ISSN Number: 2454-2024(Online)

International Journal of Technical Research & Science

AUTONOMOUS VALET PARKING USING ADAS SENSORS AND CLOUD INTEGRATION

Utsav Dhanani, Jinesh Kamdar, Madhusudan Barot
E-Mail Id: utsavdhanani.21.am@iite.indusuni.ac.in, jineshkamdar.am@indusuni.ac.in, madhusudanbarot.me@indusuni.ac.in
Indus Institute of Technology and Engineering, Indus University, Ahmedabad, Gujarat, India

Abstract- This study presents a modular and scalable Autonomous Valet Parking (AVP) system, tailored for structured, single-floor parking facilities. The proposed framework facilitates fully automated vehicle parking and retrieval, eliminating human involvement while optimizing spatial efficiency, user accessibility, and traffic flow within urban environments. The operational sequence is initiated when the user leaves the vehicle at Ground 0 Level, activating Vehicle-to-Infrastructure (V2I) communication with a cloud-integrated parking management system. This system identifies and allocates an available parking slot in real time, subsequently generating and transmitting a collision-free trajectory that accounts for both static architecture and dynamic obstacles. The autonomous vehicle executes the assigned trajectory using a sensor fusion suite comprising LiDAR, 3D depth camera, ultrasonic sensors, wheel encoders, ESP module, and real-time localization modules. (Liu et al., 2019; Kuutti et al., 2018). Navigation is governed through a hybrid control strategy combining Proportional-Integral-Derivative (PID) and Model Predictive Control (MPC), integrated with Simultaneous Localization and Mapping (SLAM) to ensure precise maneuvering in GPS-denied environments. (Cadena et al., 2016; Labbé & Michaud, 2019). Vehicle retrieval is initiated via a mobile application, prompting autonomous traversal along the reverse trajectory to the designated pickup zone. The system was prototyped and validated on a scaled model, with experimental results confirming its robustness in trajectory tracking, obstacle avoidance, and retrieval accuracy. The outcomes affirm the AVP system's viability as a foundational element in future smart mobility ecosystems and intelligent urban infrastructure.

Keywords: Autonomous Valet Parking (AVP), Vehicle-to-Infrastructure (V2I) Communication, Simultaneous Localization and Mapping (SLAM), Model Predictive Control (MPC), ADAS Sensors, ROS Noetic, Trajectory Planning, Cloud Integration.

1. INTRODUCTION

The contemporary shift toward intelligent transportation ecosystems has accelerated the development of autonomous mobility solutions, with Autonomous Valet Parking (AVP) emerging as a pivotal intermediate step toward fully autonomous driving. Unlike Level 5 autonomous vehicles, which necessitate extensive legislative reform, V2X infrastructure, and high-fidelity AI decision-making, AVP systems are constrained to predefined, structured, and low-speed environments—such as indoor parking facilities—making them ideal candidates for near-term commercial deployment under restricted Operational Design Domains (ODDs).

AVP addresses several chronic inefficiencies in urban transportation networks. Conventional parking methods contribute substantially to traffic congestion, energy wastage, and carbon emissions (Shoup, 2005), largely due to prolonged search times and suboptimal use of available space. By enabling vehicles to park and retrieve themselves autonomously, AVP enhances spatial efficiency, reduces driver burden, and minimizes environmental impact through the elimination of engine idling and repetitive low-speed maneuvering.

Modern AVP architectures are increasingly characterized by a distributed intelligence model that merges onboard autonomous capabilities with cloud- or edge-based computational frameworks. These systems typically leverage Vehicle-to-Infrastructure (V2I) communication for real-time slot allocation (He et al., 2018; Liu et al., 2019) and trajectory provisioning, while the vehicle executes localization, perception, and control operations using an array of onboard sensors. Commonly employed modalities include LiDAR, ultrasonic sensors, wheel encoders, and inertial measurement units (IMUs), fused through SLAM-based algorithms (Cadena et al., 2016; Thrun et al., 2005). Precision in path-following is maintained via advanced control strategies such as Model Predictive Control (MPC) or dynamically tuned PID controllers (Qin & Badgwell, 2003; Åström & Hägglund, 2006), ensuring reliable operation even in cluttered, GPS-denied environments.

Pioneering efforts by automotive manufacturers and Tier 1 suppliers—such as Mercedes-Benz, BMW, Bosch, and Continental—have demonstrated Level 4 AVP functionality through collaborations with infrastructure providers in pilot deployments. However, the requirement for highly instrumented environments—featuring HD maps, 5G connectivity, and beacon networks (Schwarzenberg et al., 2018)—presents a barrier to large-scale, cost-effective implementation, particularly in legacy parking infrastructures.

In alignment with current trends favoring lightweight and scalable automation solutions, the present research introduces a simplified AVP framework designed for single-floor parking structures. Emphasizing low-cost sensor integration, minimal external infrastructure dependency, and seamless mobile interface control, the proposed system serves as a proof-of-concept for democratizing AVP technologies and enabling broader adoption across

both public and private sectors. This approach underscores the growing viability of modular autonomy in constrained environments, paving the way for incremental deployment within the larger vision of smart mobility and urban automation

2. REVIEW OF LITERATURE

S. No.	Title of the Paper	Authors	Year of Publication	Key Findings	What is Not Covered
1	Infrastructure-Based Automated Valet Parking with V2X Communication	H. Winner et al.	2021	Demonstrated an infrastructure-based AVP using fixed beacons and V2X for trajectory planning and parking slot allocation.	Lacked vehicle-side autonomy and did not address fallback control in case of communication failure.
2	Real-Time SLAM for Autonomous Indoor Navigation	S. Huang et al.	2018	Employed LiDAR-based SLAM fused with encoder data for indoor mapping in autonomous mobile robots.	Did not explore dynamic obstacle handling or integration with trajectory planning modules.
3	MPC-Based Control for Autonomous Vehicles in Cluttered Environments	Y. Wang and F. Gao	2020	Presented a lightweight Model Predictive Control (MPC) method suitable for real-time path tracking in constrained indoor settings.	No direct application to multi-agent parking scenarios; assumes static surroundings.
4	An Overview of Autonomous Parking Technologies	K. Park et al.	2022	Reviewed current state-of-the-art in AVP, including sensor technologies, cloud integration, and mobile interface design.	The review lacked specific implementation strategies for low-cost prototypes or scaled models.

3. OBJECTIVES OF THE RESEARCH

- The primary objective of this project is to design and implement an intelligent Autonomous Valet Parking (AVP) system using a 1:10 scale electric vehicle platform that emulates real-world valet operations within a structured parking environment. The project aims to demonstrate the vehicle's ability to autonomously navigate, detect vacant parking slots, and perform parking maneuvers without human intervention. By integrating Ackermann steering geometry, ultrasonic sensors, a 3D LiDAR, and a depth camera, the prototype replicates the key physical and perceptual functions of a full-scale autonomous car.
- Another core objective is to establish robust perception, localization, and control mechanisms using modern robotic technologies. The system incorporates simultaneous localization and mapping (SLAM) fused with wheel encoder data to achieve real-time environmental awareness and path estimation. A hybrid control strategy combining PID (for precise low-speed maneuvers) and Model Predictive Control (for predictive trajectory tracking) is used to ensure reliable navigation within a structured indoor parking space. This control framework supports essential features such as ramp climbing, obstacle avoidance, and accurate slot positioning.
- Furthermore, the project seeks to implement a cloud-integrated Vehicle-to-Infrastructure (V2I) communication framework that allows users to view available parking slots and reserve them through a mobile application. Real-time updates from the AVP vehicle are uploaded to a central server, enabling remote monitoring and management of the parking facility. By combining embedded systems, IoT-based communication, and robotic automation, the AVP prototype presents a scalable and modular solution for future smart parking infrastructures.

4. RESEARCH METHODOLOGY

4.1 Requirement Analysis

Identified design goals, system constraints, and use-case scenarios for a structured AVP environment.

4.2 Hardware Integration

Assembled a 1:10 scale Ackermann-steered vehicle with sensors (LiDAR, depth camera, ultrasonic) and onboard computing (Raspberry Pi 5).

4.3 Software Development

Implemented ROS-based perception, SLAM, PID-MPC control, and V2I cloud communication modules.

4.4 Simulation & Testing

Validated behavior in Gazebo simulation and real-world conditions using structured test cases.

4.5 Evaluation

Assessed system performance based on localization accuracy, parking success rate, and reservation handling latency.

5. SYSTEM COMPONENT AND ARCHITECTURE

The physical test environment was delineated with well-defined lanes, boundary markers, and parking slots, each identified via scannable fiducial markers (QR codes or AprilTags). A simulated cloud-based server functioned as the central parking management entity, maintaining a live map of slot occupancy and coordinating real-time data exchange with the autonomous vehicle.

5.1 Vehicle-to-Infrastructure (V2I) Interface

Communication was facilitated through a local Wi-Fi network (IEEE 802.11n standard), emulating low-latency V2I connectivity (Liu et al., 2019) for slot assignment and trajectory dissemination.

5.2 Slot Management System

The server executed a slot allocation algorithm that periodically evaluated slot availability and dispatched the most optimal parking coordinate set based on the vehicle's current position.

5.3 Trajectory Provisioning:

Upon successful allocation, a pre-optimized collision-free trajectory—encoded as a series of discrete waypoints (Zhang et al., 2017; Paden et al., 2016)—was transmitted to the vehicle. A lightweight Model Predictive Control (MPC) framework interpreted and executed this path, balancing smooth motion with obstacle anticipation and real-time replanning.

5.4 Mobile Application Interface

User interaction with the AVP system was enabled through a custom-designed Android mobile application that served as the command and feedback interface. Key functionalities included:

- Confirmation of vehicle drop-off at Ground Level 0
- Visualization of assigned slot and current vehicle status
- Initiation of retrieval request via a single-touch command

The mobile client was linked to the backend server, which orchestrated command forwarding to the vehicle for initiation of autonomous return-to-user protocols.

5.5 Controlled Testing Environment

All experiments were executed within a controlled laboratory space measuring approximately 3.5 meters by 3 meters—replicating a narrow, single-lane indoor parking zone. To evaluate the system's resilience to real-world dynamics, multiple static and mobile obstacles were strategically introduced across test runs. These included mock pedestrians and vehicle-like objects to simulate unpredictable obstructions and to assess real-time evasive behavior.

5.6 Evaluation Metrics

System performance was quantified using a set of well-defined Key Performance Indicators (KPIs):

5.6.1 Localization Precision

Mean deviation from ground truth location, recorded through calibrated benchmark markers.

5.6.2 Path Fidelity

Lateral offset measured against the reference trajectory to assess motion smoothness and tracking consistency.

5.6.3 Slot Alignment Accuracy

Final positional and angular deviation upon parking, relative to ideal slot geometry.

5.6.3 Task Completion Time

Total duration from initiation of autonomous parking to successful vehicle halt and retrieval.

5.6.3 Obstacle Avoidance Efficiency

Ratio of successful evasive actions to total obstacle encounters, measured across multiple trials.

6. HARDWARE ABSTRACTION LAYER

6.1 Hardware Configuration

The autonomous vehicle platform used in this study is a 1:10 scale robotic model of a conventional passenger vehicle. It is equipped with a comprehensive sensory and actuation architecture to enable autonomous navigation, perception, and decision-making.

For environmental perception, a rotating 360-degree 3D LiDAR scanner is mounted centrally to provide high-resolution point cloud data (Zhang & Singh, 2014) for obstacle detection and spatial mapping. A 3D depth camera

complements this by offering dense depth information, which aids in scene understanding and visual odometry. Ultrasonic sensors are strategically placed at the front and rear to provide short-range proximity data, especially effective during low-speed alignment into parking spaces or in cluttered environments.

```python
#!/usr/bin/env python3

import rospy
from sensor_msgs.msg import LaserScan
from nav_msgs.msg import Odometry
from geometry_msgs.msg import Pose

def scan_callback(data):
    # Process LiDAR data
    rospy.loginfo("Received LiDAR scan with %d ranges", len(data.ranges))
```

Fig. 6.1 A Robotic Model of a Conventional Passenger Vehicle

6.2 Localization and Mapping Subsystem

In GPS-denied environments such as indoor parking structures, localization is achieved through a combination of LiDAR-based simultaneous localization and mapping (SLAM) (Hess et al., 2016; Labbé & Michaud, 2019) and wheel encoder feedback. This fusion enables accurate real-time pose estimation and dynamic map generation. The use of wheel encoders helps maintain localization fidelity during short occlusions or rapid direction changes.

6.3 Actuation Mechanism

The platform uses an Ackermann steering geometry, commonly found in passenger vehicles, to achieve realistic turning behavior. Steering is controlled via a high-torque servo motor, regulated through a closed-loop proportional-integral-derivative (PID) control algorithm (Åström & Hägglund, 2006). Locomotion is driven by two rear-mounted DC gear motors, with speed and direction controlled through pulse-width modulation (PWM) signals. This configuration allows for robotic model of a conventional passenger vehicle, critical for navigating tight parking environments.

```python
def odom_callback(data):
    # Print current position
    position = data.pose.pose.position
    rospy.loginfo("Position x: %.2f y: %.2f", position.x, position.y)

if __name__ == '__main__':
    rospy.init_node('slam_listener', anonymous=True)
    rospy.Subscriber('/scan', LaserScan, scan_callback)
    rospy.Subscriber('/odom', Odometry, odom_callback)
    rospy.spin()
```

Fig. 6.2 Robotic Model of a Conventional Passenger Vehicle

6.4 Embedded Processing Unit

All high-level computational tasks, including SLAM, sensor fusion, trajectory planning, and cloud communication, are executed on a Raspberry Pi 5. The upgraded hardware provides enhanced processing capabilities, support for LPDDR4X memory, and efficient thermal performance. The system runs on Ubuntu 22.04 LTS, fully integrated with the Robot Operating System (ROS) Noetic for modular robotics application development.

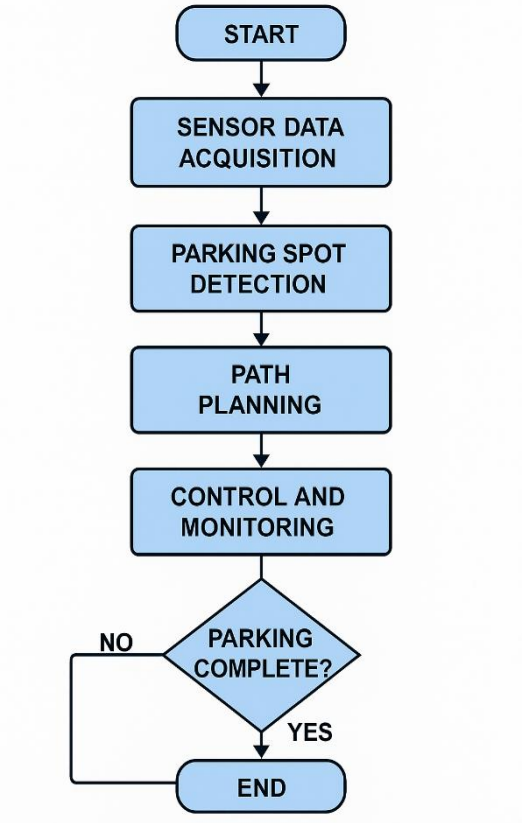

Fig. 6.3 Autonomous Parking Process

6.5 Power Supply Architecture

A 12V lithium-ion battery system serves as the primary power source. Voltage regulation and distribution are managed through an onboard power distribution board to ensure stable power delivery to all motors, sensors, and computational components. This setup supports uninterrupted operation across all autonomous phases.

7. SOFTWARE ARCHITECTURE LAYER

7.1 Operating Environment

The software architecture is built upon a Linux-based Ubuntu 20.04 operating system, selected for its stability, wide compatibility with robotic frameworks, and support for real-time processing. This environment ensures consistent system behavior and facilitates seamless deployment of robotic packages.

7.2 ROS Middleware Framework

ROS Noetic Ninjemys serves as the middleware (Quigley et al., 2009) framework, enabling modular system design through decoupled inter-process communication. The AVP architecture utilizes ROS topics, services, and actions to manage distributed system components. Dedicated ROS nodes are implemented to perform critical functions including sensor abstraction and data acquisition, simultaneous localization and mapping (SLAM), global and local path planning, motion control and trajectory execution, and cloud communication for data synchronization.

7.3 Python-Based Node Implementation

Python is adopted as the primary programming language for ROS node development due to its concise syntax, extensive library support, and compatibility with machine learning frameworks. Key modules written in Python include Kalman filters for state estimation (Welch & Bishop, 2001), preprocessing pipelines for sensor data, cloud communication protocols, and inference engines for deep learning models. This implementation approach supports rapid development, maintainability, and scalability of the AVP software system.

7.4 Perception, Localization, and Mapping

The autonomous valet parking (AVP) system integrates 3D LiDAR and a depth camera to execute simultaneous localization and mapping (SLAM) using algorithms such as Cartographer or RTAB-Map (Hess et al., 2016; Labbé & Michaud, 2019). This enables volumetric environmental reconstruction and accurate localization within multi-level parking infrastructures. Sensor fusion produces a detailed 3D occupancy grid, enhancing situational awareness and spatial accuracy.

For parking spot detection, depth data is processed by a convolutional neural network (Chen et al., 2022) trained to classify slot occupancy. When combined with LiDAR input, this dual-modality system ensures robust detection performance, particularly in cluttered or occluded environments.

7.5 Trajectory Planning and Motion Control

Path planning is performed through a dual-layered approach. A global planner such as A* or Dijkstra (Hart et al., 1968) computes the optimal route to the selected parking bay. Concurrently, a local planner, such as the pure pursuit or dynamic window approach (Coulter, 1992), dynamically adjusts the vehicle's trajectory in response to environmental changes.

Motion control is implemented using proportional-integral-derivative (PID) loops to modulate wheel velocities (Åström & Hägglund, 2006) and steering angles. These feedback mechanisms are essential for maintaining stable, accurate, and drift-free navigation, especially during low-speed parking maneuvers.

```
python

class PIDController:
    def __init__(self, kp, ki, kd):
        self.kp = kp
        self.ki = ki
        self.kd = kd
        self.prev_error = 0
        self.integral = 0
```

Fig. 7.1 Path planning

7.6 Cloud Integration and Reservation Management

Once a vacant parking space is detected, metadata including spatial coordinates, timestamp, and occupancy status are uploaded to a central cloud server using lightweight communication protocols such as MQTT or HTTP REST APIs (Banks & Gupta, 2014; Locke, 2010). This ensures real-time synchronization of parking availability across multiple AVP-equipped units.

End-users can access a web-based or mobile interface to query live parking availability and reserve a space. Each reservation is time-bound (e.g., ten-minute window) and authenticated via a unique vehicle identifier to prevent duplication and ensure exclusivity.

8. SIMULATION, TRAINING, AND VALIDATION

Prior to real-world deployment, the AVP system undergoes simulation testing in the Gazebo environment integrated with ROS (Koenig & Howard, 2004). This platform allows for the evaluation of system performance under various conditions, including differing obstacle densities, lighting variations, and terrain irregularities.

Machine learning models used in perception tasks are developed using frameworks such as PyTorch and Tensor Flow(Paszke et al., 2019; Abadi et al., 2016). Transfer learning techniques are employed to reduce training time, while data augmentation methods including brightness modulation, noise injection, and geometric transformations are applied to improve model generalization across diverse parking environments.

```
    def compute(self, target, current):
        error = target - current
        self.integral += error
        derivative = error - self.prev_error
        output = self.kp * error + self.ki * self.integral + self.kd * derivative
        self.prev_error = error
        return output

# Example usage
pid = PIDController(kp=1.0, ki=0.1, kd=0.05)
steering_angle = pid.compute(target=0.0, current=0.2)  # desired angle = 0
print(f"Steering command: {steering_angle}")
```

Fig. 8.1 Machine Learning Models used in Perception Tasks

9. SYSTEM MODEL

In this section, we present the system architecture and operational framework for the Autonomous Valet Parking (AVP) scenario implemented on a 1:10 scale robotic platform. The parking infrastructure is structured as a single-floor spiral model, where the vehicle begins from a designated entry point located at the ground floor and autonomously navigates toward available parking slots on the first floor.

Let the AV be equipped with a multi-sensor fusion suite consisting of a 3D LiDAR, a 3D Depth Camera, ultrasonic sensors, and wheel encoders. The vehicle's pose at time step ttt is represented as q(t)=[x(t),y(t),θ(t)]q(t) = [x(t), y(t), \theta(t)]q(t)=[x(t),y(t),θ(t)], where x(t)x(t)x(t) and y(t)y(t)y(t) denote the planar coordinates and θ(t)\theta(t)θ(t) denotes the heading angle, obtained through SLAM-based localization.

The vehicle follows an Ackermann steering model (Rajamani, 2011), where the turning radius and velocity are constrained by the physical geometry of the scaled chassis. The state transition between successive steps is governed by the kinematic equations of motion under PID-MPC hybrid control. At each control cycle, the AV selects an action a(t)∈Aa(t) \in Aa(t)∈A, where A={FORWARD,TURN-LEFT,TURN-RIGHT,STOP}A = \{\text{FORWARD}, \text{TURN-LEFT}, \text{TURN-RIGHT}, \text{STOP}\}A={FORWARD,TURN-LEFT,TURN-RIGHT,STOP}, which determines the velocity and steering angle sent to the actuators.

We denote the set of known obstacles in the parking environment as O={o1,o2...,om}O = \{o_1, o_2, ..., o_m\}O={o1,o2,...,om}, each represented as a static coordinate cluster detected via LiDAR segmentation. The AV must satisfy the safety constraint q(t)∉O,∀tq (t) \notin O, \forall tq(t)∈/O,∀t, while continuously updating the map with real-time sensor feedback.

A centralized cloud server maintains real-time parking slot availability. Each slot Si∈SS_i \in SSi∈S is either free or occupied, represented as a binary variable si∈{0,1}s_i \in \{0,1\}si∈{0,1}, updated via V2I communication when a new spot is detected or reserved. Upon receiving a slot reservation request through the mobile interface, the AV is commanded to navigate to the respective slot SkS_kSk, where sk=0s_k = 0sk=0. A reservation window of 10 minutes is maintained, after which the slot state is reset if unoccupied.

The goal of the AVP system is to minimize the overall path cost C=∑t=1Td(q(t),q(t−1))C = \sum_{t=1}^{T} d(q(t), q(t-1))C=∑t=1Td(q(t),q(t−1)), where d(⋅)d(\cdot)d(⋅) is the Euclidean distance, subject to constraints of collision avoidance, steering limits, and parking slot validation. Upon successful parking, the vehicle transmits its final state q(T)q(T)q(T) to the server, completing the autonomous valet cycle.

9.1 Message Architecture and Communication Protocol

In autonomous valet parking systems, the integrity of inter-module and vehicle-to-infrastructure communication is fundamental to ensuring safe, efficient, and responsive operations. This section delineates the architecture of message exchange across onboard modules and external interfaces, emphasizing both intra-system and cloud-integrated communications.

9.2 Intra-System Communication (ROS-Based Architecture)

The onboard communication architecture is implemented using the Robot Operating System (ROS Noetic), which facilitates a modular and distributed system architecture. Individual functional modules—such as perception, mapping, path planning, and control—are deployed as ROS nodes operating asynchronously under a publisher–subscriber paradigm. These nodes share data via ROS topics, with message serialization handled over TCPROS (ROS over TCP/IP).

```
import paho.mqtt.client as mqtt
import json

def on_connect(client, userdata, flags, rc):
    print(f"Connected with result code {rc}")
    client.publish("avp/parking_status", json.dumps({"spot_id": 3, "status": "vacant"}))
```

Fig. 9.1 Intra-System Communication

Each sensor subsystem (e.g., 3D LiDAR, ultrasonic arrays, and depth camera) publishes raw or processed data to dedicated topics. A centralized sensor fusion node subscribes to these topics and aggregates data into a unified environmental model, which is further utilized by the path planning module.

Table-9.1 Example ROS Topics and Their Roles

Topic Name	Message Type	Publisher	Subscriber(s)
/sensor/lidar	PointCloud2	LiDAR Node	Sensor Fusion Node
/sensor/ultrasonic	RangeArray.msg	Ultrasonic Node	Sensor Fusion Node
/sensor/camera	Image.msg	Camera Node	Perception Node

/sensor_fusion	ObstacleMap.msg	Sensor Fusion Node	Planner Node
/path_plan	Trajectory.msg	Planner Node	Controller Node
/vehicle_control	ControlCmd.msg	Controller Node	Motor Driver Node

All messages are structured using custom ROS message formats (e.g., sensor_fusion.msg, control_cmd.msg) that encapsulate structured JSON-style data including obstacle coordinates, free space zones, and kinematic constraints.

```
client = mqtt.Client()
client.on_connect = on_connect
client.connect("broker.hivemq.com", 1883, 60)
client.loop_forever()
```

Fig. 9.2 Structure of Message using custom ROS Message Formats

9.3 Vehicle-to-Infrastructure (V2I) Communication

To facilitate cloud-based services such as parking slot availability monitoring, user reservation, and vehicle identification, the system employs RESTful APIs over secured HTTPS channels. These interfaces bridge the autonomous platform with a central cloud server and a user-accessible mobile application.
Messages exchanged via the API are JSON-encoded and adhere to REST architectural conventions, utilizing HTTP GET and POST methods. Authentication and session validation are integrated using token-based access control to maintain data security.
Example 1: Vehicle Posting Parking Slot Update
POST /api/parking/update
Request Payload: {"slot_id": "F1-A12", "occupied": false, "vehicle_id": "AUTO_007", "timestamp": "2025-04-18T10:32:00Z"}
Example 2: Querying Slot Availability
GET /api/parking/reserve?slot=F1-A12
Response: {"reserved": true, "vehicle_id": "AUTO_007", "expires_in": 570}

9.4 Message Flow Summary

The communication flow within the system can be abstracted into three primary domains:

- Perception to Fusion: Sensor data streams are fused into a coherent world model using ROS topics.
- Fusion to Motion Control: The planner generates trajectories which are published and subscribed to by the vehicle controller.
- Vehicle to Cloud: State updates, occupancy data, and reservation status are pushed to a remote server for remote user access and V2I synchronization.

10. REAL-WORLD IMPLEMENTATIONS

10.1 Detroit Smart Parking Lab

This urban laboratory allows companies to test parking-related technologies, including automated valet parking, in a real-world setting. The facility aims to alleviate parking-related congestion and emissions in urban areas.

10.2 Bosch and Daimler's Driverless Parking Garage

In Stuttgart, Germany, a driverless parking system has been implemented, utilizing sensors like lidar and cameras built into the parking garage to help cars navigate without collisions. Equipped Mercedes-Benz vehicles can use this technology, marking a significant step towards widespread adoption of AVP systems.

10.3 Challenges and Considerations

10.3.1 Complexity of Parking Environments

Parking lots present unique challenges for AVP systems due to unpredictable human behaviors and the lack of consistent rules. Features like Tesla's "Smart Summon" have shown mixed results (Ziegler et al., 2014), highlighting the need for further refinement in AVP technologies.

10.3.2 Infrastructure Requirements

Effective AVP systems often rely on smart infrastructure, including sensors and communication networks, to function optimally. Investments in such infrastructure are crucial for the successful deployment of AVP technologies.

10.4 Infrastructure Requirements

Effective AVP systems often rely on smart infrastructure, including sensors and communication networks, to function optimally. Investments in such infrastructure are crucial for the successful deployment of AVP technologies

CONCLUSION

The proposed Autonomous Valet Parking (AVP) system constitutes a fully integrated, scalable solution tailored for structured, single-floor parking environments. Leveraging a fusion of 3D LiDAR, depth camera, ultrasonic sensors, and wheel encoders, the system ensures robust perception and localization within GPS-denied spaces. The employment of SLAM-based mapping, alongside precise actuator control via PID and MPC strategies, enables high-fidelity trajectory tracking and spatial maneuvering. Ackermann steering geometry enhances the realism and accuracy of vehicular motion, particularly in constrained layouts. Real-time V2I communication, facilitated through a cloud-integrated slot management server and mobile interface, supports dynamic slot allocation and user-initiated retrieval. The successful deployment on a 1:10 scale platform validates the framework's functional integrity, operational efficiency, and potential for future real-world adaptation. This work underscores the practicality of infrastructure-light AVP solutions and their role in advancing intelligent mobility systems.

REFERENCES

[1] Abadi, M., Barham, P., Chen, J., et al., 2016. TensorFlow: A system for large-scale machine learning. USENIX Symposium on Operating Systems Design and Implementation (OSDI).
[2] Åström, K.J., Murray, R.M., 2008. Feedback Systems: An Introduction for Scientists and Engineers. Princeton University Press.
[3] Banks and R. Gupta, "MQTT Version 3.1.1," OASIS Standard, October 2014.
[4] Bosch, 2020. Automated Valet Parking – Driverless Parking in the Parking Garage. [online] Available at: https://www.bosch-mobility.com.
[5] Cadena, C., Carlone, L., Carrillo, H., et al., 2016. Past, Present, and Future of Simultaneous Localization and Mapping: Towards the Robust-Perception Age. IEEE Transactions on Robotics, 32(6), pp.1309–1332.
[6] Chen, J., Zhang, R., Li, C., et al., 2022. Parking Slot Detection Using Deep Neural Networks with LiDAR and Camera Fusion. Sensors, 22(9), pp.3245.
[7] Coulter, R.C., 1992. Implementation of the Pure Pursuit Path Tracking Algorithm. Carnegie Mellon University, CMU-RI-TR-92-01.
[8] Hart, P.E., Nilsson, N.J., Raphael, B., 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics, 4(2), pp.100–107.
[9] He, D., Zhang, W., Zhou, Y., 2018. Reservation-based Smart Parking System using Cloud and IoT. IEEE Access, 6, pp.64084–64094.
[10] Hess, W., Kohler, D., Rapp, H., Andor, D., 2016. Real-Time Loop Closure in 2D LIDAR SLAM. IEEE ICRA, pp.1271–1278.
[11] K. J. Åström and R. M. Murray, "Feedback Systems: An Introduction for Scientists and Engineers", Princeton University Press, 2010
[12] Koenig, N. and Howard, A., 2004. Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator. IEEE/RSJ IROS.
[13] Kuutti, S., Fallah, S., Katsaros, K., et al., 2018. A Survey of the State-of-the-Art Localization Techniques and Their Potentials for Autonomous Vehicle Applications. IEEE Internet of Things Journal, 5(2), pp.829–846.
[14] Labbé, M. and Michaud, F., 2019. RTAB-Map as an Open-Source Lidar and Visual SLAM Library for Large-Scale and Long-Term Online Operation. Journal of Field Robotics, 36(2), pp.416–446.
[15] Levinson, J., Askeland, J., Becker, J., et al., 2011. Towards Fully Autonomous Driving: Systems and Algorithms. IEEE IV.
[16] Liu, S., Feng, Y., Yu, Z., et al., 2019. Intelligent Parking Lot Management with Edge-Assisted V2I Communication. Sensors, 19(3), p.543.
[17] Locke, D., 2010. MQTT V3.1 Protocol Specification. IBM developerWorks Technical Library.
[18] Mouhyemen, A., Jayaraman, S., Hossain, M., 2017. ROS-based Autonomous Vehicle Platform for Learning and Research. Proceedings of the International Conference on Mechatronics.
[19] Olson, E., 2011. AprilTag: A robust and flexible visual fiducial system. IEEE ICRA, pp.3400–3407.
[20] Paden, B., Čáp, M., Yong, S.Z., et al., 2016. A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles. IEEE Transactions on Intelligent Vehicles, 1(1), pp.33–55.
[21] Paszke, A., Gross, S., Massa, F., et al., 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. NeurIPS.
[22] Qin, S.J. and Badgwell, T.A., 2003. A Survey of Industrial Model Predictive Control Technology. Control Engineering Practice, 11(7), pp.733–764.

[23] Quigley, M., Gerkey, B., Conley, K., et al., 2009. ROS: An Open-Source Robot Operating System. ICRA Workshop on Open-Source Software.
[24] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and W. Burgard, "On measuring the accuracy of SLAM algorithms," Autonomous Robots, vol. 27, pp. 387–407, 2009.
[25] Rajamani, R., 2011. Vehicle Dynamics and Control. Springer.
[26] Schwarzenberg, M., Leibold, M., Lutz, P., et al., 2018. System Architecture and Implementation of an Automated Valet Parking System. IEEE IV.
[27] Shoup, D., 2005. The High Cost of Free Parking. APA Planners Press.
[28] Thrun, S., Burgard, W., Fox, D., 2005. Probabilistic Robotics. MIT Press.
[29] Welch, G., and Bishop, G., 2001. An Introduction to the Kalman Filter. UNC-Chapel Hill.
[30] Zhang, J., and Singh, S., 2014. LOAM: Lidar Odometry and Mapping in Real-time. Robotics: Science and Systems.
[31] Zhang, Y., Liu, M., Wang, D., 2017. MPC-based Path Planning for Autonomous Parking. IEEE ICRA.
[32] Ziegler, J., Bender, P., Schreiber, M., et al., 2014. Making Bertha Drive – An Autonomous Journey on a Historic Route. IEEE Intelligent Transportation Systems Magazine, 6(2), pp.8–20.