

Final Report

Team 36: Vision-based Intelligent Robotic System based on NVIDIA Jetson

Affaf Amjad

Ethan Evans

Shashank Ojha

Eric Sass

Rishi Tulsiani

Faculty Advisors: Professor Brian Surgenor and Professor Xian Wang

Due Date: Sunday December 1st

MECH 460: Team Project, Conceive & Design

Department of Mechanical & Materials Engineering




Statement of Originality

Our signatures below attest that this submission is our original work.

Following professional engineering practice, we bear the burden of proof for original work. Therefore, we have read the Departures from Academic Integrity (DFAI) information posted on the [Smith Engineering website](#) and confirm that this work is in accordance with the Policy.

Individual 1:

Signature:  Date: December 1st 2024


Name: Shashank Ojha ID #: 20219797

Individual 2:

Signature:  Date: December 1st 2024

Name: Affaf Amjad ID #: 20223725

Individual 3:

Signature:  Date: December 1st 2024

Name: Eric Sass ID #: 20219616

Individual 4:

Signature: EthanEvans Date: December 1st 2024

Name: Ethan Evans ID #: 20294376

Individual 5:

Signature:  Date: December 1st 2024

Name: Rishi Tulsiani ID #: 20207821

Contents

Table of Figures	iv
Table of Tables	iv
Executive Summary.....	v
Introduction and Background	1
Background Information	1
Hardware Setup	1
Societal, Environmental and Enterprise Considerations	2
Project Overview.....	2
Objectives and Scope	3
Criteria and Requirements.....	3
Assumptions.....	3
Constraints	3
Specifications and Metrics	4
Project Management	4
Project Timeline and Milestones	4
Team Roles and Responsibilities.....	5
Resources and Tools	5
Safety and Regulatory Requirements	6
Fail-safe Mechanisms.....	6
System Monitoring and Hardware Protection.....	6
Design Alternatives and Methodology	6
Computer Vision Approaches	7
Sensor Configuration	7
Image Processing Pipeline	7
Line Detection Methods.....	8
Parking Spot Identification and Validation	8
CNN Approaches	9
Camera Configuration.....	9
Network Architectures.....	9
Dataset Considerations.....	10

Training Strategies	10
Path Planning	10
Control System	11
Selection Methodology	11
Final Design and Implementation	15
System Architecture	15
Software Architecture	15
Image Processing	15
CV Implementation:	16
Parking Line Detection:	16
Entry Point Analysis	17
CNN Implementation	18
Data Collection	18
Training	18
CV vs CNN	18
Vehicle Control	19
UI and Safety Design	20
Testing and Results	20
Testing Methodology	20
Performance Metrics Analysis	21
Computer Vision Virtual Environment	21
Computer Vision Physical Environment	21
Convolutional Neural Network Physical Environment	22
Results	22
Challenges encountered	22
Future Improvements	23
Economic Analysis	23
Quanser QCar	23
Custom teaching tool	25
Component choices	26
Current Recommendations	28

Conclusion.....	30
Appendix	31
Appendix A - Source Evaluation	31

Table of Figures

Figure 1 shows the hardware configuration of the QCar	2
Figure 2 shows the CV method detecting the front and side parking line	17
Figure 3 shows how the curb was isolated using binary thresholding of the curb	17
Figure 4 shows the data annotation process for the 1500 captured images on Rob flow	18

Table of Tables

Table 1: Comparison of key parameters for CSI and RealSense camera	11
Table 2: Weighted Evaluation Matrix for Preprocessing Approaches	12
Table 3: Weighted Evaluation Matrix for Line Extraction Approaches.....	12
Table 4: Weighted Evaluation Matrix for Object Detection CNN Architectures.....	14
Table 5: Weighted Evaluation Matrix for Path Planning Approaches	14
Table 6: Weighted Evaluation Matrix for Control Mechanisms	15
Table 7 shows the WEM comparing the CNN and CV method	19
Table 8 Shows how the CNN and CV methods in the virtual and physical environment correlate to original functional requirements	21
Table 9: Bill of material of QCar Components.	25
Table 11 shows the bill of materials to replicate QCar using Yahboom Rosmaster kit.	29
Table 12 shows the bill of materials for a custom car replicating QCar's sensors.....	29

Executive Summary

This project developed an autonomous parallel parking system for the Quanser QCar platform, a state-of-the-art, scale-model autonomous vehicle designed to provide students and researchers with hands-on experience in developing and validating advanced algorithms. The system integrates both traditional computer vision (CV) techniques and convolutional neural networks (CNNs) to identify parking spaces and execute parallel parking maneuvers autonomously within a controlled environment.

The primary objectives included designing a parking detection system using traditional CV techniques, implementing a CNN-based detection system with MobileNetV2 architecture, and developing a robust control system for path planning and parking execution. Performance metrics included parking accuracy within 5 cm of the target position, a completion time under 15 seconds, and a minimum clearance of 3 cm from obstacles.

The traditional CV implementation, leveraging grayscale conversion, Gaussian blur, and Canny edge detection, achieved parking line detection accuracies of 70% with the front camera and 90% with the right camera. The CNN-based system, trained on an augmented dataset of 3,640 images, demonstrated superior robustness to varying lighting conditions. Testing results showed parking success rates ranging from 67% to 90%, with consistent adherence to safety constraints and minimal human intervention (<15%).

An economic analysis evaluated the feasibility of the Quanser QCar platform versus alternative options for educational tools. While the QCar offers advanced features and compatibility with existing software libraries, its high cost may limit accessibility. The Yahboom Rosmaster R2 ROS2 Robot is proposed as a cost-effective alternative that maintains similar functionality and educational value.

This project demonstrates the viability of integrating advanced CV and deep learning techniques to address real-world challenges in autonomous systems. The findings highlight the comparative strengths of traditional CV and CNN-based approaches, with the latter offering enhanced robustness at the expense of increased computational complexity. The outcomes contribute to advancements in autonomous vehicle technology and provide valuable insights for educators and researchers developing future intelligent systems.

Introduction and Background

Background Information

Parallel parking is a challenging driving maneuver that demands precision and situational awareness. Modern vehicles are increasingly utilizing advanced technologies like cameras and LiDAR (Light Detection and Ranging) sensors to assist drivers during this process [1]. Cameras provide real-time visual feedback, helping drivers monitor blind spots and gauge distances to surrounding objects. LiDAR enhances this capability by emitting laser pulses to create accurate 3D maps of the vehicle's environment, precisely measuring the distance to nearby obstacles [2]. The combination of cameras and LiDAR allows for sophisticated driver-assistance systems that can detect suitable parking spaces, assess their size, and even automate the parking maneuver.

The Quanser QCar is a 1/10th-scale autonomous vehicle platform designed to replicate the functions and challenges of full-sized autonomous cars[2]. It's primary purpose is to support research and education by providing a practical environment for developing, testing, and validating complex algorithms in real-time. Equipped with an array of sensors like cameras, lidar, and IMU, the QCar allows users to explore tasks such as path planning, sensor fusion, perception, and control, enabling realistic hands-on experience in autonomous navigation, decision-making, and control systems [2]. Quanser additionally provides Python APIs that simplify the development of algorithms for controlling the QCar by offering intuitive access to its sensors and actuators. These APIs include modules for hardware control, data acquisition, image processing, and communication, allowing users to focus on higher-level algorithm design without handling low-level hardware details. With detailed documentation and examples, Quanser's Python libraries enable educators and researchers to quickly prototype, test, and deploy autonomous system applications. The scope of this project is to use the QCar platform to explore and develop autonomous parallel parking, in a controlled environment.

Hardware Setup

The vehicle uses four CSI cameras that are configured for 360-degree coverage providing real time image acquisition for the system. The camera is calibrated extrinsically and intrinsically to provide optimal accuracy. Calibration involves using 15 images of a checkerboard in several locations to undistort the image. The design also features motor encoders for motion accuracy, an RGB camera for higher resolution processing, and a virtual environment for enhanced testing.

The chassis consists of one motor to linearly actuate steering and another motor to power the rear wheels on it.

The NVIDIA Jetson provides all the computational CV and ML tasks required for the car and acts as the main motherboard/processor for the unit. It contains an internal CPU and GPU, where libraries, resources and machine learning training can occur. The platform's increased efficiency to handle multiple sensors and CNNs ensures quick real-time operation.

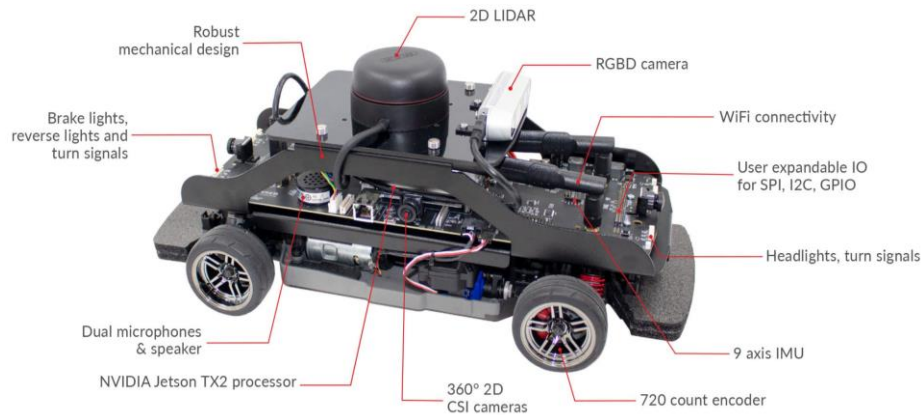


Figure 1 shows the hardware configuration of the QCar

Societal, Environmental and Enterprise Considerations

Many new drivers struggle with "parallel phobia," commonly known as the fear of parallel parking, this anxiety is driven by the pressure of locating a suitable parking spot and maneuvering into it successfully [3] [4]. Autonomous parallel parking systems, however, have the potential to alleviate this phobia by minimizing human error, reducing the risk of hitting the curb or other vehicles, and easing the stress of holding up traffic.

Several automotive companies, including Tesla, BMW, Mercedes, Audi and Ford currently offer active parking assist technologies [5]. Tesla offers features with Level 3 autonomy, allowing it to be summoned and park itself without immediate driver input, though the driver must be ready to take control in certain circumstances. These features have the potential to change infrastructure in environments like grocery stores, shopping malls, and other enterprises where parking is essential. Adoption of such technologies encourages the use of electric vehicles and optimizes parking space, helping enterprises reduce emissions over time. However, it is important to recognize that implementing new infrastructure to support these advancements will require significant upfront investment, which may lead to temporary increases in both costs and emissions.

The target audience for autonomous vehicles with parallel parking capabilities includes individuals with physical disabilities, such as those with impaired vision or limited mobility. Businesses can adopt autonomous drop-off and pickup systems, improving accessibility for individuals who require wheelchair access and the elderly. This opens opportunities for partnerships with organizations advocating for disability rights, automotive manufacturers and technology firms [6].

Project Overview

Autonomous parallel parking is a key challenge in the development of self-driving vehicles, particularly in environments requiring precision and reliability. Current systems often struggle with detecting parking spaces accurately and executing efficient parking within constrained areas. This project focuses on addressing these challenges by developing a vision-based autonomous parallel parking system using the Quanser QCar platform and NVIDIA Jetson hardware. The system is designed for use on a dedicated track, replicating real-world parking scenarios in a controlled, educational environment.

Objectives and Scope

The primary objective is to design an efficient and accurate parallel parking system that integrates both traditional CV and CNNs. The system is developed to function autonomously after manual navigation to the starting point using a gamepad. Upon activation, the system identifies an appropriate parking space and completes the parallel parking process within the dedicated track. The project also includes a comparative analysis of the performance of the CV and CNN-based methods, as well as a Simulink model to simulate the CNN-based system for further evaluation. This dual-system approach provides students with insights into both traditional and advanced methods, offering a comprehensive framework for understanding autonomous navigation systems.

Criteria and Requirements

The system's performance is evaluated based on several key criteria. It must park the QCar accurately within the designated parking space, with the vehicle's center aligning within 5 cm of the center of the parking spot. The parking process should be completed efficiently, within 15 seconds, while minimizing the number of directional adjustments. Smooth operation is a priority, with controlled acceleration and deceleration ensuring reliable performance.

The system operates autonomously after activation, detecting and parking in suitable spaces without requiring human intervention. Safety is critical, as the QCar must maintain a minimum clearance of 3 cm from obstacles on the track. The system must adapt to varying lighting conditions, ensuring consistent performance under different scenarios within the controlled environment. Energy efficiency is also emphasized to align the system's scalability with practical applications.

Assumptions

The system operates in a controlled indoor environment with a dedicated track that provides consistent and repeatable conditions for testing. It assumes static surroundings, with no moving obstacles or unexpected changes in the parking environment. Sensors such as CSI cameras are relied upon for accurate data collection, while the system is expected to perform reliably in well-lit conditions. Although the system is designed to prioritize safety and efficiency, its performance may degrade in less ideal environments, such as those involving dynamic obstacles or adverse lighting conditions. Manual intervention remains possible in case of system faults or emergencies.

Constraints

The project is constrained by the hardware capabilities of the QCar platform. The reliance on CSI cameras for visual input excludes additional sensors such as LiDAR, requiring the algorithms to be optimized for efficient processing. The computational power of the NVIDIA Jetson TX2 imposes further constraints, necessitating lightweight and resource-efficient designs for real-time performance. The QCar's mechanical capabilities, such as steering precision and movement control, are limited by its standard design and actuator response times, which demand careful consideration during the development of detection and control algorithms. Additionally, the project operates exclusively within

the boundaries of the dedicated track, ensuring that all testing and validation are conducted in a consistent environment.

Specifications and Metrics

The list below describes the quality demanded of the final product, along with quantifiable specifications.

- | | |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>1 Park in Designated Spot: The car should park within the designated space, with the final position of the car's centre within 5 cm from the centre of the parking spot.</p> | <p>2 Efficient Parking Process: The parking maneuver should be completed within 15 seconds, with an ideal number of 3 direction changes during the process.</p> |
| <p>3 Smooth Motion During Parking: The car's motion should exhibit smooth acceleration and deceleration, with a maximum speed of 0.5 km/h. Overshoot should be less than 5% (~3.4 cm) of the parking space length.</p> | <p>4 Reliable Performance: The system should maintain a success rate between 67% and 90%, like industry standards for parking systems.</p> |
| <p>5 Minimal Human Intervention: The rate of human intervention should be around 15%, with interventions happening only when necessary to ensure safety.</p> | <p>6 Safe Parking: The car should maintain a minimum clearance of 3 cm from obstacles during the parking maneuver.</p> |
| <p>7 Dynamic Lighting Conditions: Measured using the Light Meter app, with the lowest recorded value being 63 lux and the highest reaching 850 lux in the controlled environment.</p> | <p>8 Effective Traditional CV and CNN Parking: The system should have 70% accuracy in detecting parking spaces for traditional computer vision methods and 90% for CNN algorithms.</p> |

Project Management

Project Timeline and Milestones

The project is being developed in line with a structured timeline to ensure successful implementation and delivery. The timeline is divided into distinct phases, each addressing specific objectives and ensuring progress toward the project's goals.

The first phase, spanning Weeks 1 to 3, was focused on problem definition and preparing the Quality Function Deployment (QFD) document. During this phase, the team identified the project's key objectives and functional requirements while accounting for technical constraints. The QFD document provided a clear framework for guiding the design and implementation processes.

In Weeks 4 to 6, the team conducted hardware validation and baseline performance testing. This phase involved evaluating the Quanser QCar's CSI cameras and motion control systems to ensure they were

operational and capable of supporting real-time detection and parking maneuvers. The tests ensured that the hardware met the foundational requirements for algorithm integration.

Weeks 7 to 9 were dedicated to research and the initial development of detection algorithms. During this stage, preliminary versions of both the traditional CV and CNN systems were created and tested. These prototypes established the foundational framework for further refinement and optimization in subsequent phases.

The final phase, spanning Weeks 10 to 12, focuses on system integration, optimization, and validation. The team is currently refining the detection algorithms to operate in real time on the NVIDIA Jetson TX2 platform. Rigorous testing is being conducted to evaluate system reliability and performance under various conditions. The team is also collaboratively preparing the final report, documenting all aspects of the project, including the design, implementation, and results.

Team Roles and Responsibilities

Ethan Evans and Rishi Tulsiani were responsible for developing and optimizing the CNN-based detection algorithms. Shashank Ojha and Affaf Amjad focused on the implementation of traditional CV-based detection methods. Eric Sass developed and integrated path-planning algorithms. This work ensures that the QCar can execute smooth and precise parallel parking maneuvers by calculating efficient trajectories and optimizing motion control.

The entire team oversees the testing and validation process for both systems. This includes designing testing protocols, evaluating performance metrics, and identifying areas for improvement to ensure the systems meet the required benchmarks.

This distribution of responsibilities ensures that all critical aspects of the project are addressed efficiently, with each team member contributing to the overall success of the product development.

Resources and Tools

The Quanser QCar platform includes all the hardware used for this project, serving an autonomous vehicle designed for academic research and autonomous system development. The QCar is equipped with a wide array of advanced sensors and computational tools, providing a robust framework for implementing and testing autonomous parallel parking systems.

The QCar has a NVIDIA Jetson TX2, a high-performance embedded computer that enables real-time data processing and execution of complex algorithms. This onboard processor is critical for handling tasks such as image processing, path planning, and motion control. The QCar's CSI cameras provide high-resolution visual input for both the traditional CV and CNN-based systems, enabling real-time identification of parallel parking spaces.

The QCar is also equipped with a 2D LiDAR sensor, capable of scanning at resolutions between 2k and 8k at a rate of 10–15 Hz with a range of up to 12 meters. While not directly utilized for this project, the LiDAR sensor adds versatility to the platform, offering potential for future expansions or additional functionality. Other onboard sensors include a 9-axis inertial measurement unit (IMU), which monitors

the vehicle's orientation, velocity, and acceleration, ensuring stable and accurate motion control, and motor encoders that provide precise feedback on wheel movement, aiding in trajectory execution [7].

The project leverages Python as the primary programming language, utilizing its extensive libraries for computer vision and deep learning. For CNN development, PyTorch is employed to build and train models, while the Robot Operating System (ROS) facilitates seamless communication between the QCar's components.

Testing and validation are conducted in a dedicated lab environment equipped with a configurable indoor track. This track simulates a real-world parking scenario, incorporating features such as variable lighting and inconsistently marked parking spaces. This controlled environment allows for consistent and repeatable testing, ensuring thorough evaluation of the detection and control systems under diverse conditions.

Safety and Regulatory Requirements

Fail-safe Mechanisms

In the event of a system failure, the NVIDIA Jetson TX2 QCar is built in with fail-safe measures to ensure safe operation. Expensive parts of the car such as the sensors, motor suspension system, communication linkages, and cameras are all incorporated with built-in checks to monitor their functionality and prevent potential issues. The system is designed to either stop movement or return to a safe state in the event of a failure.

System Monitoring and Hardware Protection

The NVIDIA Jetson TX2 has the following system monitoring and protection:

1. Temperature Monitoring: Qcar has 9-axis IMU and temperature sensor, this will ensure it meets the environmental conditions of being tested indoors with temperatures ranging from 5C to 40C
2. Power Monitoring: If the battery voltage drops below 10.5V, a 'LOW BAT' warning message is displayed on the LCD. It is also equipped with voltage and over-current protection
3. Hardware Protection: Bumper is durable and has shock-resistant casing to protect it from physical damage
4. Data Integrity Checks: If the data is compromised due to noise, the system can stop its operation until the issue is resolved.

Design Alternatives and Methodology

This section outlines the systematic approach taken to generate, evaluate, and select solutions for implementing the parking system on the QCar. The design was broken up into three separate systems: parking space detection approaches for CV and CNN, path planning and control strategies, and safety system integration.

Computer Vision Approaches

The foundation of autonomous parallel parking lies in reliable parking space detection. Traditional computer vision and deep learning approaches each offer distinct advantages and challenges for this task. The computer vision pipeline requires careful consideration of sensor configuration, image processing methods, and geometric analysis techniques.

Sensor Configuration

The QCar platform provides multiple sensing options: CSI cameras mounted on each side, a front-mounted RealSense RGBD camera, and a LIDAR sensor. While LIDAR offers precise distance measurements, parking space detection primarily relies on visual line features, making cameras the more appropriate choice. Initial analysis identified three essential viewing angles: front view for approach alignment, left view for space detection, and rear view for maneuver monitoring.

For front view imaging, both the RealSense RGBD camera and CSI camera present viable options. The RealSense offers superior image quality with built-in distortion correction and depth information, while the CSI camera provides simpler integration and lower computational overhead. This choice significantly impacts the overall system architecture and processing requirements.

Image Processing Pipeline

Several processing pipelines were evaluated, each comprising different combinations of preprocessing, line detection, and space identification methods. Two colour space transform approaches were considered for their potential benefits in parking line detections. The first approach, grayscale conversion, is computationally efficient, making it a good option for real-time processing, but may lose useful colour information that may be essential for line detection under varying lighting. The second approach converts the image to HSV (hue, saturation, value) colour to isolate parking lines based on their colour [8]. The HSV approach may be more robust to changes in lighting but comes with a higher computational overhead compared to grayscale.

Noise reduction methods, such as CLAHE, bilateral filtering, and Gaussian blur, were also evaluated to improve image quality and reduce false detections. The choice between these methods involves balancing processing efficiency, the ability to handle different lighting conditions, and preserve edge information. CLAHE is considered to improve contrast in poorly lit areas and reduce noise to make features more distinguishable [9]. CLAHE was considered as it is particularly useful for dealing with images with uneven lighting which was thought to be beneficial for detecting parking lines, however it adds an extra processing step which may impact the overall efficiency of the pipeline. Bilateral filtering is a noise reduction technique which was considered for its ability to smooth images while maintaining the sharpness of the line edges [10]. This technique would aid with preserving the important features of the parking lines but comes with a computational cost compared to simpler filtering methods. Gaussian blur is a faster alternative for general noise reduction, which effectively reduces high frequency noise, but may also degrade the edges of the parking lines, potentially affecting the accuracy of the line detection stage.

Line Detection Methods

Once the image is filtered and preprocessed, a method to isolate and detect parking lines needed to be developed. Canny edge detection is a commonly used technique for detecting both strong and weak edges and eliminates false positives but requires careful parameter tuning for optimal results [11]. Another method considered was morphological edge detection which is a shape-based approach. This method was considered for its potential to detect parking lines based on their shapes. It uses morphological operations, such as erosion and dilation to identify edges [12]. This approach keeps the image's edge contours intact while suppressing noise but may be unnecessary for identifying parking lines.

After detecting the edges, we needed to design a way to extract lines. Several Hough Transform variations were considered for extracting parking lines from the edge-detected image. The Standard Hough Transform is a complete line detection method that ensures no potential parking lines are missed but comes with higher memory requirements and slower processing times, which may not be suitable for real-time applications [13]. The Probabilistic Hough Transform, on the other hand, offers faster processing and lower memory usage by randomly sampling points from the edge-detected image and finding lines that pass through enough of these points. This method is more efficient than the Standard Hough Transform but may potentially miss some line segments if the sampling is not dense enough. The Progressive Probabilistic Hough Transform was also considered as an adaptive approach for better handling of noise and more efficient processing. However, the implementation of this approach is more complex compared to the standard Probabilistic Hough Transform.

Parking Spot Identification and Validation

To begin the parallel parking maneuver, the potential parking spaces must be identified. The team discussed many ways to possibly achieve this which led to the evaluation of several geometric analysis approaches. Parallel line pairing is a straightforward method that detects parking space boundaries by identifying pairs of parallel lines with a specified distance between them. This method was considered for its simplicity in implementation but may be sensitive to missing or partially occluded lines. Multiple line clustering groups detected lines based on their spatial proximity and orientation. It is more robust to broken or discontinuous lines but is more computationally complex and may require additional post-processing to refine the detected parking spaces.

When working with images of the real-world to detect features on a 2D plane, a challenge arises from perspective distortion due to the way coordinates are projected. This makes it more difficult to detect and analyse features with computer vision. Inverse Perspective Mapping (IPM), a technique that transforms the camera view into a top-down, bird's eye view of the parking area, simplifies the process of detecting parking spaces by removing perspective distortion [14]. IPM can be combined with any of the detection strategies discussed to improve their accuracy and robustness, however, it requires extensive knowledge of the camera's parameters and can be often difficult to implement. Simpler space identification methods offer faster processing times but may be less reliable in complex parking scenarios, while more sophisticated techniques can improve detection accuracy but at the cost of increased computational costs and complexity.

Various validation methods were considered to reduce false positives and ensure the detected parking spaces are suitable for the vehicle. Dimension-based validation verifies that the detected parking space satisfies the minimum size requirements for the vehicle, which is straightforward to implement but relies on accurate camera calibration to estimate real-world dimensions of the parking space.

Orientation-based validation, which checks that the detected parking space is aligned with the expected parking angle, is useful for filtering out spaces detected at incorrect orientations but may require perspective correction to account for the camera's viewing angle. Rule-based validation, which combines multiple criteria such as dimension, orientation, and line angle to determine the validity of a detected parking space, can filter out false positives that satisfy some but not all the required criteria. However, designing an effective rule set may require extensive tuning and may not generalize well to different parking scenarios. Filtering by angle and length, a simple yet effective method for removing line segments that are unlikely to be part of a parking space boundary, can quickly eliminate lines that are too short, too long, or oriented at improbable angles for a parking space. This filtering can be applied as a pre-processing step to reduce the computational burden on subsequent space detection and validation stages.

CNN Approaches

Continuing with the CNN approach for parking space detection, we explored various aspects of the system, including camera configurations, network architectures, dataset collection strategies, and training methods. Each of these components plays a crucial role in developing an accurate and efficient CNN-based detection system.

Camera Configuration

For the CNN approach to parking space detection, both single-camera and multi-camera configurations were considered. Single-camera setups, such as using only the front or left camera, offer a simpler implementation but have limitations in terms of view range and context. Multi-camera configurations, like combining front and left cameras or front, left, and rear cameras, provide better spatial awareness and coverage but come with increased processing requirements and complexity in data fusion. The choice between these configurations involves a trade-off between simplicity and comprehensiveness, with multi-camera setups offering more information at the cost of higher computational demands.

Network Architectures

When exploring network architectures, we initially compared the use of segmentation and object detection approaches. Segmentation networks, such as U-Net or Mask R-CNN, provide pixel-wise classification, which can be useful for precise boundary delineation of parking spaces. However, we ultimately decided to focus on object detection networks, as they are better suited for identifying and localizing parking spaces as distinct objects within the image. Among the object detection architectures, we considered several popular choices, including YOLO (You Only Look Once), EfficientNet, ResNet, MobileNet, and Inception. Each of these architectures has its own strengths and characteristics. YOLO is known for its fast inference speed, making it suitable for real-time detection. EfficientNet offers a good balance between accuracy and efficiency, with its compound scaling method. ResNet provides a deep and residual learning framework, enabling the training of very deep networks. MobileNet is designed for

mobile and embedded devices, focusing on lightweight and efficient models. Inception introduces a module that performs convolutions at different scales and aggregates the results, enhancing the network's ability to capture multi-scale features.

Dataset Considerations

To fine-tune the chosen CNN model for parking space detection, a suitable dataset is required. Different dataset collection approaches were explored, each with its own advantages and challenges. Gathering real-world data by capturing images of various parking scenarios ensures that the dataset represents realistic conditions and variations but can be time-consuming and labor-intensive. Generating synthetic data using simulation tools offers the advantage of precise annotations but may face challenges when applied to real-world due to the difference between simulated and real environments. A hybrid approach that combines both real-world and synthetic data can help improve the model's generalization ability and robustness to different conditions.

Training Strategies

Once a suitable dataset is obtained, the next step is to train the CNN model. Different training methods were explored to optimize the model's performance. Transfer learning, where a pre-trained model is fine-tuned for the specific task of parking space detection, can lead to faster convergence and requires less training data compared to training from scratch. Training the model from scratch allows for more specialization and optimization for the target task but requires a larger dataset and longer training time. Progressive training strategies, where the model's complexity is gradually increased, enable better parameter optimization and more controlled development. The choice of training method involves a trade-off between development complexity, dataset size, and available resources.

Path Planning

After a potential parking spot is validated, the optimal path for the car to take to maneuver into the spot must be determined. When considering path planning systems for the autonomous parking task, we explored various approaches ranging from geometric methods to optimization-based techniques. Each approach has its own strengths, limitations, and trade-offs that need to be carefully evaluated in the context of our specific requirements and constraints. Geometric approaches like fixed multi-point paths and circle-based trajectories are straightforward to implement and computationally efficient but may lack adaptability and smoothness. Search-based methods such as A* and Rapidly exploring Random Trees (RRT/RRT*) were considered for their completeness and ability to handle complex environments but come with computational complexity and optimality trade-offs. Model Predictive Path Planning (MPPP) optimizes multiple objectives while considering vehicle dynamics and constraints but has high computational requirements.

Integration considerations play a crucial role in path planning selection. Complete pre-planning simplifies implementation but limits adaptability, while progressive planning allows for better adaptability at the cost of computational load and complexity. Incorporating vehicle constraints, such as kinematic and dynamic limitations, is essential for generating safe and feasible paths. The operating

environment also influences the choice between one-time planning for static scenarios and real-time adjustment for dynamic environments.

Control System

Several control techniques were explored to ensure accurate and reliable trajectory execution. Proportional-Integral-Derivative (PID) control is widely used for control but may not be optimal for all conditions and lacks consideration of vehicle dynamics. Pure Pursuit, a geometric path tracking algorithm, is computationally efficient and provides smooth steering control but relies on geometric calculations and does not consider vehicle dynamics or constraints. Model Predictive Control (MPC) optimizes control commands over a finite horizon while considering vehicle dynamics, constraints, and future predictions. MPC offers a flexible and adaptive control framework capable of handling complex system dynamics and objectives but comes with higher computational requirements and implementation complexity.

Selection Methodology

The development of an autonomous parallel parking system requires a systematic approach to evaluate and select appropriate solutions across multiple subsystems. The methodology follows a structured evaluation process consisting of four key phases: requirement analysis, solution identification, comparative evaluation, and integration validation. Each phase employs specific evaluation metrics and testing procedures to ensure optimal component selection while maintaining system cohesion.

The requirement analysis phase established performance metrics including real-time processing capabilities, computational resource constraints, and system reliability requirements. These metrics formed the foundation for our weighted evaluation matrices and guided the selection process across all subsystems.

The selection of appropriate sensors forms the foundation of our autonomous parking system, requiring careful consideration of coverage requirements and processing overhead. The decision to exclude LIDAR emerged from analysis of the parking detection task, where visual features are sufficient and the added complexity of sensor fusion outweighed potential benefits. For front view imaging, we compared the RealSense RGBD camera and CSI camera as shown in Table 1.

Table 1: Comparison of key parameters for CSI and RealSense camera

Criterion	RealSense	CSI Camera	Notes
Image Quality	High	Medium	Both sufficient
Processing Requirements	High	Low	Critical factor
Field of View	69.4°	160°	Wider FOV better
Integration Complexity	Medium	Low	CSI preferred

The selection of CSI cameras throughout the system was driven by our real-time processing requirements and the need for simplified integration. Basic line detection testing confirmed both

cameras could achieve the necessary performance, making the CSI camera's lower computational overhead and wider field of view the deciding factors. The final configuration of 820x410 at 30 FPS was selected to balance detail and processing speed.

The development of our image processing pipeline required balancing processing efficiency, detection accuracy, and environmental robustness. For colour space selection, while HSV offers superior lighting robustness, grayscale conversion provides sufficient detail for high-contrast parking lines while maintaining essential real-time performance. To improve image quality and reduce false detections, several preprocessing methods were evaluated using weighted criteria as shown in Table 2.

Table 2: Weighted Evaluation Matrix for Preprocessing Approaches

Criterion	Weight	Gaussian	Bilateral	CLAHE
Processing Speed	30%	5	2	3
Noise Reduction	25%	4	5	3
Edge Preservation	25%	3	5	4
Implementation Complexity	20%	5	2	3
Weighted Total	100%	4.3	3.5	3.3

Gaussian blur was selected for its optimal balance between noise reduction and computational efficiency, with its minor impact on edge quality deemed acceptable given subsequent processing steps. For effective parking line identification, edge detection methods were evaluated based on their strengths and weaknesses. Canny edge detection was chosen over morphological edge detection for its simplicity and superior ability to detect both strong and weak edges. Morphological edge detection was deemed unnecessary for its complex implementation of shape-based identification. To extract lines from the edge-detected image, we compared Hough Transform variations using a weighted evaluation matrix (WEM) shown in Table 3.

Table 3: Weighted Evaluation Matrix for Line Extraction Approaches

Criterion	Weight	Prob. Hough	Standard Hough	Prog. Prob. Hough
Processing Efficiency	30%	4	2	3
Detection Accuracy	25%	4	5	4
Memory Usage	20%	4	2	3
Implementation Complexity	15%	4	4	2
Adaptability	10%	3	4	5
Weighted Total	100%	3.90	3.25	3.35

The Probabilistic Hough Transform was selected for its efficient memory usage and processing speed while maintaining adequate line detection quality.

The final pipeline employs grayscale conversion, Gaussian blur, Canny edge detection, and Probabilistic Hough Transform, selected primarily for computational efficiency while maintaining reliable line detection capability.

For reliable parking space identification, we evaluated various geometric analysis approaches and validation methods. To address perspective distortion in the camera view, both Inverse Perspective Mapping (IPM) and direct analysis approaches were considered. While IPM offers superior accuracy in spatial measurements, its complex implementation and additional processing overhead led us to select direct analysis, which provides sufficient accuracy for our application while maintaining computational efficiency.

The evaluation of geometric detection strategies focused on three main approaches. Parallel line pairing offered a straightforward method with low computational cost, particularly effective for structured parking spaces. Multi-line clustering demonstrated superior handling of broken or discontinuous lines but at the cost of significant computational overhead. Intersection analysis provided an alternative approach but proved less reliable for parallel parking scenarios. Given our specific requirements, parallel line pairing emerged as the optimal choice, offering the best balance between reliability and computational efficiency.

For validation methods, we considered three complementary approaches to minimize false positives. Dimension-based validation offered straightforward implementation for verifying space size, while orientation-based validation provided important angular verification despite some perspective sensitivity. Rule-based validation, incorporating multiple criteria, was considered but ultimately rejected due to its complex parameter tuning requirements. The final detection pipeline combines dimension and orientation validation with parallel line pairing, creating a robust system that maintains computational efficiency while ensuring reliable parking space detection.

The selection of an appropriate CNN architecture required careful consideration of the NVIDIA Jetson's capabilities and our real-time processing requirements. Our evaluation focused primarily on architectures optimized for embedded systems while maintaining adequate detection accuracy. Key architectures were evaluated using a weighted evaluation matrix shown in Table 4 which emphasized deployment feasibility on the Jetson platform.

Table 4: Weighted Evaluation Matrix for Object Detection CNN Architectures

Criterion	Weight	MobileNet	YOLO	EfficientNet	ResNet
Inference Speed	30%	5	4	3	2
Memory Efficiency	25%	5	3	4	2
Jetson Optimization	20%	4	4	3	3
Detection Accuracy	15%	4	5	5	5
Implementation Complexity	10%	5	3	3	3
Weighted Total	100%	4.65	3.85	3.60	2.85

MobileNet emerged as the optimal choice through our systematic evaluation process. Its architecture, specifically designed for mobile and embedded applications, provides several key advantages that align perfectly with our system requirements. The use of depth-wise separable convolutions significantly reduces computational overhead while maintaining competitive accuracy. This efficiency is particularly valuable given our real-time processing requirements and the limited computational resources available on the Jetson platform.

In evaluating path planning approaches, our focus was on balancing computational efficiency with path quality for structured parking environments. Three main categories were assessed against critical system requirements shown in Table 5.

Table 5: Weighted Evaluation Matrix for Path Planning Approaches

Criterion	Weight	Geometric	Search-based (A*/RRT)	Model Predictive
Computational Efficiency	30%	5	3	2
Implementation Complexity	25%	5	2	1
System Integration	20%	4	3	2
Path Adaptability	15%	2	5	5
Memory Usage	10%	5	3	2
Weighted Total	100%	4.35	3.20	2.15

The geometric approach was selected due to its computational efficiency and simplicity, with its limited adaptability deemed acceptable for structured parking scenarios. This choice was further supported by our static environment assumptions and the ability to pre-plan complete trajectories. For trajectory execution, control methods were evaluated with emphasis on implementation practicality and system integration using the WEM seen below in Table 6.

Table 6: Weighted Evaluation Matrix for Control Mechanisms

Criterion	Weight	PID	Pure Pursuit	MPC
Implementation Simplicity	30%	5	3	1
Computational Efficiency	25%	5	4	2
Integration Effort	20%	5	3	2
Control Performance	15%	3	4	5
Parameter Tuning	10%	4	3	2
Weighted Total	100%	4.65	3.45	2.15

PID control emerged as the most suitable choice despite not offering the most sophisticated control capabilities. Its straightforward implementation, minimal computational requirements, and ease of integration align perfectly with our system needs. Given the structured nature of parking maneuvers and our geometric path planning approach, PID control provides sufficient performance while maintaining system simplicity.

Final Design and Implementation

System Architecture

The implementation of the autonomous parking system uses both hardware and software components to develop the final product. Each part of the architecture fulfills the specific requirements set by the client.

Software Architecture

The software employs a modular operating system using the Robot Operating System (ROS). ROS handles functions such as data acquisition, path planning and motion controls to ensure seamless data flow. The key nodes planned for the communication protocols include:

- **Perception Node:** Used to process data from the cameras to identify parking spaces
- **Path Planning Node:** Computed the trajectory needed for motion control
- **Control Node:** Executes the planned path by managing wheel actuation

Data flows from the sensors to the processing units where the raw inputs undergo preprocessing. The data is then passed through the CV and ML algorithms to detect parking spots. Control signals are used to move the vehicle to the intended target. Finally, results are relayed to path planning and control modules to create a closed loop system, where error is accounted for.

Image Processing

The car will achieve parallel parking using two dynamic methods: Traditional CV and CNN methods. The CV method utilizes basic pattern and image recognition to filter and detect the parking spot. This method is reliant on a dynamic subset of filters extracting the parking lines.

The CNN method trains the model with thousands of images to detect the spot, using various layers of pattern recognition. The data is split into a training model - where images are used to teach the model, and a testing model - where the images are tested to validate its accuracy. The following are the main parameters for image processing:

1. **Parking lines:** Horizontal white lines used to identify the parking spot
2. **Parking Spot:** Consists of the image detecting two distinct parking spots a certain distance apart
3. **Entry Point:** Consists of the direction that the car should enter to execute parallel parking

CV Implementation:

Parking Line Detection:

The pipeline for the visual system utilizes a customizable function where the location, size, and angle of the parking spot can be isolated. The following steps were taken to create the CV pipeline:

1. **Live Image Collection:** The front and left CSI camera must collect the live image
2. **Grayscale conversion:** The live image is converted to grayscale to reduce processing power by removing colour information.
3. **Smoothing:** Gaussian blur is applied to the image to remove high frequency noise that may interfere with edge detection.
4. **Edge Detection:** The Canny edge detection method identifies significant intensity changes to locate potential edges.
5. **Region of Interest (ROI):** The ROI is defined to focus on parking lines, which are extracted and masked for analysis.
6. **Line isolation:** The Hough Transform method is used to extract the parking line based off its size and angle. The right CSI camera identifies lines within the 30-to-180-degree angle, and the front CSI camera identifies lines within a 0-to-5-degree angle.
7. **Noise reduction:** To finally limit noise and remove unintended lines, a cropping line was created in the image. This can be increased in size and location to take away unintended lines such as the wires and wheels of the car as seen in [Figure 2].

After the lines are detected, a green line is drawn for visual representation and a dot is drawn on the edge of the line to identify a point for the car to drive towards as seen in [Figure 2]. The results found around 70% accuracy in detecting the parking lines with the front camera and 90% accuracy in detecting the lines with the right camera. The deviation in accuracy was a result of changing the lighting conditions, which is a major constraint with the CV method. However, the method requires minimal computational power and complexity thus allowing for more efficient parking maneuvers.



Figure 2 shows the CV method detecting the front and side parking line

Entry Point Analysis

The entry point to align with the parking spot is derived by using the curb of the field. The car will lane follow the curb using the RGB camera, until it is at the appropriate location and detects the parking spot. The curb is followed by finding the slope and intercept from the camera to the curb using the following equation:

$$\text{steering} = \text{factor} * (\text{slope}_{\text{binary image}} - \text{adjustment value}) + \text{distance from curb}$$

The equation changes the steering accordingly to provide accurate curb following.

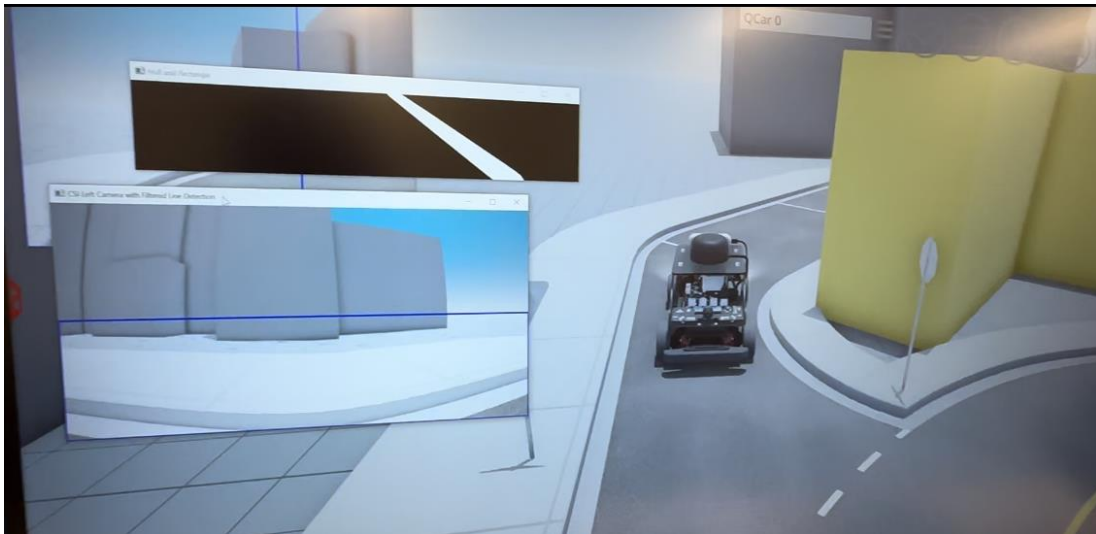


Figure 3 shows how the curb was isolated using binary thresholding of the curb

CNN Implementation

Data Collection

To train the CNN model, data must first be collected. The process begins with a data collection phase, where 1500 images are captured with the front CSI camera at an 820x410 resolution. The data is then annotated manually on Robo flow to identify the entry point, parking spot and parking lines as seen in [Figure 4].

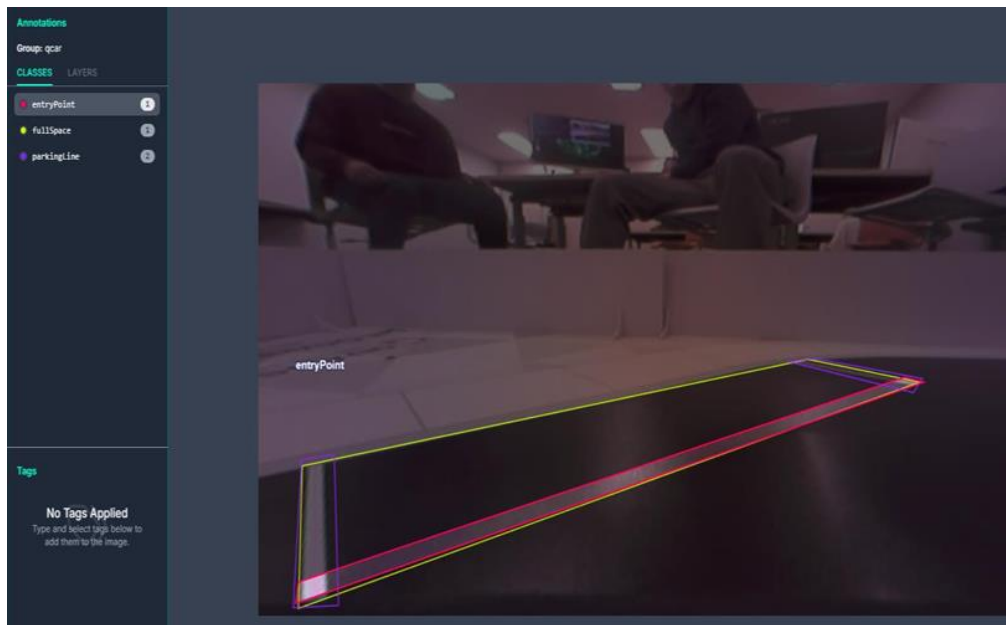


Figure 4 shows the data annotation process for the 1500 captured images on Robo flow

Training

The MobileNetV2 model was fine-tuned by retraining it on a custom dataset. The dataset was augmented to include 3,640 images through techniques such as flipping, tilting, adjusting brightness, and adding noise. The training process was implemented using PyTorch, with a batch size of 32 over 100 epochs, balancing memory efficiency and performance. After training, the model was converted to ONNX format and optimized with TensorRT, reducing latency and enabling real-time image processing. This approach resulted in a reliable model capable of identifying parking spots with high accuracy. Additionally, the model demonstrated robustness and adaptability to various lighting conditions and noise levels encountered in real-world scenarios.

CV vs CNN

The CV and CNN method both provided a dynamic range of benefits and drawbacks. The main advantages of the CNN method include its robustness and adaptability, as it can park in a dynamic range of parking environments and lighting conditions. Some disadvantages include its complexity, and time-intensive training. As the course is designed for students to complete in a semester, training a neural network may be too complex.

The CV method's main pros include its simplicity, low computational usage, and transparency. Since the outputs can be tuned faster and are easier to debug, it is easier to achieve parallel parking when time is sensitive. The main disadvantages for the CV method include its limited scope, environment sensitivity, and labor-intensive calibration. Although it is simpler and easier to tune, the method requires a significant amount of time tuning and filtering noise to make it effective. Furthermore, the CV method is extremely sensitive to light.

The following Weighted Evaluation Matrix (WEM) analyzes which method is more optimal for the scope of the project, using the functional requirements defined by the client:

Table 7 shows the WEM comparing the CNN and CV method

Criterion	Weight	CNN Score	CV Score	Explanation
Accuracy	30%	4.5	3.5	CNNs achieve high accuracy, whereas CV struggles in adverse lighting conditions.
Real-Time Performance	25%	3.6	5	CV's lower computational load ensures better real-time performance.
Robustness	20%	4.5	3	CNNs handle varying conditions better due to data-driven learning.
Implementation Effort	15%	3.5	4.5	CV requires simpler implementation; CNN demands significant data preparation.
Scalability	10%	4	3	CNN models can generalize better to new scenarios after retraining.
Weighted Total	100%	4.1	3.9	

As visible in Table 7, the CNN method outperforms the CV method marginally. However, due to its robust accuracy, the CNN method is preferred over the CV method.

Vehicle Control

Utilizing the CNN and CV method with path planning, the car can successfully execute the parking maneuver using the following steps:

1. The car is manually driven by the user using a Logitech joystick until close to the parking zone.
2. Once in the parking region, lane following will occur until the QCar can confidently detect the parking spot.
3. The car will follow the path of the entry point until it passes the first and second parking line
4. Once the right CSI camera is aligned with the second line, the car will turn backwards to the right until the center of the parking line is around a 45-degree angle from the car.
5. The car will then turn backwards to the left, until the front and back CSI cameras are aligned with both the front and back lines.
6. The car will adjust its throttle until it is evenly spaced between the regions of the parking lines.

These methods are used both in the CNN and CV methods to ensure that the car effectively parks in the intended space. The control system was implemented using a state machine where state . The state-based architect allowed for simple debugging and optimization of each process of parallel parking. Additionally, each state of the parking can further be optimized in the future by other developers that continue the project.

UI and Safety Design

As the QCar has an estimated value exceeding \$20,000, it was integral that a safety plan was used to reduce damage to the field, user and car. The vehicle always had a manual joystick override through the to ensure that the autonomous period can always be cancelled. A series of processes were created to ensure that safety was ensured while executing tasks:

1. **Virtual environment:** Prior to code being tested on the physical car, it was tested on a virtual environment. This ensured that the user could predict the actions of the car prior to field testing. The car must perform the intended action at least 10 times in a row prior to testing physically.
2. **Jack Stand Usage:** A jack stand was used to test and tune the throttle prior to entering the field. The stand consists of propping the car so that the wheels do not touch the car. This allows the user to test various speeds of the car and ensure that the throttle coefficient is slow enough to manage.
3. **Cable and cord management:** A 40ft HDMI cable was carefully routed to ensure that there is no strain on the Nvidia motherboard, while still allowing the user to monitor car performance.
4. **Kill Switch:** A button was added on the joystick to ensure that the code can be disrupted and stopped prior to testing
5. **Buddy system:** The team ensured that there was always a tester and a spotter to ensure that the car performance can be monitored.

Testing and Results

Testing Methodology

The table below presents three testing methods: the QCar in a virtual environment using computer vision, the physical QCar on the track using computer vision, and the physical QCar on the track using a Convolutional Neural Network (CNN). The testing methodology was based on the performance metrics, including detection accuracy, parking success rate, time efficiency, and safety compliance. These metrics were developed to align closely to the engineering specifications; therefore, the specifications will be used in the analysis below to determine the system's success. This is shown by the red ticks and crosses as shown in [Table 8].

Table 8 Shows how the CNN and CV methods in the virtual and physical environment correlate to original functional requirements

Engineering Specifications	Virtual Computer Vision	Physical Computer Vision	Physical CNN
Park in Designated Spot	Parking position of Qcar is within 5 cm from centre of parking area	Parking position of Qcar is within 5 cm from centre of parking area	Parking position of Qcar is within 5 cm from centre of parking area
	✓	✓	✓
Efficient Parking Process	3 direction changes	3 direction changes	3 direction changes
	✓	✓	✓
Smooth Motion During Parking	Max speed 0.5 km/h	Max speed 0.5 km/h	Max speed 0.5 km/h
	✓	✓	✓
Reliable Performance	Success Rate: 70%	Success Rate: 60%	Success Rate: 84%
	✓	×	✓
Minimal Human Intervention	Human Control: 10%	Human Control: 15%	Human Control: 5%
	✓	✓	✓
Safe Parking	Min clearance: 3 cm	Min clearance: 3 cm	Min clearance: 3 cm
	✓	✓	✓

Performance Metrics Analysis

Computer Vision Virtual Environment

The car requires three directional changes to successfully execute the parallel parking mechanism. Although the car in the virtual environment can park in three directional changes, adding a PID control could improve its movement between states. Currently, the system is programmed based on time and the three motion states, making it a less reliable method for conducting parking. It uses only the RGB and Left CSI cameras to detect the parking lines and then performs the time-based parking. However, this approach has a drawback: it cannot adjust to minimize the errors between the desired and actual positions. Instead, it just relies on camera detection without correcting for any inaccuracies in its angle or position.

The virtual environment only has a single lighting condition; therefore, the canny algorithm was able to detect all edges accurately. For curb detection, a fixed range of white RGB values was used to identify the white color of the lane. However, this does not meet the engineering requirements as the system needs to respond to various lighting conditions.

Computer Vision Physical Environment

The CV method in the physical environment is less accurate than the CNN method because of the differences between how they process data. The CV method relies on real-time data from the Left CSI

and RGB cameras to determine the car's position relative to reference coordinates in the frame. This makes it more likely for errors to arise due to any changes in lighting, camera angles, or noise. Overall, the CV method depends heavily on fixed algorithms, which don't work as well in dynamic environments. The difference between the two methods displays the variation in performance accuracy.

The room's lighting intensity was tested from bright to dim, the results showed that Canny detection could locate the parking lines, but the detection became less accurate as the car moved around the track. The dim lighting of the room caused more noise, which caused gaps in the line detection. The maximum speed of the vehicle was determined by the throttle variables. When backing into a parking spot, the throttle value was set higher than when aligning the car parallel to the line. Finally, the system must be started and occasionally monitored by a human to ensure it functions safely so that it does not damage other equipment or any internal hardware.

Convolutional Neural Network Physical Environment

As outlined in the Implementation section above, the initial three main classes are "Entry Point", "Parking Line", and "Entry Space." The data training results for the CNN method show varying performance across the different classes.

During the initial implementation, the mean average precision (mAP) is 0.5509, indicating moderate overall accuracy for all classes. The Entry Point class has the highest mAP at 0.5318; however, the confidence scores are quite low, with nearly all detections falling below 50%. This suggests that the model struggles to detect Entry Points with good confidence. In contrast, the Full Space class has a low mAP of 0.3324 and even lower confidence scores than the Entry Point class, indicating significant challenges in detecting these objects. The Parking Line class has the best detection performance, with an mAP of 0.7885, despite low confidence for most detections. These results underscore the need for improvements in the model's training process. This can be done by annotating more images or training the data for longer to improve overall accuracy.

After retraining the model with a focus solely on parking lines, detection accuracy improved to around 90%. With the new confidence scores, the CNN method was able to preform the parallel parking maneuver. Using the CNN for only one classification, the model was able to preform the parking manoeuvre with a high level of accuracy. However, the maneuver does not currently integrate an entry point – which could be refined in further iterations for more autonomy.

Results

Challenges encountered

Though the NVIDIA Jetson TX2 has a powerful processor, it still has a hard time producing and processing high-resolution images or running complex neural networks in parallel. There are also issues with calibrating the sensors for appropriate environmental mapping, this issue makes it hard to integrate the sensor and camera to behave in conjunction with another. In addition to this, the lack of

calibration can also make it hard to undistort the camera frame. Another challenge is the variation in lighting conditions and sensor noise, this can impact the reliability of the computer vision algorithms.

For the CNN method, the model should have been trained on diverse sets of data, such as various lighting conditions, different types of parking spaces, and parking angles. Lastly, the steering angles and movement when executing the autonomous parking maneuver required a lot of fine tuning of the control system when backing up into the parking space with smooth motion.

Future Improvements

To further improve the model in the following semester, the functional requirements given by the clients were benchmarked to the performance of the car. The following improvements could be made:

Fully Autonomous Parking

The parking provided still requires human input. As cars are trending towards fully autonomous driving, the next step would be to create a localization algorithm on the car, where the car can start at any point on the field, lane follow to the required area, find the parking area, and execute the parking maneuvers derived in the project.

Improved Annotations

As 1500 images are annotated by 5 members, the model was confused at times due to discrepancies of annotations. To solve this problem, it is recommended that annotation time is accounted for in the timeline and done by one member to minimize human error and discrepancies.

Improved Entry Point Analysis

During the CV method, the QCAR followed the path of the curb to perfectly align at the correct parking spot. However, this method causes deviations in areas where the curb is significantly non-linear. To further improve the computer vision model, other methods such as binding a full parking space when two lines are detected, or refining the entry point model for the CNN approach could be considered to make the model more autonomous.

PATH Planning Execution

The path planning model was never fully tuned during the project, which caused the car to be less dynamic in the various ways that it can park. To improve the model, it is recommended that the predecessors of the project spend more resources on detecting the path and mitigating errors.

Economic Analysis

Quanser QCar

As artificial intelligence and autonomous systems rapidly evolve, educators at Queens need improved tools that bridge theoretical knowledge and practical application. Robotics platforms like Quanser's QCar provide academia with important resources that offer students hands-on experience with machine learning algorithms and sensor integration while they solve real-world problems. These tools deepen the

understanding of complicated concepts and prepare future engineers and scientists to tackle the challenges of an increasingly automated world.

Quanser's QCar advances education and research in robotics and AI and autonomous systems as a state-of-the-art, scale-model autonomous vehicle. Its primary purpose is to provide a practical platform for students and researchers to develop, test, and validate algorithms related to computer vision, navigation, and control in real-world scenarios. The QCar simulates the complications of full-scale autonomous vehicles and bridges the gap between theory and practice, so it improves learning outcomes and promotes innovation in autonomous technology. The key components include:

- **Processing Unit:** The QCar is powered by an NVIDIA Jetson TX2 supercomputer, offering exceptional speed and power efficiency for running complex algorithms and processing sensor data in real-time. This costs about \$880 [15].
- **LiDAR Sensor:** The QCar utilizes the RP-LiDAR A2, a 2D planar LiDAR capable of generating up to 8,000 samples per second with a range of 18 meters, ideal for environmental mapping and obstacle detection. This costs about \$460 [16].
- **Depth Camera:** An Intel RealSense D435 RGB-D camera provides both depth and RGB imaging, essential for object detection, scene reconstruction, and navigation tasks. This costs about \$510 [17].
- **360° Vision System:** The platform features four CSI cameras, each equipped with wide-angle lenses offering approximately 160° horizontal and 120° vertical fields of view. This configuration ensures comprehensive visual coverage, used for tasks like parallel parking. These cost less than \$20 each [18].
- **Inertial Measurement Unit (IMU):** A 9-axis IMU comprising an accelerometer, gyroscope, and magnetometer enables accurate motion tracking and orientation estimation, crucial for navigation and control algorithms. This costs less than \$50 [19], however the exact model is not released by Quanser.
- **Drive Motor:** The motor that drives the car is 21W, and costs about \$25 [20], however the exact model is not released by Quanser.
- **Motor Encoder:** High-resolution encoders are attached to the motor to provide precise measurements of rotation, facilitating accurate odometry calculations for position tracking. This costs less than \$60 [21].
- **Communication Interfaces:** The platform includes Wi-Fi and Bluetooth modules to support wireless communication, enabling remote control, data transmission, and integration with other devices or networks. Similar systems cost less than \$100 [22].
- **Power System:** A robust power management system, including high-capacity batteries, ensures stable and reliable operation of all components. The battery costs about \$100 [23], the charger costs \$100 [24].
- **Additional components** like turn signals, the servo motor for steering, and microphones are a negligible cost.

The QCar costs approximately \$20,000 to purchase, which is significantly higher than the sum of these components, plus the chassis which houses everything and allows the car to drive. The APIs that

Quanser provides significantly contribute to the purchase price. A custom PCB is also included to support expanded I/O capabilities. The costs of these components are difficult to estimate, since they are dependant on the volume of cars being produced; they will be left out of the bill of materials (BOM), however \$300 of miscellaneous expenses has been added to account for this. A BOM containing QCar Components is shown below in [Table 9].

Table 9: Bill of material of QCar Components.

Part #	Description	Qty.	Unit Cost
1	Jetson TX2	1	\$880
2	RP-LiDAR A2	1	\$460
3	Intel RealSense D435 RGB-D camera	1	\$510
4	CSI Camera	4	\$20
5	IMU	1	\$50
6	Drive Motor	1	\$25
7	Motor Encoder	1	\$60
8	Communication Interface	1	\$100
9	Power System	1	\$200
10	Miscellaneous Items	1	\$300
Total			\$2,665

Custom teaching tool

It is desired to create a budget-friendly autonomous vehicle platform because of the high purchase price of the QCar. The \$20,000 price makes it impractical for widespread use within our programs. While the QCar is a fantastic tool, its cost limits how many units can be afforded, which can restrict access for faculty members who want to use it for teaching or research. By designing a more affordable alternative, we can ensure our faculty have the tools they need to teach cutting-edge concepts and conduct innovative research without being constrained by budget limitations.

It is crucial to use the same sensors as the QCar to take full advantage of its existing Python libraries. These libraries make it easier to implement algorithms and run experiments without having to develop custom software from scratch. By utilizing the same sensor suite—such as the Intel RealSense D435, RP-LiDAR, and 360° CSI camera suite - we can ensure the platform is fully compatible with these tools. This compatibility will save time and effort for faculty, letting them focus on teaching and research rather than troubleshooting or writing new code.

Component choices

The NVIDIA Jetson platform, such as the Jetson TX2, stands out compared to alternatives like Raspberry Pis and Arduinos due to its superior computational power, AI-focused hardware, and energy-efficient design. While Raspberry Pis are excellent for general-purpose applications and Arduino boards excel in low-level hardware control, neither is well-suited for the intensive demands of autonomous systems. The Jetson TX2 features a Pascal GPU with CUDA cores, enabling real-time AI processing for tasks like object detection, path planning, and sensor fusion—capabilities that Raspberry Pis and Arduinos lack due to limited computational resources and absence of GPU acceleration. Additionally, the Jetson provides extensive support for AI frameworks like TensorFlow and PyTorch, which are essential for autonomous vehicle algorithms, whereas Raspberry Pis often struggle with such workloads, and Arduinos are not designed for them at all [25]. Moreover, the Jetson's compact size, low power consumption, and rich I/O interfaces make it ideal for embedded systems, offering seamless integration with advanced sensors like LiDAR, cameras, and IMUs, which would be challenging for a Raspberry Pi or Arduino to handle effectively. By choosing a Jetson platform, the system gains the robust performance and scalability needed for demanding AI and robotics applications that simpler boards cannot support. When comparing multiple Raspberry Pis to a Jetson, several key differences emerge. While you can network multiple Raspberry Pis to boost overall computational power, they lack dedicated GPUs and AI acceleration capabilities [25]. The Jetson Nano, on the other hand, is specifically designed for AI and machine learning tasks, featuring an NVIDIA GPU with 128 CUDA cores. This allows it to efficiently process complex algorithms essential for real-time autonomous functions like object detection and path planning. In contrast, Raspberry Pis are general-purpose single-board computers without specialized hardware for AI workloads, making them less efficient for handling intensive tasks required in autonomous systems.

While the NVIDIA Jetson TX2 is a high-performance computing platform tailored for demanding AI applications like autonomous vehicles. It uses a 256-core GPU capable of handling complex parallel processing tasks, a dual-core NVIDIA Denver 2 CPU combined with a quad-core ARM Cortex-A57 CPU for robust multitasking, 8 GB of LPDDR4 RAM to smoothly run memory-intensive applications, and extensive connectivity options including USB 3.0, HDMI, and PCIe for versatile sensor and peripheral integration [26]. The NVIDIA Jetson Nano is the lowest budget Jetson option, which offers more affordable yet capable alternative with a 128-core GPU that supports CUDA and AI processing, a quad-core ARM Cortex-A57 CPU, and 4 GB of LPDDR4 RAM, along with essential connectivity features [27]. The Nano significantly reduces costs, its price is less than half of the TX2, while maintaining compatibility with NVIDIA's rich AI software ecosystem, making it suitable for educational purposes in autonomous vehicle technology by enabling students to develop and deploy AI algorithms, integrate with the same sensors, and gain practical experience with machine learning and computer vision tasks. Although it may not match the Jetson TX2 in terms of real-time processing capabilities and performance under heavy computational loads, the Jetson Nano's balance between performance and affordability makes it an appropriate and feasible choice.

A large consideration is whether to design the chassis from scratch and order each sensor individually, or order a kit containing all required components. Designing a chassis from scratch and ordering each

sensor individually offers unparalleled flexibility and customization for an autonomous vehicle platform. This approach allows the designer to tailor every aspect of the platform, from the size and weight distribution to the specific placement and integration of sensors like LiDAR, cameras, and IMUs. Building from scratch also provides an opportunity to experiment with innovative designs, optimize for particular use cases, and select components with precision to meet desired performance criteria. However, this approach requires significant time and expertise, as it involves complex tasks such as CAD modeling, material selection, manufacturing, and ensuring compatibility between hardware and software components. The cost of individually sourcing sensors and fabrication can quickly add up, and unforeseen challenges during assembly or testing may further increase expenses and development time. A breakdown of potential product choices is shown below.

- Processing Unit: The Nvidia Jetson Nano is chosen to be the optimal budget choice, which costs about \$400 [28]. This is 55% less than the Jetson in the QCar.
- LiDAR Sensor: LD 19 D300 is chosen, which costs less than \$150 [29]. This is approximately 67% less than the LiDAR in the QCar.
- Depth Camera: YDLIDAR OS30A 3D Depth Camera is chosen based on its low cost of \$200 [30]. This is about 61% less than the camera used on the QCar.
- 360° Vision System: The platform will feature the same four CSI cameras as the QCar, which cost less than \$20 each [18]
- Inertial Measurement Unit (IMU): A 9-axis IMU will be used, the BNO055 from RobotShop.com costs less than \$50 [19].
- Drive Motor: The motor that drives the car is chosen to be the Luzkey 21W motor, and costs about \$25 [20].
- Motor Encoder: The encoder chosen is the US Digital E8T-720-125, which is the same used on the QCar. This costs less than \$60 [21].
- Communication Interfaces: The platform must include Wi-Fi and Bluetooth modules to support wireless communication, the system costs less than \$100 [22].
- Power System: The same battery management system as the QCar will be used. This costs about \$200 in total [23], [24].
- Chassis: The chassis can be purchased from RobotShop.com, which contains everything required for driving, except the servo motor for steering. It costs approximately \$220 [31].
- Servo Motor: A servo motor is required to steer the car; this costs approximately \$30 [32].
- Additional components like turn signals and microphones are a negligible cost.

Alternatively, purchasing a pre-assembled kit like the Yahboom Rosmaster R2 ROS2 Robot can be a simpler and more cost-effective solution, particularly for those prioritizing functionality over custom design. Kits like the Rosmaster R2 come with a ready-made chassis and pre-integrated sensors that are already tested for compatibility, significantly reducing the time needed to get the platform up and running. These kits also typically include well-documented software and examples, making them easier to implement in educational or research settings. While they may not offer the same level of customization as a bespoke platform, the reduced development time, lower cost, and convenience of a kit make it an attractive choice compared to designing from scratch. This is why it is recommended to use a pre-assembled kit. Sensors would still have to be purchased to match the exact suite found on the

QCar. This may prove to be a challenge, however less design decisions are required to be made which would cut down on the development time.

There are multiple kit options available to purchase, three of the most popular/applicable options are the Yahboom Rosmaster R2, Waveshare JetRacer AI Kit, and the Waveshare JetRacer Pro AI Kit B.

The Yahboom Rosmaster R2 ROS2 Robot, which costs about \$1300 [33] without the Jetson Nano is an educational robotics platform designed to teach advanced concepts in robotics and autonomous systems. Powered by the NVIDIA Jetson Nano, it supports ROS2 (Robot Operating System 2), which is widely used in academic and research settings for robot control and simulation. The Rosmaster R2 comes equipped with a variety of sensors, including LiDAR for precise mapping and navigation, a depth camera for 3D perception, and ultrasonic sensors for obstacle avoidance. The sensors that would need to be integrated, are the 4 CSI cameras, and the 9 - axis IMU.

The Waveshare JetRacer AI Kit, which costs about \$250 [34] without the Jetson Nano is a DIY racing robot also powered by the NVIDIA Jetson Nano but is specifically designed for high-speed autonomous racing applications. It focuses on teaching AI-based autonomous driving through practical, hands-on experience. The kit includes components such as a high-quality camera for vision processing, a motorized chassis optimized for speed, and pre-configured software that supports deep learning frameworks.

The Waveshare JetRacer Pro AI Kit B, costing about \$550 [34] is an enhanced version of the JetRacer AI Kit, offering upgraded hardware for improved performance. It features a more powerful motor system for higher speeds, a better suspension system for stability, and additional sensors for more accurate environmental perception. Like its predecessor, it is powered by the Jetson Nano and is geared towards advanced autonomous racing applications. The Pro version allows for more complex algorithm development and experimentation with sophisticated AI models due to its superior hardware capabilities.

Current Recommendations

The Yahboom Rosmaster R2 ROS2 Robot stands out as an optimal choice for replicating the sensor suite of the Quanser QCar while remaining budget-friendly and minimizing additional design/assembly work. One of the primary reasons is that it comes with a built-in LiDAR sensor, which is a critical component of the QCar's sensor array. In addition to LiDAR, the Rosmaster R2 is equipped with other sensors that closely match those used in the Quanser QCar, such as depth cameras and Wi-Fi connectivity. By integrating the remaining sensors to mirror the QCar's sensors, the Rosmaster R2 ensures full compatibility with existing Python libraries developed for the QCar. A BOM in **Error! Reference source not found.** shows all components required to replicate the QCar.

Table 10 shows the bill of materials to replicate QCar using Yahboom Rosmaster kit.

Part #	Description	Qty.	Unit Cost
1	Jetson Nano	1	\$400
2	Yahboom Rosmaster R2 ROS2 Robot	1	\$1,300
3	CSI Camera	4	\$20
4	IMU	1	\$50
5	Miscellaneous Expenses	1	\$200
Total			\$2,030

A BOM for the other option, which is to purchase a chassis and every component separately, is shown below in Table 11.

Table 11 shows the bill of materials for a custom car replicating QCar's sensors.

Part #	Description	Qty.	Unit Cost
1	Jetson Nano	1	\$400
2	LD 19 D300 (LiDAR)	1	\$150
3	YDLIDAR OS30A 3D Depth Camera	1	\$200
4	CSI Camera	4	\$20
5	IMU	1	\$50
6	Drive Motor	1	\$25
7	Motor Encoder	1	\$60
8	Communication Interface	1	\$100
9	Chassis	1	\$220
10	Servo Motor	1	\$30
11	Power System	1	\$200
12	Miscellaneous Expenses	1	\$200
Total			\$1,715

This option should be chosen due to its lower cost, and the ability to customize every component on the car.

Conclusion

The development and implementation of the autonomous parallel parking system for the QCar platform has successfully demonstrated the viability of both traditional computer vision and CNN-based approaches for autonomous vehicle applications. Through systematic evaluation and testing, our research has yielded significant insights into autonomous system development and implementation.

The traditional CV methods achieved reliable performance with 70-90% accuracy while maintaining minimal computational overhead. In comparison, the CNN-based detection demonstrated superior robustness with an 84% success rate and greater adaptability to environmental variations. The combination of both approaches provided complementary strengths for robust system operation, suggesting potential benefits in hybrid implementations for future developments.

System integration proved successful across multiple domains, incorporating various sensor inputs including CSI cameras, RGB-D camera, and motor encoders. The implementation of efficient processing pipelines enabled real-time operation on the Jetson TX2 platform, while the modular software architecture established a foundation for future expansions and improvements. The control system met all key performance metrics, achieving parking accuracy within 5 cm and maintaining safety clearance of 3 cm minimum, while demonstrating reliable state transitions and error handling capabilities.

The project has also revealed several promising areas for future enhancement. Implementation of full autonomy from initial positioning represents a natural evolution of the current system. Enhanced annotation consistency for CNN training could significantly improve detection accuracy, while integration of advanced path planning algorithms could optimize trajectory execution. Additionally, upgrading to newer machine learning frameworks could provide access to improved detection capabilities and processing efficiency.

From an educational perspective, the project has provided invaluable practical implementation experience with advanced robotics concepts, demonstrating both the challenges and solutions inherent in autonomous system development. The economic analysis supports the feasibility of developing more cost-effective platforms for widespread adoption, potentially expanding access to autonomous vehicle education and research.

Looking ahead, this work establishes a robust framework for continued advancement in autonomous vehicle education and research, with particular emphasis on the integration of traditional engineering approaches with modern machine learning techniques. The methodologies developed and challenges overcome provide valuable guidance for future implementations, while the documented solutions offer a solid foundation for subsequent research initiatives. The success of this implementation demonstrates not only the technical feasibility of autonomous parking systems but also their potential for scaled deployment in educational settings.

Appendix

Appendix A - Source Evaluation

Source 1 - Github, Jetson Inference [35]

This source is the GitHub repository for Jetson Inference, an open-source project developed by NVIDIA. It provides tools, examples, and pretrained models for deploying deep learning inference on NVIDIA Jetson platforms. The repository includes support for tasks like image recognition, object detection, semantic segmentation, and pose estimation, with detailed documentation and code samples designed for both beginners and advanced users in AI and robotics.

Currency: Jetson Inference supports NVIDIA Jetson devices, and the repository's updates typically align with the latest NVIDIA hardware and software advancements, making it relevant and current for the AI and machine learning fields.

Relevance: This source provides examples, pretrained models, and tools for deploying deep learning models. The repository balances accessibility for beginners (through examples and documentation) and depth for advanced users (e.g., detailed configurations and customizations).

Authority: The repository is maintained by Dusty-Nv, an official NVIDIA contributor and recognized expert in Jetson development. Dusty-Nv is associated with NVIDIA, a leading company in AI and hardware development, ensuring authoritative and credible content.

Accuracy: The repository includes documentation, example codes, and links to pretrained models, all of which are demonstrable and reproducible. Users can verify the results directly. Contributions and updates go through community feedback on GitHub, ensuring quality and accuracy over time. Any errors in the repository are often identified and resolved quickly.

Purpose: The repository's purpose is to facilitate the deployment of AI models on Jetson devices. It provides tools, examples, and educational resources to enhance usability. The content is technical and factual, focusing on practical implementation. It is free of opinion or bias.

Source 2 - Geeksforgeeks, OpenCV Overview [36]

The article on GeeksforGeeks titled "OpenCV Overview" provides an introduction to OpenCV, a popular open-source computer vision library. It covers the key features, applications, and functions of OpenCV, making it a useful starting point for beginners and developers interested in image and video processing.

Currency: The article is regularly updated to reflect changes in OpenCV's features and functionality, which is important given the library's frequent updates. It was last updated on April 15, 2024.

Relevance: This resource is relevant for individuals learning about computer vision or developing projects requiring OpenCV. It provides a high-level overview, making it suitable for beginners, while its links to other tutorials and detailed functions cater to intermediate users.

Authority: GeeksforGeeks is a widely recognized platform for technical education, especially in programming and development.

Accuracy: The information provided in the article is factual and consistent with OpenCV's official documentation.

Purpose: The purpose of the article is to educate and introduce readers to OpenCV in an accessible way. It is unbiased and purely informational, focusing on helping users understand the library's capabilities and applications without promoting any commercial interest.

References

- [1 N. R. C. Canada, "New LiDAR technology for autonomous vehicles picks up speed in Canada,"
] Government of Canada, 18 11 2021. [Online]. Available: <https://nrc.canada.ca/en/stories/new-lidar-technology-autonomous-vehicles-picks-speed-canada>.
- [2 Quanser, "Qcar Sensor-rich autonomous vehicle for self-driving applications," 2024 Quanser,
] [Online]. Available: <https://www.quanser.com/products/qcar/#details>.
- [3 T. Covington, "Nearly half of Americans have “parallelphobia”, " The Zebra, 14 03 2024. [Online].
] Available: <https://www.thezebra.com/resources/driving/parallel-parking-fear-survey/>.
- [4 T. Covington, "Nearly half of Americans have “parallelphobia”, " The Zebra, 14 03 2024. [Online].
] Available: <https://www.thezebra.com/resources/driving/parallel-parking-fear-survey/>. [Accessed 01 12 2024].
- [5 E. Cook, "Top 10: Best Self-Parking Cars in 2023," Evens Halshaw, 05 01 2023. [Online]. Available:
] <https://www.evanshalshaw.com/blog/best-self-parking-cars/>. [Accessed 01 12 2024].
- [6 E. Kassens-Noor, M. Cai, Z. Kotval-Karamchandani and T. Decaminada, "Autonomous vehicles and
] mobility for people with special needs," 04 07 2021. [Online]. Available:
<https://www.sciencedirect.com/science/article/abs/pii/S0965856421001622>. [Accessed 01 12 2024].
- [7 "QCar," Quanser, [Online]. Available: <https://www.quanser.com/products/qcar/>. [Accessed 01 12
] 2024].
- [8 "Color Extraction by HSV," [Online]. Available: [https://justin-
\] liang.com/tutorials/hsv_color_extraction/](https://justin-liang.com/tutorials/hsv_color_extraction/).
- [9 T. K. P. M. E. W. a. M. K. M. Widyaningsih, "Optimization Contrast Enhancement and Noise
] Reduction for Semantic Segmentation of Oil Palm Aerial Imagery," [Online]. Available:
<https://oaji.net/articles/2022/3603-1672282521.pdf>.
- [1 P. K. J. T. a. F. D. S. Paris, "Bilateral Filtering: Theory and Applications," [Online]. Available:
0] <https://doi.org/10.1561/06000000020>.
- [1 "Canny Edge Detection," [Online]. Available:
1] https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html.
- [1 "Image Processing — Edge Detection, Morphological Operations & Template Matching," [Online].
2] Available: <https://sybernix.medium.com/image-processing-edge-detection-morphological-operations-template-matching-part-2-of-3-8b28d699736c>.

- [1] "OpenCV: Hough Line Transform," [Online]. Available:
 3] https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html.
- [1] "Transform Road Image to Bird's-Eye-View Image," [Online]. Available:
 4] <https://www.mathworks.com/help/driving/ref/birdseyeview.html>.
- [1] "NVIDIA Jetson TX2 Module," RobotShop, [Online]. Available:
 5] <https://www.robotshop.com/products/nvidia-jetson-tx2-module>. [Accessed 1 December 2024].
- [1] "RPLIDAR A2M8 360° Laser Scanner," RobotShop, [Online]. Available:
 6] [robotshop.com/products/rplidar-a2m8-360-laser-scanner](https://www.robotshop.com/products/rplidar-a2m8-360-laser-scanner).
- [1] "Intel RealSense Depth Camera D435," Best Buy, [Online]. Available: https://www.bestbuy.ca/en-ca/product/intel-realsense-depth-camera-d435/12943518?cmp=seo-12943518&cmp=knc-s-12556806400&&utm_source=google&utm_id=240828cac943&utm_medium=troas&utm_creativeformat=shopping&gad_source=1&gclid=Cj0KCQiAr7C6BhDRARIsAOUKifiJbXR. [Accessed 1 December 2024].
- [1] "Arducam IMX219 Wide Angle Camera Module," RobotShop, [Online]. Available:
 8] https://ca.robotshop.com/products/arducam-imx219-wide-angle-camera-module?gad_source=1&gclid=CjwKCAiArva5BhBiEiwA-oTnXXKi1UHSTu7t6dwtr4Y4SDcAvpj4Dh0pi7sBISkC7CfegZ9ml1oBPRoCf_4QAvD_BwE. [Accessed 1 December 2024].
- [1] "BNO055 9 DOF Absolute Orientation IMU Fusion Breakout Board," RobotShop, [Online]. Available:
 9] <https://ca.robotshop.com/products/bno055-9-dof-absolute-orientation-imu-fusion-breakout-board>. [Accessed 1 December 2024].
- [2] "DC 12V 21W Motor 8000 RPM Shaft Low Noise Motor For Electric Toys," Walmart, [Online].
 0] Available: <https://www.walmart.ca/en/ip/DC-12V-21W-Motor-8000-RPM-Shaft-Low-Noise-Motor-For-Electric-Toys/1AB01PDPKKS1>. [Accessed 1 December 2024].
- [2] "E8T Miniature Optical Kit Encoder," US Digital, [Online]. Available:
 1] <https://www.usdigital.com/products/encoders/incremental/kit/e8t/>. [Accessed 1 December 2024].
- [2] "TP-Link WiFi 6 AX3000 PCIe WiFi Card for PC with Heat Sink (Archer TX50E) - Bluetooth 5.0,
 2] 802.11AX Dual Band Wireless Adapter with MU-MIMO, Ultra-Low Latency, Supports Windows 11, 10 (64bit) Only," Amazon, [Online]. Available: https://www.amazon.ca/TP-Link-Bluetooth-Ultra-Low-Archer-TX50E/dp/B089FCX3C3/ref=asc_df_B089FCX3C3/?tag=googleshopc0c-20&linkCode=df0&hvadid=706724917350&hvpos=&hvnetw=g&hvrand=8944643488339847450&hvponetw=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocph. [Accessed 1 December 2024].

- [2] "Gens Ace G-Tech 3300mAh 4S 14.8V 45C LiPo EC3 Plug," Canada Hobbies, [Online]. Available: 3] <https://www.canadahobbies.ca/product/hobby-brands/gens-ace/gens-ace-g-tech-3300mah-4s-148v-45c-lipo-ec3-plug/?srsId=AfmBOOp4aVXzpWJCYEuKbOZo0pHqJFnEblWMSlZvLqAqjp1XI3SDSfEf>. [Accessed 1 December 2024].
- [2] "EV-PEAK E4 110 - 240V input, 50W 4A 2-4S Lipo Charger," Grandado, [Online]. Available: 4] https://can.grandado.com/products/ev-peak-e4-110-240v-input-50w-4a-2-4s-lipo-charger?variant=UHVjZHVjdFZhcmllbnQ6NzA5MTY0MzE1&gad_source=1&gclid=CjwKCAiArva5BhBiEiwA-oTnXaSOQVEhVxeg67h_z-0Kv9fOggFmMDWMtt1KtZVwkFrb_EBXWPbUAxoCLgIQAvD_BwE. [Accessed 1 December 2024].
- [2] A. Özen, "JETSON NANO VS RASPBERRY PI 4," Medium, 11 January 2022. [Online]. Available: 5] <https://medium.com/@anil.ozenn/jetson-nano-vs-raspberry-pi%CC%87-4-b1f6bf5a00e>. [Accessed 1 December 2024].
- [2] "Jetson TX2 Module," Nvidia, [Online]. Available: <https://developer.nvidia.com/embedded/jetson-tx2>. [Accessed 1 December 2024].
- [2] "Jetson Nano," Nvidia, [Online]. Available: <https://developer.nvidia.com/embedded/jetson-nano>. 7] [Accessed 1 December 2024].
- [2] "Jetson NANO 4GB Developer Kit (SUB) With Official Module For Artificial Intelligence Python 8] Programming," Nvidia, [Online]. Available: https://ca.robotshop.com/products/yahboom-jetson-nano-4gb-developer-kit-sub-with-official-module-for-artificial-intelligence-python-programming?gad_source=1&gclid=Cj0KCQiAr7C6BhDRARIsAOUKifi2GqCw53usQcYfJfT57TMSY53mPpE89oAAyU_OBumxCzuilq4Mh44aAp5dEALw_wcB. [Accessed 1 December 2024].
- [2] "LD19 D300 LiDAR Developer Kit, 360 DToF Laser Scanner, Supports ROS1/2, Raspberry Pi & Jetson 9] Nano," RobotShop, [Online]. Available: https://ca.robotshop.com/products/hiwonder-ld19-d300-lidar-developer-kit-360-dtof-laser-scanner-supports-ros1-2-raspberry-pi-jetson-nano?gad_source=1&gclid=Cj0KCQiAr7C6BhDRARIsAOUKifjs6yJwCDWUBWN_rz8MMziulYMCiA6Gej4ysyoOfECOR6h5HDWgYMaAkGUEALw_wcB. [Accessed 1 December 2024].
- [3] "YDLIDAR OS30A 3D Depth Camera," RobotShop, [Online]. Available: 0] https://ca.robotshop.com/products/ydlidar-os30a-3d-depth-camera?gad_source=1&gclid=Cj0KCQiAr7C6BhDRARIsAOUKifg4zYH14h4HO8m0aWS2UXZoBCplXNxOfDNk77wioNVxUnt5elp4qckaAnKPEALw_wcB. [Accessed 1 December 2024].
- [3] "Yahboom Aluminum Alloy ROS Robot Car Chassis--Ackerman steering Chassis," RobotShop.com, 1] [Online]. Available: <https://ca.robotshop.com/products/yahboom-yahboom-aluminum-alloy-ros-robot-car-chassis-ackerman-steering-chassis>. [Accessed 1 December 2024].

- [3] "Hiwonder LD 20MG Full Metal Gear Digital Servo w/ 20kg High Torque Aluminium Case," RobotShop, [Online]. Available: https://ca.robotshop.com/products/hiwonder-hiwonder-ld-20mg-full-metal-gear-digital-servo-w-20kg-high-torque-aluminium-case?gad_source=1&gclid=Cj0KCQiAr7C6BhDRARIsAOUKifhKg1mSXec66bLiu8AtKjqUshW4npjGdCCFnDSry9iAVC_0wLdiv6oaAgRsEALw_wcB. [Accessed 1 December 2024].
- [3] "Yahboom Rosmaster R2 ROS2 Robot w/ Ackermann Structure (Jetson Standard Version w/o Board)," RobotShop, [Online]. Available: https://ca.robotshop.com/products/yahboom-yahboom-rosmaster-r2-ros2-robot-w-ackermann-structure-jetson-standard-version-w-o-board?gad_source=1&gclid=Cj0KCQiAr7C6BhDRARIsAOUKifjZ9CNW89CsViwqkwZ04ABIIUrHhXK4vGZZXI9yLnHakAF8EXULvYaAhA0EALw_wcB. [Accessed 1 December 2024].
- [3] "JetRacer AI Kit, AI Racing Robot Powered by Jetson Nano (NOT included)," WaveShare, [Online]. Available: <https://www.waveshare.com/product/jetracer-ai-kit.htm?sku=17607>.
- [3] "dusty-nv / jetson-inference," Github, [Online]. Available: <https://github.com/dusty-nv/jetson-inference/tree/master>. [Accessed 1 December 2024].
- [3] "What is OpenCV Library?," Geeksforgeeks, [Online]. Available: <https://www.geeksforgeeks.org/opencv-overview/>. [Accessed 1 December 2024].
- [3] National Research Council Canada, "New LiDAR technology for autonomous vehicles picks up speed in Canada," Government of Canada, 18 07 2022. [Online]. Available: <https://nrc.canada.ca/en/stories/new-lidar-technology-autonomous-vehicles-picks-speed-canada>. [Accessed 20 11 2024].
- [3] Quanser, "Supported Software and APIs," [Online]. Available: <https://www.quanser.com/products/qcar/#details>. [Accessed 20 11 2024].
- [3] Advanced Buildings, "Using this Guide," 2021. [Online]. Available: <https://www.patternguide.advancedbuildings.net/using-this-guide.html>.
- [4] Government of Canada, "New LiDAR technology for autonomous vehicles picks up speed," 18 07 2022. [Online]. Available: <https://nrc.canada.ca/en/stories/new-lidar-technology-autonomous-vehicles-picks-speed-canada>. [Accessed 01 12 2024].