DEPARTMENT OF ELECTRICAL
AND ELECTRONICS
ENGINEERING

EE492 PROJECT REPORT

PROJECT TITLE

Autonomous Parking System based on Autoware

STUDENT NAME, SURNAME AND NUMBER

Boğaçhan Ali Düşgül , 270206051

Cem Tolga Münyas , 270206042

ADVISOR NAME SURNAME

Sibel Demir

Prof Dr. Barış Atakan / Utku Karakaya

DATE: 29/05/2024

ABSTRACT

Nowadays, autonomous driving vehicles use advanced technologies to navigate and perform driving tasks without human intervention. Autonomous vehicles can make decisions based on their programming and by processing the sensor inputs which they are equipped. Autonomous vehicles have become a common place in today’s world. On the other hand, finding a suitable parking space in modern urban environments has become a significantly hard mission for drivers for many reasons.

The main goal of this paper is to propose an efficient solution to the parking problem in urban areas by using an open-source autonomous driving stack which is especially developed for autonomous vehicles called as Autoware. By using the functionalities of Autoware and the parking space planner algorithm that we developed using the Robot Operating System architecture, the autonomous vehicles could determine the nearest parking facility and the nearest parking space to park the vehicle autonomously. In this way, the effort will be minimized for the drivers to search for empty parking spaces, and it will decrease the minor accidents happen in parking lots, carbon emissions and additional fuel consumption.

TABLE OF CONTENTS

ABBREVIATIONS

GNSS :	Global Navigation Satellite System
IMU :	Inertial Measurement Unit
ROS :	Robot Operating System
LIDAR :	Light Detection and Ranging

LIST OF FIGURES

1. INTRODUCTION

Many of the scientific studies have shown that semi-autonomous parking systems are commercially available from several automotive manufacturers. Moreover, automated valet parking systems have been introduced in a way that drivers can control their autonomous vehicles by the mobile app, which enables an autonomous car to drive to a parking spot and parks itself. Despite the extensive research in the field of autonomous parking for vehicles, generating a free-obstacle trajectories for the autonomous vehicles in tight urban environments remains as a difficult task. Furthermore, the semi-autonomous parking systems has still under development, so they still work slightly slower compared to the fully automated parking systems. From the perspective of time efficiency, and performance, semi-autonomous parking systems still depends on the existence of driver and the car should be under control until it is placed on the entrance of the parking facility.

1.1. General Description of the Project

The goal of this project is to design an autonomous parking system that utilizes the capabilities of Autoware and the Robot Operating System (ROS 2) to identify and navigate to the nearest available parking space within the nearest parking lot relative to the current position of the ego-vehicle. The system aims to enhance urban mobility by providing an efficient, safe, and user-friendly solution to the problem of finding and utilizing parking spaces.

We believe that our autonomous parking system is a critical improvement based on the current parking technology such as automated valet parking systems and parking assistance systems. Current technologies still depend on the existence of the driver on the steering wheel, on the other hand, the autonomous parking system that we developed can determine the nearest parking space inside of the nearest parking lot fully-autonomously by utilizing the stacks in Autoware.

2. PROBLEM DEFINITON

Finding an empty parking spaces in modern urban environments has become a significant challenge for the drivers. Number of vehicles in the traffic is being increased day to day. On the other hand, since there are limited parking spaces in urban areas, it has become a serious problem to find an empty parking space. According to the research in UK, an average British driver spends 44 hours a year searching for a parking space. In addition to this waste of time, this process increases the emissions and the carbon footprint. According to another research which is held in United States, it shows that %20 of all car accidents happen in parking facilities. Looking for a parking space can sometimes be frustrating for the drivers, so that they can take extra risks while parking. As a result, minor accidents are very likely to occur in tight parking lots.

2.1 Technological Challenges

As another aspect of the problem, the existing parking assistance technologies are sometimes limited. The automotive industry brands use the parking assistance technology for a while. As general, the current parking assistance technologies in industry requires the driver to select the parking space and then drive in near this space. This process is not fully automated at all. The only point that the current parking assistance systems work autonomously is that after the driver selects the suitable parking space. As a result, the current parking assistance systems do not work fully autonomously because of the changing conditions of the environment, tights spaces, difficulty of detecting and avoiding other obstacles such as pedestrians in real-time.

Overall, we believe that finding suitable parking spaces in modern urban environments with the existing parking assistance systems are not the effective way of parking. We designed a fully autonomous parking system that it finds the nearest parking space relative to the current position of the autonomous vehicle. We believe that our parking system is a serious improvement on the existing parking assistance systems.

3. Autonomous Parking

The main aim of this autonomous parking system project is to optimize the parking space usage and to enable safe and efficient parking. We developed our parking system based on Autoware which is an open-source autonomous driving stack especially developed for self-driving vehicles. Autoware provides fully autonomous driving by enabling real-time communication between different nodes. In other words, Autoware needs a middleware platform as a basis in order to make the connection between the real word hardware and the software which is Autoware. At this point, we used the Robot Operating System (ROS2) as a basis for the communication between different nodes and also for the hardware on the vehicle.

3.1 Robot Operating System (ROS2)

The Robot Operating System (ROS) is a set of software libraries and tools for building robot applications. From drivers and state-of-the-art algorithms to powerful developer tools, ROS has the open-source tools you need for your next robotics project. Powerful libraries, easy real-time communication system and infrastructure applicable to every robot are the reasons why ROS2 is preferred.

3.1.1 ROS2 Packages

There are different packages for each task in ROS2. As seen in Figure X, 1 package for Hardware control, one package for Camera and one package for Motion Planning. For each package there are executable C++ or Python files called Node. You should decide in advance in which software language your nodes will be created and create your package in this way. Different packages can work in different software languages. This is one of the most important features of ROS2.

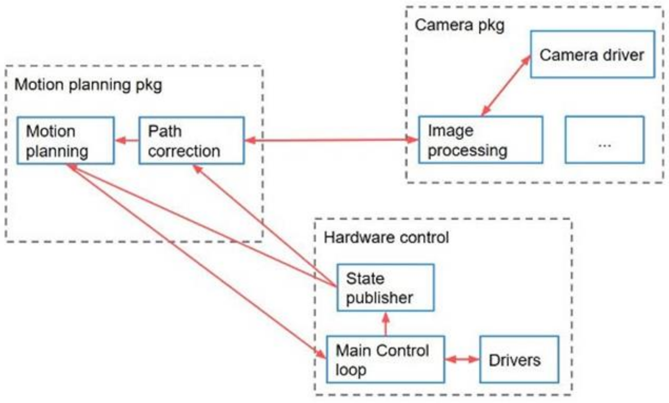

Figure 1. Design blocks of ROS2

3.1.2 ROS2 Nodes

Nodes are C++ or Python files created for a single task. For example, "Path Correction" shown in Figure 1 is a node and "Motion Planning" is a node. Nodes can transfer data to each other through communication channels called topics. As seen in Figure 2, nodes provide message flow through topics. Topics provide a publish-subscribe mechanism to ensure the communication between different nodes.

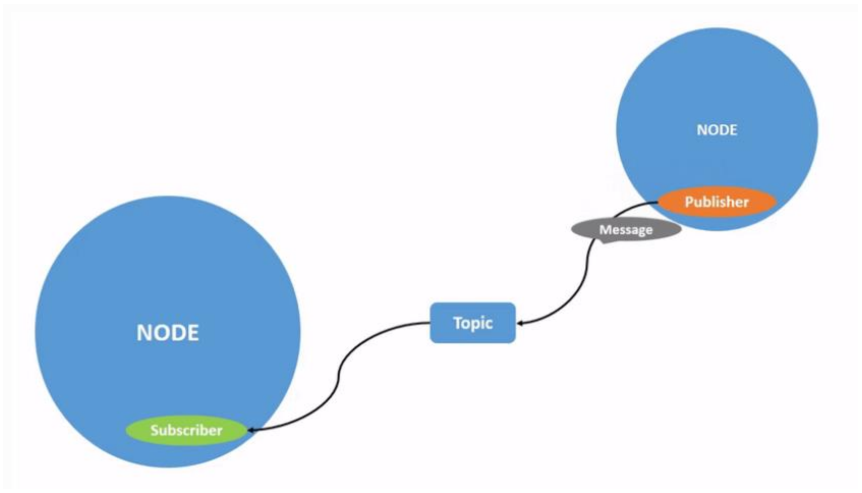

Figure 2. Communication system between nodes through topics

3.1.3 ROS2 Topics

We can demonstrate the Publisher-Subscriber structure as the base station communication visualized in Figure 3. Publishers send messages to topics; subscribers listen to topics and receive data. It's like your car radio listening to a specific frequency.

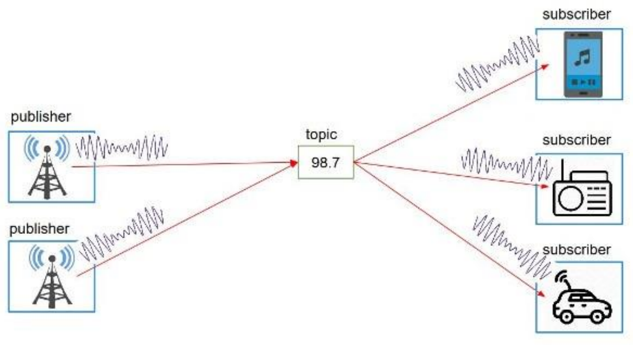

Figure 3. Publisher – Subscriber mechanism in ROS2

3.1.4 Service – Client in ROS2

Service-Client communication system, unlike Publisher and Subscriber, works with request and response. Client Service sends a request and receives the message as a response. While the Publisher and Subscriber systems provide a continuous data flow, this system only occurs on request. Figure 4 shows how the system works between nodes.

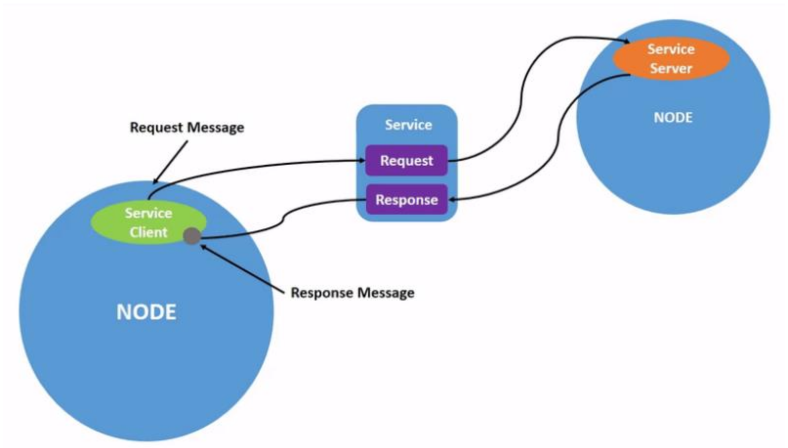

Figure 4. Service – Client architecture

3.2 Relation of Autonomous Vehicle Technology and Robots

The autonomous vehicles are the specialized robots which is especially used for transportation and mobility. The relationship between an autonomous driving vehicle and a robot lies in their shared characteristics and common technology beneath. They mainly have 4 main characteristics in common. The first one is that they perform tasks without human intervention. It means that they both can make decisions and execute actions based on their programming and the sensor inputs. The second common feature is that they are both equipped with various sensors and actuators. These sensors can be LIDARs, RADARs, GNSS, IMU, cameras, etc. Actuators mainly control the velocity and the steering angle values for the robot and the autonomous vehicle as well. Additionally, autonomous vehicles and robots both need specific control systems to process the sensor data, make decisions and control the movements. These control systems can be PID controllers, differential controllers, or Ackermann kinematic model. For the last common characteristics, they both use some navigation and path finding algorithms in order to determine the optimum trajectory, reaching to the goal pose by considering other dynamic obstacles in the environment.

Overall, we can clearly observe that autonomous vehicles are actually mobile robots specifically designed for transportation. Therefore, understanding the principles of robotics provides a solid foundation to develop autonomous vehicle technologies.

3.3 Autoware

While developing our node and algorithm, we used the Autoware as the software which has many autonomous driving packages inside. Autoware consists of 6 stacks and the communication of these 6 stacks simultaneously makes the autonomous driving possible. Here is the architecture of the Autoware. The 6 components which is essential for autonomous driving are the sensing, mapping, localization, perception, planning and control.

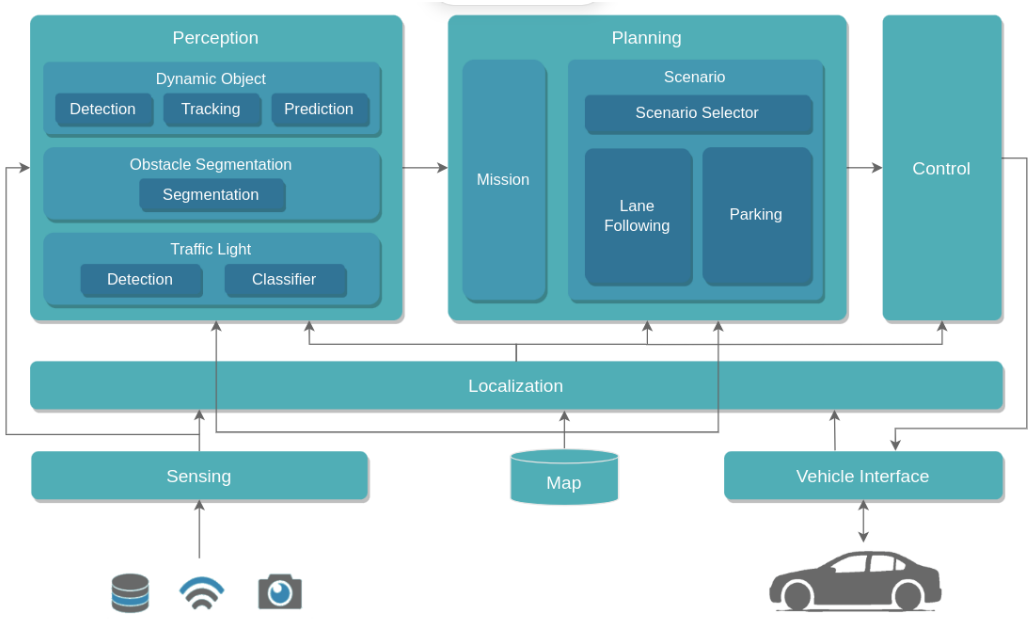

Figure 5. Architecture of Autoware

3.3.1 Sensing

Sensing is basically collecting data from the environment with various kinds of sensors. Those sensors are mainly LIDARs, cameras, GNSS, and IMU. Sensing is a crucial component in autonomous since these collected data from the environment can be used for localization, generating trajectories, and correcting these trajectories in order to reach to the desired goal pose.

Autoware mainly uses laser scanners (LIDARs) and cameras to recognize the environment, lanelets, objects and other vehicles. LIDAR scanners measure the distance to objects by illuminating a target with pulsed lasers and measuring the time of the reflected pulses.

Data from other sensors such as GNSS (Global Navigation Satellite System) and IMU (Internal Measurement Unit) are used to enhance the information about mapping, localization, and detection. While developing our parking system, we used odometry messages from the navigation stack in order to determine the positioning information on the map precisely. The whole components in Autoware are basically connected to each other in order to communicate as a whole system and execute the actions based on this communication. Here is the high-level architecture of the sensing component that shows how each sensor on the vehicle is related to other components of Autoware.

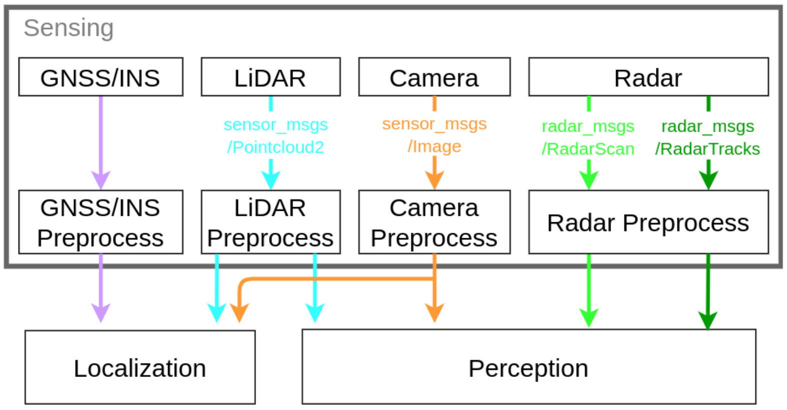

Figure 6. High – Level Architecture of Sensing

3.3.2 Mapping

Autoware relies on high-definition point cloud maps and vector maps of the driving environment to perform various tasks such as localization, decision making, generating trajectories, etc. Mapping in Autoware provides two types of information to the rest of the stack. It provides semantic information about roads as a vector map, and geometric information about the environment as point cloud map. Vector map basically contains highly precise information about the lane geometry, road network and the traffic lights. It is crucial submodule for predicting the trajectories of other vehicles or pedestrians. On the other hand, 3D point cloud map is primarily used for LIDAR-based localization. In order to determine the current pose of the vehicle, an accurate point cloud map is crucial for accurate localization.

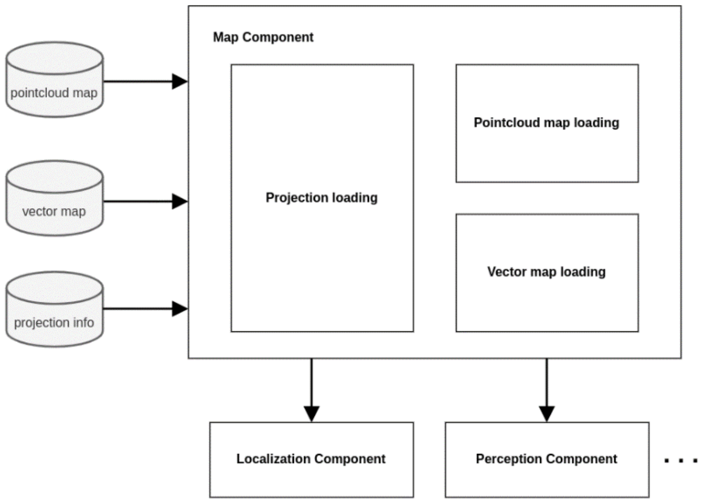

Figure 7. High – Level Architecture of Mapping

3.3.3 Localization

Localization module is one of the most crucial components of autonomous driving. Localization basically aims to estimate the pose, velocity, and acceleration of the autonomous vehicle. Many other perception modules rely on the location of the vehicle. The localization algorithms that are used in Autoware make use of scan matching between 3D maps and LIDAR scanners. By using the Kalman filter and sensor fusion, the system model can estimate the vehicle's location on the point cloud map with the error of ~10cm. Additionally, we used odometry messages from the vehicle kinematic state topic in order to satisfy the localization of the vehicle.

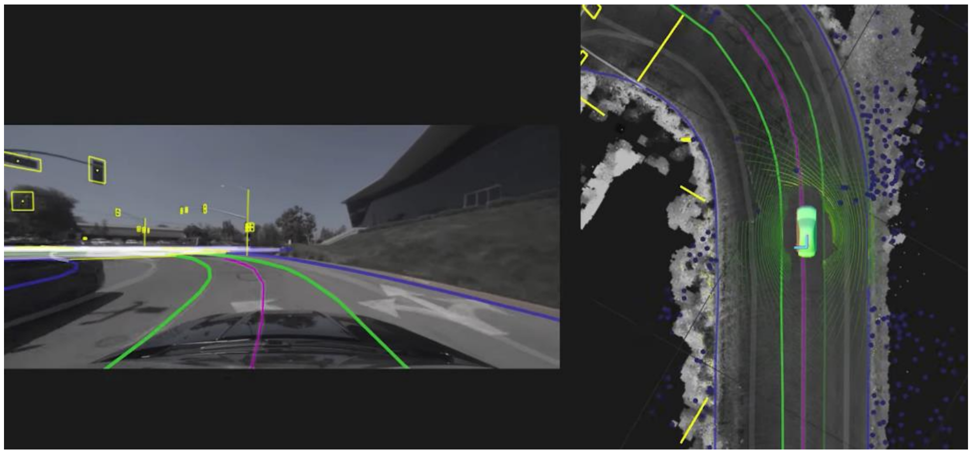

Figure 8. Localization of Autonomous Vehicle on the 3D Map

3.3.4 Perception

Perception module in Autoware is basically used for detection and classification of other objects or obstacles around the environment so that the autonomous driving vehicle does not crash into them. The perception component receives inputs from sensing, localization, and mapping components and adds semantic information on top of these such as object recognition, obstacle segmentation, traffic light recognition. Then the output of the perception component is transmitted to the planning component. Since the perception component receives data from different kinds of sensors such as laser scanners, cameras, GNSS and maybe IMU, a filter is needed in order to combine and process such data. This operation of combining different kinds of sensor data is known as sensor fusion. The filter used for this operation is called the Kalman Filter.

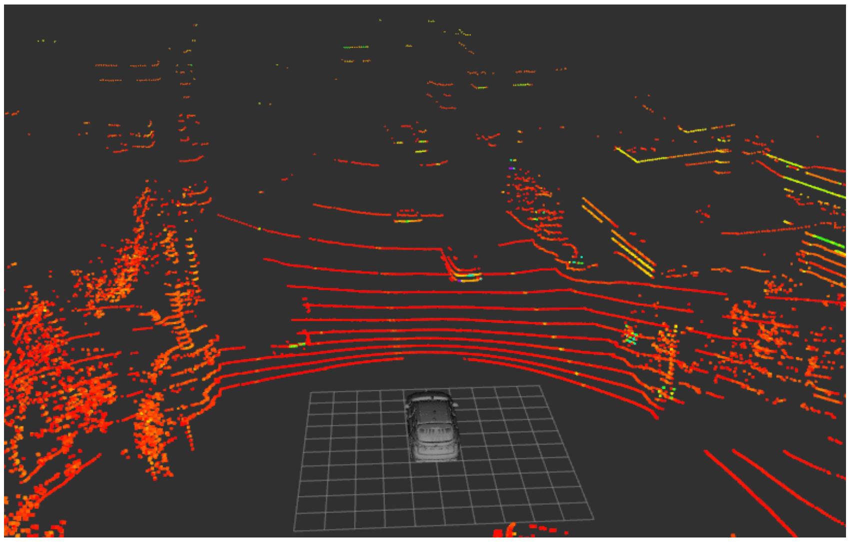

Figure 9. Detecting objects around the scene using LIDAR

When we are developing our parking space planner algorithm, we needed the information of inside of the parking space, in order to determine the availability of the it. We used “Predicted Objects” submodule for an area with defined boundaries. By utilizing the perception messages, the autonomous vehicle determined that whether the parking space is empty or there is any other object or obstacle such as other vehicles, pedestrians, etc.

3.3.5 Planning

The planning component in autonomous driving systems plays a crucial role in generating a target trajectory, determining a goal pose while ensuring all of the safety standards and following the traffic rules. The planning stacks in Autoware generates trajectories based on outputs from all of the components that we mentioned earlier. Path planning is separated into two classes: mission and motion planning. Autoware plans the trajectory based on the current position of the ego-vehicle and the desired position. Search algorithms are used in order to generate the efficient trajectory to the goal position. In our case, we used the A* path finding algorithm since it provides the most efficient and shortest path to the goal pose.

Mission Planning: Depending on the driving states, mission planner employs a rule-based approach to calculate path trajectories. The mission planning also includes navigation from the current position to the destination. Special planning algorithms are required in more complex cases, like parking, in order to produce the right path for the parking lot.

Motion Planning: It is responsible for generating efficient and feasible trajectories by considering the states of vehicle, drivable areas indicated by 3D point cloud map, surrounding objects, and the desired goal.

3.3.6 Control / Actuation

The autonomous vehicle is controlled to follow the generated trajectories by the control signals that are created by the control component. The autonomous vehicle's actuators must follow the trajectories when they have been established and the desired pose is known. As a result, control signals are generated and subscribed to by vehicle actuators.

Path Following: The commands for actuation that move the ego vehicle are generated by the pure pursuit algorithm. Using this strategy, the path is divided into several waypoints, each of which represents the path in a discrete manner. The program looks for the next closest waypoint in the ego vehicle's going direction during each control cycle. Autonomous vehicle follows the selected waypoints by adjusting the velocity and the steering angle values.

Vehicle Control: The target waypoint is continuously updated until the goal is reached. In other words, the ego vehicle follows the determined waypoints until it reaches the final destination. In order to execute the velocity and the steering wheel command “Ackermann Kinematic Model” is applied as a controller in our project.

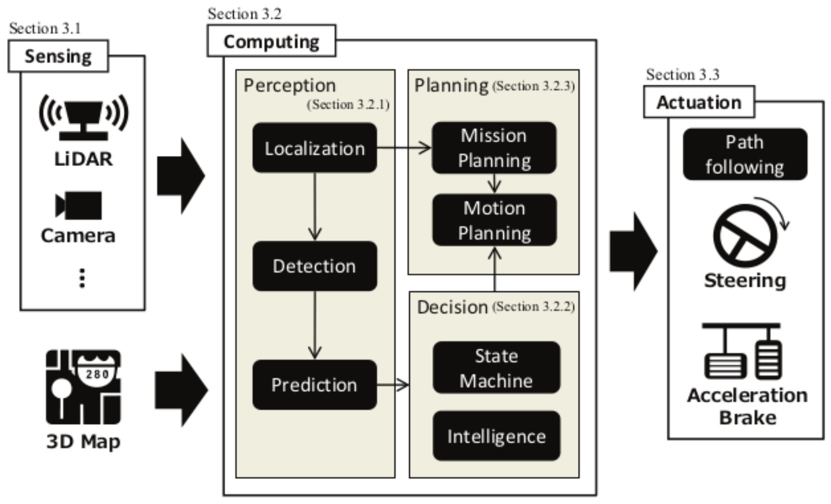

Figure 10. Basic control and data flow of autonomous vehicles in Autoware

3.4 Sensor Fusion and Kalman Filter

One of the most important concepts in autonomous driving is the sensor fusion. The importance of sensor fusion in autonomous driving comes from the dependence of multiple sensors that are used on the vehicle to make autonomous driving possible. The vehicle relies on the sensor data from which is collected from different kinds of sensors all around the vehicle and it should extract a meaning from the collected sensor data. As we mentioned earlier, the sensors which we used in our project was 1 LIDAR, 1 GNSS, 2 side cameras, and 1 IMU. The ego-vehicle collects data from these sensors on every moment so that the localization and perception of the vehicle could be reliable.

Each sensor on the vehicle cannot work well for all tasks and in all conditions, so the sensor fusion is necessary to provide redundancy for autonomous driving functions. Sensor measurements are generally noisy data signals. The noises are basically caused by error of measurements. Sensor fusion basically combines different kinds of sensor data and fuses them. In this way, the system can predict the parameter of interest in any conditions.

Autoware uses Kalman Filter in order to fuse the output of different sensors. Kalman Filter is an optimal estimation algorithm which predicts a parameter of interest such as location, speed and heading of the vehicle in the presence of measurement errors. The fundamental principle of the Kalman Filter is to use a model of system being measured and update it with new measurements. Based on the previous state estimate, and the system model, the Kalman filter predicts the current state of the system. Then, it combines this prediction with a fresh measurement to obtain an updated state estimate. Here is a figure shows that how sensor fusion basically works by utilizing the Kalman Filter.

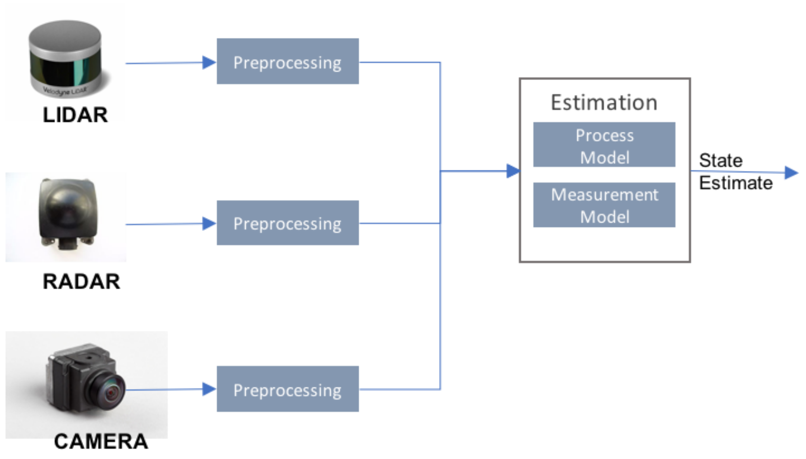

Figure 11. Sensor fusion working mechanism

Kalman Filter is based on Bayes' Theorem. This is an iterative process and tries to predict the state at each step and update the state based on the measurement data from the sensors. The figure below shows how Kalman filter tries to predict and update the state by using the previous sensor measurements. In this figure, the parameter of interest is the position of the autonomous vehicle. This parameter could also be the velocity, acceleration, etc.

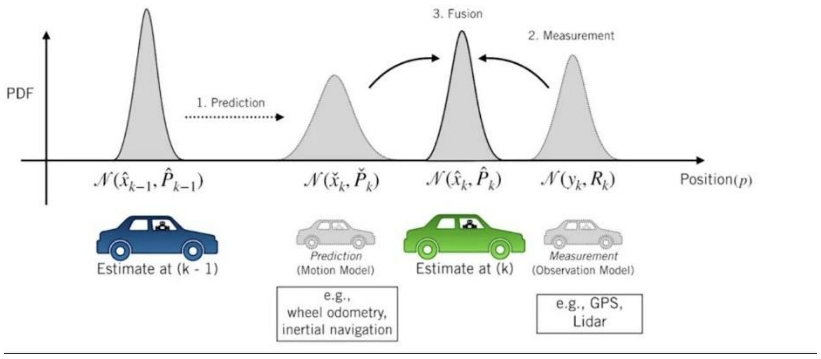
PDF
1. Prediction
3. Fusion
2. Measurement
$\mathcal{N}(\hat{x}_{k-1}, \hat{P}_{k-1})$
$\mathcal{N}(\check{x}_k, \check{P}_k)$
$\mathcal{N}(\hat{x}_k, \hat{P}_k)$
$\mathcal{N}(y_k, R_k)$
Position(p)
Estimate at (k - 1)
Prediction (Motion Model)
Estimate at (k)
Measurement (Observation Model)
e.g., wheel odometry, inertial navigation
e.g., GPS, Lidar

Figure 12. Kalman Filter's prediction and correction

In our project, we used the Kalman filter and the sensor fusion for localization of our autonomous vehicle, generating the efficient trajectory to reach the goal pose, tracking other vehicles, objects, or obstacles, and finally fusing data from multiple sensors. Here is the comparison of true trajectory and the estimated trajectory of our vehicle. True trajectory is the actual path followed by the vehicle in real world. It can only be measured through sensors on the vehicle. On the other hand, estimated trajectory is the path which is calculated by the Kalman filter using a combination of observed measurements and predictions based on the system model.

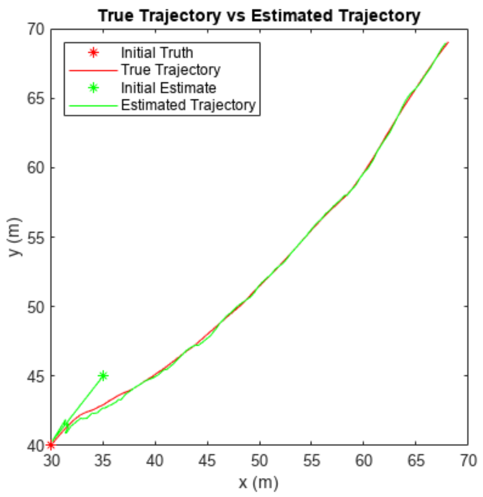

Figure 13. Comparison of true trajectory and the estimated trajectory

3.5 A* Path Finding Algorithm

We must find the shortest path to calculate the path our vehicle will follow. A* is a search algorithm that allows us to find the shortest path. The algorithm uses a combination of the cost to reach the node and a heuristic estimate of the cost from the node to the goal.

The blue grid is shown as our starting point and the orange grid is shown as our target point. A* measures the cost only between connected grids, starting from our current location. Shown on the top right is the distance of the current grid to the target gray and shown on the top left is the distance of the next grid to the target gray. A* finds the shortest path by adding these two costs.

There are many ways to represent a map and the position of the vehicle within the map. An occupancy grid is a matrix that corresponds to a region of 2-dimensional space. Elements containing zeros are free space where the robot can move, and those with ones are obstacles where the robot cannot move.

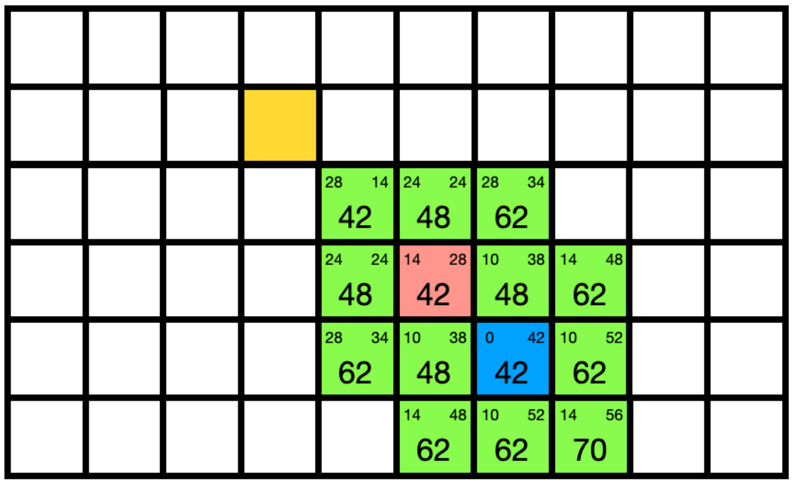

Figure 14. Demonstration of A path finding algorithm*

3.6 Ackermann Kinematic Model

Kinematic models are mathematical models that allow us to create control and movement on our vehicle. The Ackermann Kinematic Model is a mathematical representation used to describe the steering geometry of vehicles. While the vehicle makes a circular motion, its inner and outer wheels rotate in different radii. In order to prevent our vehicle from skidding and the wheels from being damaged, we must find different rotation angles of the wheels.

Let:

θi be the steering angle of the inner wheel.

θo be the steering angle of the outer wheel.

L be the wheelbase of the vehicle (distance between the front and rear axles).

W be the track width of the vehicle (distance between the left and right wheels).

The Ackermann principle states that for a vehicle to turn smoothly:

$$\boldsymbol{cot}(\theta\mathrm{i}) - \cot(\theta\mathrm{o}) = \frac{W}{L}$$

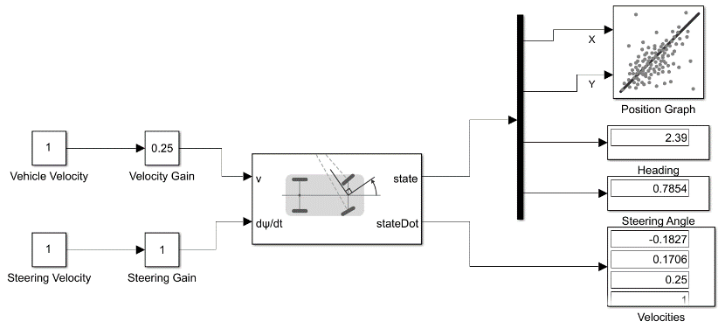

Figure 15. Ackermann Kinematic Model

Thanks to the Ackermann Kinematic model formulation, the wheels of the vehicle can rotate at different angles, so that the vehicle does not skid on the road.

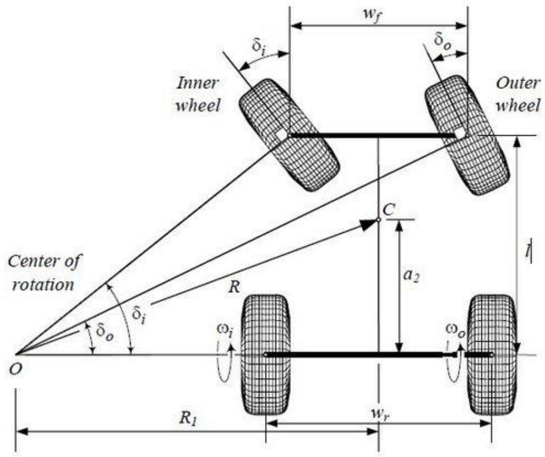

Figure 16. Rotation of angles in Ackermann

3.7 Parking Space Planner

3.7.1 Publisher-Subscriber Model

Until this point, we tried to explain the most crucial concepts in Autoware such as A* path finding algorithm, Ackermann kinematic controller, sensor fusion and Kalman filter. At the beginning of the report, we indicated that Autoware actually runs on our computer based on ROS2.

While developing our node and algorithm, our parking space planner package needs to follow the publisher-subscriber model. Inside of the parking space planner package, we created an executable node so that it could communicate with other nodes in Autoware. Our executable node needs to subscribe to the specific nodes via topics in order to utilize the necessary messages. Basically, our parking space planner node subscribed to 3 different nodes. These are "Vector Map" for mapping, "Odometry" for localization and "Predicted Objects" to check the availability of the parking spaces. By using all these data, fusing them, and adding our planner algorithm, it finally publishes the exact goal pose of the autonomous vehicle on the map. Here is the overall architecture of the publisher-subscriber model of our package that we designed from scratch.

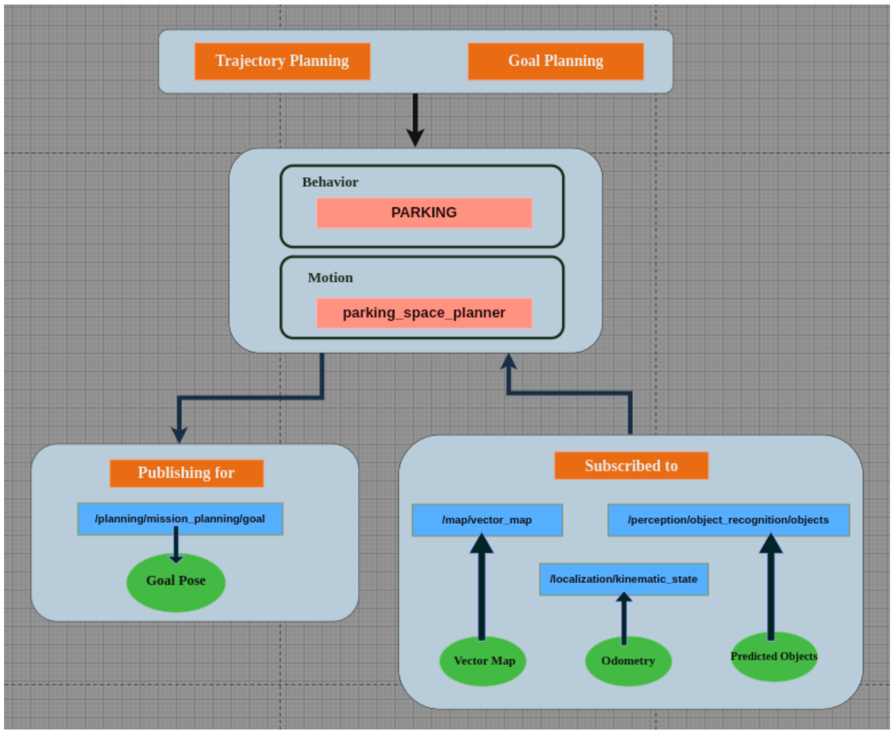

Figure 17. Publisher-Subscriber model of our package

3.7.2 Algorithm

We used the sample map which is provided by the Autoware. In Autoware, parking lots and parking spaces are defined in a way that our autonomous vehicle can define them. The parking lots are the big parking facilities has many parking spaces inside of it and it is defined with a polygon in Autoware. On the other hand, parking spaces are the areas which the vehicle should park inside, and it is defined with two-line strings in Autoware.

Our algorithm basically determines the nearest parking lot relative to the current position of the ego-vehicle. Then using the mapping, localization, and sensing components, it determines the positions of all parking spaces inside of the nearest parking lot. Among these parking spaces, it determines the nearest parking space relative to the current position of our ego-vehicle. Following to this step, it generates a trajectory to reach in front of the nearest parking space. When it arrives to this position, by using the “Predicted Objects” module from the perception component, it checks for the availability of the parking space. In other words, it checks that if there is any other object, obstacle, or pedestrian inside of the nearest parking space. If there is no object detected inside the nearest parking space, then the new goal pose will be inside the parking space and the parking operation will be completed. Otherwise, if there is an object detected inside of the nearest parking space, it basically generates a new trajectory in order to reach in front of the second nearest parking space. The autonomous vehicle keeps doing this operation until it finds the nearest empty parking space. Here is the flowchart of our algorithm.

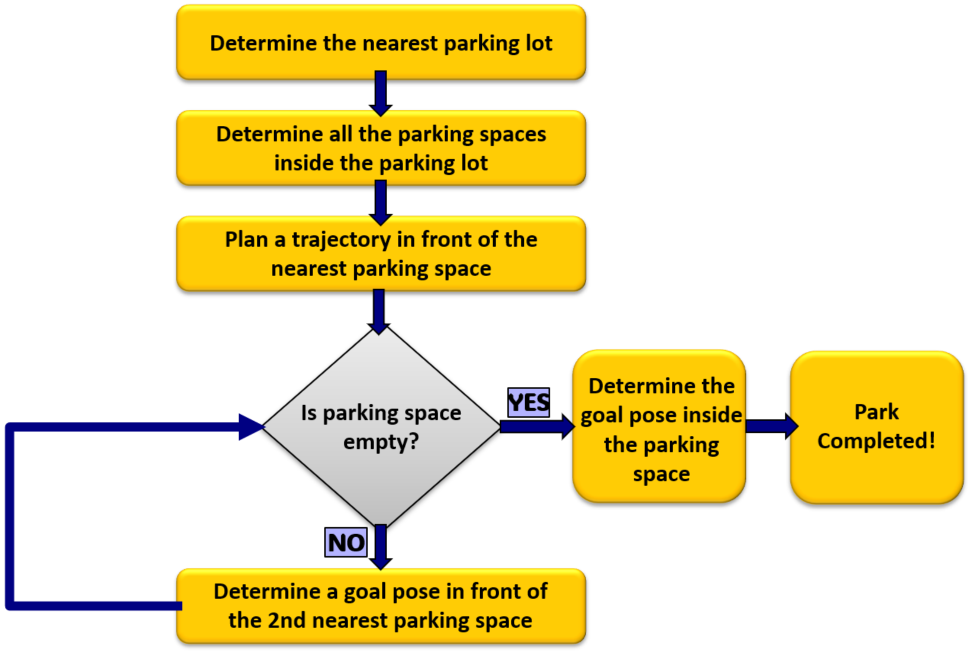
Determine the nearest parking lot
Determine all the parking spaces inside the parking lot
Plan a trajectory in front of the nearest parking space
Is parking space empty?
YES
Determine the goal pose inside the parking space
Park Completed!
NO
Determine a goal pose in front of the 2nd nearest parking space

Figure 18. Flowchart of parking space planner

ENGINEERING STANDARDS and DESIGN CONSTRAINTS

1) **Safety Standards**
    - ISO 26262: Functional safety for automotive equipment
    - IEC 61508: Functional safety of programmable electronics devices
2) **Performance Standards**
    - ISO 22737: Requirements for low-speed autonomous driving systems
    - SAE J3016: Classification the levels of automated driving
3) **Communication Standards**
    - V2X (Vehicle-to-Everything): Vehicle can share information through sensors with other vehicles, pedestrians, and infrastructure.
4) **Cybersecurity Standards**
    - ISO/SAE 21434: Cybersecurity engineering for road vehicles

In the development phases of this project, various realistic design constraints were considered in order to ensure that the project is economically feasible, environmentally friendly, socially beneficial, adoptable to industry, sustainable and so on. Here is the detailed description of the selected design constraints in the scope of this project :

Budget Limitations

In automotive industry, budget constraints play a crucial role in determining the choice of both software and hardware. Since we only use an open-source autonomous driving software called “Autoware” the cost of this project was nearly zero. We tested our product on the visualization tool “RVIZ”. At the hardware part of the autonomous vehicle, it cannot be considered as a cost-effective solution because 3D LIDAR sensors are really expensive in real life.

Benchmarking

The project is evaluated with the other existing semi-autonomous and fully autonomous parking systems available in the market. By benchmarking against these products, we determined the needs of the market and developed our competitive solution which offers advanced features at lower costs.

Environment – Friendly

At the beginning of this report, we mentioned that drivers release much more emission to the air while they were looking for a parking space. Our solution offers decreasing the air pollution by reducing the time it takes for searching for an empty parking space.

Assisted Parking for Society

Our autonomous parking system was designed to help society so that it minimizes the total waste of time, prevents minor accidents inside of parking lots due to taking extra risks and provides all the safety factors such as pedestrian safety, driver safety and other vehicles as well as following the traffic rules inside and outside of the parking facilities.

Information Security and Privacy

Since ROS2 provides robust encryption and communication in real time, secure communication is an important factor while collecting data from the environment by using different kinds of sensors. Ensuring the privacy of the drivers is a key point, especially in preventing unauthorized access to location information of the ego-vehicle.

Feasibility

Feasibility of such an improvement is a key point in autonomous-driving technology. Since Autoware could equip their own vehicle with the same sensors, and gets the expected outcomes, we used Autoware and its packages as a basis of our autonomous parking system, so that we can actually implement this parking space planner system in real-life with a suitable vehicle equipped with necessary sensors.

Sustainability

The autonomous parking systems need to be reliable and has to have high durability. It should not affect from the environmental conditions such as rain, temperature variation, dust, etc. The necessary protection on the sensors is crucial for durability and reliability of the autonomous parking system.

Adoptable to Industry

Since we have been developing this project with one of the most well-known car manufacturers, TOFAŞ which is jointly owned by Stellantis Group, we believe that this project will be implemented in real-life by using advanced sensors on the vehicle. The brand is supporting for us to generate our own map inside of the campus by using the point cloud data which was collected by the 3D Ouster OS1 LIDAR. Then we will use this map, draw the necessary lanelets and parking space line strings for Autoware to recognize. At the final phase of the project, these progresses will be implemented by using a real vehicle and in real-time.

In conclusion, the design of autonomous parking system was guided by a set of constraints mainly aimed at its feasibility, budget limitations, environmentally friendly, sustainability, safety, easily adoptable to industry, and reliability of the system. By addressing these parameters, our autonomous parking system aims to deliver robust, efficient and user-friendly solution to the autonomous parking problems in modern urban environments.

4. RESULTS AND DISCUSSIONS

The main aim of this project was to determine the nearest empty parking space relative to the current position of our autonomous vehicle and park the car autonomously inside of it. The map which we used for our experiments was provided by the Autoware itself. The parking lots are indicated with slightly faded yellow polygons. Inside of the parking lots, there are many parking spaces which are indicated with the yellow-colored areas. There could be some dynamic or static obstacles inside of these parking spaces sometimes. In this case scenario, there are some obstacles inside of the first, second and the fourth parking spaces.

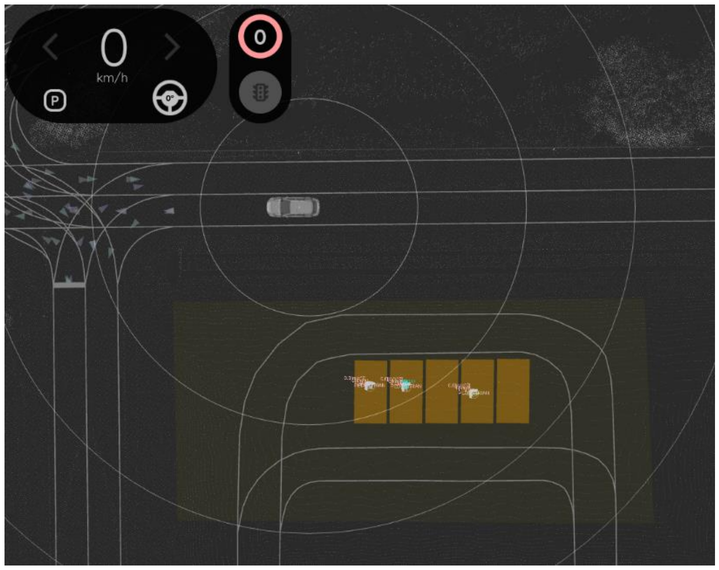

Figure 19. Ready to launch node

When we launch our parking space planner node, it should find the position of nearest parking space relative to the current position of our ego-vehicle which is the first parking space if we start counting from the left-hand side of the parking facility.

When it determines the position in front of the nearest parking space, by using the Kalman filter, sensor fusion and A* path finding algorithm, autonomous vehicle generates the most efficient and shortest path to this position and follows the generates trajectory autonomously.

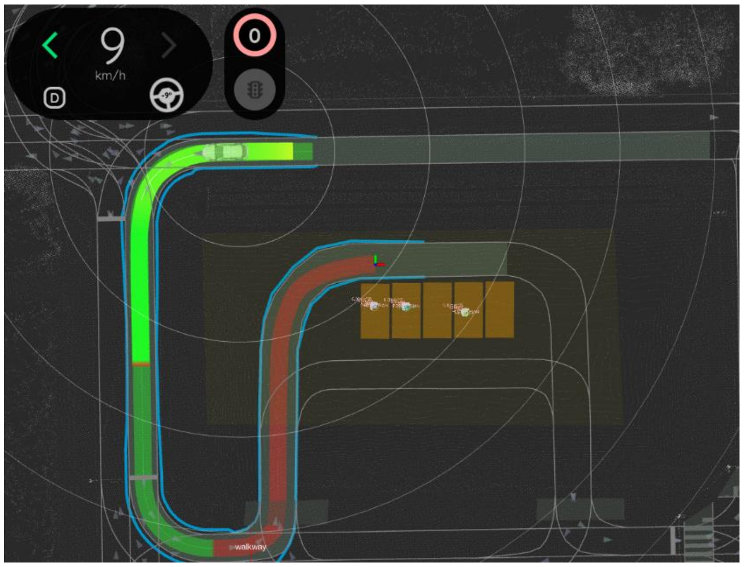

Figure 20. Following the trajectory to reach in front of the nearest parking space

Whenever the autonomous vehicle reaches in front of the nearest parking space, the algorithm simply checks that if there is any static or dynamic obstacle inside of this parking space by using the "Predicted Objects" module in Autoware. Since there is a pedestrian inside the nearest parking space, the algorithm basically determines the second nearest parking space. The new goal pose is the position in front of the second nearest parking space. By executing the algorithm, it simply follows the same steps.

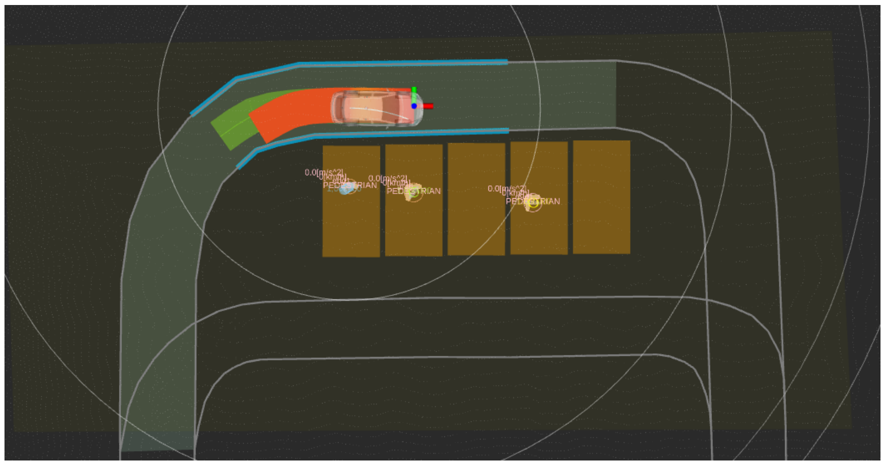

Figure 21. Determining a new goal pose

Again, there is a pedestrian inside the parking space, so it should go for the third one. Our autonomous vehicle does not recognize any other objects or obstacles inside of the third nearest parking space, so that the parking operation is now enabled. The goal pose as the final point will be given inside of the parking space.

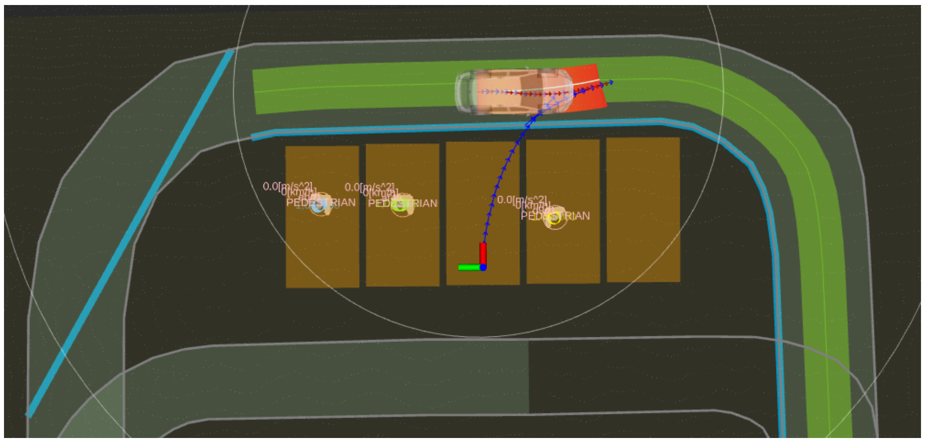

Figure 22. Finding the nearest empty parking space

Our autonomous vehicle parks neatly inside the nearest empty parking space using the A* search algorithm.

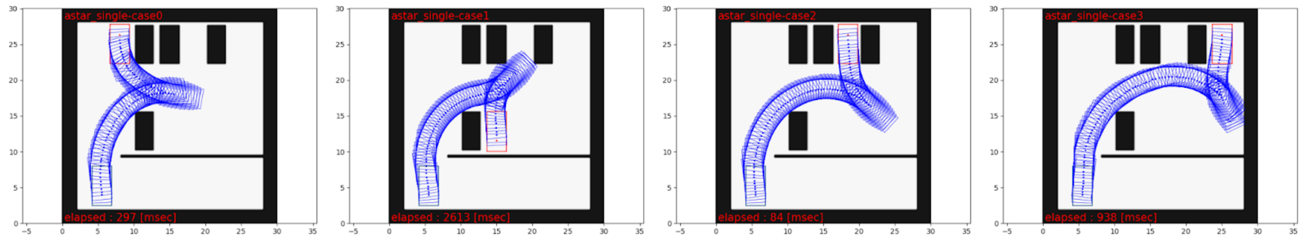

Figure 23. A* Search Algorithm (Single Curvature Case)

Here is the overall result of the autonomous parking system that we designed from scratch. It basically found the nearest empty parking space relative to the current position of our ego-vehicle and by using the sensor data our car is parked inside of the nearest empty parking space with high accuracy.

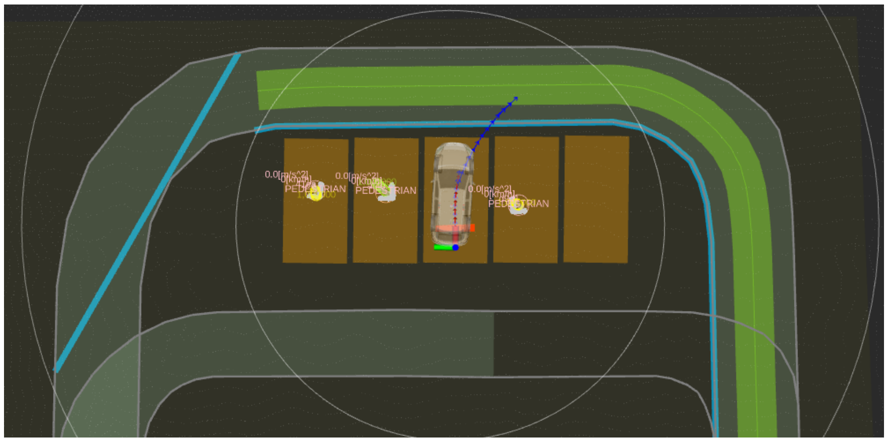

Figure 24. Park Completed

5. CONCLUSIONS

The autonomous driving technology has become very popular and an effective solution among different situations. The main problem that we focused on was the parking issue especially in modern urban environments. Utilizing the features of a fully autonomous driving stack “Autoware” and the and Robot Operating System to enable communication among many nodes within the overall system, we have developed a parking space planner algorithm which basically determines the nearest parking facility inside of the map relative to the current position of the autonomous vehicle. Then, the autonomous vehicle generates a trajectory by utilizing solid concepts such as sensor fusion, Kalman filtering, A* path finding algorithm, etc. While it follows that trajectory for reaching to the position in front of the nearest parking space, it also detects and classifies other objects or obstacles on the scene so that it avoids crashing. In a way, it follows all the traffic rules by using the sensors that places on the vehicle. These sensors basically collect different kinds of data from the environment and fuses them in order to extract a better meaning from the scene or environment.

As a result of the parking space planner algorithm that we designed from scratch, it first determines the nearest parking lot and then determines the nearest parking space inside of it. Following these operations, it checks the condition of parking space availability. This is a key point because there could possibly be some other vehicles, obstacles, or objects inside of the parking spaces. This condition is checked by the predicted objects module in Autoware, so that ego-vehicle could determine the objects inside of a specific area which are the parking spaces.

To sum up, parking space planner node communicates with other necessary nodes and topics in Autoware by utilizing the functions of Robot Operating System and as an output it returns the nearest empty parking space inside of the nearest parking lot relative to the current position of the ego-vehicle.

REFERENCES

[1] S. Kato et al., "Autoware on Board: Enabling Autonomous Vehicles with Embedded Systems," 2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)

[2] Ibrahim, Hossam El-Din, Car Parking Problem in Urban Areas, Causes and Solutions (November 25, 2017). 1st International Conference on Towards a Better Quality of Life, 2017

[3] Autoware, “Planning - Autoware Documentation,” autowarefoundation.github.io. https://autowarefoundation.github.io/autoware-documentation/main/design/autoware-interfaces/components/planning/ (accessed May 28, 2024).

[4] MATLAB, “Understanding Sensor Fusion and Tracking, Part 1: What Is Sensor Fusion?,” YouTube. Oct. 21, 2019. Accessed: Nov. 13, 2022. [Online]. Available: https://www.youtube.com/watch?v=6qV3YjFppuc

[5] MathWorks, “Understanding Kalman Filters, Part 1: Why Use Kalman Filters? Video,” Mathworks.com, 2017. https://www.mathworks.com/videos/understanding-kalman-filters-part-1-why-use-kalman-filters--1485813028675.html

[6] P. Corke, “1. Introduction,” Peter Corke. https://petercorke.com/rvc/home/1-introduction/ (accessed May 28, 2024).

[7] MathWorks, “Car-like vehicle motion using Ackermann kinematic model - Simulink,” www.mathworks.com. https://www.mathworks.com/help/robotics/ref/ackermannkinematicmodel.html (accessed May 28, 2024).

[8] “Understanding topics — ROS 2 Documentation: Humble documentation,” docs.ros.org. https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Topics/Understanding-ROS2-Topics.html (accessed May 28, 2024).

[9] yodayoda, “Localization with Autoware,” Map for Robots, Jul. 21, 2021. https://medium.com/yodayoda/localization-with-autoware-3e745f1dfe5d (accessed May 28, 2024).

[10] Open Robotics, “Why ROS 2?” Available: https://docs.ros.org/en/foxy/_downloads/2a9c64e08982f3709e23d20e5dc9f294/ros2-brochure-ltr-web.pdf