. Letort, "Multi-sensors people detection system for heavy machines," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2014, pp. 867–872.

[5] A. Hekmatmanesh, V. Zhidchenko, K. Kauranen, *et al.*, "Biosignals in human factors research for heavy equipment operators: A review of available methods and their feasibility in laboratory and ambulatory studies," *IEEE Access*, 2021.

[6] H. Singh, V. Kushwaha, A. Agarwal, and S. Sandhu, "Fatal road traffic accidents: Causes and factors responsible," *Journal of Indian Academy of Forensic Medicine*, vol. 38, no. 1, pp. 52–54, 2016.

[7] J. Terken and B. Pfleging, "Toward shared control between automated vehicles and users," *Automotive Innovation*, vol. 3, no. 1, pp. 53–61, 2020.

[8] M. Rohani-Rasaf, Y. Mehrabi, S. S. Hashemi-Nazari, M. Azizmohammad Looha, and H. Soori, "The role of interaction-based effects on fatal accidents using logic regression," *Archives of Trauma Research,*, vol. 7, no. 4, pp. 140–145, 2018.

[9] D. Sam, C. Velanganni, and T. E. Evangelin, "A vehicle control system using a time synchronized hybrid vanet to reduce road accidents caused by human error," *Vehicular communications*, vol. 6, pp. 17–28, 2016.

[10] S. S. Shadrin, O. O. Varlamov, and A. M. Ivanov, "Experimental autonomous road vehicle with logical artificial intelligence," *Journal of advanced transportation*, vol. 2017, 2017.

[11] I. Nastjuk, B. Herrenkind, M. Marrone, A. B. Brendel, and L. M. Kolbe, "What drives the acceptance of autonomous driving? an investigation of acceptance factors from an end-user's perspective," *Technological Forecasting and Social Change*, vol. 161, p. 120 319, 2020.

[12] O. Vermesan, R. John, P. Pype, *et al.*, "Automotive intelligence embedded in electric connected autonomous and shared vehicles technology for sustainable green mobility," *Frontiers in Future Transportation*, vol. 2, p. 688 482, 2021.

[13] S. Moe, W. Caharija, K. Y. Pettersen, and I. Schjølberg, “Path following of underactuated marine surface vessels in the presence of unknown ocean currents,” in *2014 American Control Conference*, IEEE, 2014, pp. 3856–3861.

[14] S. Moe, “Path following of underactuated marine vessels in the presence of ocean currents,” M.S. thesis, Institutt for teknisk kybernetikk, 2013.

[15] M. Geimer, *Mobile Working Machines*. SAE, 2020.

[16] D. Horton and D. Crolla, “Theoretical analysis of the steering behaviour of articulated frame steer vehicles,” *Vehicle System Dynamics*, vol. 15, no. 4, pp. 211–234, 1986.

[17] P. Dudziński and A. Skurjat, “Directional dynamics problems of an articulated frame steer wheeled vehicles,” *Journal of KONES*, vol. 19, pp. 89–98, 2012.

[18] A. Pazooki, D. Cao, S. Rakheja, and P.-É. Boileau, “Ride dynamic evaluations and design optimisation of a torsio-elastic off-road vehicle suspension,” *Vehicle System Dynamics*, vol. 49, no. 9, pp. 1455–1476, 2011.

[19] Y. Yin, S. Rakheja, J. Yang, and P.-E. Boileau, “Effect of articulated frame steering on the transient yaw responses of the vehicle,” *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 232, no. 3, pp. 384–399, 2018.

[20] E. E. Cepolina, M. Przybyłko, G. B. Polentes, and M. Zoppi, “Design issues and in field tests of the new sustainable tractor locostra,” *Robotics*, vol. 3, no. 1, pp. 83–105, 2014.

[21] P. Yu, M. Wu, J. She, K.-Z. Liu, and Y. Nakanishi, “An improved equivalent-input-disturbance approach for repetitive control system with state delay and disturbance,” *IEEE Transactions on Industrial Electronics*, vol. 65, no. 1, pp. 521–531, 2017.

[22] V. Filaretov and D. Yukhimets, “Synthesis method of control system for spatial motion of autonomous underwater vehicle,” *International Journal of Industrial Engineering and Management*, vol. 3, no. 3, p. 133, 2012.

[23] B. Lantos, P. Klatsmanyi, L. Ludvig, and F. Tél, “Intelligent control system of a robot with dextrous hand,” in *Proceedings of IEEE International Conference on Intelligent Engineering Systems*, IEEE, 1997, pp. 129–134.

[24] Y. Takahara, “Multi-level approach to dynamic optimization,” CASE INST OF TECH CLEVELAND OHIO SYSTEMS RESEARCH CENTER, Tech. Rep., 1964.

[25] K. Bipin, *Robot Operating System Cookbook: Over 70 recipes to help you master advanced ROS concepts*. Packt Publishing Ltd, 2018.

[26] H. Abu-Rub, A. Iqbal, and J. Guzinski, *High performance control of AC drives with Matlab/Simulink*. John Wiley & Sons, 2021.

[27] S. Macenski, F. Martin, R. White, and J. G. Clavero, “The marathon 2: A navigation system,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 2718–2725.

[28] R. R. Quintero, C. Vidal-Abarca, H. M. Garrido, *et al.*, “Revision of the eu green public procurement criteria for transport,” *Luxembourg. doi*, vol. 10, p. 700 836, 2019.

[29] M. Ahopelto, “Towards automation and improved fuel economy with system architecture design of a non-road working machine,” 2019.

[30] N. D. Manring and R. C. Fales, *Hydraulic control systems*. John Wiley & Sons, 2019.

[31] S. Pramanik, “Kinematic synthesis of a six-member mechanism for automotive steering,” *J. Mech. Des.*, vol. 124, no. 4, pp. 642–645, 2002.

[32] J. Reimpell, H. Stoll, and J. Betzler, “The automotive chassis: Engineering principles, 2. baskı,” *Society of Automotive Engineers, Inc*, 2002.

[33] P. Simionescu, D. Beale, and I. Talpasanu, “Dynamic effect of the bump steer in a wheeled tractor,” *Mechanism and machine theory*, vol. 42, no. 10, pp. 1352–1361, 2007.

[34] K. Kozłowski and D. Pazderski, “Modeling and control of a 4-wheel skid-steering mobile robot,” *International journal of applied mathematics and computer science*, vol. 14, no. 4, pp. 477–496, 2004.

[35] L. Caracciolo, A. De Luca, and S. Iannitti, “Trajectory tracking control of a four-wheel differentially driven mobile robot,” in *Proceedings 1999 IEEE international conference on robotics and automation (Cat. No. 99CH36288C)*, IEEE, vol. 4, 1999, pp. 2632–2638.

[36] Y. Yin, S. Rakheja, J. Yang, and P.-E. Boileau, “Design optimization of an articulated frame steering system,” *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 232, no. 10, pp. 1339–1352, 2018.

[37] A. Pazooki, “Ride and directional dynamic analysis of articulated frame steer vehicles,” Ph.D. dissertation, Concordia University, 2012.

[38] N. Lashgarian Azad, “Dynamic modelling and stability controller development for articulated steer vehicles,” 2007.

[39] M. Lopatka and T. Muszynski, “Research of the snaking phenomenon to improve directional stability of remote controlled articulated wheel tool-carrier,” *ISARC2003 The Future Site*, p. 95, 2003.

[40] E. Frazzoli, M. A. Dahleh, and E. Feron, “Real-time motion planning for agile autonomous vehicles,” *Journal of guidance, control, and dynamics*, vol. 25, no. 1, pp. 116–129, 2002.

[41] T. Kanade, C. Thorpe, and W. Whittaker, “Autonomous land vehicle project at cmu,” in *Proceedings of the 1986 ACM fourteenth annual conference on Computer science*, 1986, pp. 71–80.

[42] R. S. Wallace, A. Stentz, C. E. Thorpe, H. P. Moravec, W. Whittaker, and T. Kanade, “First results in robot road-following.,” in *IJCAI*, 1985, pp. 1089–1095.

[43] E. D. Dickmanns and A. Zapp, “Autonomous high speed road vehicle guidance by computer vision,” *IFAC Proceedings Volumes*, vol. 20, no. 5, pp. 221–226, 1987.

[44] S. Baskutis, V. Gružauskas, P. Leibl, and L. Obcarskas, “Agent-based modelling approach for autonomous vehicle influence on countries’ welfare,” *Journal of Cleaner Production*, vol. 374, p. 134 008, 2022.

[45] M. Rodrigues, G. Gest, A. McGordon, and J. Marco, “Adaptive behaviour selection for autonomous vehicle through naturalistic speed planning,” in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2017, pp. 1–7.

[46] J. M. Bennett, K. L. Challinor, O. Modesto, and P. Prabhakharan, “Attribution of blame of crash causation across varying levels of vehicle automation,” *Safety Science*, vol. 132, p. 104 968, 2020.

[47] F. Sagir and S. V. Ukkusuri, “Mobility impacts of autonomous vehicle systems,” in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2018, pp. 485–490.

[48] A. C.-C. Liu, O. M. K. Law, and I. Law, “Autonomous vehicle,” 2022.

[49] C. B. S. T. Molina, J. R. De Almeida, L. F. Vismari, R. I. R. Gonzalez, J. K. Naufal, and J. Camargo, “Assuring fully autonomous vehicles safety by design: The autonomous vehicle control (avc) module strategy,” in *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, IEEE, 2017, pp. 16–21.

[50] A. Reda and J. Vásárhelyi, “Model-based control strategy for autonomous vehicle path tracking task,” *Acta Universitatis Sapientiae, Electrical and Mechanical Engineering*, vol. 12, no. 1, pp. 35–45, 2020.

[51] G. Wang, W. Deng, S. Zhang, J. Wang, and S. Yang, “A comprehensive testing and evaluation approach for autonomous vehicles,” SAE Technical Paper, Tech. Rep., 2018.

[52] J. Minguez, F. Lamiraux, and J.-P. Laumond, “Motion planning and obstacle avoidance,” in *Springer handbook of robotics*, Springer, 2016, pp. 1177–1202.

[53] J. Kocić, N. Jovičić, and V. Drndarević, “Sensors and sensor fusion in autonomous vehicles,” in *2018 26th Telecommunications Forum (TELFOR)*, IEEE, 2018, pp. 420–425.

[54] M. Rokonuzzaman, N. Mohajer, S. Nahavandi, and S. Mohamed, “Review and performance evaluation of path tracking controllers of autonomous vehicles,” *IET Intelligent Transport Systems*, vol. 15, no. 5, pp. 646–670, 2021.

[55] Q. Zhu, C. Huang, R. Jiao, *et al.*, “Safety-assured design and adaptation of learning-enabled autonomous systems,” in *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*, IEEE, 2021, pp. 753–760.

[56] H. Zhao and Z. Wang, “Motion measurement using inertial sensors, ultrasonic sensors, and magnetometers with extended kalman filter for data fusion,” *IEEE Sensors Journal*, vol. 12, no. 5, pp. 943–953, 2011.

[57] A. Jouybari, A. Ardalan, and M. Rezvani, “Experimental comparison between mahoney and complementary sensor fusion algorithm for attitude determination by raw sensor data of xsens imu on buoy,” *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 42, pp. 497–502, 2017.

[58] M. C. Rivers, A. A. Trusov, S. A. Zotov, and A. M. Shkel, “Micro imu utilizing folded cube approach,” in *IMAPS Device Packaging Conference*, 2010, pp. 154–158.

[59] T. Islam, M. S. Islam, M. Shajid-Ul-Mahmud, and M. Hossam-E-Haider, “Comparison of complementary and kalman filter based data fusion for attitude heading reference system,” in *AIP Conference Proceedings*, AIP Publishing LLC, vol. 1919, 2017, p. 020 002.

[60] B. Schabron, A. Reust, J. Desai, and Y. Yihun, “Integration of forearm semg signals with imu sensors for trajectory planning and control of assistive robotic arm,” in *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, IEEE, 2019, pp. 5274–5277.

[61] N. Ahmad, R. A. R. Ghazilla, N. M. Khairi, and V. Kasi, “Reviews on various inertial measurement unit (imu) sensor applications,” *International Journal of Signal Processing Systems*, vol. 1, no. 2, pp. 256–262, 2013.

[62] S. Campbell, N. O’Mahony, L. Krpalcova, *et al.*, “Sensor technology in autonomous vehicles: A review,” in *2018 29th Irish Signals and Systems Conference (ISSC)*, IEEE, 2018, pp. 1–4.

[63] D. Nistér, O. Naroditsky, and J. Bergen, “Visual odometry,” in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, Ieee, vol. 1, 2004, pp. I–I.

[64] Z. Sun, G. Bebis, and R. Miller, “On-road vehicle detection: A review,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 5, pp. 694–711, 2006.

[65] T. Yamazato, I. Takai, H. Okada, *et al.*, “Image-sensor-based visible light communication for automotive applications,” *IEEE Communications Magazine*, vol. 52, no. 7, pp. 88–97, 2014.

[66] E. Dagan, O. Mano, G. P. Stein, and A. Shashua, “Forward collision warning with a single camera,” in *IEEE Intelligent Vehicles Symposium, 2004*, IEEE, 2004, pp. 37–42.

[67] H. Kurihata, T. Takahashi, I. Ide, *et al.*, “Rainy weather recognition from in-vehicle camera images for driver assistance,” in *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.*, IEEE, 2005, pp. 205–210.

[68] J. Shackleton, B. VanVoorst, and J. Hesch, “Tracking people with a 360-degree lidar,” in *2010 7th IEEE International Conference on Advanced Video and Signal Based Surveillance*, IEEE, 2010, pp. 420–426.

[69] J. R. V. Rivero, I. Tahiraj, O. Schubert, *et al.*, “Characterization and simulation of the effect of road dirt on the performance of a laser scanner,” in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2017, pp. 1–6.

[70] A. T. Hudak, M. A. Lefsky, W. B. Cohen, and M. Berterretche, “Integration of lidar and landsat etm+ data for estimating and mapping forest canopy height,” *Remote sensing of Environment*, vol. 82, no. 2-3, pp. 397–416, 2002.

[71] A. Topol, M. Jenkin, J. Gryz, *et al.*, “Generating semantic information from 3d scans of crime scenes,” in *2008 Canadian Conference on Computer and Robot Vision*, IEEE, 2008, pp. 333–340.

[72] D. Fernandes, A. Silva, R. Névoa, *et al.*, “Point-cloud based 3d object detection and classification methods for self-driving applications: A survey and taxonomy,” *Information Fusion*, vol. 68, pp. 161–191, 2021.

[73] E. Dayangac, F. Baumann, J. Aulinas, and M. Zobel, “Target position and speed estimation using lidar,” in *International Conference on Image Analysis and Recognition*, Springer, 2016, pp. 470–477.

[74] M. E. Russell, C. A. Drubin, A. S. Marinilli, W. G. Woodington, and M. J. Del Checcolo, “Integrated automotive sensors,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 50, no. 3, pp. 674–677, 2002.

[75] P. Lowbridge, “Low cost millimeter-wave radar systems for intelligent vehicle cruise control applications,” *Microwave Journal*, vol. 38, no. 10, pp. 20–27, 1995.

[76] B. Zohuri, “Fundaments of radar,” in *Radar Energy Warfare and the Challenges of Stealth Technology*, Springer, 2020, pp. 1–110.

[77] K. Miyashita, T. Takahashi, and M. Yamanaka, “Features of a magnetic rotary encoder,” *IEEE Transactions on Magnetics*, vol. 23, no. 5, pp. 2182–2184, 1987.

[78] K. Miyashita, T. Takahashi, S. Kawamata, H. K-anai, and T. Seki, “Integrated magnetic rotary encoders,” *IECON-84, Tokyo, Japan*, 1984.

[79] M. Quigley, K. Conley, B. Gerkey, *et al.*, “Ros: An open-source robot operating system,” in *ICRA workshop on open source software*, Kobe, Japan, vol. 3, 2009, p. 5.

[80] A. A. Cabrera-Ponce and J. Martinez-Carranza, “Convolutional neural networks for geo-localisation with a single aerial image,” *Journal of Real-Time Image Processing*, vol. 19, no. 3, pp. 565–575, 2022.

[81] K. A. Wyrobek, E. H. Berger, H. M. Van der Loos, and J. K. Salisbury, “Towards a personal robotics development platform: Rationale and design of an intrinsically safe personal robot,” in *2008 IEEE International Conference on Robotics and Automation*, IEEE, 2008, pp. 2165–2170.

[82] *Ros history*, www.theconstructsim.com/history-ros, Accessed 1st Aug 2022.

[83] *"ros 1.0 – ros robotics news"*, https://www.ros.org/news/2010/01/ros-10.html, Accessed 17th November 2022.

[84] *Wiki (ros goals)*, http://wiki.ros.org/ROS/Introduction, Accessed 1st Aug 2022.

[85] S. Chitta, E. Marder-Eppstein, W. Meeussen, *et al.*, "Ros_control: A generic and simple control framework for ros," *The Journal of Open Source Software*, vol. 2, no. 20, pp. 456–456, 2017.

[86] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige, "The office marathon: Robust navigation in an indoor office environment," in *2010 IEEE international conference on robotics and automation*, IEEE, 2010, pp. 300–307.

[87] D. Coleman, I. Sucan, S. Chitta, and N. Correll, "Reducing the barrier to entry of complex robotic software: A moveit! case study," *arXiv preprint arXiv:1404.3785*, 2014.

[88] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, eabm6074, 2022.

[89] K. Birman and T. Joseph, "Exploiting virtual synchrony in distributed systems," in *Proceedings of the eleventh ACM Symposium on Operating systems principles*, 1987, pp. 123–138.

[90] G. Mühl, L. Fiege, and P. Pietzuch, *Distributed event-based systems*. Springer Science & Business Media, 2006.

[91] M. Anderson, "Introduction to the robot operating system (ros) middleware," in *Embedded Linux Conference & OpenIOT Summit North America*, 2018.

[92] A. Cowley and C. J. Taylor, "Stream-oriented robotics programming: The design of roshask," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2011, pp. 1048–1054.

[93] S.-Y. Jeong, I.-J. Choi, Y.-J. Kim, *et al.*, "A study on ros vulnerabilities and countermeasure," in *Proceedings of the Companion of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, 2017, pp. 147–148.

[94] R. Mishra and A. Javed, "Ros based service robot platform," in *2018 4th International Conference on Control, Automation and Robotics (ICCAR)*, IEEE, 2018, pp. 55–59.

[95] *Ros node)*, https://roboticsbackend.com/what-is-a-ros-node, Accessed 1st Aug 2022.

[96] *Ros2 galactic tutorials, ros official web page)*, https://docs.ros.org/en/galactic/Tutorials.html, Accessed 20th October 2022.

[97] *Ros2 designs, ros official web page)*, http://design.ros2.org/articles/actions.html, Accessed 20th October 2022.

[98] D. Hähnel, W. Burgard, and G. Lakemeyer, “Golex—bridging the gap between logic (golog) and a real robot,” in *Annual Conference on Artificial Intelligence*, Springer, 1998, pp. 165–176.

[99] S. Chik, C. Yeong, E. Su, T. Lim, Y. Subramaniam, and P. Chin, “A review of social-aware navigation frameworks for service robot in dynamic human environments,” *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 8, no. 11, pp. 41–50, 2016.

[100] M. Kulkarni, P. Junare, M. Deshmukh, and P. P. Rege, “Visual slam combined with object detection for autonomous indoor navigation using kinect v2 and ros,” in *2021 IEEE 6th International Conference on Computing, Communication and Automation (ICCCA)*, IEEE, 2021, pp. 478–482.

[101] X. Wang, J. Zhang, D. Zhang, and L. Shen, “A virtual force guidance law for trajectory tracking and path following,” in *International Conference on Intelligent Autonomous Systems*, Springer, 2016, pp. 417–432.

[102] N. Mohajer, S. Nahavandi, H. Abdi, and Z. Najdovski, “Enhancing passenger comfort in autonomous vehicles through vehicle handling analysis and optimization,” *IEEE Intelligent Transportation Systems Magazine*, vol. 13, no. 3, pp. 156–173, 2020.

[103] O. Amidi, “Integrated mobile robot control,” Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-90-17, May 1990.

[104] S. Thrun, M. Montemerlo, H. Dahlkamp, *et al.*, “Winning the darpa grand challenge,” *Journal of Field Robotics*, 2006, accepted for publication.

[105] G. Oriolo, A. De Luca, and M. Vendittelli, “Wmr control via dynamic feedback linearization: Design, implementation, and experimental validation,” *IEEE Transactions on control systems technology*, vol. 10, no. 6, pp. 835–852, 2002.

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

[107] F. Yakub and Y. Mori, “Comparative study of autonomous path-following vehicle control via model predictive control and linear quadratic control,” *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of automobile engineering*, vol. 229, no. 12, pp. 1695–1714, 2015.

[108] L. L. Scharf, W. P. Harthill, and P. H. Moose, “A comparison of expected flight times for intercept and pure pursuit missiles,” *IEEE Transactions on Aerospace and Electronic Systems*, no. 4, pp. 672–673, 1969.

[109] R. S. Wallace, A. Stentz, C. E. Thorpe, H. P. Moravec, W. Whittaker, and T. Kanade, “First results in robot road-following.,” in *IJCAI*, 1985, pp. 1089–1095.

[110] R. C. Coulter, “Implementation of the pure pursuit path tracking algorithm,” Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, Tech. Rep., 1992.

[111] S. Thrun, M. Montemerlo, H. Dahlkamp, *et al.*, “Stanley: The robot that won the darpa grand challenge,” in *The 2005 DARPA grand challenge*, Springer, 2007, pp. 1–43.

[112] G. M. Hoffmann, C. J. Tomlin, M. Montemerlo, and S. Thrun, “Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing,” in *2007 American control conference*, IEEE, 2007, pp. 2296–2301.

[113] B. Tearle, K. P. Wabersich, A. Carron, and M. N. Zeilinger, “A predictive safety filter for learning-based racing control,” *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7635–7642, 2021.

[114] C. Altafini, “Following a path of varying curvature as an output regulation problem,” *IEEE Transactions on Automatic Control*, vol. 47, no. 9, pp. 1551–1556, 2002.

[115] C. Altafini, A. Speranzon, and B. Wahlberg, “A feedback control scheme for reversing a truck and trailer vehicle,” *IEEE Transactions on robotics and automation*, vol. 17, no. 6, pp. 915–922, 2001.

[116] A. Astolfi, P. Bolzern, and A. Locatelli, “Path-tracking of a tractor-trailer vehicle via lyapunov techniques,” in *2001 European Control Conference (ECC)*, IEEE, 2001, pp. 451–456.

[117] P. Bolzern, R. DeSantis, and A. Locatelli, “An input-output linearization approach to the control of an n-body articulated vehicle,” *J. Dyn. Sys., Meas., Control*, vol. 123, no. 3, pp. 309–316, 2001.

[118] P. Bolzern, R. M. DeSantis, A. Locatelli, and D. Masciocchi, “Path-tracking for articulated vehicles with off-axle hitching,” *IEEE Transactions on control systems technology*, vol. 6, no. 4, pp. 515–523, 1998.

[119] R. M. DeSantis, “Path-tracking for a tractor-trailer-like robot: Communication,” *The International Journal of Robotics Research*, vol. 13, no. 6, pp. 533–544, 1994.

[120] F. Lamiraux and J.-P. Laumond, “A practical approach to feedback control for a mobile robot with trailer,” in *ICRA*, Citeseer, 1998, pp. 3291–3296.

[121] M. Sampei, T. Tamura, T. Kobayashi, and N. Shibui, “Arbitrary path tracking control of articulated vehicles using nonlinear control theory,” *IEEE Transactions on Control Systems Technology*, vol. 3, no. 1, pp. 125–131, 1995.

[122] C. Samson, “Control of chained systems application to path following and time-varying point-stabilization of mobile robots,” *IEEE transactions on Automatic Control*, vol. 40, no. 1, pp. 64–77, 1995.

[123] K. Tanaka and T. Kosaki, “Design of a stable fuzzy controller for an articulated vehicle,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 27, no. 3, pp. 552–558, 1997.

[124] P. Terry and J. Schuster, “Tractor-trailer reverse movement path,” *ITE Journal*, vol. 66, no. 11, 1996.

[125] Y. Bian, M. Yang, X. Fang, and X. Wang, “Kinematics and path following control of an articulated drum roller,” *Chinese Journal of Mechanical Engineering*, vol. 30, no. 4, pp. 888–899, 2017.

[126] G. C. Rains, A. G. Faircloth, C. Thai, and R. L. Raper, “Evaluation of a simple pure pursuit path-following algorithm for an autonomous, articulated-steer vehicle,” *Applied engineering in agriculture*, vol. 30, no. 3, pp. 367–374, 2014.

[127] K. Fue, W. Porter, E. Barnes, C. Li, and G. Rains, “Autonomous navigation of a center-articulated and hydrostatic transmission rover using a modified pure pursuit algorithm in a cotton field,” *Sensors*, vol. 20, no. 16, p. 4412, 2020.

[128] J. K. Kolb, G. Nitzsche, and S. Wagner, “A simple yet efficient path tracking controller for autonomous trucks,” *IFAC-PapersOnLine*, vol. 52, no. 8, pp. 307–312, 2019.

[129] M. Murillo, G. Sánchez, N. Deniz, L. Genzelis, and L. Giovanini, “Improving path-tracking performance of an articulated tractor-trailer system using a non-linear kinematic model,” *Computers and Electronics in Agriculture*, vol. 196, p. 106 826, 2022.

[130] B. Hejase and U. Ozguner, “Physics-based simulation and automation of a load-haul-dump operation for an articulated dump truck,” *Vehicles*, vol. 4, no. 1, pp. 167–181, 2022.

[131] C. Altafini, “A path-tracking criterion for an lhd articulated vehicle,” *The International Journal of Robotics Research*, vol. 18, no. 5, pp. 435–441, 1999.

[132] R. Ghabcheloo and M. Hyvonen, “Modeling and motion control of an articulated-frame-steering hydraulic mobile machine,” in *2009 17th Mediterranean Conference on Control and Automation*, IEEE, 2009, pp. 92–97.

[133] T. Nayl, “On autonomous articulated vehicles,” Ph.D. dissertation, Luleå tekniska universitet, 2015.

[134] M. A. Haidekker, *Linear feedback controls: the essentials*. Elsevier, 2020.

[135] H. Zhou, L. Güvenç, and Z. Liu, “Design and evaluation of path following controller based on mpc for autonomous vehicle,” in *2017 36th Chinese Control Conference (CCC)*, IEEE, 2017, pp. 9934–9939.

[136] A. AbdElmoniem, A. Osama, M. Abdelaziz, and S. A. Maged, “A path-tracking algorithm using predictive stanley lateral controller,” *International Journal of Advanced Robotic Systems*, vol. 17, no. 6, p. 1 729 881 420 974 852, 2020.

[137] D. Göhring, D. Latotzky, M. Wang, and R. Rojas, “Semi-autonomous car control using brain computer interfaces,” in *Intelligent Autonomous Systems 12*, Springer, 2013, pp. 393–408.

[138] A. TOMAR, R. KUSUMAKAR, and A. NAREN, “Path following bi-directional controller for articulated vehicles,”

[139] Z. Lu, B. Shyrokau, B. Boulkroune, S. Van Aalst, and R. Happee, “Performance benchmark of state-of-the-art lateral path-following controllers,” in *2018 IEEE 15th*

International Workshop on Advanced Motion Control (AMC), IEEE, 2018, pp. 541–546.

[140] Z. Leng and M. Minor, “A simple tractor-trailer backing control law for path following,” in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2010, pp. 5538–5542.

[141] S. Bacha, R. Saadi, M. Y. Ayad, A. Aboubou, and M. Bahri, “A review on vehicle modeling and control technics used for autonomous vehicle path following,” in *2017 International Conference on Green Energy Conversion Systems (GECS)*, IEEE, 2017, pp. 1–6.

[142] D. R. Nelson, D. B. Barber, T. W. McLain, and R. W. Beard, “Vector field path following for small unmanned air vehicles,” in *2006 American Control Conference*, IEEE, 2006, 7–pp.