# PARKING SPACE PLANNER ALGORITHM ON LIDAR BASED MAP WITH AUTONOMOUS VEHICLE

**Cem Tolga Münyas**
TOFAŞ R&D İzmir Branch, TÜRKİYE
cemtolga2000@gmail.com

**Sibel Demir**
TOFAŞ R&D İzmir Branch, TÜRKİYE
sibelz@live.com

**Boğaçhan Ali Düşgül**
Izmir Institute of Technology, Engineering Faculty, Department of Electrical & Electronics Engineering, TÜRKİYE
bogachandusgul@gmail.com

## ABSTRACT

This paper proposes an effective solution to the shortcomings of urban parking inside the university campus (IZTECH) by developing an autonomous decision-making method in the parking space planner algorithm within the framework of an open-source autonomous driving stack called Autoware. Through the use of the libraries and components provided by Autoware, along with the Robot Operating System (ROS2) architecture, an efficient algorithm is developed that enables to locate the nearest parking facility and closest available parking space on the PointCloud map that we created using the Ouster OS-1 LIDAR scans within the campus. The main goal is to park the vehicle fully-autonomously inside the nearest empty parking space relative to the current position of the self-driving vehicle, considering dynamic objects, obstacles, pedestrians, and other vehicles. The results for autonomous vehicle to locate and park inside the closest empty parking space are observed on the 3D visualization tool called Rviz. The parking space planner node which is specifically designed for this operation is subscribed to three different nodes in Autoware which provides the mapping, localization and perception information for the autonomous vehicle. On the other hand, it publishes messages to the goal pose topic to keep update the goal position and orientation of the vehicle. Thus, the autonomous vehicle makes driving decision meticulously to identify and navigate to the nearest empty parking space on the map. To sum up, this parking space planner solution minimizes the time and effort for drivers to search for empty parking spaces in urban environments.

**Keywords**: Autonomous Parking, Decision-Making Algorithm, Robot Operating System, Urban Parking Solutions

## INTRODUCTION

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 with. On the other side, finding a suitable parking space in modern urban environments has become a significantly hard mission for drivers for many reasons [1]. The most significant reason is the human error in parking [1]. One bad parking in tight urban environments could effect other drivers to park their cars properly between the predetermined parking areas. In order to minimize this kind of human error, several automotive manufacturers have developed and released semi-autonomous parking systems as commercial products [2]. Moreover, automated valet parking systems have been introduced in a way that drivers can control their autonomous vehicles by the mobile application, which enables an autonomous car to drive to a parking spot and parks itself [2]. Despite the extensive research in the field of autonomous parking for vehicles, generating a free-obstacle trajectories for the autonomous vehicles in tight urban areas remains as a difficult task. Furthermore, this kind of 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 or the vehicle should be in suitable position for parallel-parking [2]. The main goal of this project is to design an autonomous parking algorithm that utilizes the capabilities of Autoware and the Robot Operating System (ROS2) to identify and navigate the autonomous vehicle to the nearest available parking space within the nearest parking lot relative to the current position of the vehicle in IZTECH (Izmir Institute of Technology) map. We called this algorithm parking space planner and it can be applied and executed in any environment with a LIDAR scanned map. The overall project provides an effective solution to the shortcomings of urban parking by providing an efficient, safe and user-friendly solution to the problem of finding and utilizing parking spaces especially in narrow urban areas. We believe that this autonomous parking project provides a critical improvement in comparison with the current parking technologies since it does not need the existence of driver activity.

## MATERIALS AND METHOD

The autonomous parking algorithm is developed on Autoware which is an open source autonomous driving stack software for self-driving vehicles. Autoware was used as the primary software framework for managing various aspects of autonomous driving and autonomous parking. Autoware's modular architecture facilitated the integration of different subsystems such as mapping, localization, motion planning, sensing, and control, into a unified system tailored for autonomous parking [3]. Autoware needs a middleware platform as a basis so that different nodes in Autoware can communicate in real-time [3]. Robot Operating System (ROS2) was used for this operation to ensure efficient communication between mapping, planning, perception etc. nodes.

ROS2 is a highly flexible, distributed framework especially designed for real-time robotics applications [4]. It enables communication between different components known as nodes, through its publisher-subscriber architecture [5]. For the project, ROS2 serves as the middleware that provides the real-time communication infrastructure, runtime environment and low-latency data transfer, while Autoware utilizes this foundation to implement higher-level features needed for autonomous driving [3]. Autoware also provides sample vehicle and sample sensor kit which composed of 3D LIDAR scanner, Global Navigation Satellite System (GNSS) and Inertial Measurement Unit (IMU) mainly used for mapping, localization and perception [3]. Autoware relies on high-definition point cloud and maps and vector maps of the driving environment to perform various tasks such as localization, decision-making, generating trajectories, etc [3]. Mapping in Autoware provides two types of information to the rest of the stack. It provides geometric information (3D visualization of other vehicles, nearby buildings, etc.) of the environment as point cloud map and semantic information about roads as a vector map. Vector map basically contains highly precise information about the lane geometry, road networks, parking lots, parking spaces and the traffic lights. One of the most significant contribution of this project to Autoware is mapping. We recorded the point cloud data in IZTECH campus by using a high-resolution LIDAR sensor Ouster-OS1 on top of the vehicle. Then, 3D pointcloud map is generated from recorded LIDAR scans which was placed on top of the vehicle. Here is an image taken from the generated point cloud map inside the campus.

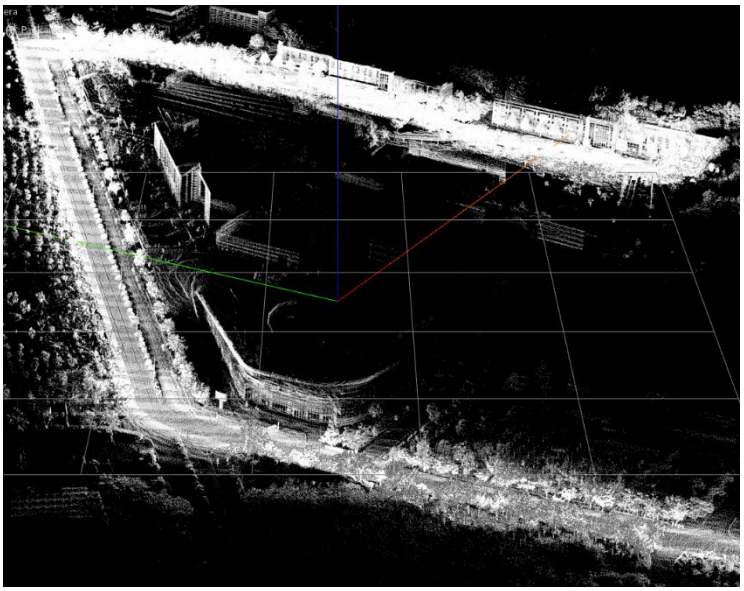


Figure 1. 3D point cloud map generated from LIDAR scans

3D Ouster-OS1 LIDAR is the only sensor that is used for creating a virtual environment from reality. Since the LIDAR has also equipped with an inner IMU, the mapping results were precise. On the software basis, sample vehicle is equipped with LIDAR, GNSS and IMU for basically localization and perception. Here is an image of the Ouster-OS1 LIDAR sensor mounted on top of the vehicle in order to collect data from the IZTECH environment and create a 3D map.

Figure 2. Ouster-OS1 LIDAR mounted on top of the vehicle to collect data

In addition to these sensors, software, and middleware; a 3D visualization tool for ROS is used to visualize and verify the autonomous driving and parking features of the self-driving vehicle. As mentioned earlier in the abstract, the most significant improvements to enhance such parking algorithm are the redesign and construction of mapping and planning components. On the other hand, perception packages in Autoware was utilized in order to enable object detection and path prediction. Each component in Autoware is connected each other and ROS2 provides the communication basis between different nodes within different packages [3]. Here is the high-level architecture of Autoware which consists of six stacks which the autonomous vehicle utilizes.

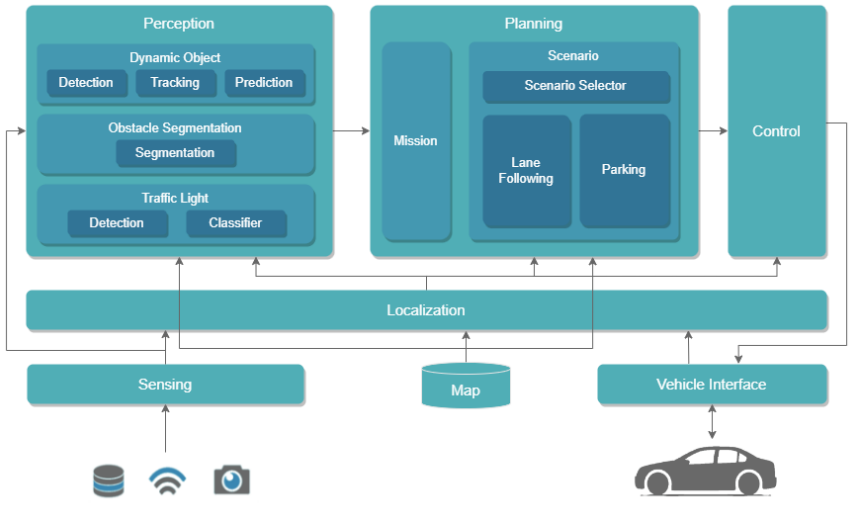



Diagram 1. High-level architecture of Autoware [6]

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 [7]. 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 [7]. Autoware plans the trajectory based on the current position of the self-driving 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 [8]. Planning module basically consists of two submodules which are mission planning and motion planning [7]. Mission planner employs a rule-based approach to calculate path trajectories depending on the driving states [7]. The mission planning also includes navigation from the current position to the destination [7]. On the other hand, motion planner is responsible for generating efficient and feasible trajectories by considering the states of the vehicle, drivable areas indicated by 3D point cloud map, surrounding objects, and desired goal pose [7]. The main aim of this project is to design and implement a planner package and a parking space planner node so that it could communicate with other nodes in Autoware. When this node is executed, it basically locates the nearest available parking space inside the nearest parking lot depending on the current position of the self-driving vehicle inside the IZTECH campus map. Since ROS2 is based on publisher-subscriber model, the parking space planner node is subscribed to four different topics in order to utilize these messages to update the goal pose of the self-driving vehicle. The subscribed nodes are “Vector Map” for mapping, “Odometry” for localization, and also “Predicted Objects” and “Detected Objects” for perception. By utilizing the messages published by the related nodes through topics, parking space planner node publishes messages to the goal pose topic so that the goal pose of the vehicle keeps updating itself while executing the parking maneuvers. Here is the publisher-subscriber architecture of the parking space planner node.

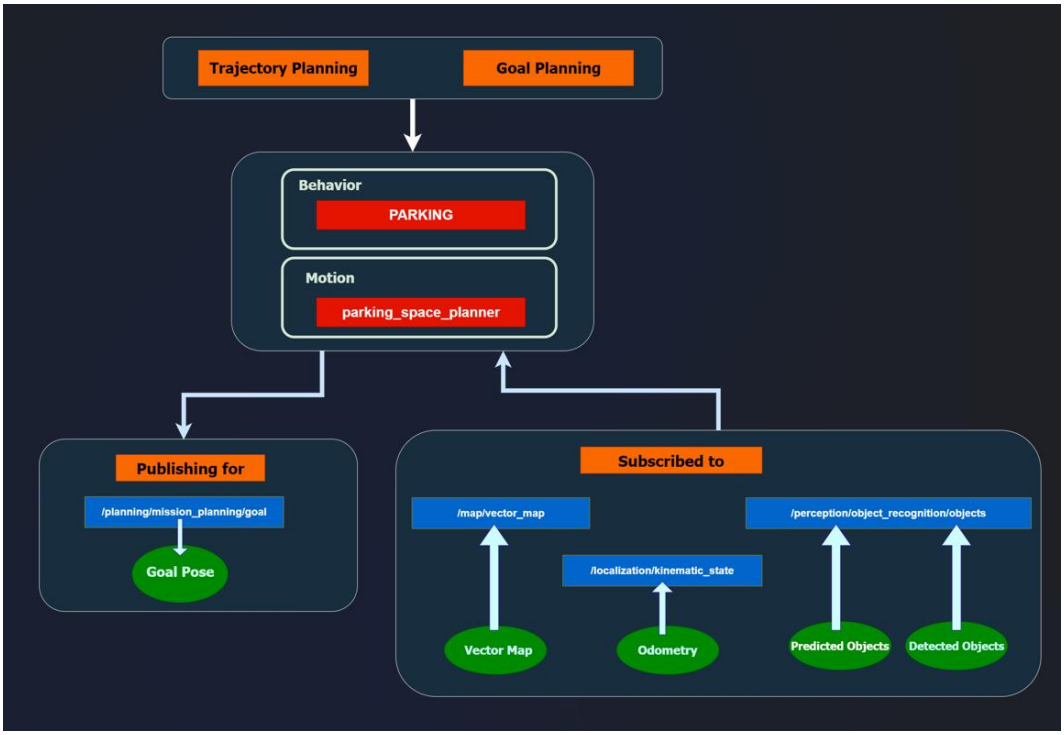



Diagram 2. Architecture of publisher-subscriber model of the parking space planner node

## RESULTS AND DISCUSSION

The main purpose of the parking space planner algorithm is to locate the nearest empty parking space inside of the nearest parking facility (parking lot) on the IZTECH campus map. As explained earlier on the “Materials & Method” part, the map of IZTECH campus has been generated by the collected 3D point cloud data scans with Ouster-OS1 LIDAR sensor. A consistent point cloud data has been collected from the environment, especially nearby the parking lots. Here is the 3D view of parking lot in campus which is generated from recorded LIDAR scans which was placed on top of the vehicle.

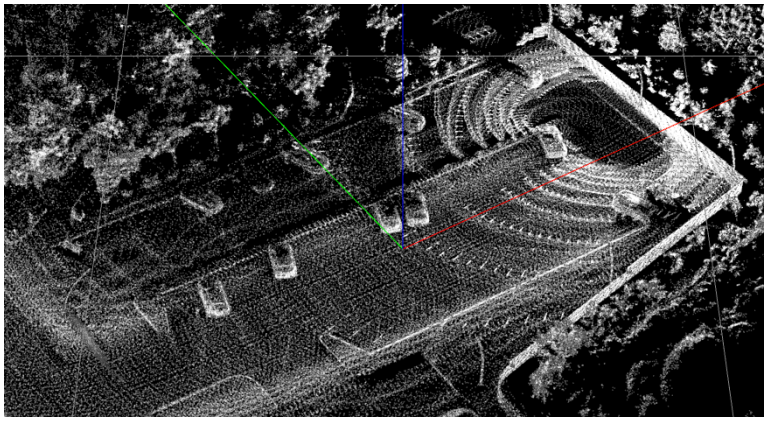


Figure 3. 3D view of point cloud map of the parking lot in campus

In order to spot the differences of the point cloud map versus the parking facility in real-life, the bird-eye-views of the parking lots are compared. Here is the bird-eye-view of the same parking lot in real-life.

Figure 4. Bird-eye-view of the parking lot in IZTECH campus

On the other hand, here are the results for point cloud map generated from LIDAR scans for the same parking lot.

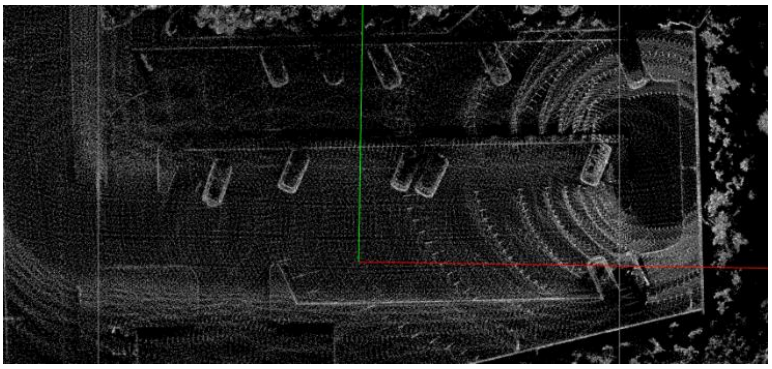


Figure 5. 3D point cloud map generated from LIDAR records

As mentioned earlier, Autoware recognizes the lanes, traffic rules, parking lots and parking spaces with a vector map [3]. In a web-based tool called Vector Map Builder, the lanelets (lanes), parking lots which are the parking facilities with many parking spaces inside, and the parking spaces are indicated. The parking lots are defined with polygons and the parking spaces are the areas which the self-driving vehicle should park inside, indicating with a line-string in Vector Map Builder. This process is the manual annotation to label specific features as in our case parking spaces and parking lots. These annotations help the autonomous vehicle better understand the boundaries and rules of the parking area, ensuring accurate planning.

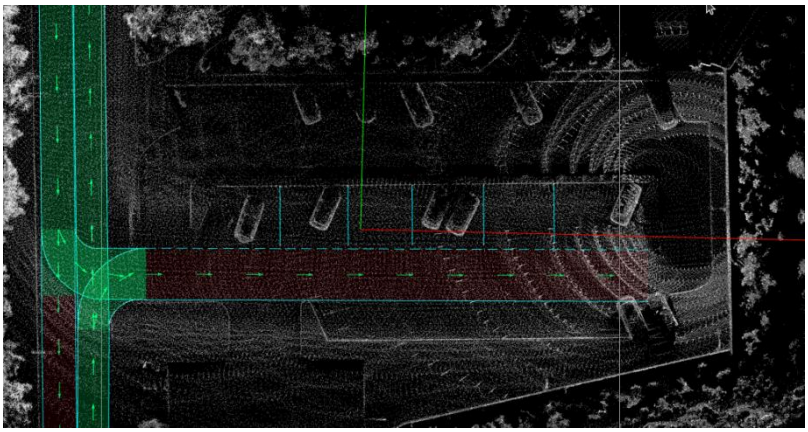


Figure 6. Vector map of the parking lot in campus

The mapping has been completed with the combination of point cloud map and the vector map (lanelet2 map). Once the map is created, the vehicle uses it for precise localization. In this project, a combination of NDT (Normal Distributions Transform) matching and GNSS (Global Navigation Satellite System) was used for localization within the overall map [9]. The NDT algorithm matches real-time sensor data from the LIDAR with the pre-built map to estimate the vehicle's position with high accuracy [9]. In other words, the NDT matching process uses the point cloud data to align the self-driving vehicle's current environment with the stored map which provides precise localization even in GNSS denied environments, such as indoor parking lots [9]. The generated vector map is used not only for localization but also for path planning. At this point, parking space planner package takes the control. This package plans the vehicle's behavior to find the nearest empty parking space inside the nearest parking lot including target trajectory and concerning traffic rules, avoiding other objects, etc.

The vector map includes predefined information about the parking lots and the parking spaces, which self-driving vehicle uses to locate available spots and navigate to them. Parking spaces are predefined on the vector map based on their physical locations in the real world. These parking spaces are marked with specific attributes such as their width, coordinates and boundaries. The physical boundaries of each parking space is included it allows planner to asses whether a particular space is suitable for the vehicle. Here is a visual from Rviz which shows five parking spaces inside of a parking lot.

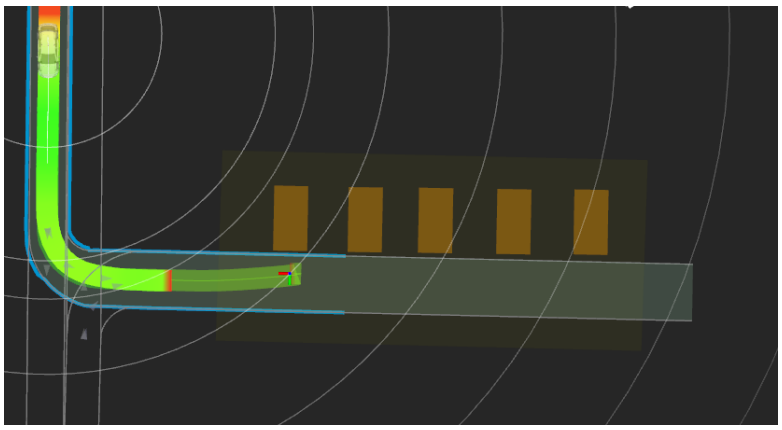


Figure 7. Pre-built parking spaces marked with yellow rectangles visualized on Rviz

The vector map also includes parking lot areas where groups of parking spaces are clustered. Each parking lot is predefined with attributes such as its coordinates of the boundaries. Since the parking lots can be created as polygons on vector map, boundaries define the limits of the parking lot area, ensuring that the vehicle understands where it is allowed to navigate and search for parking spaces. The predefinition of parking lots is crucial since Autoware allows parking spaces to be inside of the parking lots only [10].

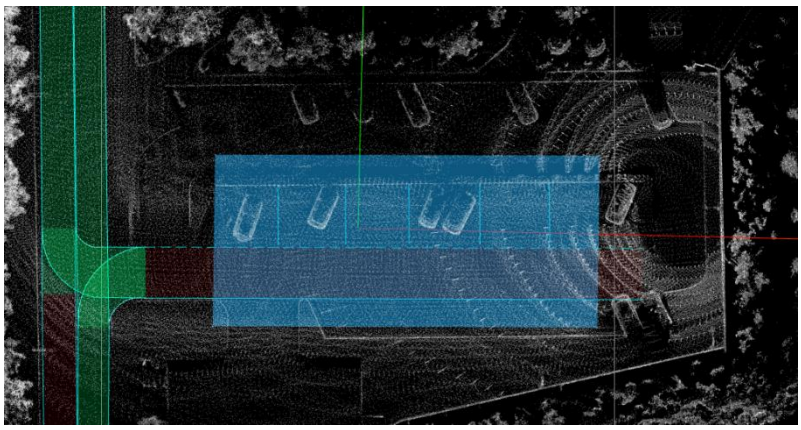
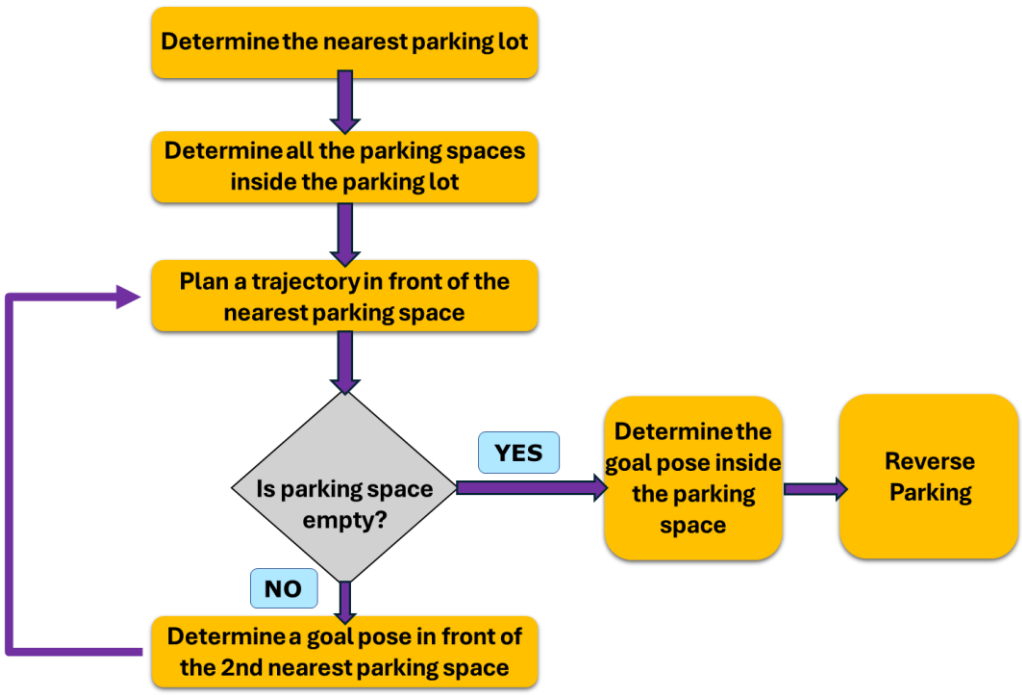


Figure 8. Pre-built parking lot marked with a blue polygon on vector map

As mentioned earlier, the self-driving vehicle uses LIDAR and GNSS sensors to determine its current position on the map, continuously updating its location relative to the IZTECH campus map. Based on the current position, the parking space planner node first identifies the closest predefined parking lot on the map among the other parking lots. Once the nearest parking lot is identified, the node searches for all the parking spaces within that parking lot and defines their

coordinations on the map. Then, the algorithm prioritizes parking spaces based on proximity to the vehicle's current position. After that step, the node plans a trajectory to reach in front of the closest parking space. Once the vehicles reaches in front of the nearest parking space, the decision-making mechanism takes over control. The perception modules "Detected Objects" and "Predicted Objects" check the availability of the parking space. If algorithm decides that the parking space is not occupied with any other vehicles, pedestrians, or obstacles, parking space planner node determines a new goal pose inside of the parking space which enables reverse parking. Otherwise, if the parking space is occupied with any other objects, obstacles, pedestrians, etc., the node determines a new goal pose in front of the second closest parking space. It again checks the availability of the parking space and decides to move on to the third one or to park inside of it depending on the occupancy. Once the parking space is determined as occupied, the node generates the trajectory in order to reach to the new determined goal pose. In some cases, parking space availability may change dynamically based on sensor data or external inputs such as arrival and departure of other vehicles. The mapping messages continuously feed the vector map topic, so that self-driving vehicle is aware of the dynamic changings in environment. The flowchart given below indicates the processes and the decision-making mechanism of the parking space planner node.




Flowchart 1. Decision making algorithm of parking space planner node

The perception module plays a critical role in determining the availability of parking spaces. This is done by processing 3D LIDAR data and utilizing two perception modules [11]. As mentioned earlier, "Predicted Objects" and "Detected Objects" messages in perception module were used check the availability of the parking spaces.

While object detection module detects the pose and velocity of dynamic objects on the environment, object prediction module predicts the possible paths of the dynamic objects. “Detected Objects” message contains information about objects that have been directly detected through LIDAR data [11]. The detection pipeline classifies and segments the sensor data, transforming it into clusters that represent various objects, such as cars, pedestrians, or obstacles in the parking lot [11]. The parking space planner node uses this message to look for potential vehicles, pedestrians, and obstacles inside or near parking spaces to determine if they are occupied or not. The 3D detection algorithm can recognize these objects based on their shape, size, and motion which enables self-driving vehicle to determine occupancy. Contributively, “Predicted Objects” messages provide predictions about the possible future positions and behavior of the detected objects [11]. For example, if a vehicle is detected approaching a parking space, the prediction module might estimate its trajectory and determine whether it will occupy the parking space shortly [11]. These predictions are especially important in dynamic environments such as parking lots where vehicles are moving in and out of parking spaces. In combination, the detected and predicted objects data allow the self-driving vehicle to check the current and future availability of parking spaces by considering both the static and dynamic environments. As an example from the IZTECH campus map, the first parking space was occupied by a pedestrian. The self-driving vehicle detects the pedestrian existence and predicts their possible future paths.

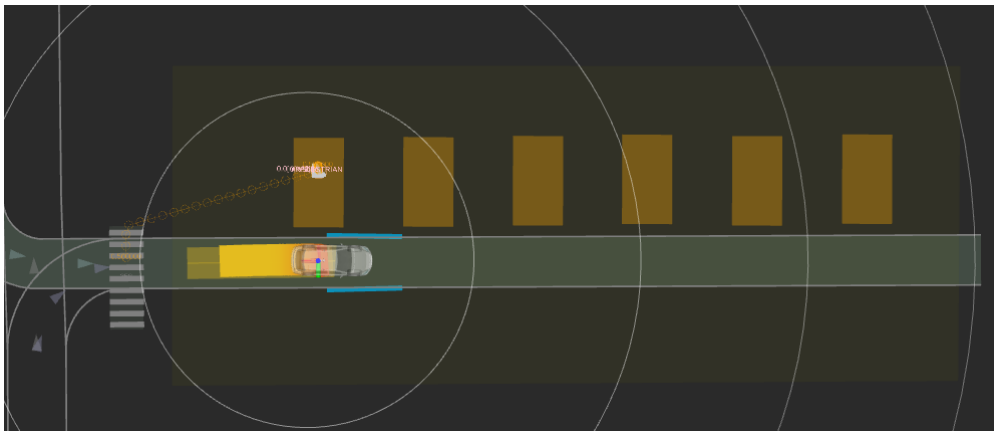


Figure 9. Parking space occupancy detection

Once the self-driving vehicle spots that the first parking space is occupied by a pedestrian, the goal pose is updated in front of the second one.

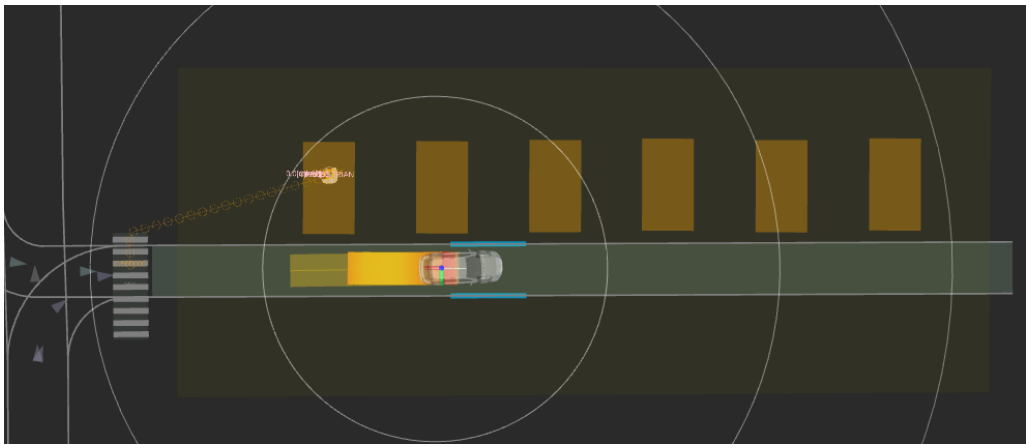


Figure 10. Updating goal pose in front of the second nearest parking space

In combination, cooperation of object detection and path prediction algorithms allow the self-driving vehicle to check the current and future availability of parking spaces by understanding both the static environment (e.g., parked cars) and dynamic factors (e.g., pedestrians on move, cars about to park). The detected objects are processed in the segmentation phase of the perception stack, while prediction is applied to analyze the future movements of the objects [11]. By leveraging these messages, parking space planner node efficiently evaluates the parking availability and guides the self-driving vehicle to the nearest unoccupied space. As a final demonstration of all the work done combined together, the parking space planner algorithm will be in action on the IZTECH map generated with real-life LIDAR scans and drawn lanelets, parking lot, and parking spaces. All figures are collected from Rviz while parking space planner node was active. The system begins by identifying the self-driving vehicle's initial pose and recognizing the parking lot layout, which includes five linked parking spaces. The first two parking spaces are occupied with the pedestrians.

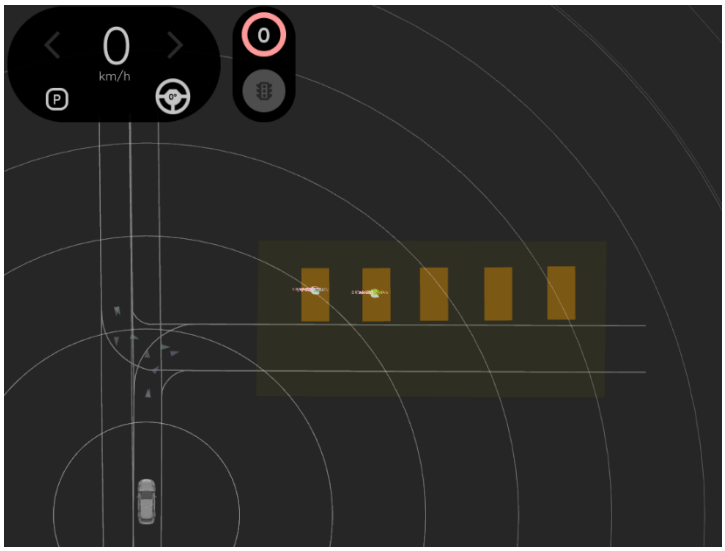



Figure 11. Setting initial pose of the self-driving vehicle on the IZTECH map

Once the parking space planner node is initialized, the self-driving vehicle locates the nearest parking space relative to its current position on the map. It determines a goal pose in front of this parking space and approaches it autonomously.

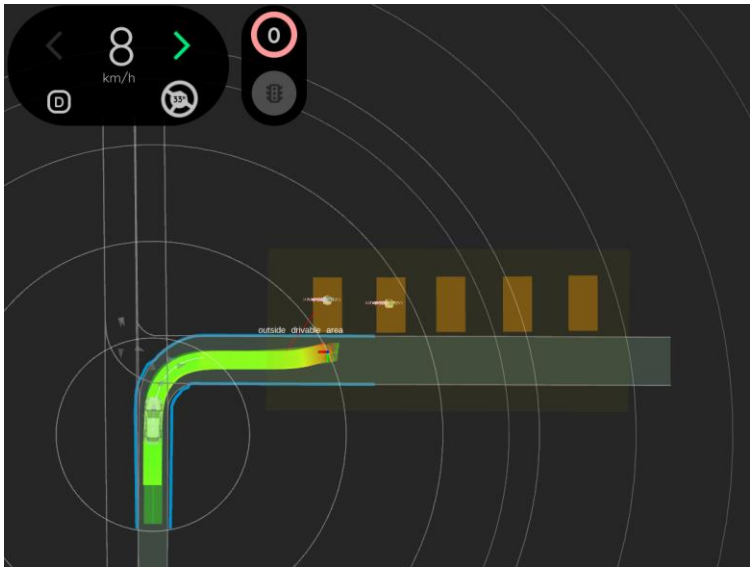



Figure 12. Determining the goal pose in front of the nearest parking space

When the self-driving vehicle reaches to that goal pose, it conducts a space check based on the data from the perception module using the “Detected Objects” and “Predicted Objects” messages and determining that it is occupied with a pedestrian. The vehicle then locates the second closest parking space regarding to its initial position, and it changes the driving mode from “Stop” to “Autonomous”. The vehicle again checks the availability of the space when it arrives in front of second nearest parking space, which is also deemed unavailable due to the presence of another pedestrian.

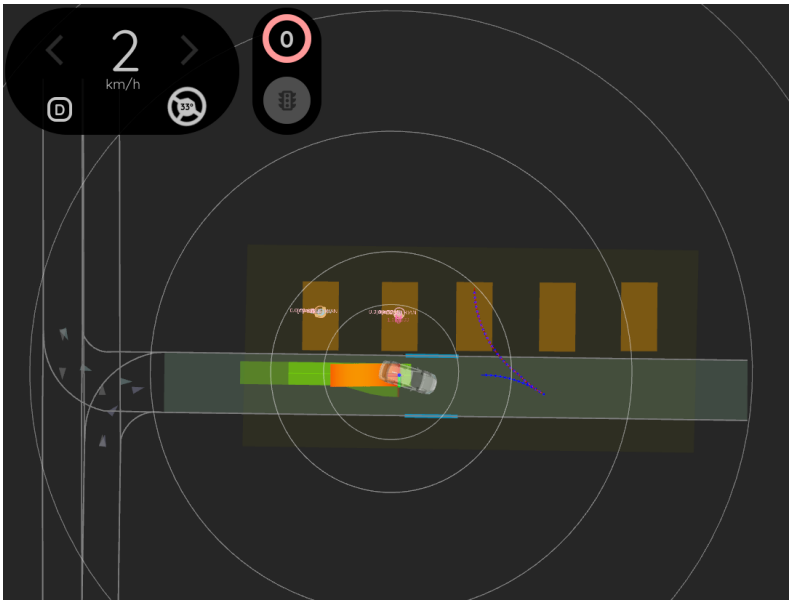



Figure 13. Arriving in front of the second closest parking space and checking its availability

Finally, the vehicle proceeds to the third closest parking space. Once the messages from the perception module feed the subscription module in the parking space planner node, the algorithm decides that the space is unoccupied.

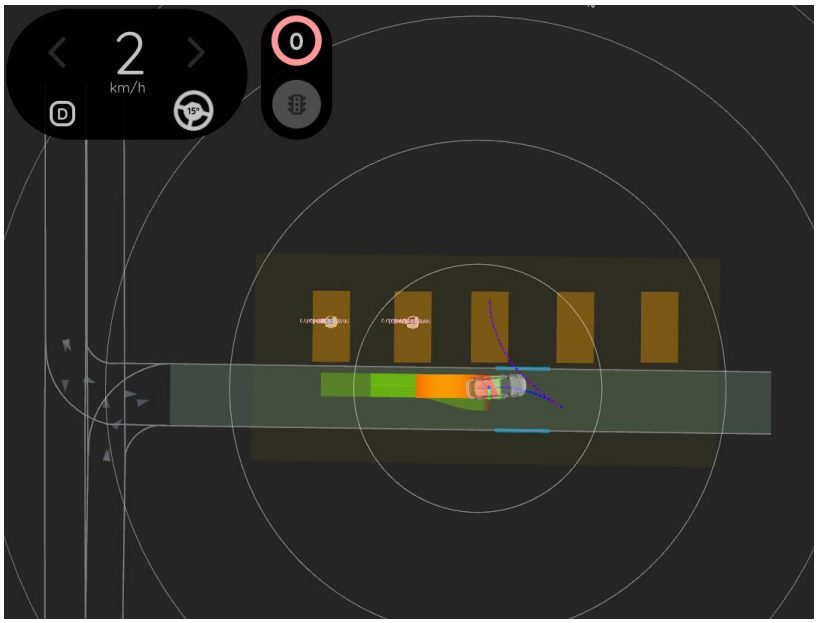



Figure 14. Arriving in front of the third closest parking space and checking its availability

At this point, the parking space planner node computes a goal pose for the vehicle inside of this parking space, taking into account the vehicle's dimensions as well as the length and width of the parking space.

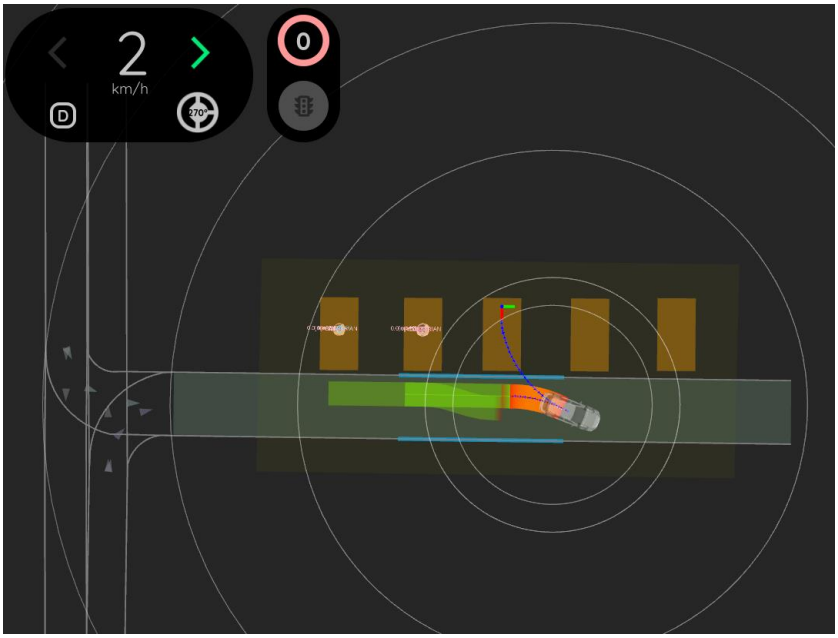



Figure 15. Determining the goal pose inside the nearest empty parking space

Once the criteria for safe parking are met, the self-driving vehicle performs a reverse parking maneuver by using the A* search algorithm (single curvature case) to shorten the distance while parking [12]. The A* search algorithm is used to calculate an optimal trajectory that guides the self-driving vehicle into the parking space while considering the constraints of the vehicle's steering capabilities, represented by a single fixed curvature [12]. A* also helps by ensuring the selected path avoids collisions while adhering to the vehicle's turning constraints. This is particularly important when parking in tight spaces or reversing into a spot like in our parking scenario.

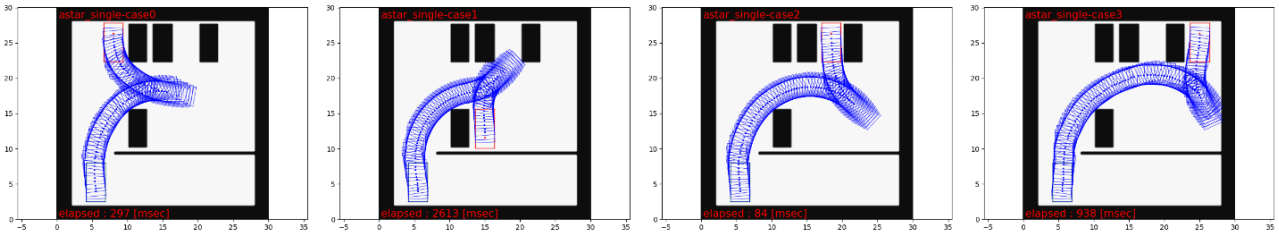



Figure 16. A* Path finding algorithm (single curvature case) [12]

When the self-driving vehicle finishes its reverse parking maneuver and successfully parks in the closest available parking space, it changes the driving state from "Autonomous" to "Stop" and sends a message to the planner node that the parking operation is successfully completed. Here is the final output of the successful execution of the parking space planner algorithm.

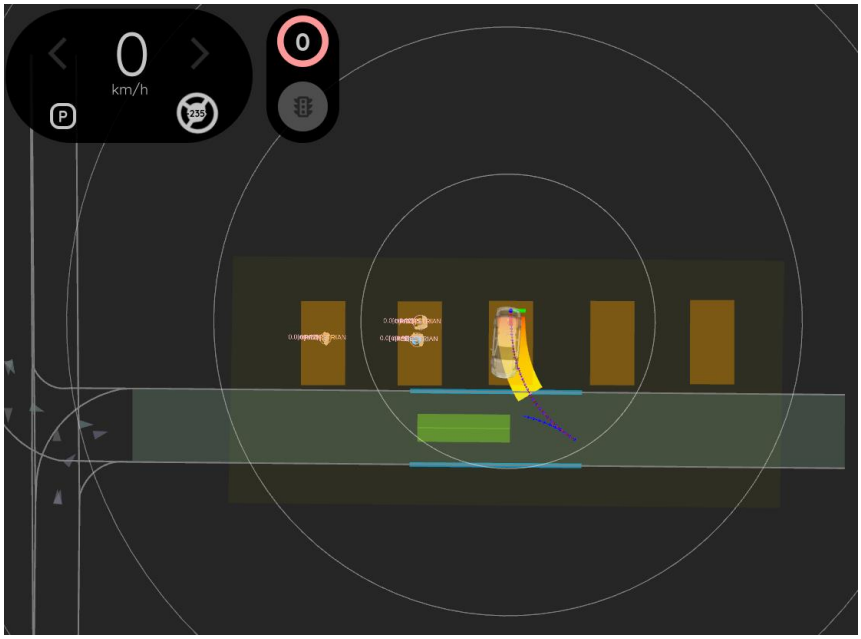



Figure 17. Autonomous parking is completed into the closest empty parking space on IZTECH map

These sequences illustrate the parking space planner algorithm's ability to dynamically evaluate parking space availability and adjust its path to ensure a safe and efficient parking operation.

## CONCLUSION

The overarching goal of this project was to develop a cost-effective, fully-autonomous, and safe urban parking solution that could compete with existing market solutions. By leveraging a comprehensive open-source autonomous driving stack software Autoware, built on top of ROS2, a system has been built which which is capable of performing autonomous parking in real-world conditions with a focus on safety, efficiency, and affordability. Autoware provides essential components for autonomous driving, such as localization, decision-making, perception, control, sensing, and generation of feasible trajectories [6]. Utilizing the Autoware architecture, an autonomous parking solution was developed that integrates real-time sensor-data, perception algorithms, behavior and motion planning, to ensure safe and efficient parking. One of the key elements in this implementation was the creation of a custom map, which was generated from point cloud data recorded at the IZTECH campus using an Ouster OS-1 3D LIDAR sensor. Then, the vector map was generated onto the point cloud map which includes the actual lanes (lanelets), crosswalks, parking lot and parking spaces indications, traffic lights, etc. This map was critical for the vehicle's localization, enabling the system to understand its position relative to the parking lot and parking spaces. The mapping module combined with the perception messages "Predicted Objects" and "Detected Objects", allowed the vehicle to identify obstacles and predict their possible paths like pedestrians or parked vehicles, ensuring accurate decision-making when scanning the parking spaces. The parking space planner algorithm was crucial in enabling the self-driving vehicle to locate the nearest parking lot, and within that lot, the closest unoccupied parking space. The planner relied on the localization and perception modules to process 3D LIDAR data and the messages from three different ROS2 topics helped vehicle to understand the surroundings, localize on every point of the lanelet map, and anticipate the future movements of detected obstacles providing self-driving vehicle to make forward-looking decisions about the parking space availability. In this step, the localization and the perception informations should be precise for safety considerations in order to enable operation in tight urban environments with dynamic obstacles, such as pedestrians moving near the parking area, vehicles entering or exiting the parking lot, etc [11]. To conclude, the system successfully demonstrated its ability to navigate within a complex parking environment, detect obstacles and predict their paths, and autonomously park in the closest available space. The use of Autoware and ROS2 proved to be an effective combination for achieving the objectives of the project, and the custom-designed parking space planner algorithm allowed self-driving vehicle to present an efficient and safe autonomous parking system in urban environments. This work demonstrates the potential of open-source autonomous driving software in practical applications, showing that it is possible to achieve high levels of performance without the need for expensive, proprietary systems.

**REFERENCES**


[1] H. Ibrahim , “Car Parking Problem in Urban Areas, Causes and Solutions,” *papers.ssrn.com*, Nov. 25, 2017. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3163473 (accessed May 15, 2024).

[2] A. Suppé, L. Navarro-Serment, and A. Steinfeld, “Semi-Autonomous Virtual Valet Parking.” [Online]. Available: https://www.auto-ui.org/10/proceedings/p139.pdf (accessed Aug. 14, 2024)

[3] S. Kato et al., “Autoware on Board: Enabling Autonomous Vehicles with Embedded Systems,” IEEE Xplore, Apr. 01, 2018. https://ieeexplore.ieee.org/document/8443742 (accessed Nov. 08, 2023).

[4] “Why ROS 2?” Available: https://docs.ros.org/en/foxy/_downloads/2a9c64e08982f3709e23d20e5dc9f294/ros2-brochure-ltr-web.pdf (accessed Nov. 11, 2023).

[5] “Understanding nodes — ROS 2 Documentation: Humble documentation,” docs.ros.org. https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Nodes/Understanding-ROS2-Nodes.html (accessed Nov. 11, 2023).

[6] Tier4, “Architecture overview - Autoware Documentation,” autowarefoundation.github.io. https://autowarefoundation.github.io/autoware-documentation/main/design/autoware-architecture/ (accessed Feb 08, 2024).

[7] Tier4, “Planning component design - Autoware Documentation,” autowarefoundation.github.io. https://autowarefoundation.github.io/autoware-documentation/main/design/autoware-architecture/planning/ (accessed Feb 08, 2024).

[8] Amit’s Thoughts on Pathfinding, “Introduction to A*,” theory.stanford.edu. https://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html (accessed Mar. 04, 2024).

[9] S. Chen et al., “NDT-LOAM: A real-time lidar odometry and mapping with weighted NDT and LFA,” IEEE Sensors Journal, vol. PP, p. 12, Jan. 2022, doi: https://doi.org/10.1109/JSEN.2021.3135055. (accessed May 12, 2024).

[10] Tier4, “The `freespace_planner` - Autoware Universe Documentation,” Github.io, 2022. https://autowarefoundation.github.io/autoware.universe/pr-2609/planning/freespace_planner/ (accessed Mar. 01, 2024).

[11] Tier4, “Perception Component Design - Autoware Documentation,” Github.io, 2023. https://autowarefoundation.github.io/autoware-documentation/main/design/autoware-architecture/perception/ (accessed Feb. 12, 2024).

[12] Tier4, “freespace planning algorithms - Autoware Universe Documentation,” Github.io, 2014. https://autowarefoundation.github.io/autoware.universe/main/planning/autoware_freespace_planning_algorithms/ (accessed Feb. 08, 2024).

[13] yodayoda, “Localization with Autoware,” Map for Robots, Jul. 21, 2021. https://medium.com/yodayoda/localization-with-autoware-3e745f1dfe5d (accessed Apr. 17, 2024).