NAVIGATION AND OBSTACLE AVOIDANCE FOR AUTONOMOUS VEHICLE

21102031 Shubhankar Kaushik
21102033 Saksham Saha

Name of Supervisor: Mr. Ritesh Kumar Sharma

May-2025

Submitted in partial fulfilment of the Degree of
Bachelors of Technology

DEPARTMENT OF ELECTRONICS
JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, SEC-62,
NOIDA

TABLE OF CONTENTS

CERTIFICATE

This is to certify that the work titled “Navigation and Obstacle Avoidance for Autonomous Vehicles” submitted by “**SHUBHANKAR KAUSHIK (21102031)**” and “**SAKSHAM SAHA (21102033)**” in partial fulfilment for the award of degree of Bachelors of Technology of Jaypee Institute of Information Technology, Noida has been conducted under my supervision. This work has not been submitted partially or completely to any other University or Institute for the award of this or any other degree or diploma.

(Prof. Shweta Srivastava)

(Mr. Ritesh Kumar Sharma) **(Prof. Shweta Sristava)**

Associate Professor HOD - ECE

Date: 10/5/2025

DECLERATION

I/We declare that this written submission represents my/our ideas in my/our own words and in where others ideas or words have been included, I have adequately cited and referenced the original sources. I/We also declare that I/we have adhered to all principles of academic honesty and integrity and have not misrepresented of fabricated or falsified any idea/ data/fact/source in my submission. I/We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Signature:	Kaushik	
Name:	Shubhankar Kaushik	Saksham Saha
Enrolment:	21102031	21102033

Date: 10/5/2025

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our supervisor **Mr. Ritesh Sharma** for providing his invaluable guidance, comments, and suggestions throughout the course of the project. He consistently motivated and guided us towards the completion of the project. We are highly indebted to him for his constant supervision as well as providing necessary information regarding the project. However, it would not have been possible without the kind support and encouragement of our parents and colleagues who have helped us out with their abilities in developing the project.

Signature:	Kaushik	
Name:	Shubhankar Kaushik	Saksham Saha
Enrolment:	21102031	21102033

Date: 10/5/2025

SUMMARY

Autonomous vehicles have revolutionized mobility, offering innovative solutions to meet diverse needs. One such application is developing autonomous vehicles tailored for disabled individuals, ensuring they can navigate their surroundings effortlessly and independently. These vehicles rely on advanced object detection and motion control systems to operate effectively in real-world environments.

In this project, we focus on integrating vision-based object detection and motion planning algorithms to create a reliable autonomous vehicle prototype. Initially, we implemented a detection system capable of identifying obstacles, pathways, and points of interest using advanced computer vision techniques. Next, we added motion control features that enable smooth navigation to predefined destinations. Finally, the prototype was tested in a simulated environment with dynamic obstacles to validate its ability to avoid collisions and adapt to real-time challenges. This project aims to enhance accessibility while maintaining efficiency and safety for users with limited mobility.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AV	Autonomous Vehicle
UAV	Unmanned Aerial Vehicle
ROS	Robot Operating System
LiDAR	Light Detection and Ranging
RADAR	Radio Detection and Ranging
GPS	Global Positioning System
IMU	Inertial Measurement Unit
DWA	Dynamic Window Approach
DWB	Dynamic Window Based
SLAM	Simultaneous Localization and Mapping
GMapping	Grid Mapping (SLAM algorithm using particle filters)
AMCL	Adaptive Monte Carlo Localization
VSLAM	Visual Simultaneous Localization and Mapping
MCL	Monte Carlo Localization
PID	Proportional-Integral-Derivative Controller
TF	Transform Frame (used in ROS)
PCL	Point Cloud Library
FPS	Frames Per Second
CPU	Central Processing Unit
GPU	Graphics Processing Unit
RRT	Rapidly-exploring Random Tree
RRT*	Optimal Rapidly-exploring Random Tree
A*	A-Star Search Algorithm
D*	Dynamic A-Star
HMI	Human-Machine Interface
VLP-16	Velodyne LiDAR Puck-16 (16 channel)
CAD	Computer-Aided Design
SL	Supervised Learning
IL	Imitation Learning
DRL	Deep Reinforcement Learning
SDV	Self-Driving Vehicle
QOS	Quality of Service (in ROS2)
FPS	Frames Per Second
ROI	Region of Interest
TTS	Time-To-Stop
TTC	Time-To-Collision
RViz	ROS Visualization

CHAPTER 1
INTRODUCTION

1.1 Introduction

Autonomous navigation and obstacle avoidance form the backbone of intelligent robotic systems, enabling machines to traverse environments without human intervention. As the demand for automation in industries, delivery systems, and assistive technologies continues to grow, the need for reliable and efficient autonomous navigation solutions becomes increasingly critical.

This project focuses on the design and implementation of a self-navigating vehicle capable of making intelligent decisions in real-time using environmental feedback. At its core, the system integrates localization, mapping, path planning, and motion control to ensure smooth and collision-free movement. By leveraging sensor data to perceive its surroundings, the vehicle constructs an understanding of its environment and dynamically adjusts its path in response to obstacles and changes in terrain.

The architecture of the system is built around a continuous feedback loop: sensors provide data about the world, the onboard planners determine viable paths, and controllers execute motion commands based on those decisions. Central to this process is the use of real-time map updates and planning algorithms that account for the robot's immediate surroundings. The robot evaluates potential routes, selects the optimal path, and continuously refines its strategy to avoid collisions while progressing toward its goal.

This project not only explores the technical intricacies of autonomous navigation but also emphasizes the practical challenges of building a system that is both responsive and reliable in real-world scenarios. Through iterative development and testing, the vehicle is trained to navigate complex environments, showcasing the effectiveness of modern planning techniques in real-time robotic applications.

1.2 Overview of Autonomous Navigation Systems

Autonomous navigation systems are built on the integration of perception, localization, mapping, path planning, and control. Each component plays a critical role in enabling a robot to move from point A to point B without human guidance.

Perception involves the use of sensors like LiDAR, cameras, and IMUs to understand the environment. This sensory data is used to identify obstacles, recognize terrain types, and detect changes in surroundings. Localization helps the robot determine its position within a global or local map, which is crucial for effective planning.

Mapping is the process of creating a representation of the environment, often in the form of occupancy grids or costmaps. These maps help in assessing obstacle density and traversable areas. Path planning algorithms use this map and localization data to find a feasible path to the destination. Control systems then translate the planned path into physical movement by sending commands to motors or actuators.

This layered architecture allows for modularity and scalability, making it easier to upgrade or replace individual components without overhauling the entire system. The integration of these layers in a feedback-driven architecture makes autonomous navigation systems both intelligent and adaptable.

1.2.1 Costmaps and Environmental Representation

In autonomous navigation, the costmap is a critical data structure that represents the environment around the robot. It provides a real-time spatial understanding of obstacles, free space, and zones that may be traversable with caution. A costmap is essentially a 2D grid where each cell is assigned a cost value representing how "safe" or "risky" it is to move through that cell.

There are typically two types of costmaps used in the navigation stack:

- **Global Costmap**

  Created based on a prebuilt map or SLAM-generated map, this costmap covers a larger area and is used by the global planner (like NavFn) to find an optimal route from the robot's current location to the goal.

- **Local Costmap:**
  Continuously updated using real-time sensor data (e.g., from LiDAR), the local costmap focuses on the robot's immediate surroundings. It allows for dynamic obstacle avoidance and short-term trajectory adjustments, especially useful in unstructured or crowded environments.

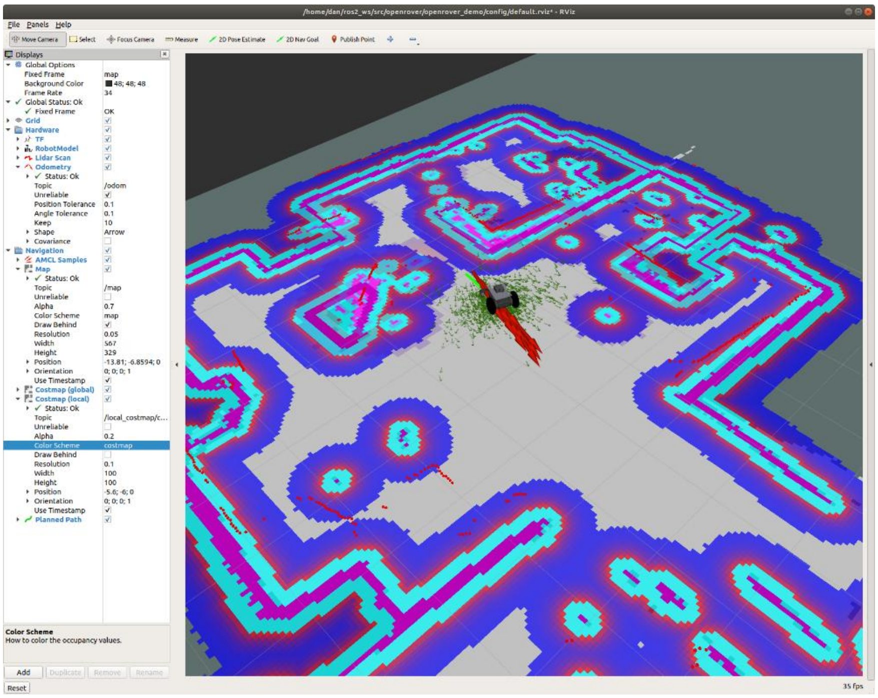
1.2.2 Costmap Visualization (RViz):

In RViz, costmaps appear as colored overlays:

- Black: Unknown areas
- White: Free space

- Shades of gray: Areas near obstacles
- Red: Occupied or impassable zones

These representations help the planner identify safe and unsafe zones and determine efficient detours in real time.



```
Global Options
    Fixed Frame              map
    Background Color         48; 48; 48
    Frame Rate               34
Global Status: Ok
    Fixed Frame              OK
Grid                         ✓
Hardware                     ✓
    TF                       ✓
    RobotModel               ✓
    Lidar Scan               ✓
    Odometry                 ✓
        Status: Ok
        Topic                /odom
        Unreliable           ✓
        Position Tolerance   0.1
        Angle Tolerance      0.1
        Keep                 10
        Shape                Arrow
        Covariance           □
Navigation                   ✓
    AMCL Samples             ✓
    Map                      ✓
        Status: Ok
        Topic                /map
        Unreliable           □
        Alpha                0.7
        Color Scheme         map
        Draw Behind          ✓
        Resolution           0.05
        Width                567
        Height               329
        Position             -13.81; -6.8594; 0
        Orientation          0; 0; 0; 1
        Use Timestamp        ✓
    Costmap (global)         ✓
    Costmap (local)          ✓
        Status: Ok
        Topic                /local_costmap/c...
        Unreliable           □
        Alpha                0.2
        Color Scheme         costmap
        Draw Behind          □
        Resolution           0.1
        Width                100
        Height               100
        Position             -5.6; -6; 0
        Orientation          0; 0; 0; 1
        Use Timestamp        ✓
    Planned Path             ✓

Color Scheme
How to color the occupancy values.
```

Figure 1.1 *Rviz Costmap*

1.2.3 Role of Behavior Trees in Navigation

Another vital aspect of autonomous navigation in modern ROS2-based systems like Nav2 is the use of **Behavior Trees (BTs)**. Unlike traditional state machines, behavior trees provide a modular, hierarchical, and easily extensible structure for robot decision-making.

Behavior trees execute in discrete update steps known as **ticks**. When a BT is ticked, usually at some specified rate, its child nodes recursively tick based on how the tree is constructed. After a node ticks, it returns a **status** to its parent, which can be ***Success***, ***Failure***, or ***Running***.

Execution nodes, which are leaves of the BT, can either be **Action** or **Condition** nodes. The only difference is that condition nodes can only return *Success* or *Failure* within a single tick, whereas action nodes can span multiple ticks and can return *Running* until they reach a terminal state. Generally, condition nodes represent simple checks (e.g., “is the gripper open?”) while action nodes represent complex actions (e.g., “open the door”).

Control nodes are internal nodes and define how to traverse the BT given the status of their children. Importantly, children of control nodes can be execution nodes or control nodes themselves. **Sequence**, **Fallback**, and **Parallel** nodes can have any number of children, but differ in how they process said children. **Decorator** nodes necessarily have one child, and modify its behavior with some custom defined policy.

In the context of autonomous vehicles, a BT orchestrates high-level tasks such as:

- Planning a path
- Replanning on failure
- Waiting for a path to clear
- Handling recovery behaviors

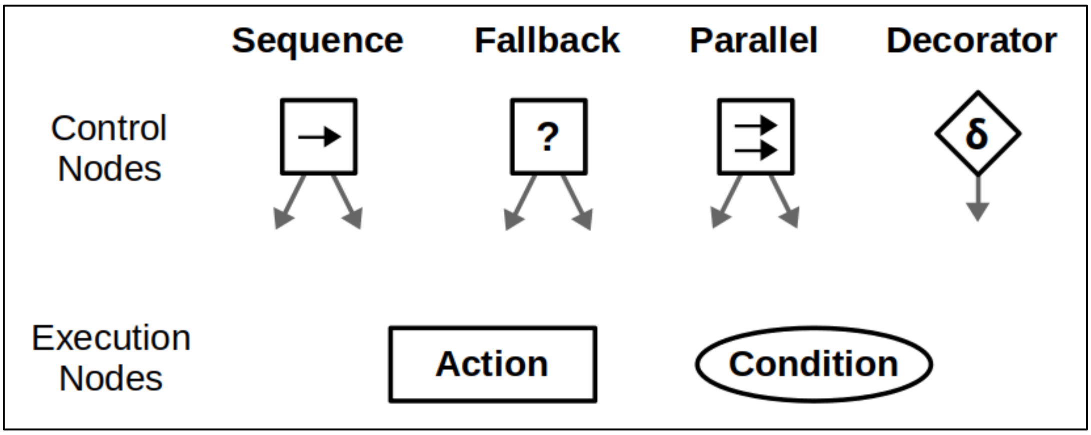


Figure 1.2 Behavior Trees in Navigation

1.3 Obstacle Avoidance Techniques in Mobile Robots

Obstacle avoidance is a crucial function of autonomous vehicles, ensuring safety and operational efficiency. It involves real-time detection and evasion of obstacles encountered during navigation. Techniques for obstacle avoidance can be broadly classified into reactive and deliberative approaches.

Reactive methods rely on immediate sensor data to avoid obstacles without constructing a complete map. These methods are fast and efficient for simple environments but may struggle in complex scenarios. Common reactive techniques include potential fields and vector field histograms.

Deliberative methods, on the other hand, involve building a map of the environment and planning paths accordingly. This allows the robot to foresee obstacles and plan optimal routes. These methods include algorithms like Dijkstra, A*, and their variants.

Modern systems often use a hybrid approach, combining the quick response of reactive methods with the foresight of deliberative planning. This hybrid model enables the robot to handle both known and unknown obstacles more effectively. Real-time updates to local costmaps ensure that new obstacles are accounted for without disrupting the global navigation plan.

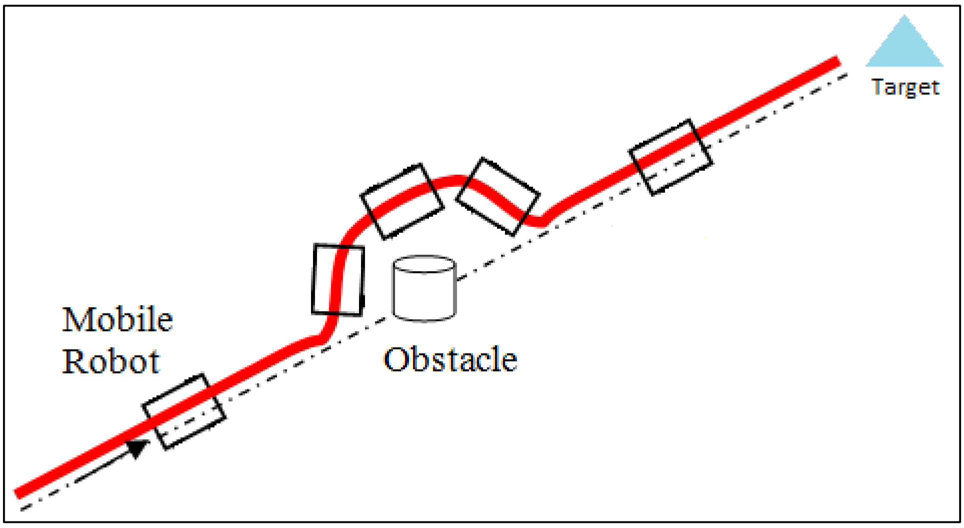


Figure 1.3 Object Avoidance by a Robot

1.4 Navigation Stack and Architecture

The navigation stack forms the backbone of autonomous movement in robotic systems. It encompasses the software architecture responsible for integrating perception, planning, and control.

In this project, we utilize the ROS 2-based Nav2 stack, which provides a modular framework for autonomous navigation. The stack includes several key components: the global planner, local planner, costmap server, and controller server. These modules communicate via ROS topics and services to share information such as sensor data, planned paths, and movement commands.

The global planner generates a path from the robot's current position to the destination using the known map. The local planner adjusts the path in real time based on updated sensor data and dynamic obstacles. Costmap servers create representations of the environment that include obstacle information, allowing planners to assess traversability.

This architecture is designed to be extensible, allowing for the inclusion of custom planners, controllers, and sensor drivers. The use of ROS 2 also brings improved performance, enhanced security, and better real-time capabilities compared to its predecessor, ROS 1.

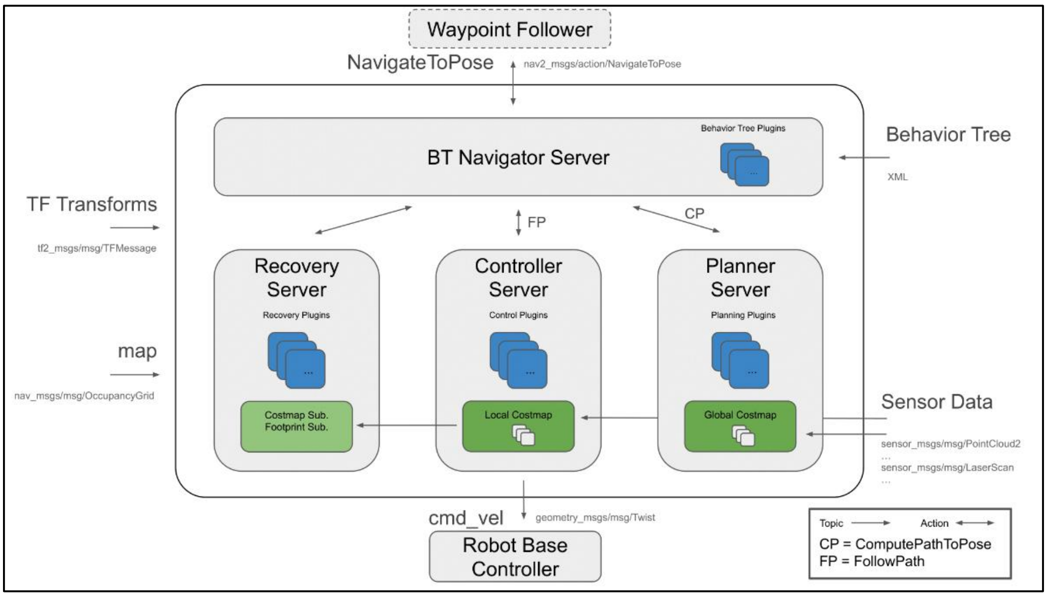


***Figure 1.4** Overview of ROS Navigation Stack 2*

1.5 Path Planning Algorithms (Dijkstra, A*, etc.)

Path planning is at the core of autonomous navigation, enabling the robot to calculate an optimal route from its current location to a specified goal. Different algorithms offer trade-offs between optimality, computational efficiency, and suitability for dynamic environments.

The Dijkstra algorithm is a classic graph-based planning method that finds the shortest path by exploring all possible routes. It guarantees an optimal solution but can be computationally intensive.

A* (A-star) is an extension of Dijkstra's algorithm that introduces heuristics to guide the search, significantly improving efficiency while still guaranteeing an optimal solution. The heuristic function estimates the cost from a node to the goal, allowing the algorithm to prioritize more promising paths.

Feature	**Dijkstra's Algorithm**	**A* Algorithm**
Search Strategy	Uniform cost search (uninformed)	Heuristic-based informed search
Uses Heuristic	No	Yes (e.g., Euclidean or Manhattan distance)
Optimal Path	Always (if all edge weights are non-negative)	Yes (if heuristic is admissible and consistent)
Speed (Efficiency)	Slower due to exhaustive search	Faster due to directed search using heuristic

Feature	Dijkstra’s Algorithm	A* Algorithm
Explored Area	Explores all possible nodes	Explores fewer nodes (focuses on goal direction)
Suitable For	Maps with uniform or unknown costs	Maps where a good heuristic can be estimated
Implementation Complexity	Relatively simple	Slightly more complex due to heuristic integration
Computational Cost	Higher (especially in large maps)	Lower (heuristic helps reduce unnecessary exploration)
Memory Usage	Higher (stores distances to all nodes)	Lower (fewer nodes may be stored depending on heuristic)
Common Usage	Base algorithm in NavFn Planner in ROS2	Used in Smac Planner (hybrid-A*) for more efficient pathfinding

Table 1 *Dijkstra’s vs A* algorithm*

1.6 Role of Sensor Data and Perception (LiDAR, IMU, etc.)

Perception is at the heart of autonomous navigation, acting as the "eyes and ears" of the system. In this project, sensor fusion—especially from LiDAR and IMU—plays a pivotal role in enabling real-time decision-making for obstacle detection and path planning.

LiDAR (Light Detection and Ranging)

LiDAR provides 2D or 3D information about the environment by emitting laser pulses and measuring their return time after hitting surfaces. This results in a point cloud or scan, published on a topic such as /scan, which is consumed by the Nav2 stack to generate costmaps.

- **Key Use:** Used for obstacle detection and building local costmaps.
- **Topic:** /scan (sensor_msgs/LaserScan)

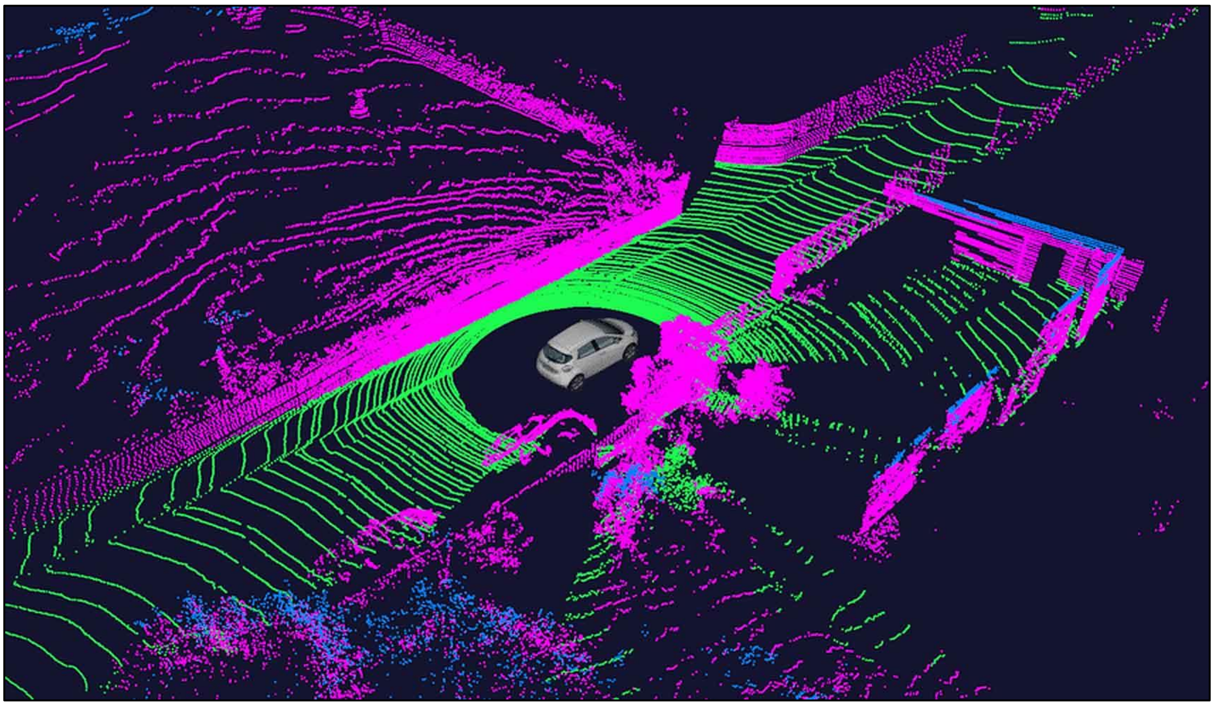


Figure 1.5 3D point cloud using a LIDAR

IMU (MPU6050)

The IMU helps track orientation, angular velocity, and acceleration. It is essential for localization and for correcting drift in odometry data.

- **Key Use:** Provides inertial data used in odometry fusion.
- **Publishing Rate:** Generally, around 50–100 Hz.

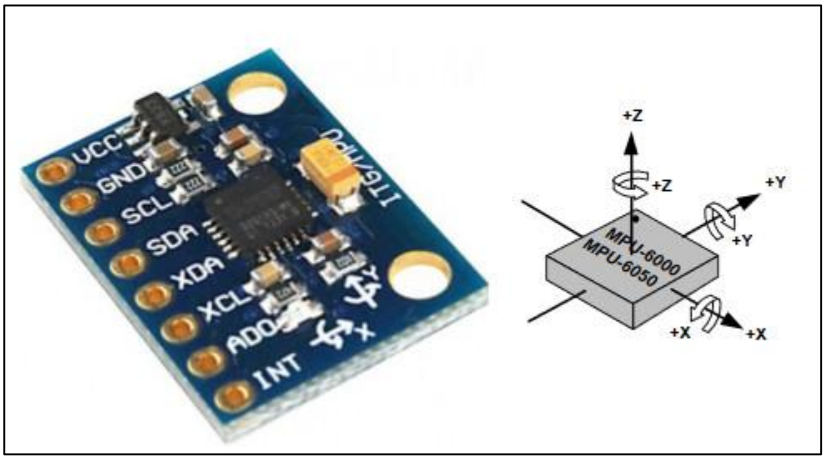


Figure 1.6 MPU 6050(3Axis - 6DOF)

1.7 Motion Control and Decision-Making

Once a path is planned, the robot must convert that plan into physical movement. This process is handled by **controllers** within the Nav2 stack, which calculate velocity commands (cmd_vel) to drive the motors.

DWB (Dynamic Window Approach) Controller

DWB evaluates possible velocity commands based on:

- Obstacle proximity
- Heading to the goal
- Smoothness and safety

How It Works:

- Predicts the trajectory for each velocity pair (v, ω)
- Scores trajectories based on cost functions
- Picks the one with the lowest cost (safe and efficient)

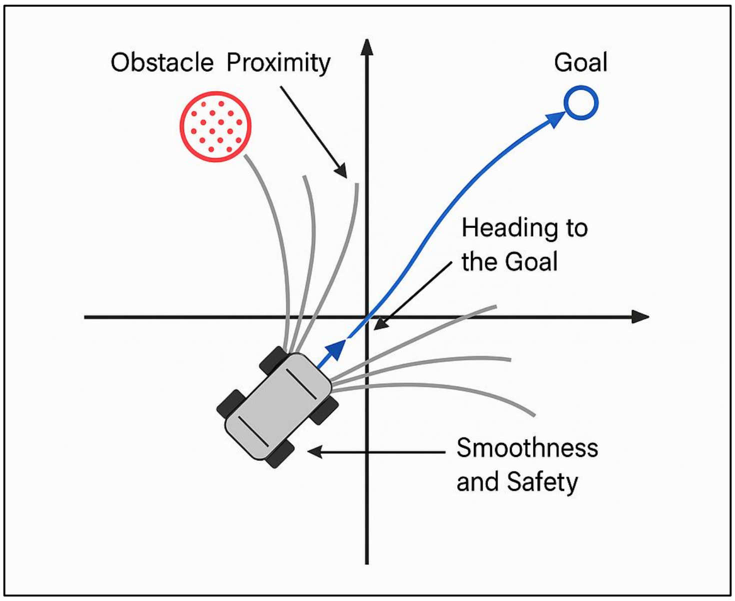


Figure 1.7 Motion Control and Decision-Making DWB Controller

1.7.1 Integration with ROS 2 and Nav2 Framework

ROS 2 (Robot Operating System 2) provides a modular and scalable architecture for robotic applications. In this project, **Nav2 (Navigation 2)** is used as the navigation stack.

Key Components Used:

- **Lifecycle Nodes**: Allow controlled activation/deactivation of planners, controllers.

- **Behavior Tree (BT) Navigator**: Manages tasks like reaching a goal or avoiding collisions.
- **Costmaps**: Represent the environment with obstacles and free space.

System Flow Diagram:

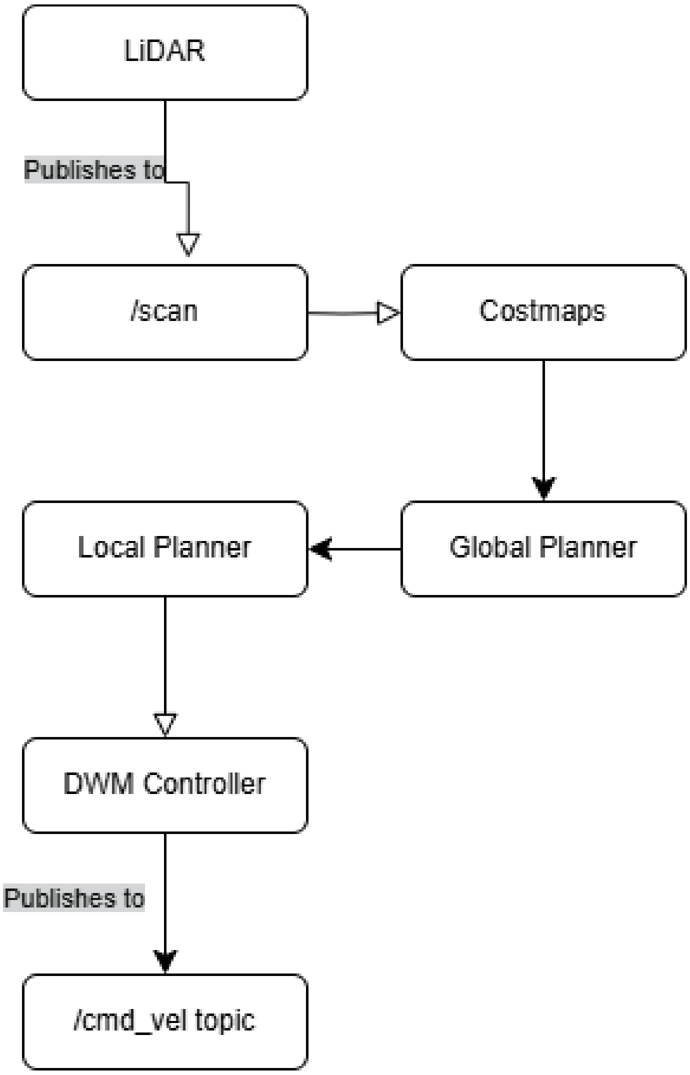


Figure 1.8 System Flow Diagram

1.8 Real-Time Route Recalculation and Adaptation

A key strength of modern autonomous systems is the ability to adapt to changing environments. The local planner recalculates paths periodically based on:

- Obstacle movement
- Updated LiDAR data

- Drift or deviation from the path

How Adaptation Works:

- Costmaps are updated in real-time.
- The local planner replans the trajectory every few milliseconds.
- If the global goal is unreachable, the behavior tree may reissue the goal or retry navigation.

1.9 Testing and Environment Setup

Testing is conducted in both simulated and real-world environments. Initial configurations and algorithm tuning are performed using simulation tools like **Gazebo** and **RViz**.

Steps for Testing:

1. Launch robot base with sensors and URDF.
2. Use slam_toolbox for map generation (if not using a pre-built map).
3. Launch Nav2 with customized planner and controller configs.
4. Issue navigation goals via RViz or /navigate_to_pose.

Launch Command:

```
ros2 launch nav2_bringup bringup_launch.py use_sim_time: =false
```

CHAPTER 2

LITERATURE REVIEW

2.1 LiDAR-Based Obstacle Avoidance for the Autonomous Mobile Robot

Authors: Dony Hutabarat, Djoko Purwanto, Muhammad Rivai, Harjuno Hutomo

This paper introduces an autonomous mobile robot equipped with LiDAR for obstacle avoidance, aimed at addressing limitations of traditional sensors like ultrasonic and infrared. LiDAR's ability to consistently measure distances regardless of object color or lighting conditions makes it an ideal solution for dynamic environments. The robot's control system is based on the Braitenberg vehicle algorithm, implemented on a Raspberry Pi 3.

The experimental setup involves a differential drive mobile robot, equipped with YDLIDAR X4 for 360° scanning, enabling the detection of objects within a range of 0.12 m to 10 m. Experimental results demonstrate reliable obstacle avoidance for objects of varying colors and sizes, regardless of ambient lighting. Furthermore, the robot's navigation was smooth and collision-free in a controlled indoor environment.

This study highlights the advantages of using LiDAR, such as high spatial resolution and immunity to environmental factors like light intensity, while emphasizing the simplicity of integrating LiDAR with cost-effective hardware like Raspberry Pi. The research provides a solid foundation for deploying LiDAR-based systems in real-world scenarios that require precise obstacle detection and navigation.

2.2 The Dynamic Window Approach to Collision Avoidance

Authors: Dieter Fox, Wolfram Burgard, and Sebastian Thrun

This paper introduces the Dynamic Window Approach (DWA), a real-time motion control technique designed to prevent collisions in dynamic environments. Tailored for mobile robots with velocity and acceleration constraints, the DWA narrows down possible motion trajectories to a dynamically feasible set of velocity commands, known as the dynamic window.

The DWA is grounded in the robot's motion dynamics, where only velocity pairs (linear and angular) that can be achieved within a short time window are considered. These admissible velocities are evaluated against an objective function that incorporates goal-directed movement (heading), obstacle clearance (distance), and forward speed (velocity). The best velocity command is selected by maximizing this function over the restricted velocity space.

The paper discusses how this method was successfully implemented on real robots like the RHINO and RWI B21, demonstrating reliable performance in cluttered indoor environments at speeds up to 95 cm/s. Experimental results show smooth, reactive behavior, with the robot able to maneuver around unexpected obstacles in real time.

DWA's ability to handle physical constraints such as acceleration and inertia, along with its low computational overhead, makes it particularly suitable for embedded and low-cost robotic systems. The authors compare this approach favorably against global planners, emphasizing its robustness in situations requiring fast, local decision-making.

2.3 Towards Fully Autonomous Driving: Systems and Algorithms

Authors: Jesse Levinson, Sebastian Thrun, and others

This paper presents Stanford's ongoing research toward achieving safe, fully autonomous driving in urban settings, building upon their previous work in the DARPA Urban Challenge. The system integrates modules for perception, localization, planning, and control, designed to function under varied environmental conditions such as day/night cycles and adverse weather.

The vehicle platform, "Junior," is a modified Volkswagen Passat outfitted with an extensive sensor suite including a 64-beam Velodyne LiDAR, multiple cameras, radar, and a high-precision GPS/IMU system. A notable contribution of the paper is the development of an unsupervised LiDAR calibration system, which improves sensor accuracy significantly without manual intervention.

The authors also detail a probabilistic mapping and localization approach using reflectivity maps, which enables centimeter-level localization. The object recognition module can classify pedestrians, vehicles, and cyclists using shape and motion descriptors, tracked via Kalman filters. Trajectory planning is handled by generating and evaluating thousands of paths per second using cost functions that balance comfort, speed, and safety.

Real-world testing over hundreds of miles confirms the system’s robustness. The paper represents a comprehensive integration of state-of-the-art algorithms and hardware for real-time autonomous driving and sets the foundation for further research on adaptive, intelligent behavior in complex traffic scenarios.

2.4 Autonomous Driving System based on Deep Q Learning

Authors: Takafumi Okuyama, Tad Gonsalves, Jaychand Upadhay

This paper presents a simulation-based study of an autonomous vehicle learning to navigate using Deep Q Networks (DQN), a form of deep reinforcement learning. Unlike conventional approaches that rely heavily on costly sensors such as LiDAR or GPS, this work focuses on learning to drive using only a single front-facing camera. The vehicle learns to steer through visual input by associating states with discrete steering actions through reward-based feedback.

The proposed system processes camera input through a Convolutional Neural Network (CNN) to compute Q-values for different steering actions. These actions are simplified into three discrete options: steer left, steer right, or drive straight. A reward mechanism guides the learning process, where the vehicle is penalized for collisions or lane violations and rewarded for successful obstacle avoidance and lane following.

The simulation environment, created in Unity and interfaced with Python through the Chainer deep learning framework, contains randomly placed static obstacles like human figures, vehicles, and roadblocks. Results show that after training, the model enables the car to avoid obstacles effectively and drive smoothly at different speeds. Notably, the performance drops at higher speeds, highlighting the trade-off between learning time and control fidelity.

This study emphasizes the viability of vision-only learning for basic driving tasks and demonstrates the potential of DQN as a lightweight, scalable approach for autonomous navigation systems that do not require heavy sensor infrastructure.

2.5 Deep Multi-Modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges

Authors: Di Feng, Christian Haase-Schütz, Lars Rosenbaum, Heinz Hertlein, Claudius Gläser, Fabian Timm, Werner Wiesbeck, Klaus Dietmayer

This review paper provides an in-depth survey of recent advancements in deep multi-modal perception for autonomous driving, focusing on object detection and semantic segmentation. The authors explore how combining multiple sensor modalities—such as cameras, LiDARs, and Radars—improves the robustness and accuracy of scene understanding in autonomous systems. They analyze challenges in multi-modal fusion design, highlighting the critical questions of **what to fuse, when to fuse, and how to fuse**, which remain open research problems.

The paper covers a broad range of sensing techniques and data processing pipelines, from voxel-based and PointNet-based LiDAR representations to various camera image fusion strategies. The authors also present a comprehensive summary of modern datasets (e.g., KITTI, nuScenes, ApolloScape) and outline their suitability for training deep learning models. Key topics include voxel encoding, 2D/3D detection, feature fusion methods (early, middle, late), and the lack of large-scale, radar-inclusive benchmarks.

This work is particularly valuable for its taxonomy of fusion methodologies and its emphasis on real-time, robust AV perception. It serves as a foundation for developing next-generation sensor fusion frameworks and calls for more exploration into radar-based deep learning, robust temporal alignment, and generalization to open-world conditions

2.6 nuScenes: A Multimodal Dataset for Autonomous Driving

Authors: Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, Oscar Beijbom

The nuScenes dataset is a pioneering large-scale benchmark for autonomous vehicle perception, specifically designed to support **3D object detection and tracking** using multimodal sensors. Unlike earlier datasets such as KITTI, nuScenes offers full 360° coverage with **6 cameras, 5 radars, 1 LiDAR**, and GPS/IMU data across over **1,000 scenes**, with annotations for **23 object classes** and **8 attributes**. The dataset captures varied environmental conditions, including **rain, nighttime, and urban complexity** across locations in Boston and Singapore.

nuScenes is notable for introducing 3D bounding box annotations, semantic maps, and high-frequency capture rates (LiDAR at 20Hz, Radar at 13Hz, and Cameras at 12Hz), which are crucial for temporal modeling. It also pioneers novel evaluation metrics, such as Average Translation Error (ATE) and Average Orientation Error (AOE), for better AV benchmarking beyond simple IoU scores.

This dataset has spurred significant research in multimodal fusion, long-tail object detection, weather adaptation, and trajectory prediction. The authors highlight the importance of radar data in complementing LiDAR and camera inputs, especially in adverse conditions. nuScenes continues to shape AV research by encouraging community-wide dataset standardization and releasing detailed devkits and evaluation tools.

CHAPTER 3
AUTONOMOUS VEHICLES

3.1 Introduction

Autonomous vehicles (AVs), also known as self-driving cars or driverless vehicles, represent a transformative shift in transportation technology. By eliminating the need for human input in driving tasks, AVs promise to enhance road safety, reduce traffic congestion, and provide mobility solutions for people who are unable to drive. At the heart of autonomous vehicles lies a sophisticated interplay of perception, planning, and control systems—all of which work together to navigate complex real-world environments.

Unlike traditional vehicles, AVs rely on an array of sensors—such as LiDAR (Light Detection and Ranging), radar, ultrasonic sensors, high-definition cameras, and Inertial Measurement Units (IMUs)—alongside powerful onboard computing units to interpret their environment. These sensory inputs are used to detect lanes, road signs, traffic signals, pedestrians, vehicles, and obstacles, enabling the vehicle to make decisions and adapt to changing conditions in real-time.

Levels of Autonomy

The Society of Automotive Engineers (SAE) defines six levels of driving automation, ranging from Level 0 (no automation) to Level 5 (full automation). Most commercially available AVs today operate at Level 2 or Level 3, where the vehicle can manage steering, acceleration, and braking under specific conditions, but human intervention is still required. Level 5 vehicles, capable of fully autonomous operation in any environment, remain under active development.

Core Functionalities

At the foundation of AV systems are several tightly integrated modules:

- **Perception:** Interprets sensor data to understand the environment.
- **Localization:** Determines the vehicle’s precise position within a map.
- **Mapping:** Generates and updates high-resolution maps for navigation.
- **Planning:** Charts both high-level routes and detailed short-term trajectories.
- **Control:** Executes the planned path via actuation of throttle, brakes, and steering.

Each module operates under strict real-time constraints, ensuring decisions are made within milliseconds to avoid hazards and comply with traffic rules.

Technological Ecosystem

The development of autonomous vehicles draws from various disciplines, including computer vision, robotics, control systems, artificial intelligence (especially machine learning and deep learning), and embedded systems. Open-source frameworks like the Robot Operating System (ROS and ROS2), NVIDIA's Drive platform, and the Autoware AV stack have accelerated research and prototyping.

Several companies—such as Waymo, Tesla, Cruise, and Baidu—are at the forefront of deploying AVs on public roads. Their test vehicles are logging millions of miles in diverse environments to train and validate their systems. Some deployments are limited to geo-fenced urban zones (Level 4 autonomy), while others aim for broad generalization.

Benefits and Societal Impact

The potential benefits of AVs are profound:

- **Improved safety:** By eliminating human error—which accounts for over 90% of traffic accidents—AVs can drastically reduce fatalities and injuries.
- **Accessibility:** AVs can empower individuals who are elderly, disabled, or visually impaired by providing independent mobility.
- **Efficiency:** Autonomous fleets can reduce fuel consumption, optimize traffic flow, and minimize idle time.
- **Productivity:** Passengers can engage in other activities while commuting, turning travel time into productive time.

Ethical and Regulatory Considerations

Despite the promise, AV deployment raises important ethical and legal questions. For instance, how should an AV respond in an unavoidable crash scenario? Who is liable in the event of an accident—the manufacturer, software developer, or vehicle owner? Governments and regulatory bodies worldwide are still drafting guidelines to address these issues while balancing innovation and public safety.

3.2 Components of Autonomous Vehicles

An autonomous vehicle integrates multiple hardware and software components, each fulfilling a distinct function to ensure safe and reliable operation. These components can be broadly categorized as follows:

- **Perception System**
  This includes LiDARs, cameras, ultrasonic sensors, and radars. These sensors collect data about the vehicle's surroundings, enabling object detection, road surface recognition, and situational awareness.

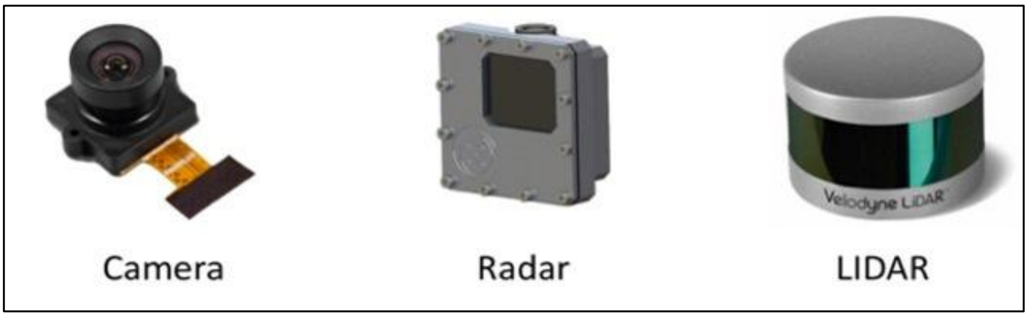


Figure 3.1 *Perception Sensors*

- **Localization System**
  Localization determines the precise position of the vehicle on a map using data from GPS, IMUs, and LiDAR. Highly accurate localization (often to the centimeter level) is critical for lane-keeping and route planning.

- **Mapping and Environment Representation**
  High-definition (HD) maps provide detailed representations of the road, including lane markings, curbs, traffic signs, and more. AVs update their internal maps using SLAM (Simultaneous Localization and Mapping) techniques.

- **Path Planning and Decision-Making**
  Path planning modules use algorithms (like A* or Dijkstra's) to find safe routes, taking into account dynamic obstacles. Decision-making modules handle scenarios such as overtaking, stopping at intersections, and merging.

- **Motion Control System**
  This includes low-level controllers (e.g., PID or MPC) that translate high-level commands into actuator signals—like steering, throttle, and braking.

- **Communication and Networking**
  V2V (Vehicle-to-Vehicle) and V2X (Vehicle-to-Everything) technologies enable AVs to communicate with other vehicles and infrastructure to enhance safety and traffic coordination.

- **Computing Hardware**
  Processing is performed on embedded systems like NVIDIA Jetson or Intel-based automotive servers. These must support real-time inference from neural networks and sensor fusion algorithms.

3.3 Navigation, Control, and Sensor Fusion in Autonomous Vehicle Technology

Autonomous navigation requires the seamless integration of sensor data with control algorithms to generate reliable and safe driving behavior. This section elaborates on how navigation, control, and sensor fusion work in tandem.

Navigation and Path Planning

Navigation involves determining a route from the vehicle's current position to a predefined goal. This includes:

- **Global planning** (e.g., A*, Dijkstra): Plans a path using a static map.
- **Local planning** (e.g., DWB, TEB): Adjusts the path in real time based on dynamic obstacles and sensor data.

The planning process continuously updates using local and global costmaps. These costmaps are built from the perception layer, integrating obstacle data and free-space detection.

Motion Control

Once a path is generated, control algorithms translate this path into physical motion. This includes:

- **Lateral control** (steering) to follow the lane or trajectory.
- **Longitudinal control** (throttle and braking) to maintain speed and avoid collisions.

Model Predictive Control (MPC) is widely used in AVs for handling constraints and predicting system behavior. Simpler systems may use PID control for basic steering and speed adjustments.

Sensor Fusion

Sensor fusion combines data from multiple sensors to create a more accurate and robust understanding of the environment. It addresses issues like:

- Inconsistencies in sensor readings.
- Occlusions or blind spots in camera data.
- Low lighting conditions affecting vision sensors.

A common approach is the **Kalman Filter** or **Extended Kalman Filter (EKF)**, which estimates the vehicle's position and velocity by fusing data from GPS, IMU, and LiDAR. Deep learning-based fusion methods are also emerging, enabling more adaptive and data-driven integration of multimodal sensor inputs.

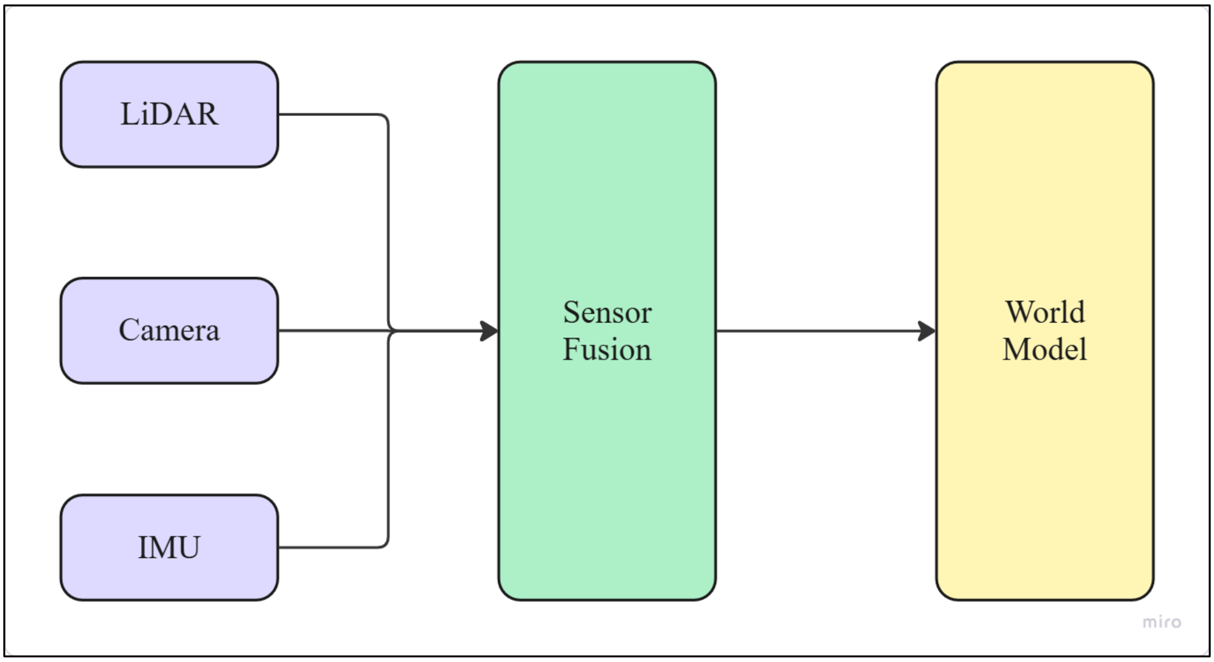


Figure 3.2 *Sensor Fusion Overview*

3.4 Challenges in Designing and Implementing Autonomous Vehicle Systems

Designing and deploying autonomous vehicles (AVs) involves overcoming numerous technical, logistical, and ethical challenges. Although the idea of self-driving cars has existed for decades, turning this vision into reality at scale remains one of the most complex engineering feats in modern times. These challenges span several domains, from hardware limitations and algorithmic complexity to real-world unpredictability and regulatory constraints.

3.4.1 Perception in Unstructured and Dynamic Environments

Autonomous vehicles must accurately perceive their surroundings under all lighting, weather, and environmental conditions. Common perception challenges include:

- Poor visibility due to rain, fog, or snow affecting camera and LiDAR performance.
- Glare or shadows that interfere with image-based detection.
- Dynamic obstacles such as pedestrians, cyclists, and erratic drivers.
- Unexpected scenarios like construction zones, fallen debris, or animals on the road.

Perception systems must therefore be robust, fault-tolerant, and capable of reasoning under uncertainty.

3.4.2 Real-Time Processing and System Latency

AVs rely on a massive amount of sensor data, often in the range of gigabytes per second. This data must be processed in real time for accurate decision-making. Delays in perception, planning, or control can lead to unsafe driving behavior. Ensuring low-latency, high-throughput processing requires efficient software architecture and powerful embedded computing systems.

- **Localization and GPS Drift**
  While localization techniques using LiDAR and GPS-IMU fusion can achieve centimeter-level accuracy, signal loss in urban canyons, tunnels, or under dense foliage can degrade performance. Drift in GPS and IMU data can lead to incorrect vehicle positioning, jeopardizing navigation and safety.

- **Sensor Fusion and Calibration**
  Multi-sensor fusion is essential for comprehensive environmental awareness, but synchronizing data streams from different modalities (e.g., radar, LiDAR, IMU) is non-trivial. Calibration errors, sensor misalignment, or timing discrepancies can introduce inconsistencies in the world model.

- **Planning and Decision-Making Under Uncertainty**
  Planning safe and efficient trajectories in unpredictable environments—such as urban intersections or highways—is complex. The system must continuously predict the future behavior of surrounding agents (cars, pedestrians) and update its strategy in real time. Scenarios involving multiple agents, ambiguous road markings, or uncommon traffic situations can confuse even advanced planners.

- **Safety, Redundancy, and Failover Mechanisms**
  AVs must include redundancy across critical subsystems (braking, steering, perception) to ensure safety in the event of hardware or software failure. Developing fault detection, diagnostic, and failover strategies is critical for reliability.

- **Ethical Dilemmas and Regulatory Hurdles**
  AVs may encounter moral dilemmas—such as choosing between two harmful outcomes in a collision scenario. Programming ethical decision-making into machines is controversial and remains unsolved. Moreover, different countries have varying regulations regarding testing, liability, and data usage, complicating global deployment.

- **Public Trust and Human-AI Interaction**
  Gaining public acceptance is as important as solving technical issues. AVs must behave predictably and communicate their intent (e.g., stopping for a pedestrian) in ways that humans can intuitively understand.

Addressing these challenges requires not only advances in technology but also interdisciplinary collaboration between engineers, policymakers, legal experts, and ethicists.

3.5 Localization and Mapping Techniques for Autonomous Vehicles

Accurate localization is a cornerstone of autonomous driving. An autonomous vehicle must know its precise position within the environment to follow lanes, obey traffic laws, avoid obstacles, and reach destinations reliably. Localization is often tightly coupled with mapping—the vehicle must localize itself relative to a known map or generate one on the fly in unstructured environments.

3.5.1 Global vs. Local Localization

- **Global localization** involves determining the vehicle's position on a pre-existing map, often referred to as the "map matching" problem.
- **Local localization** focuses on tracking small changes in position relative to a known starting point or previously mapped region, using sensors like wheel encoders or IMUs.

Most AV systems combine both approaches to ensure accurate pose estimation in varying environments.

3.5.2 Techniques for Localization

Several methods are used in AV systems to achieve high-precision localization:

- **GNSS (Global Navigation Satellite Systems)**
  GNSS-based positioning (e.g., GPS) offers global coverage but suffers from low accuracy (~3–10 meters), signal blockage in tunnels or dense urban areas, and multipath errors. High-precision variants like RTK-GPS (Real-Time Kinematic) can improve accuracy to within centimeters but are expensive and require base stations.

- **Inertial Odometry**
  IMUs track the vehicle's motion using accelerometers and gyroscopes. While useful for short-term motion estimation, IMUs tend to drift over time and must be corrected using other data sources.

- **LiDAR-Based Localization**
  LiDAR systems capture high-resolution 3D scans of the environment. These are matched with a pre-built HD map using techniques like ICP (Iterative Closest Point) or NDT (Normal Distributions Transform) to compute precise pose. LiDAR-based localization is robust but computationally intensive.

- **Visual Odometry (VO)**
  VO uses monocular or stereo cameras to estimate motion by tracking keypoints across frames. While cost-effective and rich in information, it is sensitive to lighting changes, motion blur, and textureless environments.

- **Sensor Fusion Techniques**
  To overcome the limitations of individual sensors, AVs use **sensor fusion**, typically employing an Extended Kalman Filter (EKF) or a Particle Filter to combine GPS, IMU, LiDAR, and vision data for accurate pose estimation.

3.5.3 Mapping in Autonomous Vehicles

Autonomous vehicles rely on **High Definition (HD) maps**, which contain detailed semantic and geometric information such as:

- Lane boundaries and traffic signs
- Curbs, sidewalks, and stop lines
- 3D landmarks and surface reflectivity

Mapping can be done using **SLAM (Simultaneous Localization and Mapping)**, particularly in environments where pre-mapping is not feasible. LiDAR SLAM and Visual SLAM are popular approaches that build maps while estimating the vehicle's location.

3.5.4 Challenges in Localization

- **Drift correction:** Maintaining accuracy over long distances without GPS.
- **Dynamic environments:** Handling changes due to construction or shifting objects.
- **Computational load:** Real-time processing of complex sensor data and map alignment.
- **Map maintenance:** HD maps need to be updated frequently to reflect real-world changes.

3.6 Machine Learning in Autonomous Vehicles

Machine learning (ML) and deep learning (DL) are central to the success of modern autonomous vehicles. These techniques enable systems to recognize patterns, learn behaviors, make predictions, and adapt to complex real-world driving scenarios that are difficult to handle through rule-based programming alone.

3.6.1 Key Applications of Machine Learning in AVs

a. Perception

Machine learning models—especially Convolutional Neural Networks (CNNs)—are widely used in:

- **Object Detection and Classification** (e.g., cars, pedestrians, signs, traffic lights)
- **Lane Detection**
- **Semantic Segmentation** of road surfaces and drivable regions

Popular models include YOLO, Faster R-CNN, and DeepLab for real-time image-based understanding.

b. Behavior Prediction

Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) models are used to predict the motion of surrounding agents, such as the path a pedestrian might take at a crosswalk.

c. Sensor Fusion

Deep learning-based sensor fusion architectures combine data from LiDAR, cameras, and radar to generate robust environmental models, outperforming traditional filter-based methods in complex scenarios.

d. Reinforcement Learning

Reinforcement Learning (RL) allows an AV to learn optimal driving strategies by interacting with a simulated or real-world environment. Algorithms like Deep Q-Networks (DQN) and Proximal Policy Optimization (PPO) have been used to train vehicles in obstacle avoidance, merging, and lane keeping.

e. End-to-End Learning

Some research efforts, notably by NVIDIA, focus on end-to-end driving models where a deep network directly maps input images to steering, throttle, and brake commands. While promising, these models often lack interpretability and generalizability.

3.6.2 Training and Deployment Challenges

- **Data Requirements:** Training ML models requires enormous datasets with diverse driving conditions.

- **Edge Deployment:** Running large models in real-time on in-vehicle hardware requires quantization, pruning, and hardware acceleration (e.g., TensorRT, CUDA).
- **Generalization:** Models trained in specific environments may fail in unseen scenarios, necessitating domain adaptation and continual learning techniques.
- **Safety and Explainability:** Regulators and developers require interpretable models to understand failure cases and validate decisions.

3.7 Applications of Autonomous Vehicles

The impact of autonomous vehicles extends across various sectors, promising to revolutionize not only personal mobility but also logistics, public safety, and infrastructure planning.

- **Personal Transport**
  Self-driving cars are poised to redefine the commuting experience. Companies like Waymo, Tesla, and Cruise are testing autonomous taxi services that offer door-to-door mobility with minimal human input.

- **Autonomous Delivery and Logistics**
  Autonomous vans and small ground robots are being deployed for last-mile delivery of goods. Startups like Nuro and large companies like Amazon are developing fleets to transport parcels and groceries without human drivers.

- **Public Transportation**
  Autonomous shuttles are being piloted in smart city projects and university campuses. These systems operate on pre-defined routes at low speeds, offering safe and energy-efficient transit.

- **Industrial and Agricultural Applications**
  Self-driving vehicles are used in:
  - **Mining:** Autonomous haul trucks in mines reduce accidents and improve efficiency.
  - **Agriculture:** Autonomous tractors, harvesters, and drones enable precision farming by navigating fields and applying inputs optimally.

- **Emergency and Military Use**
  Autonomous ground vehicles can be deployed in hazardous zones for surveillance, search and rescue, or payload delivery in disaster or combat zones.

- **Smart Infrastructure and Urban Planning**
  AVs can communicate with traffic lights, smart signs, and road sensors, helping cities optimize traffic flow and reduce emissions.

3.8 6DOF Simulation and Control

To safely test autonomous driving algorithms, simulations are indispensable. A **6 Degrees of Freedom (6DOF)** simulation accurately models all possible motion in 3D space—translation along X, Y, Z axes and rotation around roll, pitch, and yaw.

1. What is 6DOF?

The 6DOF model accounts for:

- **Linear Motion:** Forward/backward (X), side-to-side (Y), up/down (Z)
- **Rotational Motion:** Roll (X-axis), Pitch (Y-axis), Yaw (Z-axis)

This is essential for high-fidelity simulations where vehicle dynamics, terrain, and sensor behavior must be accurately represented.

2. Simulation Platforms

Common simulation environments for AVs include:

- **CARLA:** Open-source simulator for AV research supporting camera, LiDAR, and radar simulation.
- **Gazebo + ROS2:** Offers physics-based simulation integrated with real-time ROS2 control.
- **LGSVL Simulator:** Developed by LG for high-fidelity sensor simulation in realistic urban environments.

3. Dynamics and Control Modeling

In simulation, vehicle dynamics are modelled using differential equations that describe acceleration, braking, and steering behavior. Controllers like PID and MPC are applied to follow planned trajectories while obeying constraints.

4. Testing Scenarios

Simulations can model:

- Lane changing
- Roundabout navigation
- Overtaking in highways
- Emergency braking
- Sensor failure and adverse weather

3.9 Future Directions and Advancements in Autonomous Vehicles

Autonomous vehicle (AV) technology is progressing rapidly, but it is still far from reaching its full potential. Current AV systems are limited in scope, constrained by pre-mapped environments, weather, and regulatory boundaries. However, ongoing advancements in artificial intelligence, hardware design, connectivity, and human-machine interaction are opening new horizons for AV capabilities.

1. Generalization and Robust AI Models

Future AVs must function reliably in unpredictable environments without being explicitly programmed for every scenario. This calls for:

- Self-supervised and unsupervised learning to reduce dependence on labeled data.
- Meta-learning and continual learning to enable models to adapt on the fly.
- Multimodal perception models that integrate audio, radar, camera, and LiDAR in a holistic manner.

2. Edge AI and Onboard Intelligence

The next generation of autonomous systems will feature AI at the edge, enabling faster inference with less dependency on cloud connectivity. Companies are developing dedicated AV chips, such as NVIDIA's Drive Orin and Tesla’s Dojo, to accelerate onboard neural processing with lower energy consumption.

3. Vehicle-to-Everything (V2X) Communication

As smart cities emerge, AVs will increasingly rely on V2X communication, allowing them to exchange information with:

- Traffic infrastructure (e.g., signals, signs)
- Other vehicles (V2V)
- Pedestrians and mobile devices (V2P)

This enhanced communication will improve reaction time, reduce latency, and increase overall safety by sharing data beyond line-of-sight.

4. High-Fidelity, Real-Time Digital Twins

Integration of digital twins—virtual representations of physical environments—will allow AVs to simulate and plan ahead using real-time data. Combined with edge computing and 5G, this could enable proactive responses to traffic and environmental changes.

5. Ethical AI and Transparent Decision-Making

One of the major hurdles to widespread adoption is ethical decision-making. Future AVs will need to:

- Justify their actions in case of an incident.
- Ensure fair and unbiased perception and planning.
- Make decisions aligned with societal expectations (e.g., the "trolley problem").

Researchers are exploring explainable AI (XAI) and ethical policy layers for integration into AV decision frameworks.

6. Full Autonomy and Level 5 Vehicles

The ultimate goal is to build Level 5 AVs capable of operating without any human intervention in all environments, conditions, and terrains. To reach this:

- Sensor fusion systems must evolve to handle extreme edge cases.
- Planning algorithms must become more predictive and probabilistic.
- Regulatory approval and global standardization must be achieved to enable deployment.

Currently, companies like Waymo, Zoox, Cruise, and Aurora are testing prototypes, and pilot deployments are expected to gradually expand from geo-fenced urban areas to wider geographic regions.

CHAPTER 4

HARDWARE IMPLEMENTATION

4.1 Hardware Model

The mobile robot developed for this project is a four-wheeled differential drive platform designed with the intent to serve as an AMR (Autonomous Mobile Robot) prototype. The hardware components have been selected and integrated to support autonomous navigation and sensor fusion capabilities, forming the foundation for ROS 2-based software development and real-world testing.

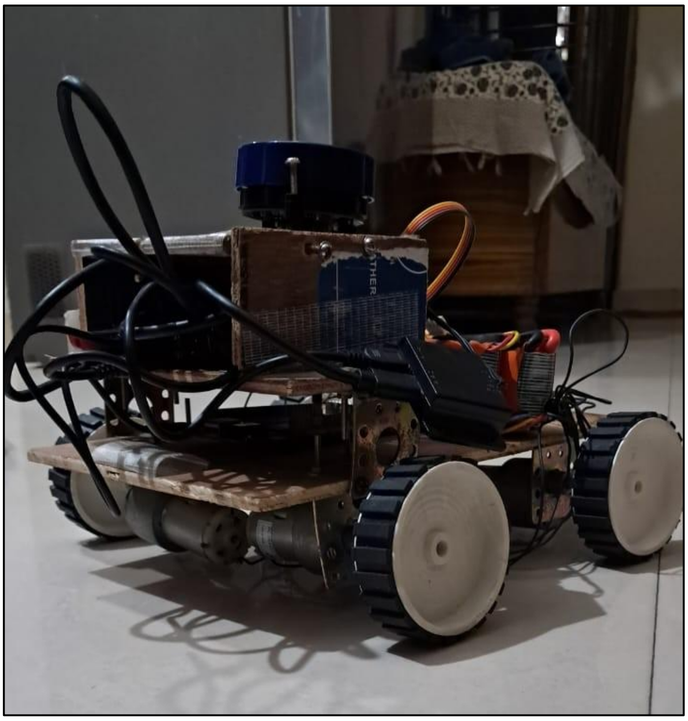


Figure 4.1 *Hardware Model*

The robot chassis is a custom-built, two-tier structure made of lightweight material to accommodate components efficiently. It is equipped with four DC gear motors, two on each side, allowing for differential drive motion. This configuration enables precise manoeuvring in constrained environments and supports basic movement patterns essential for autonomous navigation.

4.2 Sensor Suite

Effective obstacle detection is the cornerstone of autonomous vehicle safety. To ensure reliable navigation in dynamic and uncertain environments, autonomous vehicles (AVs) employ a range of sensors to perceive their surroundings. These sensors offer complementary data, enabling the vehicle to detect obstacles in various weather, lighting, and road conditions.

4.2.1. LiDAR (Light Detection and Ranging)

LiDAR is one of the most popular sensors for 3D environmental mapping and obstacle detection. It uses pulsed lasers to measure distances and generate detailed point clouds of the environment.

- **Pros:** High accuracy, excellent depth perception, works in low light.
- **Cons:** Expensive, performance can degrade in fog or rain.

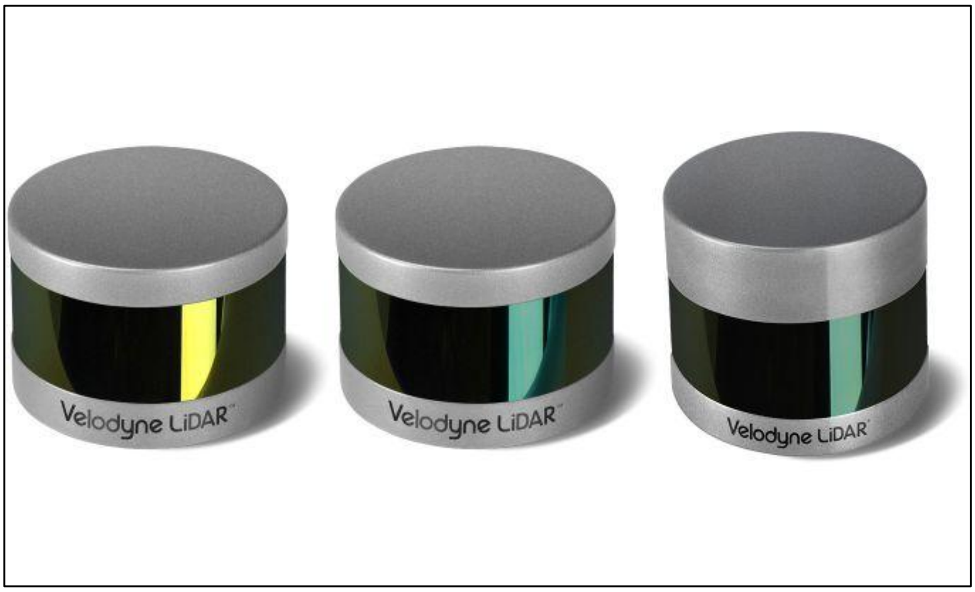


Figure 4.2 *LiDAR Sensor*

4.2.2. RADAR (Radio Detection and Ranging)

Radar uses radio waves to detect objects and is especially effective at measuring speed and detecting obstacles in adverse weather conditions.

- **Pros:** Works in fog, rain, and snow; good for long-range detection.
- **Cons:** Lower spatial resolution than LiDAR or cameras.

Figure 4.3 Corner Radar Sensor

4.2.3. Cameras (Mono & Stereo)

Cameras provide rich visual information, enabling recognition of objects, signs, and lane markings. Stereo cameras can estimate depth.

- **Pros:** Inexpensive, lightweight, ideal for classification tasks.
- **Cons:** Affected by lighting and weather; depth perception is limited.

Sensor	Range	Accuracy	Best For	Limitations
LiDAR	10–200 m	High	Mapping, 3D Obstacle Detection	Cost, Weather Sensitivity
RADAR	20–250 m	Medium	Speed, Distance	Resolution
Camera	0–80 m	High (2D)	Lane, Traffic Signs	Poor Depth, Light Sensitivity

Table 2 Difference between Perception Sensors

4.2.4. Ultrasonic Sensors

These sensors are used for short-range detection, such as during parking or slow-speed maneuvers.

- **Range:** Typically, under 5 meters.
- **Application:** Parking assistance, close obstacle detection.

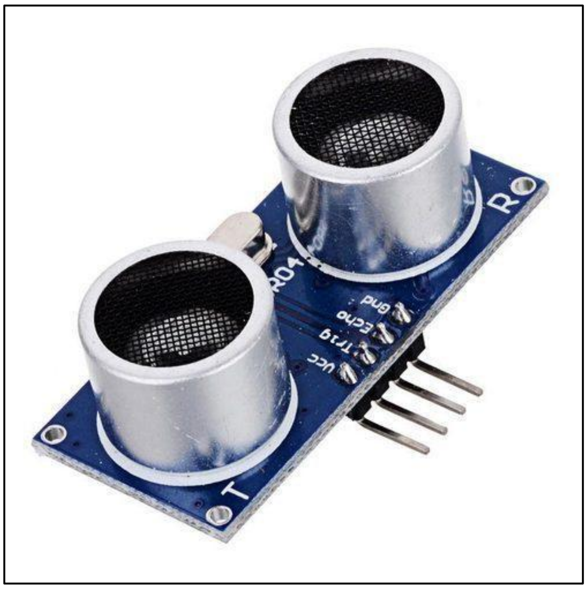


Figure 4.4 Ultrasonic Sensor

4.2.5. IMU (Inertial Measurement Unit)

An IMU measures acceleration and angular velocity and is primarily used for motion tracking and short-term dead reckoning. While it doesn't detect obstacles directly, it supports obstacle-aware motion planning.

4.3 Techniques for Obstacle Avoidance

Obstacle avoidance refers to the vehicle's ability to detect and respond to objects in its path by altering its trajectory safely. Various techniques have been developed, ranging from classical rule-based algorithms to advanced learning-based approaches.

4.3.1 Reactive Obstacle Avoidance

This method focuses on immediate sensor feedback to make short-term decisions. Reactive approaches typically don’t rely on global maps.

- **Example:** The Dynamic Window Approach (DWA), which evaluates safe velocities based on real-time obstacles and selects the best one.
- **Use Case:** Fast maneuvering in dynamic environments (e.g., crowded parking lots).

4.3.2 Deliberative Obstacle Avoidance (Planner-Based)

These methods involve constructing a map and making long-term decisions. The AV uses a global planner to find the best route and a local planner to adjust the trajectory for nearby obstacles.

- **Example:** A* or Dijkstra for global planning; DWB or TEB for local avoidance.

4.3.3 Hybrid Methods

Combining reactive and deliberative approaches enables AVs to plan efficiently while adapting to real-time changes. For instance:

- Global planner provides a rough path.
- Local planner refines it using updated sensor data.

4.3.4 Predictive Avoidance using Machine Learning

Advanced systems use prediction models to estimate the future motion of pedestrians, vehicles, and cyclists. These predictions help the planner decide whether to stop, yield, or change lanes.

4.4 Processing Units

4.4.1 Arduino Mega 2560

This microcontroller acts as an interface between the lower-level hardware (motors and IMU) and the higher-level processing unit. It is responsible for collecting encoder data, reading IMU values, and transmitting this data via serial communication to the Raspberry Pi.

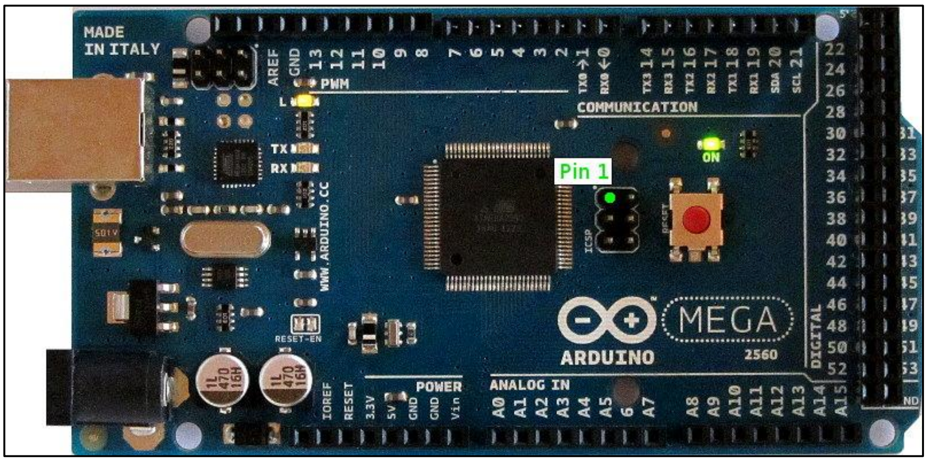


Figure 4.5 Arduino Mega 2560

4.4.2 Raspberry Pi 4

The Raspberry Pi serves as the main computational unit running the ROS 2 stack. It handles sensor data fusion, LiDAR processing, and higher-level decision making for autonomous navigation.

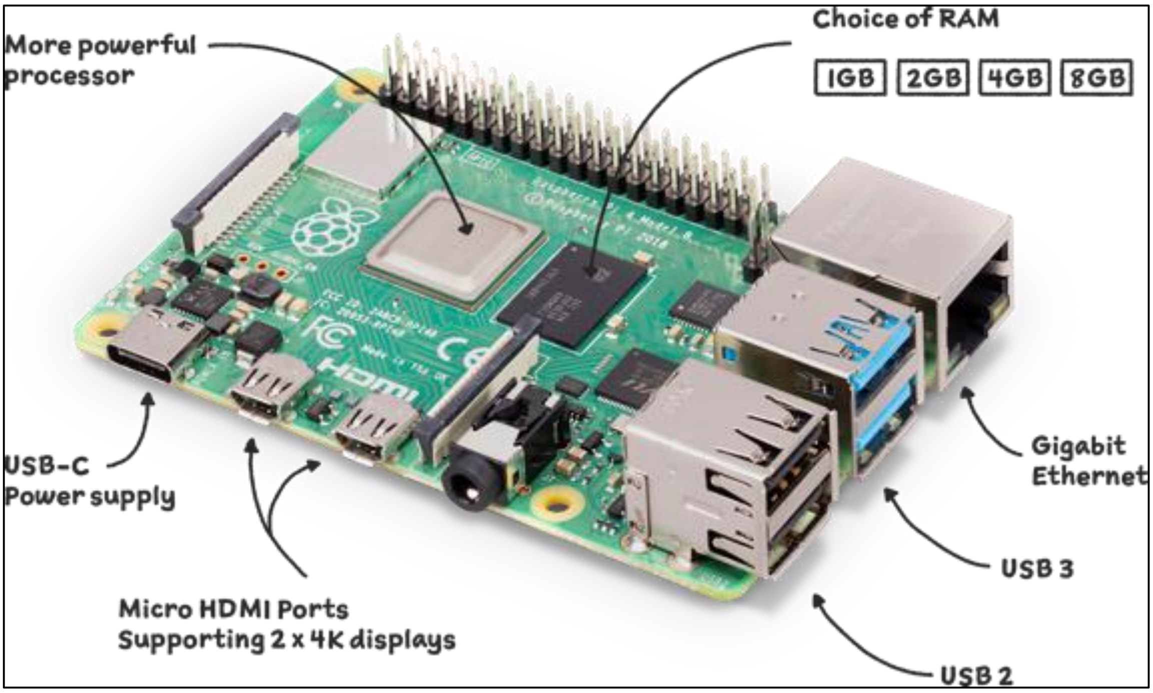


Figure 4.6 Raspberry Pi 4

4.5 Dynamic Window Approach (DWA)

The **Dynamic Window Approach (DWA)** is a real-time collision avoidance and local planning algorithm used widely in mobile robotics and autonomous vehicles. It is particularly suited for differential drive robots and vehicles operating under velocity and acceleration constraints.

4.5.1 Overview

DWA evaluates a set of admissible velocity pairs (v, ω) and simulates their resulting trajectories over a short time window. It scores each trajectory using a cost function and selects the one that balances safety and goal alignment.

4.5.2 How DWA Works

- **Velocity Sampling**: DWA samples possible linear and angular velocities within the vehicle's dynamic limits.
- **Trajectory Simulation**: For each velocity pair, a trajectory is predicted over a short duration (e.g., 1–2 seconds).
- **Cost Evaluation**: Each trajectory is scored based on:
  - **Obstacle proximity (obstacle cost)**: Penalizes paths close to obstacles.
  - **Goal heading (heading cost)**: Encourages alignment with the goal.
  - **Forward velocity (velocity cost)**: Favors faster progress.

The command with the lowest combined cost is selected and sent to the vehicle as a motion command.

4.5.3 Integration of DWA with ROS2 and Costmap Navigation

In ROS2 and Nav2, DWA is tightly integrated with the **local costmap** and operates as a real-time local planner. This integration ensures dynamic obstacle avoidance based on live sensor data.

4.5.4 Local Costmap in Nav2

The **local costmap** is a rolling window around the robot that represents nearby obstacles based on sensor data. It uses:

- LiDAR scans (from /scan)
- Inflation layers (to expand obstacles)
- Obstacle layers (to mark lethal zones)
- Voxel or 2D grid representations

4.5.5 ROS2 Navigation Pipeline with DWA

As the costmap is updated, DWA recalculates viable velocities and adjusts motion in real-time.

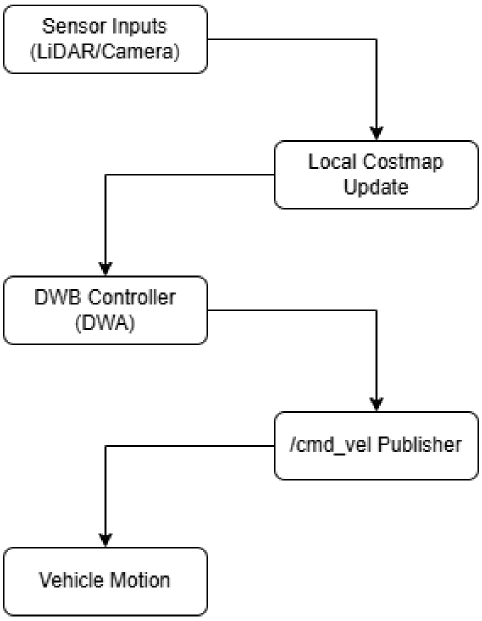


Figure 4.7 *ROS2 Navigation Pipeline with DWA*

4.6 Advancements in Obstacle Avoidance for Autonomous Vehicles

Recent years have seen significant advancements in the field of obstacle avoidance, pushing the limits of perception and decision-making in autonomous systems.

1. Deep Learning for Predictive Obstacle Avoidance

- **CNNs and RNNs** are now used to predict pedestrian and vehicle motion, enabling AVs to plan avoidance maneuvers before obstacles become dangerous.
- **Behavior cloning** and **imitation learning** allow vehicles to mimic expert human responses to complex obstacle scenarios.

2. Semantic Costmaps

Traditional costmaps mark space as occupied or free. Semantic costmaps go further by tagging objects (e.g., "pedestrian," "cyclist," "cone") and adjusting planner behavior accordingly.

3. Multi-Agent Planning

AVs are being trained to consider the actions of multiple surrounding agents:

- Predict interactions (e.g., lane merging, yielding)
- Generate cooperative avoidance behaviors

4. Reinforcement Learning (RL)-Based Avoidance

RL allows AVs to learn avoidance strategies in simulation and transfer them to real-world scenarios. These systems improve with time and exposure.

Feature	**Classical (DWA)**	**ML-Based (RL, DL)**
Real-time Replanning	✅	✅
Predictive Behavior	❌	✅
Explainability	✅	❌ (black-box)
Adaptivity	Moderate	High
Training Requirement	None	Extensive

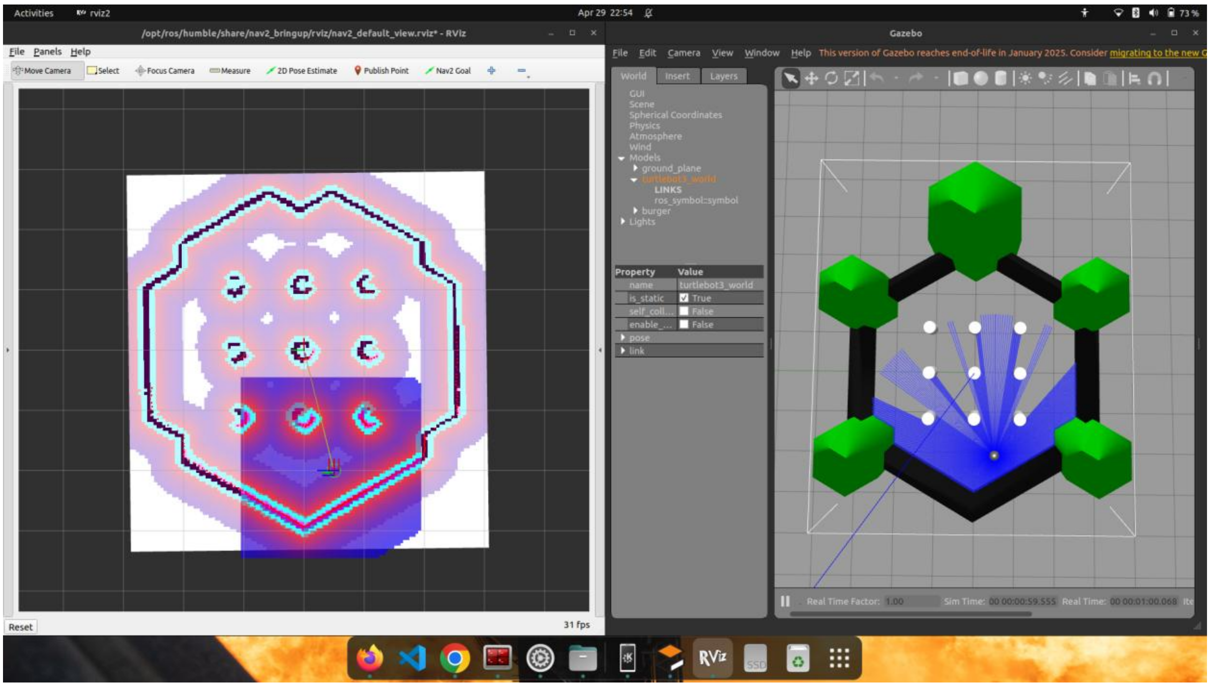
Table 3 *Classic (CWA) vs ML-Based (RL, DL) Avoidance*

CHAPTER 5
RESULTS AND CONCLUSION

5.1 Results

The autonomous navigation system was tested in a simulated Gazebo environment integrated with the ROS 2-based Nav2 stack and RViz for visualization. The system demonstrated effective mapping and path planning capabilities, utilizing SLAM for real-time environment mapping and obstacle-aware navigation.

In the **mapping phase**, SLAM was used to generate a 2D occupancy grid of the environment while the robot explored autonomously. The map accurately represented the walls, obstacles, and free space in the Gazebo world, enabling further navigation and planning. Sensor data from LiDAR was successfully integrated into the local and global costmaps to build and update the map dynamically.



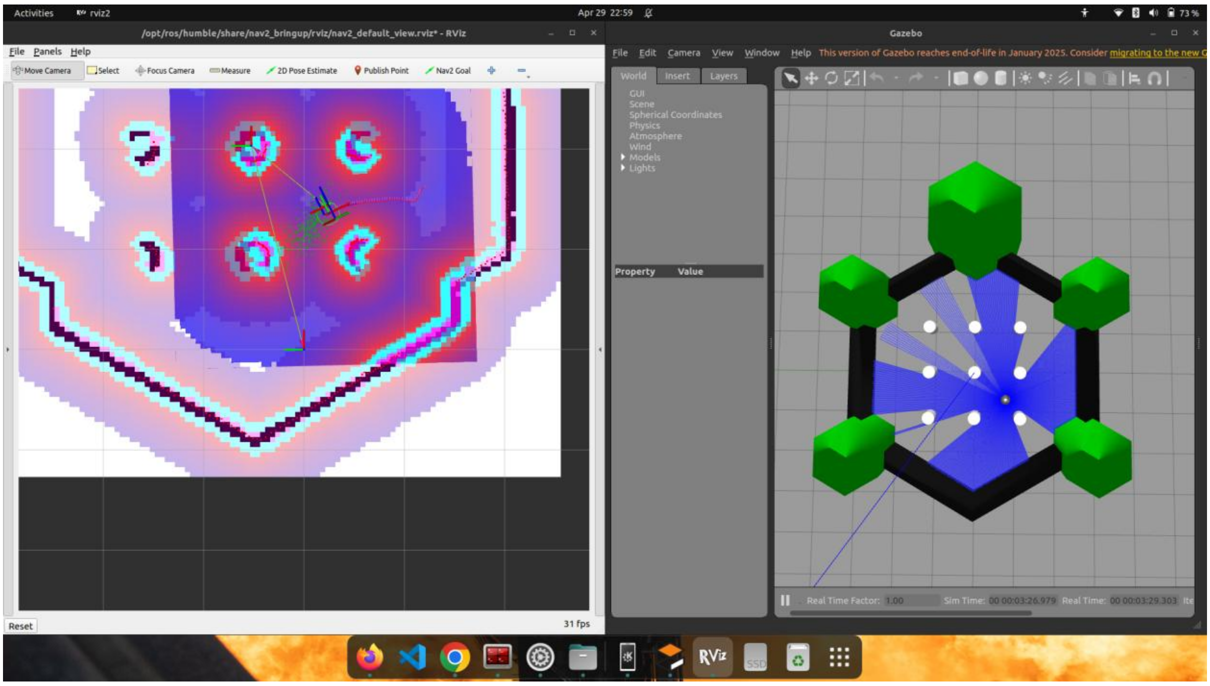
This version of Gazebo reaches end-of-life in January 2025. Consider migrating to the new G

Property	Value
name	turtlebot3_world
is_static	☑ True
self_coll...	□ False
enable_...	□ False

Real Time Factor: 1.00 Sim Time: 00 00:00:59.555 Real Time: 00 00:01:00.068 Ite

Figure 5.1 *Mapping Phase using Cartographer*

For the **navigation phase**, the robot was given a goal pose in RViz. Using A* for global planning and the Dynamic Window Approach (DWA) for local planning, the robot navigated from its start point to the destination while avoiding both static and dynamic obstacles in the environment. The local planner continuously recalculated the path in real-time using updated costmap data, ensuring smooth and collision-free motion.



This version of Gazebo reaches end-of-life in January 2025. Consider migrating to the new G

Figure 5.2 Navigation Phase (Rviz and Gazebo)

Throughout the simulation, the robot maintained accurate localization using *slam_toolbox*, and the planner successfully adapted to environmental changes, demonstrating real-time obstacle avoidance and route adjustment. The integration of behavior trees facilitated recovery behaviors in case of temporary navigation failures.

5.2 Conclusion

This project successfully implemented an autonomous navigation system using ROS 2 and the Nav2 stack, demonstrating real-time mapping, localization, planning, and obstacle avoidance in a simulated environment. Through the combination of SLAM, costmap-based planning, and behavior trees, the system achieved robust and dynamic navigation suitable for autonomous vehicles.

The mapping and navigation experiments validated the system's ability to perceive and adapt to complex environments, marking a significant step toward scalable and modular autonomous mobility solutions. The results suggest that with minor adaptations, the same system can be deployed on real hardware for real-world testing and applications in logistics, assistive mobility, or service robotics.

Future work could involve integrating vision-based semantic mapping, improving obstacle prediction using machine learning, and testing the system in larger and more dynamic environments.

5.3 Ethical and Safety Considerations in Autonomous Vehicle Deployment

As autonomous vehicles (AVs) transition from controlled environments to public roads, ethical and safety considerations become paramount. These systems are responsible for making real-time decisions that can directly impact human lives, and thus require a robust framework to ensure reliability, accountability, and public trust.

5.3.1 Safety Assurance

Safety is the foundation of autonomous systems. AVs must be rigorously tested in both simulated and real-world scenarios before deployment. Safety assurance includes:

- **Redundancy in critical systems** like braking, steering, and perception modules to handle hardware or software failures.
- **Fail-safe behaviors**, such as stopping in place or returning to a known location when system faults occur.
- **Continuous real-time monitoring** of the environment and internal diagnostics to pre-emptively detect and respond to unsafe conditions.

Additionally, thorough validation and verification of planning algorithms—especially under edge-case scenarios like sudden pedestrian crossings or unstructured road layouts—are essential.

5.3.2 Ethical Decision-Making

AVs may face moral dilemmas in situations where harm is unavoidable, such as choosing between hitting one pedestrian or swerving into another. These scenarios raise questions that currently lack universally accepted answers:

- **Who decides what is an ethically "correct" action**? Is it the developer, manufacturer, regulatory body, or society at large?
- **Can AVs be programmed to reflect cultural or regional ethical values**, which may vary?
- **Should AVs prioritize the safety of passengers over pedestrians**, or vice versa?

While frameworks like value-aligned AI and explainable decision models are being explored, ethical reasoning in real-time remains an open challenge.

5.3.3 Privacy and Data Security

Autonomous vehicles generate and process vast amounts of sensor data—including video, LiDAR, GPS, and V2X communication—that could potentially infringe on individual privacy. Therefore:

- **Data anonymization** and **secure transmission protocols** must be enforced.
- Access to sensitive driving data should be **strictly controlled and logged**.

Real-time diagnostics or camera feeds must **not compromise bystander privacy**, especially in residential areas or private spaces.

5.3.4 Legal and Regulatory Framework

The deployment of AVs must comply with evolving legal standards governing liability, insurance, and usage rights. Key questions include:

- **Who is liable in case of an accident**—the manufacturer, software provider, or user?
- **How are regulations standardized across different regions** and road systems?
- **What auditing mechanisms exist to ensure compliance with safety standards**?

Until a uniform global regulatory framework exists, AVs must be developed with configurable safety settings to adapt to local laws.

5.3.5 Public Acceptance and Transparency

Building public trust is as important as achieving technical excellence. This involves:

- **Transparent decision-making**, where AVs can explain their actions post-incident.
- **Community involvement** in policy creation and deployment trials.
- Ensuring that AV behavior is **predictable and intuitive** to human drivers and pedestrians.

CHAPTER 6

REFERENCES

[1] J. Levinson et al., "Towards fully autonomous driving: Systems and algorithms," in Proc. 2011 IEEE Intelligent Vehicles Symp. (IV), Baden-Baden, Germany, 2011, pp. 163–168, doi: 10.1109/IVS.2011.5940562.

[2] D. Feng et al., "Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges," IEEE Trans. Intell. Transp. Syst., vol. 22, no. 3, pp. 1341–1360, Mar. 2021, doi: 10.1109/TITS.2020.2972974.

[3] T. Okuyama, T. Gonsalves, and J. Upadhay, "Autonomous driving system based on deep Q learning," in Proc. 2018 Int. Conf. Intell. Autonomous Syst. (ICoIAS), Singapore, 2018, pp. 201–205, doi: 10.1109/ICoIAS.2018.8494053.

[4] H. Caesar et al., "nuScenes: A multimodal dataset for autonomous driving," in Proc. 2020 IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Seattle, WA, USA, 2020, pp. 11618–11628, doi: 10.1109/CVPR42600.2020.01164.

[5] "Arduino Uno," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Arduino_Uno. [Accessed: Dec. 8, 2024].

[6] "Raspberry Pi Documentation," Raspberry Pi Foundation, [Online]. Available: https://www.raspberrypi.com/documentation/. [Accessed: Dec. 8, 2024].

[7] H. Kim et al., "Vision-based real-time obstacle segmentation algorithm for autonomous surface vehicle," IEEE Access, vol. 7, pp. 179420–179428, Dec. 2019, doi: 10.1109/ACCESS.2019.2959312.

[8] L. Chen et al., "Deep neural network based vehicle and pedestrian detection for autonomous driving: A survey," IEEE Trans. Intell. Transp. Syst., vol. 22, no. 6, pp. 3234–3246, Jun. 2021, doi: 10.1109/TITS.2020.2993926.

[9] D. Hutabarat, M. Rivai, D. Purwanto, and H. Hutomo, "Lidar-based obstacle avoidance for the autonomous mobile robot," in Proc. 12th Int. Conf. Inf. Commun. Technol. Syst. (ICTS), Surabaya, Indonesia, 2019, pp. 197–202, doi: 10.1109/ICTS.2019.8850952.

[10] H. Cho, P. E. Rybski, and W. Zhang, "Vision-based bicyclist detection and tracking for intelligent vehicles," in Proc. IEEE Intell. Vehicles Symp. (IV), La Jolla, CA, USA, 2010, pp. 454–461, doi: 10.1109/IVS.2010.5548063.

[11] VISO.AI, "LiDAR: Light Detection and Ranging - Working & Application," [Online]. Available: https://www.elprocus.com/lidar-light-detection-and-ranging-working-application/. [Accessed: Dec. 8, 2024].

[12] "LiDAR," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Lidar. [Accessed: Dec. 8, 2024].

[13] MaxBotix, "How ultrasonic sensors work," [Online]. Available: https://maxbotix.com/blogs/blog/how-ultrasonic-sensors-work. [Accessed: Dec. 8, 2024].

[14] Standing Tech, "What is GPS tracking system?" [Online]. Available: https://standingtech.com/2016/01/07/what-is-gps-tracking-system/. [Accessed: Dec. 8, 2024].

[15] "Global Positioning System," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Global_Positioning_System. [Accessed: Dec. 8, 2024].

[16] Advanced Navigation, "Inertial measurement unit (IMU): An introduction," [Online]. Available: https://www.advancednavigation.com/tech-articles/inertial-measurement-unit-imu-an-introduction/. [Accessed: Dec. 8, 2024].

[17] "Inertial measurement unit," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Inertial_measurement_unit. [Accessed: Dec. 8, 2024].

[18] "Behavior Tree Basic Structure," GitHub, [Online]. Available: https://user-images.githubusercontent.com/119948/64462441-048d9380-d0c6-11e9-9e40-a28c3c39e059.png

[19] "Behavior Tree Node Types," Robohub, [Online]. Available: https://robohub.org/wp-content/uploads/2021/08/bt_node_types.png

[20] "Obstacle avoidance trajectory of wheeled mobile robot platform," ResearchGate, [Online]. Available: https://www.researchgate.net/publication/320744124/figure/fig3/AS:556055168864256@1509585291990/Obstacle-avoidance-trajectory-of-wheeled-mobile-robot-platform.png

[21] "ROS2 Navigation," ROS2 Industrial Workshop Documentation, [Online]. Available: https://ros2-industrial-workshop.readthedocs.io/en/latest/_source/navigation/ROS2-Navigation.html

[22] "3D Point Cloud," Keymakr Blog, [Online]. Available: https://keymakr.com/blog/content/images/size/w1200/2023/06/3D-Point-cloud.jpg

[23] "MPU6050 with the pin layout and axes orientation," ResearchGate, [Online]. Available: https://www.researchgate.net/publication/337664623/figure/fig3/AS:831395677093888@1575231584108/MPU6050-with-the-pin-layout-and-axes-orientation.ppm

[34] "AI-Generated Image Using ChatGPT," ChatGPT Image Generation Tool.

[25] "Example of 3D Perception in Robotics," Google Images, [Online]. Available: https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQARV5eY0L91u7LgoLmYWEVb1vivURhnVFNdg&s

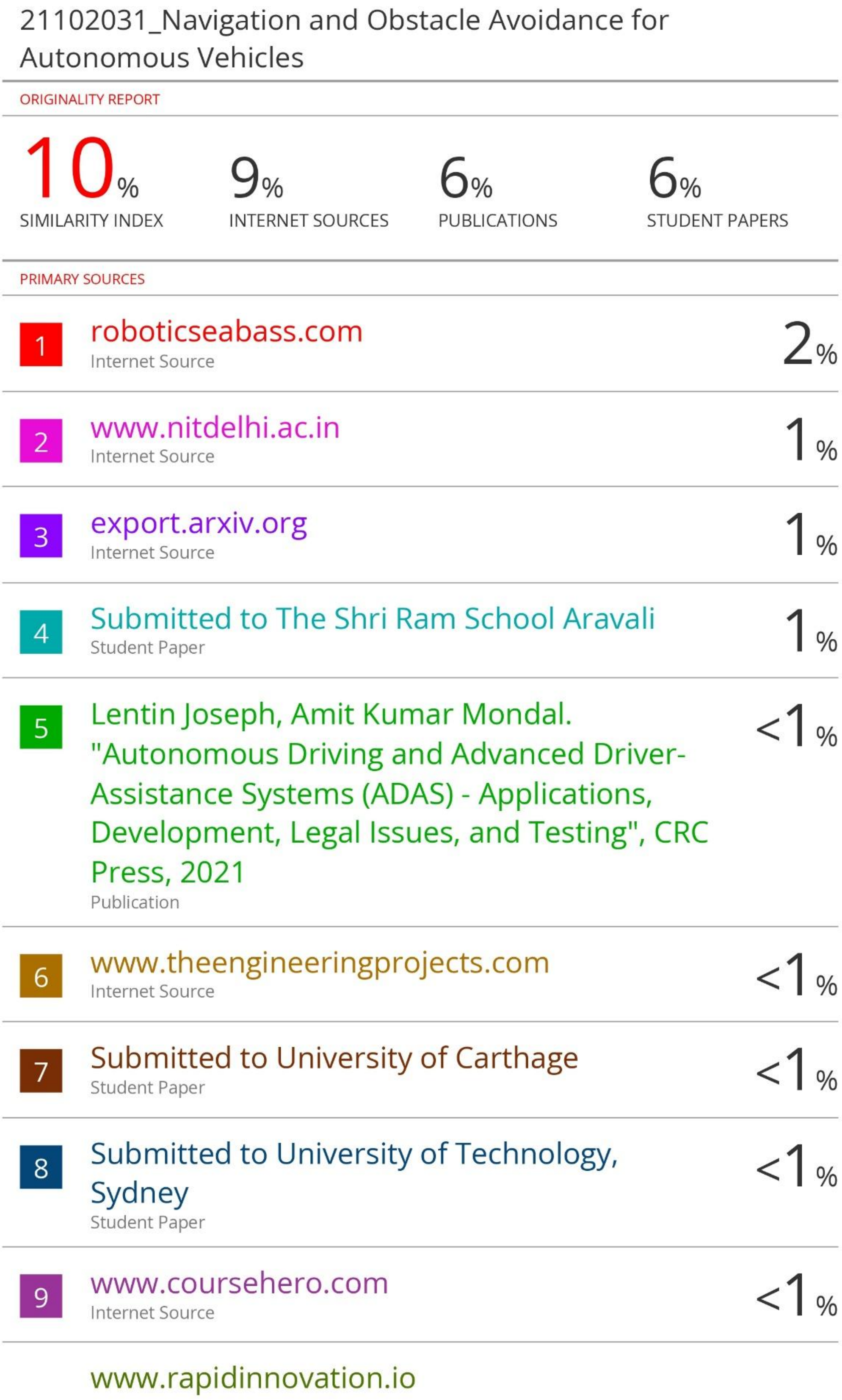
[26] "3D Target Adds Three LiDARs from Velodyne to Its Roster," GIM International, [Online]. Available: https://www.gim-international.com/content/news/3d-target-adds-three-lidars-from-velodyne-to-its-roster?output=pdf

[27] "Corner Radar Sensor – Predictive Emergency Braking System," Bosch Mobility, [Online]. Available: https://www.bosch-mobility.com/media/global/solutions/passenger-cars-and-light-commercial-vehicles/driver-assistance-systems/predictive-emergency-braking-system/corner-radar-sensor/corner-radar_stage.jpg

[28] "Featured Image," Havi, [Online]. Available: https://www.havi.co/wp-content/uploads/2024/08/featured-image.jpg

[29] Raspberry Pi Foundation, "Raspberry Pi 4 Model B," Raspberry Pi, [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/. [Accessed: May 16, 2025].

[30] W.S. Publishing, "AVR ISP connection to Arduino Mega," W.S. Publishing, [Online]. Available: https://wspublishing.net/avr-c/arduino-mega-avrisp-connection/. [Accessed: May 16, 2025].



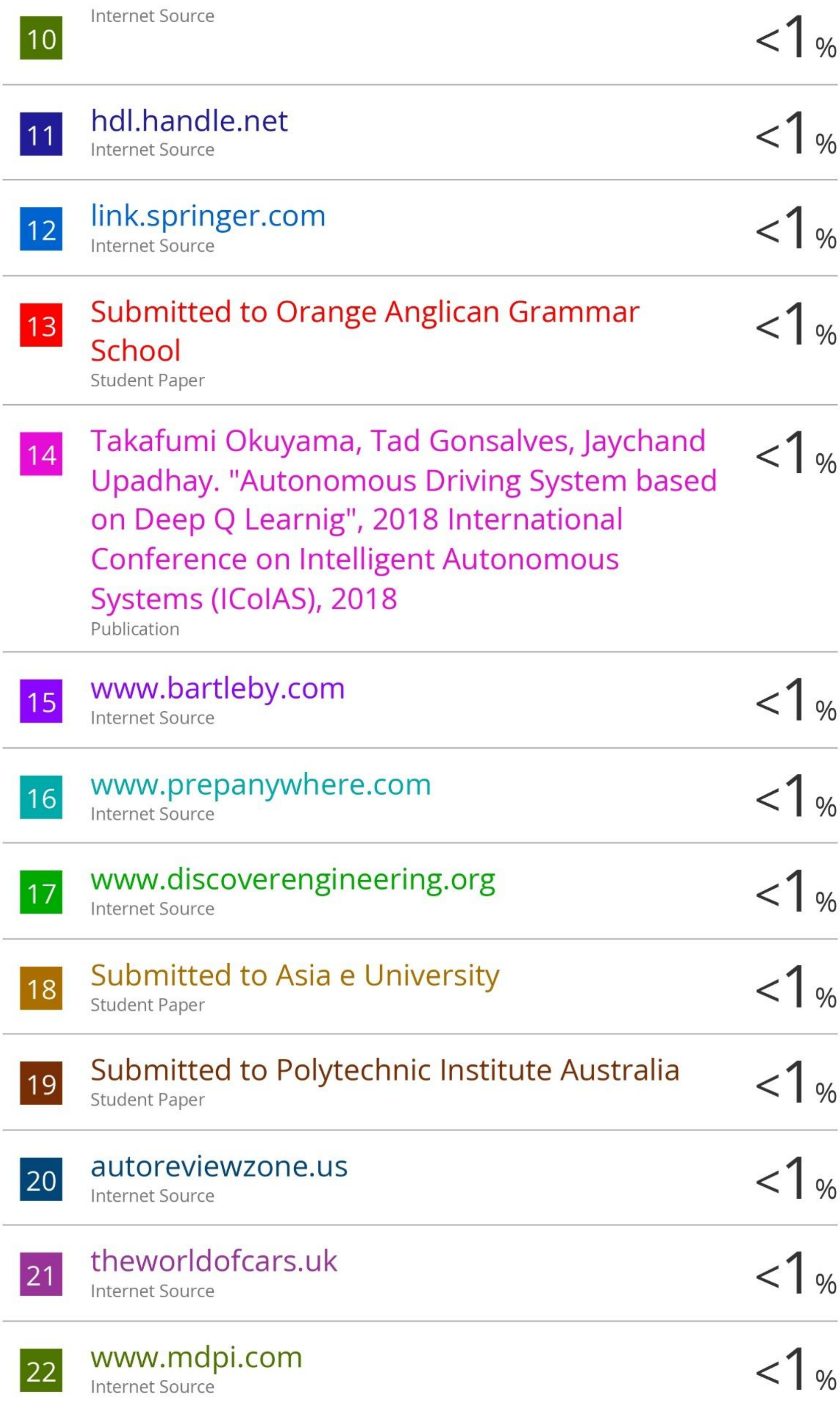
21102031_Navigation and Obstacle Avoidance for Autonomous Vehicles

ORIGINALITY REPORT

10%	9%	6%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	roboticseabass.com Internet Source	2%
2	www.nitdelhi.ac.in Internet Source	1%
3	export.arxiv.org Internet Source	1%
4	Submitted to The Shri Ram School Aravali Student Paper	1%
5	Lentin Joseph, Amit Kumar Mondal. "Autonomous Driving and Advanced Driver-Assistance Systems (ADAS) - Applications, Development, Legal Issues, and Testing", CRC Press, 2021 Publication	<1%
6	www.theengineeringprojects.com Internet Source	<1%
7	Submitted to University of Carthage Student Paper	<1%
8	Submitted to University of Technology, Sydney Student Paper	<1%
9	www.coursehero.com Internet Source	<1%
	www.rapidinnovation.io	



10	Internet Source	<1 %
11	hdl.handle.net Internet Source	<1 %
12	link.springer.com Internet Source	<1 %
13	Submitted to Orange Anglican Grammar School Student Paper	<1 %
14	Takafumi Okuyama, Tad Gonsalves, Jaychand Upadhay. "Autonomous Driving System based on Deep Q Learnig", 2018 International Conference on Intelligent Autonomous Systems (ICoIAS), 2018 Publication	<1 %
15	www.bartleby.com Internet Source	<1 %
16	www.prepanywhere.com Internet Source	<1 %
17	www.discoverengineering.org Internet Source	<1 %
18	Submitted to Asia e University Student Paper	<1 %
19	Submitted to Polytechnic Institute Australia Student Paper	<1 %
20	autoreviewzone.us Internet Source	<1 %
21	theworldofcars.uk Internet Source	<1 %
22	www.mdpi.com Internet Source	<1 %

23	www.arxiv.org Internet Source	<1 %
24	Submitted to CSU, San Jose State University Student Paper	<1 %
25	Submitted to Concordia University Student Paper	<1 %
26	Submitted to University of Limerick Student Paper	<1 %
27	umpir.ump.edu.my Internet Source	<1 %
28	Submitted to Institute of Technology, Sligo Student Paper	<1 %
29	Joko Slamet Saputro, Hanif Wisti Juliatama, Feri Adriyanto, Hari Maghfiroh, Esa Apriaskar. "Collision Avoidance in Mini Autonomous Electric Vehicles Using Artificial Potential Fields for Outdoor Environment", International Journal of Robotics and Control Systems, 2025 Publication	<1 %
30	arxiv.org Internet Source	<1 %
31	ravishreyas.github.io Internet Source	<1 %

Exclude quotes	On	Exclude matches	< 14 words
Exclude bibliography	On		