People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University of Abderrahmane Mira Bejaia
Computer science Department

RESEARCH MASTER'S THESIS

in Computer Science

Option :

Artificial Intelligence

Research Topic

Multi-sensor fusion for enhanced object detection and tracking in autonomous driving

Produced by: Amine ZERARGA

Defended on 03/07/2024 in front of the jury composed of :

Djamila BOUKREDERA	M.C. A	President	UAMB - Bejaia
Mouloud ATMANI	M.C.A	Supervisor	UAMB - Bejaia
Samira AIT KACI AZZOU	M.A.A	Co-supervisor	UAMB - Bejaia
Hayette KHALED	M.C.B	Examiner	UAMB - Bejaia
Sahar BOULKABOUL	M.R.B	Internship supervisor	CERIST, Algiers

Academic Year 2023 – 2024

Dedication

In the name of Allah, the Most Powerful,
I would like to express my deepest gratitude to my parents, who have always been there for me, to my dear brothers and sisters, and to my friends, who have stood by me throughout this period at university. I also wish to extend my heartfelt thanks to my supervisors for their trust in me and for allowing me to learn so much during this experience.

Acknowledgements

I would like to express my deepest gratitude to my parents for their unwavering support. I also wish to extend my heartfelt thanks to my supervisors, Mr. Atmani Mouloud and Mrs. AIT KACI AZZOU SAMIRA Epse BOUKERRAM, for all their support and shared ideas. I also wish to express my gratitude to the CERIST Research Center, Algiers, and my internship director, Mrs. Sahar Boulkaboul, for their trust and for providing us with the necessary resources and for all the knowledge and experience we gained thanks to them. We are also grateful to the jury members, Mrs. Djamila BOUKREDERA and Mrs. Hayette KHALED, for their time, remarks, and constructive comments that greatly improved this work. Finally, we are grateful to the A.Mira University of Béjaïa for providing the necessary teachings and resources for this research. Our sincere thanks also go to all the people not mentioned here but who played an important role in this project.

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
AssA	Association Accuracy
AssPr	Association Precision
AssRe	Association Recall
CAN	Controller Area Network
CNN	Convolutional Neural Network
DeepSORT	Deep Simple Online and Realtime Tracking
DetA	Detection Accuracy
DetPr	Detection Precision
DetRe	Detection Recall
EKF	Extended Kalman Filter
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
HOTA	Higher Order Tracking Accuracy
IDF1	Identification F1 Score
IDFN	False Negatives for Identification
IDFP	False Positives for Identification
IDP	Identification Precision
IDR	Identification Recall
IDTP	True Positives for Identification
IMU	Inertial Measurement Unit
IoT	Internet of Things
LiDAR	Light Detection and Ranging

LocA	Localization Accuracy
MOTA	Multiple Object Tracking Accuracy
MOTP	Multiple Object Tracking Precision
MOT	Multiple Object Tracking
PID	Proportional-Integral-Derivative
ROS	Robot Operating System
SLAM	Simultaneous Localization and Mapping
sMOTA	Scaled Multiple Object Tracking Accuracy
V2G	Vehicle-to-Grid
V2H	Vehicle-to-Home
V2I	Vehicle-to-Infrastructure
V2P	Vehicle-to-Pedestrian
V2V	Vehicle-to-Vehicle
V2X	Vehicle-to-Everything
YOLO	You Only Look Once

General Introduction

The field of autonomous vehicles has seen tremendous advancements over the past decade, driven by innovations in artificial intelligence (AI) and machine learning. Autonomous vehicles aim to revolutionize transportation by providing safer, more efficient, and more accessible mobility solutions. Central to their operation is the ability to perceive and understand their environment accurately. This is achieved through a combination of sensors, including cameras, LiDAR, and radar, which collectively contribute to the vehicle's situational awareness and decision-making processes. The integration of these sensors allows autonomous vehicles to detect, classify, and track objects in real-time, enabling them to navigate complex environments and make informed decisions.

Despite these advancements, one of the critical challenges in autonomous vehicle technology remains sensor fusion, the process of integrating data from multiple sensors to create a cohesive and accurate representation of the environment. Effective sensor fusion is essential for robust object detection, tracking, and obstacle avoidance. However, this task is fraught with difficulties such as sensor calibration, data synchronization, and handling varying data quality and resolution. Inaccuracies in any of these aspects can significantly affect the vehicle's ability to make safe and reliable decisions. Additionally, the differing characteristics of sensor data, such as the dense 3D point clouds from LiDAR and the high-resolution images from cameras, add to the complexity of the fusion process. Overcoming these challenges is crucial for advancing the reliability and safety of autonomous vehicles.

To address these challenges, our project, conducted in collaboration with the Cerist Research Center and the University of Bejaia, proposes a sensor fusion algorithm that leverages both camera and LiDAR data to improve object detection and tracking accuracy. Our approach focuses on integrating state-of-the-art object detection algorithms, such as YOLOv8 for camera images and Complex YOLO for LiDAR point clouds, with a tracking framework using DeepSORT and Kalman Filters. By fusing the outputs from these different sensor modalities, we aim to enhance the overall performance of the tracking system, ensuring more reliable and precise detection and tracking of objects in the vehicle's environment. This fusion approach allows us to exploit the complementary strengths of each sensor type, providing a more comprehensive understanding of the surrounding environment, thereby enhancing decision-making capabilities in autonomous driving applications.

Our contributions include the development of a robust sensor fusion algorithm, the implementation of an Extended Kalman Filter for data fusion, and a comprehensive evaluation of our approach using the KITTI dataset. This comprehensive approach allows for the accurate fusion of data from multiple sensors, addressing the challenges of sensor calibration and synchronization, and providing a more reliable system for autonomous vehicle navigation. Furthermore, our method incorporates advanced machine learning techniques to adaptively improve the fusion process, ensuring that the system can handle dynamic and unpredictable driving scenarios effectively.

The organization of this thesis is as follows: In Chapter 1, we provide an overview of autonomous vehicle technology, including the role of AI and sensor fusion. Chapter 2 reviews the current state-of-the-art techniques in object detection and tracking for autonomous driving, highlighting recent advancements and ongoing challenges. Chapter 3 details our proposed sensor fusion algorithm, including the methodologies for object detection, tracking, and data fusion. Finally, Chapter 4 evaluates the performance of our approach using the KITTI MOT Evaluation metrics and compares our results with existing methods, demonstrating the effectiveness and improvements offered by our sensor fusion solution. This structured approach ensures a clear and thorough examination of the topic, leading to a comprehensive understanding of our contributions to the field.

Chapter 1

General Informations in Autonomous Vehicles

Chapter 1

General Informations in Autonomous Vehicles

1.1 Introduction

Autonomous vehicles represent a transformative innovation in modern transportation, promising to revolutionize how people and goods move. By leveraging advanced technologies such as artificial intelligence, machine learning, and sensor fusion, these vehicles aim to enhance safety, efficiency, and convenience on the roads. This chapter provides a comprehensive overview of autonomous vehicles and intelligent transportation systems, setting the foundation for the detailed discussion on sensor fusion for enhanced object tracking in subsequent chapters.

1.2 Autonomous Vehicles

1.2.1 Definition and Overview

Autonomous vehicles, also known as self-driving cars, are vehicles capable of sensing their environment and operating without human intervention. They utilize a combination of sensors, cameras, radar, LIDAR, and advanced algorithms to navigate and respond to their surroundings.

1.2.2 History and Evolution

The concept of autonomous vehicles dates back to the early 20th century, but significant progress began in the late 20th and early 21st centuries. Key milestones include the development of early prototypes by companies like Carnegie Mellon University and DARPA's Grand Challenges, which spurred significant advancements in autonomous driving technologies [13].

1.2.3 Levels of Autonomy in Autonomous Vehicles

The SAE (Society of Automotive Engineers) [14] defines six levels of driving automation , from Level 0 (no automation) to Level 5 (full automation). Each level represents a step towards full autonomy, with increasing capabilities and decreasing reliance on human intervention as shown in the figure below 1.1.

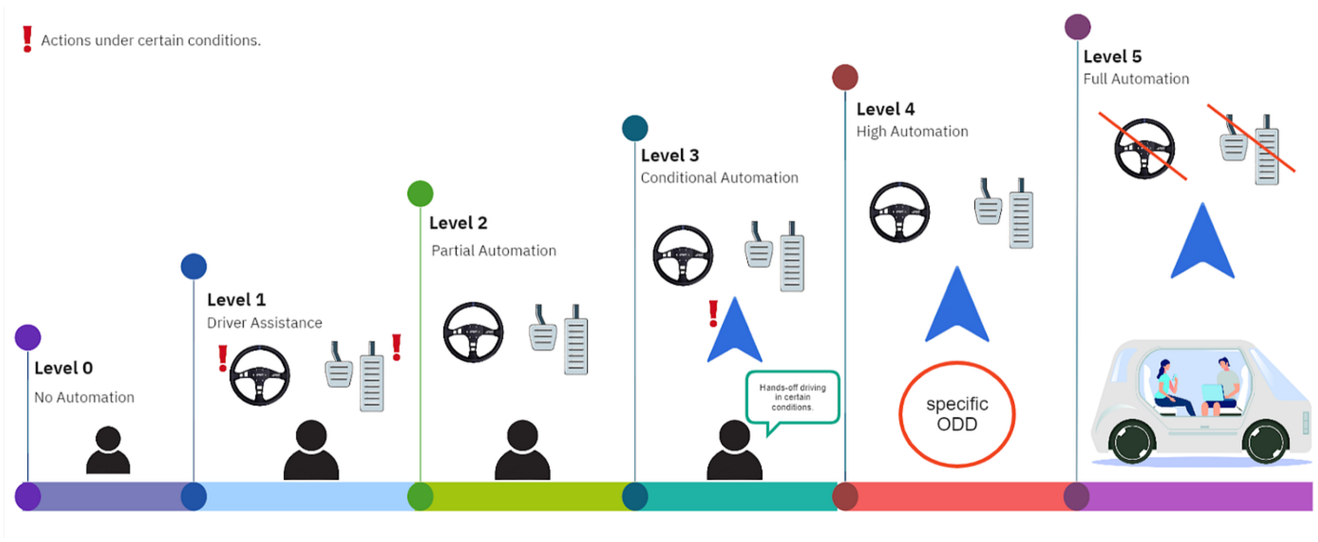


Figure 1.1: Levels of Vehicle Automation [1]

– **Level 0: No Automation**

Driver Responsibility: The human driver is responsible for all aspects of driving. The vehicle may have features that provide warnings or momentary assistance, but these do not control the vehicle.

Examples: Automatic emergency braking, blind-spot warning.

– **Level 1: Driver Assistance**

Driver Responsibility: The human driver handles most driving tasks but can be assisted by the vehicle in specific situations.

Vehicle Capability: The vehicle can assist with either steering or acceleration/deceleration, but not both simultaneously.

Examples: Adaptive cruise control, lane-keeping assistance.

– **Level 2: Partial Automation**

Driver Responsibility: The human driver must monitor the driving environment and be ready to take over at any moment.

Vehicle Capability: The vehicle can control both steering and acceleration/deceleration simultaneously under certain conditions.

Examples: Tesla Autopilot, GM Super Cruise.

– **Level 3: Conditional Automation**
Driver Responsibility: The vehicle handles all aspects of driving in certain conditions, but the human driver must be available to take control if the system requests.

Vehicle Capability: The vehicle can perform all driving tasks under specific conditions (e.g., highway driving), but the driver must intervene when the system cannot handle the situation.

Examples: Audi Traffic Jam Pilot (not widely available).

– **Level 4: High Automation**
Driver Responsibility: The vehicle can handle all driving tasks within specific conditions or environments without human intervention.

Vehicle Capability: The vehicle can perform all driving tasks and monitor the driving environment in designated areas or situations (e.g., geofenced areas, urban environments).

Examples: Waymo self-driving taxis (in limited areas).

– **Level 5: Full Automation**
Driver Responsibility: The vehicle can handle all driving tasks in all conditions without any human intervention. The human driver becomes a passenger.

Vehicle Capability: The vehicle is capable of driving autonomously in any environment and under any conditions that a human driver could manage.

Examples: This level represents the goal of full autonomy and is not yet achieved by any current vehicles.

1.3 Intelligent Transportation Systems (ITS)

1.3.1 Definition and Components

Intelligent Transportation Systems (ITS) integrate information and communication technologies with transportation infrastructure and vehicles to improve traffic management, safety, and efficiency. Key components include traffic management systems, vehicle-to-everything (V2X) communication, and automated enforcement systems.

1.3.2 Benefits and Challenges

ITS offer numerous benefits, such as enhanced road safety, reduced traffic congestion, and improved environmental sustainability. However, challenges such as high implementation costs, data privacy concerns, and the need for robust cybersecurity measures must be addressed.

1.4 Sensor Fusion in Autonomous Vehicles

1.4.1 Definition and Techniques

Sensor fusion involves combining data from multiple sensors to create a more accurate and comprehensive understanding of the environment. Techniques such as Kalman filtering, Bayesian networks, and neural networks are commonly used in sensor fusion [15].

1.4.2 Importance in Autonomous Driving

Sensor fusion is crucial for autonomous vehicles as it enhances the accuracy and reliability of object detection and tracking, leading to safer and more efficient navigation. By integrating data from various sensors, autonomous systems can compensate for the limitations of individual sensors and achieve a higher level of situational awareness [2] the figure below 1.2 shows the different sensors in an automated vehicle.

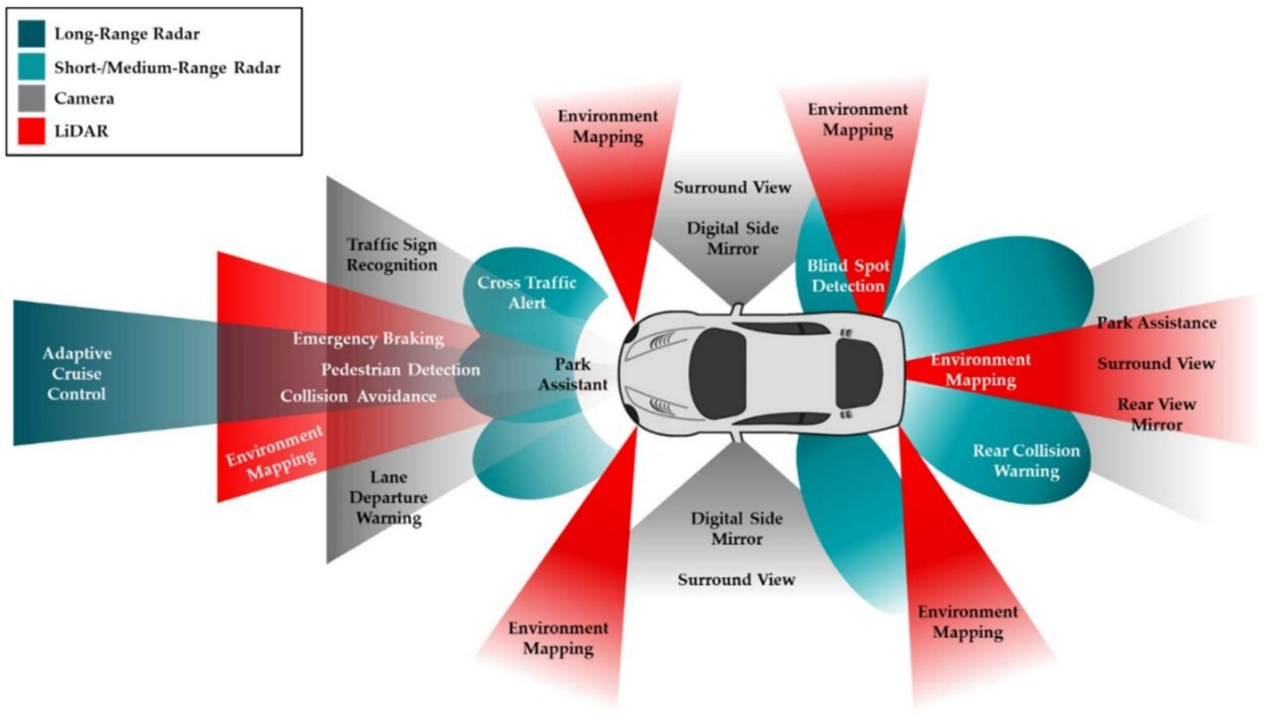


Figure 1.2: An example of the type and positioning of sensors in an automated vehicle [2]

1.5 IoT for Autonomous Vehicles

The Internet of Things (IoT) plays a crucial role in the development and functioning of autonomous vehicles. IoT enables vehicles to connect and communicate with each other and with the surrounding infrastructure, creating a more integrated and intelligent transportation system.

1.5.1 IoT and Vehicle Communication

The figure 1.3 illustrates how IoT technology facilitates Vehicle-to-Everything (V2X) communication, which includes Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I), and Vehicle-to-Pedestrian (V2P) communications. These connections [3] enhance situational awareness and safety by allowing vehicles to exchange information about road conditions, traffic signals, and potential hazards in real-time.

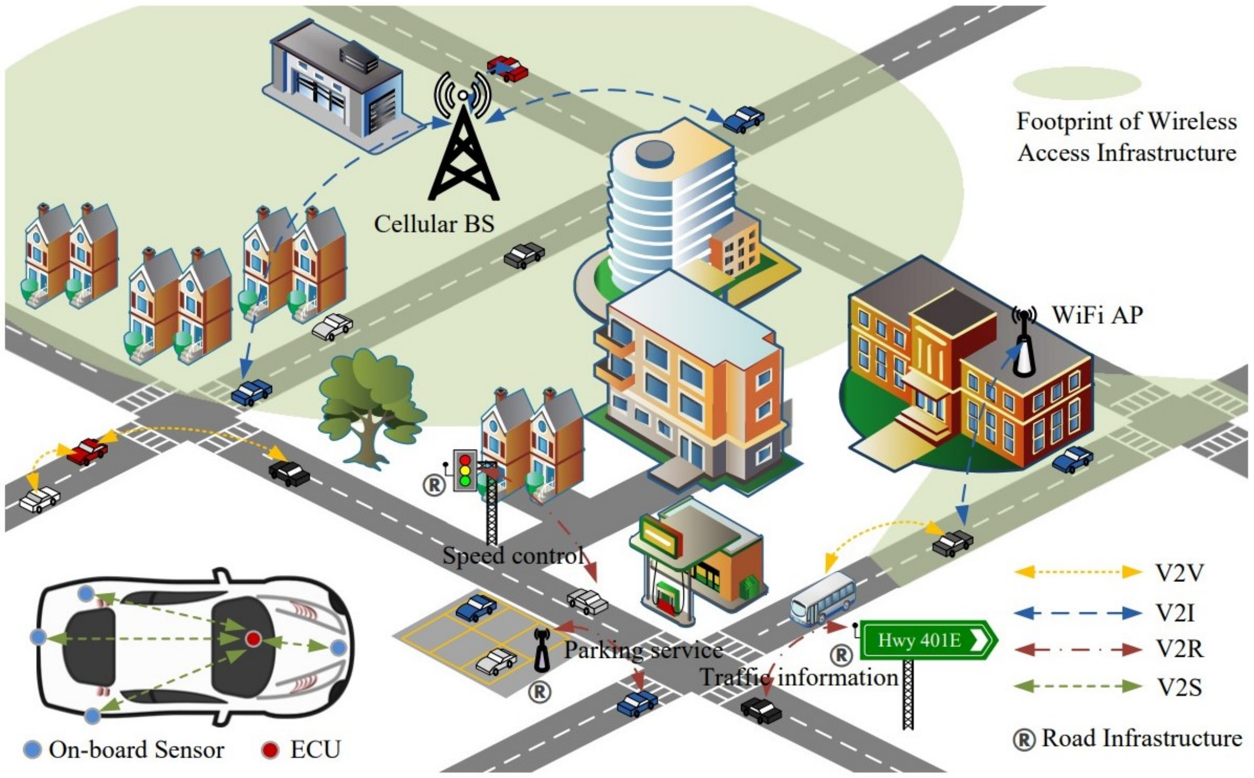


Figure 1.3: Overview of Connected Vehicles [3]

1.5.2 Sensors and Data Collection

Autonomous vehicles are equipped with a variety of sensors, including LIDAR, radar, cameras, and ultrasonic sensors. These sensors collect vast amounts of data about the vehicle's surroundings. IoT systems process this data to provide actionable insights that help the vehicle make informed decisions.

1.5.3 Cloud Computing and Edge Computing

IoT in autonomous vehicles relies on both cloud computing and edge computing [16]. Cloud computing provides the infrastructure for data storage and processing, while edge computing allows for real-time data analysis closer to the source. This dual approach ensures that critical decisions can be made quickly, while more complex analyses can be handled by the cloud.

1.6 AI for Autonomous Vehicles

Artificial Intelligence (AI) is the backbone of autonomous vehicle technology. It enables vehicles to perceive their environment, make decisions, and learn from experience to improve performance over time.

1.6.1 Machine Learning and Deep Learning

Machine learning algorithms, particularly deep learning, are used to process sensor data and recognize patterns. These algorithms are trained on large datasets to identify objects such as pedestrians, other vehicles, and road signs. Neural networks, a subset of deep learning, are especially effective in image and speech recognition tasks.

1.6.2 Decision-Making and Path Planning

AI is critical for decision-making and path planning in autonomous vehicles. Algorithms such as reinforcement learning enable vehicles to navigate complex environments by learning optimal driving strategies through trial and error. AI systems can evaluate various scenarios and make real-time decisions to ensure safety and efficiency [17].

1.6.3 Natural Language Processing

Natural Language Processing (NLP) allows autonomous vehicles to understand and respond to voice commands, enhancing user interaction. NLP technologies enable drivers and passengers to communicate with the vehicle using natural language, making the experience more intuitive and user-friendly.

1.7 Databases for Autonomous Vehicles

Databases are essential for storing and managing the vast amounts of data generated by autonomous vehicles. These databases support various functions, including mapping, sensor data storage, and machine learning model training.

1.7.1 High-Definition Maps

High-definition (HD) maps provide detailed information about road geometry, traffic signs, lane markings, and other static features of the environment. These maps are continuously updated to reflect real-time changes in road conditions and construction activities [18].

1.7.2 Sensor Data Storage

Autonomous vehicles generate massive amounts of sensor data, including images, LIDAR point clouds, and radar signals. Efficient storage solutions are required to manage this data and make it accessible for real-time processing and long-term analysis [19].

1.7.3 Training Data for Machine Learning

Large annotated datasets are necessary for training machine learning models used in autonomous vehicles. These datasets include labeled images, videos, and sensor data that help improve object detection, classification, and tracking algorithms. Examples of such datasets include the KITTI [20] dataset , which provides a variety of sensor data including stereo images and 3D point clouds, and the Waymo Open Dataset [21], which offers high-resolution sensor data and annotations for objects, enabling advanced research and development in autonomous driving.

1.8 Leading Companies in Autonomous Vehicles and Their Approaches

Several major automotive and technology companies are leading the development of autonomous vehicles, each with its unique approach and technology stack.

– **Tesla**

Tesla [22] uses a combination of computer vision, radar, and ultrasonic sensors for its autonomous driving technology. Tesla's Autopilot and Full Self-Driving (FSD) systems rely heavily on deep learning and neural networks, which are continuously updated through over-the-air software updates.

Figure 1.4: Tesla Autopilot

– **Waymo**

Waymo [23], a subsidiary of Alphabet Inc., focuses on a sensor suite that includes LIDAR, radar, and high-resolution cameras. Waymo's approach emphasizes redundancy and safety, with extensive testing and validation in real-world conditions. Waymo also uses HD maps to enhance navigation accuracy.

Figure 1.5: Waymo autonomous car

– **Zoox**

Zoox [24], acquired by Amazon in 2020, is developing a fully autonomous, bidirectional vehicle designed specifically for ride-hailing services. Zoox's approach includes a symmetrical design that eliminates the need for a front or back, allowing the vehicle to navigate and park more efficiently in urban environments. The vehicle is equipped with LIDAR, radar, cameras, and an array of sensors to ensure 360-degree coverage. Zoox focuses on creating a comprehensive, integrated system that includes custom hardware and software solutions to optimize safety and performance.

Figure 1.6: zoox autonmous vehicle

– **Mercedes-Benz**

Mercedes-Benz has been a pioneer in integrating advanced driver assistance systems (ADAS) and autonomous driving technologies into their vehicles [25]. The brand's approach emphasizes luxury and safety, combining cutting-edge technology with high-quality engineering. Mercedes-Benz's DRIVE PILOT system, which is designed for Level 3 autonomy, allows hands-free driving in certain conditions. The system uses a combination of LIDAR, radar, cameras, and ultrasonic sensors to monitor the vehicle's surroundings and make informed driving decisions. Mercedes-Benz is also involved in various collaborations and partnerships to advance autonomous vehicle technology further.

Figure 1.7: Mercedes-Benz Drive Pilot

1.9 Tesla vs Waymo - Two Opposite Visions

The world of self-driving cars has two giants: Tesla vs Waymo. Each represents one family of startups, with their own strategies and technical choices [26].

Tesla and Waymo represent two fundamentally different approaches to autonomous driving. Tesla relies heavily on a vision-based system, utilizing cameras and advanced neural networks to interpret and navigate the environment. Their approach, known as ”full self-driving” (FSD), emphasizes incremental improvements through software updates that leverage real-world data from Tesla’s extensive fleet of vehicles already on the road.

In contrast, Waymo employs a more conservative and comprehensive sensor suite, combining LIDAR, radar, and high-resolution cameras. Waymo’s strategy focuses on creating highly detailed 3D maps of the areas their vehicles operate in, allowing for precise navigation and obstacle detection. This approach aims for high levels of safety and reliability before scaling to wider geographic areas.

While Tesla pursues a scalable vision-based model, betting on the rapid advancement of AI, Waymo emphasizes robustness and precision with a multi-sensor approach. Both strategies reflect their distinct philosophies and technological bets on the future of autonomous vehicles.

1.10 Working of Autonomous Vehicles

Autonomous vehicles, or self-driving cars, rely on a complex integration of sensors, algorithms, and computing technologies to navigate and operate without human intervention. Key components include sensors such as cameras, LIDAR, radar, and ultrasonic sensors illustrated in the figure of Waymo’s Hardware Infrastructure 1.8, object detection and tracking systems, sensor fusion techniques, and Simultaneous Localization and Mapping (SLAM).

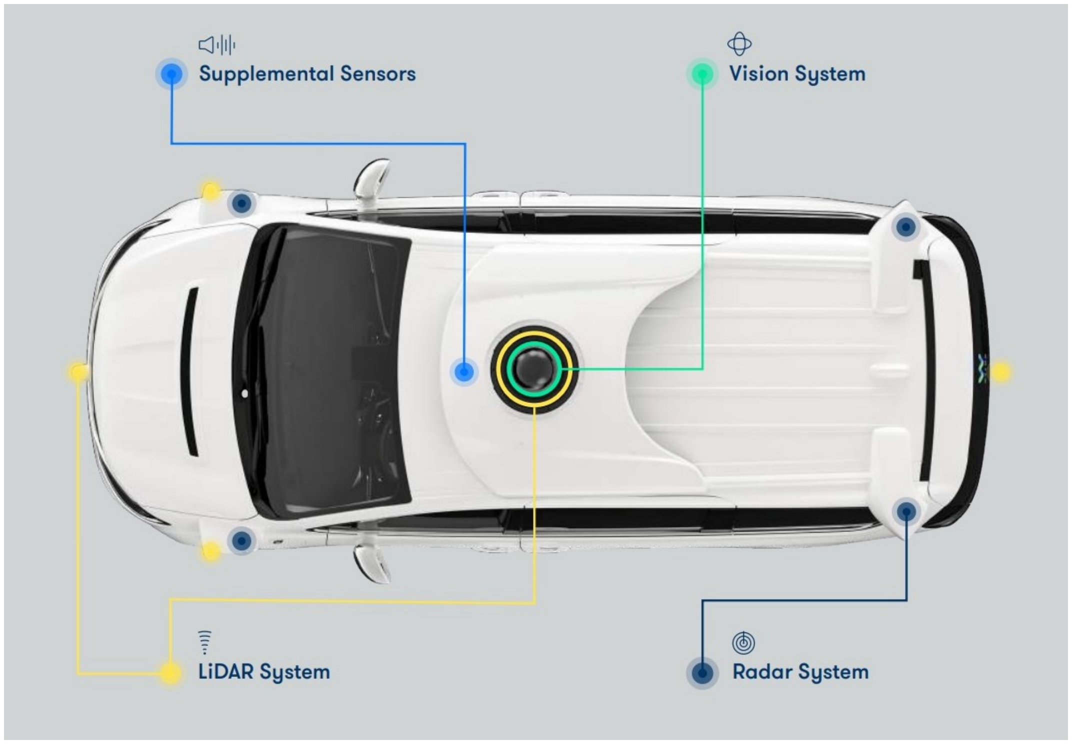


Figure 1.8: Waymo's Hardware Infrastructure

1.10.1 Sensors

- **Cameras**
  Cameras are essential for visual perception, providing high-resolution images that are crucial for recognizing objects, detecting lane markings, and reading traffic signs. Monocular and stereo cameras are commonly used in autonomous vehicles to capture detailed visual information. Monocular cameras provide a single perspective, while stereo cameras use two lenses to create depth perception, mimicking human vision.

- **LIDAR**
  LIDAR (Light Detection and Ranging) [27] uses laser beams to create detailed 3D maps of the vehicle's surroundings as shown in figure 1.9. By measuring the time it takes for the laser light to return after hitting an object, LIDAR generates precise distance measurements. This technology excels in detecting and mapping objects in various lighting conditions and provides a comprehensive view of the environment.

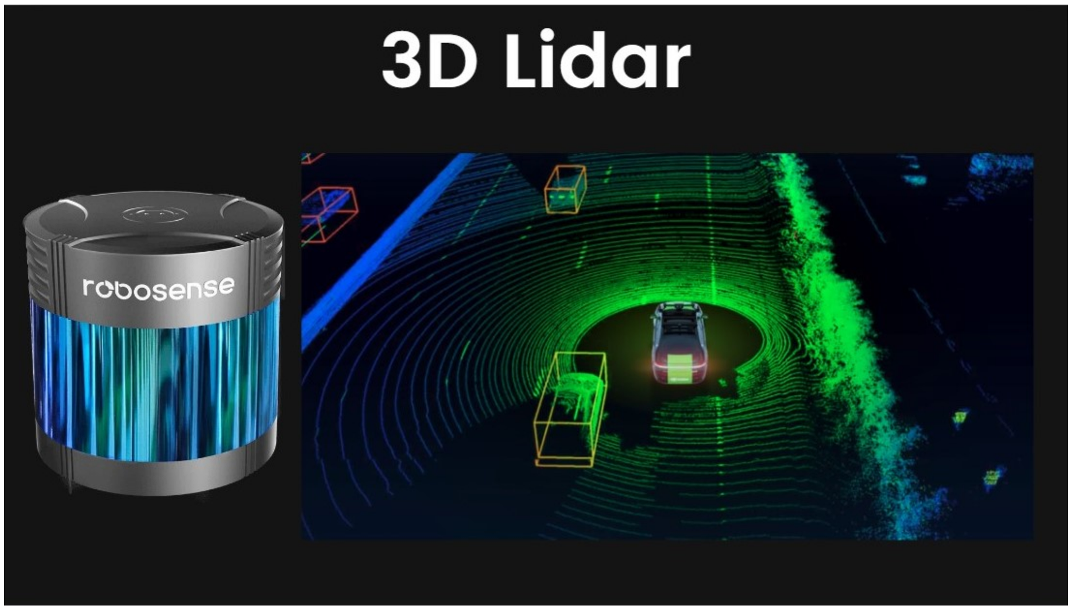


Figure 1.9: 3D Lidar Map

- **Radar**
  Radar (Radio Detection and Ranging) uses radio waves to detect objects and measure their distance, speed, and direction. Radar is highly effective in all weather conditions, including fog, rain, and snow, where optical sensors like cameras and LIDAR might struggle. It is particularly useful for detecting moving objects and providing long-range detection.

- **Ultrasonic Sensors**
  Ultrasonic sensors use high-frequency sound waves to detect objects at close range. They are commonly used for parking assistance and low-speed maneuvers, helping to detect obstacles that are very close to the vehicle. Ultrasonic sensors provide accurate distance measurements in short ranges and are effective in various environmental conditions.

1.10.2 Object Detection with CNNs

- **Convolutional Neural Networks (CNNs)**
  CNNs are widely used for image processing tasks due to their ability to automatically learn spatial hierarchies of features [28]. In autonomous vehicles, CNNs are employed for object detection, recognizing and classifying objects such as pedestrians, vehicles, and obstacles from camera images.

  - **One-Stage Object Detection: YOLO**
    YOLO (You Only Look Once) is a real-time object detection system that divides the input image into a grid and applies a single neural network to the entire image [29]. It predicts bounding boxes and class probabilities directly, making it fast and efficient for real-time applications in autonomous driving.
  - **Two-Stage Object Detection: Faster R-CNN**
    Faster R-CNN is a two-stage object detection framework that first generates region proposals and then classifies these proposals [30]. It is known for its accuracy and

is used in scenarios where precision is critical, such as detecting smaller or occluded objects.

1.10.3 Object Tracking

- **Kalman Filters**
  Kalman filters [31] are used for tracking the state of a moving object over time by predicting its future position and updating these predictions with new measurements. They are particularly useful in filtering out noise from sensor data and providing smooth and accurate object trajectories.

- **Deep SORT**
  Deep SORT (Simple Online and Realtime Tracking with a Deep Association Metric) [32] extends the SORT algorithm by incorporating appearance information through a deep learning-based association metric. This enhances the tracker's ability to maintain consistent identities of objects across frames, even in crowded scenes.

1.10.4 Lane Tracking

Lane tracking [19] is a critical functionality in autonomous driving systems, enabling the vehicle to stay within its lane by detecting and following lane markings on the road. This task involves using various sensors, such as cameras, and applying deep learning algorithms to accurately identify lane boundaries and predict the vehicle's path.

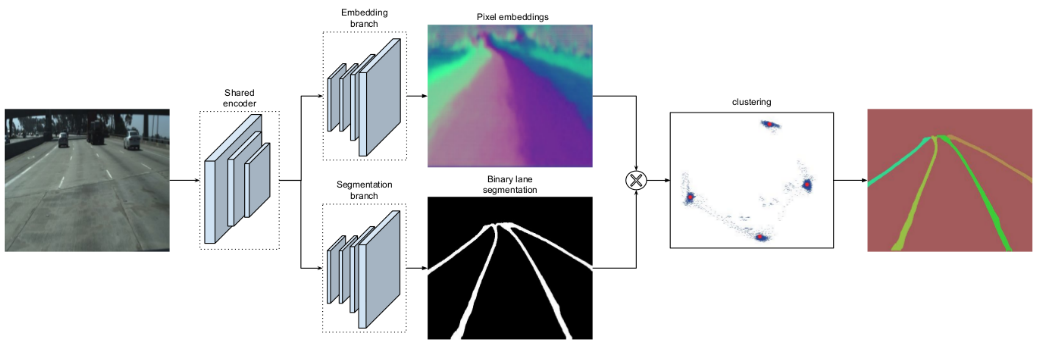


Figure 1.10: Lane Detection and Tracking: LaneNet [4]

1.10.4.1 Deep Learning Architectures for Lane Tracking

- **Convolutional Neural Networks (CNNs):** CNNs [28] are widely used for image processing tasks, including lane detection and tracking. They excel at extracting features from camera images and can be trained to recognize lane markings under various conditions. Architectures such as VGGNet and ResNet have been effectively applied to lane tracking problems.

- **Recurrent Neural Networks (RNNs):** RNNs, particularly Long Short-Term Memory (LSTM) networks [33], are used in lane tracking to capture temporal dependencies in sequences of images. By considering the context of previous frames, RNNs can improve the stability and accuracy of lane tracking over time.

- **Hybrid Approaches:** Combining CNNs and RNNs, hybrid architectures leverage the strengths of both networks. CNNs are used for feature extraction from individual frames, while RNNs handle the temporal aspect, ensuring consistent lane tracking across multiple frames.

1.10.4.2 Examples of Lane Tracking Models

- **LaneNet:** LaneNet [4] is a popular architecture that combines semantic segmentation and instance segmentation to detect lanes and separate them into different instances. This dual approach allows for robust lane detection even in complex driving scenarios.

- **LaneATT:** LaneATT (Lane Attention Network) [34] is a deep learning architecture designed for precise lane detection in autonomous driving. It uses attention mechanisms to focus on important lane features, enhancing localization and prediction of lane markings under varying conditions. LaneATT integrates CNNs with attention modules for robust performance in complex road environments, ensuring high accuracy and efficiency in real-time applications.

Lane tracking systems employing these deep learning architectures can significantly enhance the safety and reliability of autonomous driving by ensuring precise and continuous lane adherence. These models are trained on extensive datasets containing diverse road conditions to achieve robustness and generalization in real-world scenarios.

1.10.5 Obstacle Avoidance

Obstacle avoidance is a critical component of autonomous driving systems, ensuring safe navigation by detecting and maneuvering around obstacles in real-time [19]. It integrates multiple functionalities:

- **Lane Tracking:** Lane tracking provides information about the vehicle's position relative to lane boundaries. This data helps in planning safe trajectories and determining permissible paths around obstacles.

- **Object Detection:** Object detection identifies obstacles such as vehicles, pedestrians, and other objects in the vehicle's path. It utilizes sensors like cameras and LiDAR to perceive the environment and assess potential collision risks.

- **Sensor Fusion:** Sensor fusion combines data from multiple sensors (e.g., cameras, LiDAR, radar) to generate a comprehensive and accurate representation of the surroundings. It enhances obstacle detection by cross-verifying information from different sensor modalities, improving reliability and reducing false alarms.

By integrating lane tracking, object detection, and sensor fusion, autonomous vehicles can effectively detect obstacles in their path, assess potential risks, and make informed decisions to navigate safely through dynamic environments.

1.10.6 Sensor Fusion: Camera and LIDAR Fusion Systems

Sensor fusion involves combining data from multiple sensors to create a more accurate and comprehensive understanding of the environment. In autonomous vehicles, camera and LIDAR fusion are particularly effective as they complement each other's strengths and mitigate weaknesses.

1.10.6.1 Camera and LIDAR Fusion

Cameras provide high-resolution color images, which are excellent for recognizing texture and color but are affected by lighting conditions. LIDAR provides precise 3D spatial information, regardless of lighting. By fusing data from both sensors like figure 1.11 shows, autonomous vehicles can achieve robust object detection and environmental mapping [5].

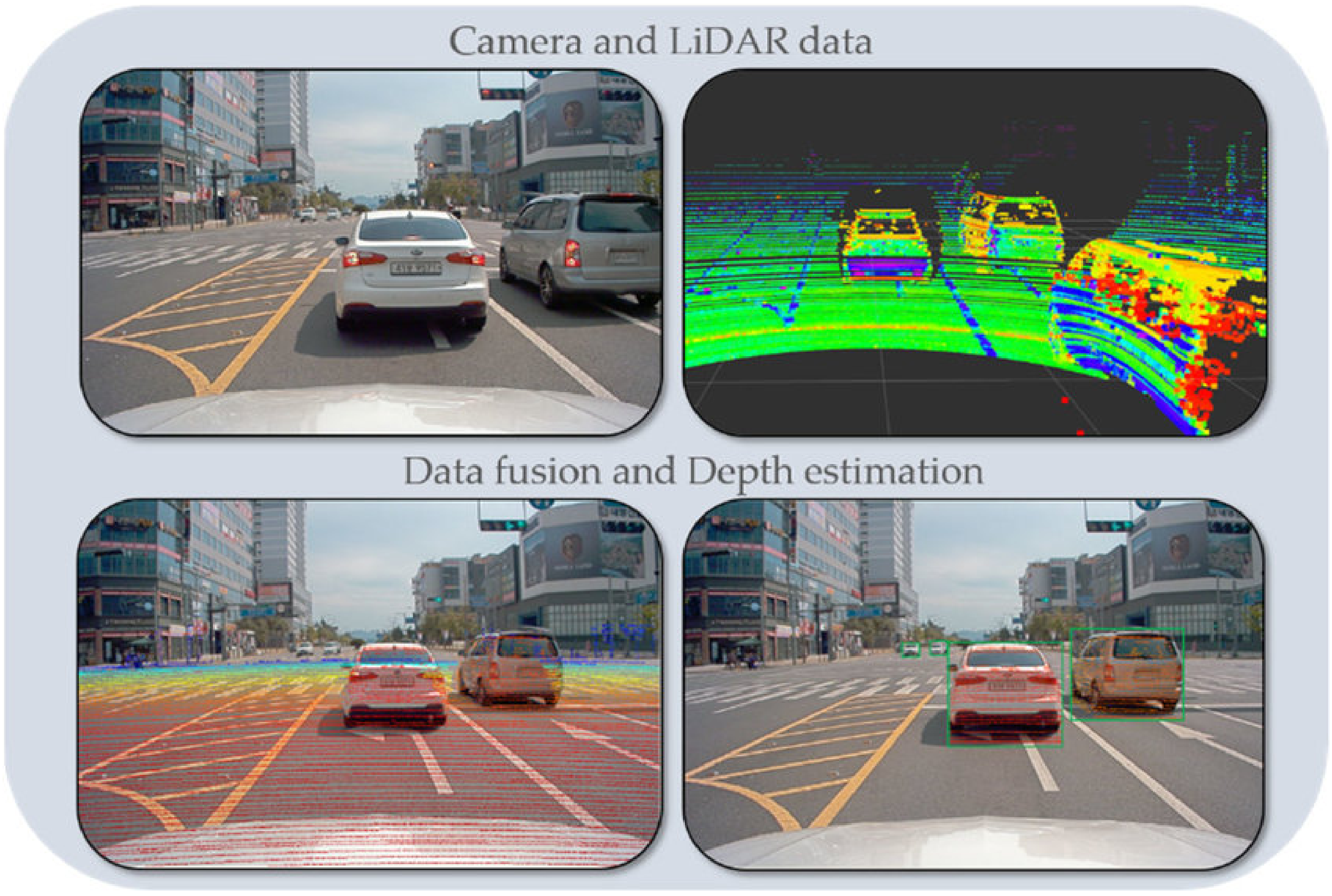


Figure 1.11: Camera and Lidar Fusion [5]

1.10.6.2 SLAM and Its Relation to Sensor Fusion

Simultaneous Localization and Mapping (SLAM) [35] is a critical technique that enables autonomous vehicles to construct a map of an unknown environment while simultaneously keeping track of their location within it. SLAM integrates data from various sensors, including cameras, LIDAR, and IMUs (Inertial Measurement Units), to build and update maps in real time.

1.10.6.3 How SLAM Works

SLAM algorithms typically involve two main processes: mapping and localization. Mapping involves creating a representation of the environment, while localization determines the vehicle's position within this map. Advanced SLAM systems use sensor fusion techniques to combine data from multiple sources, improving accuracy and reliability.

1.10.6.4 Integration with Sensor Fusion

Sensor fusion enhances SLAM by providing more comprehensive data inputs, leading to more accurate maps and better localization performance. For instance, combining camera images with LIDAR point clouds can help resolve ambiguities and improve the robustness of SLAM systems in complex environments.

1.11 Embedded Systems for Autonomous Driving

Autonomous vehicles rely heavily on embedded systems for processing sensor data, making real-time decisions, and executing control commands. Key hardware components include microcontrollers and single-board computers that provide the necessary computational power for these tasks. This section identifies popular platforms such as Arduino, Raspberry Pi, and Jetson Nano, and evaluates their suitability for deep learning tasks. Additionally, it discusses NVIDIA's deep learning accelerators, including DeepStream, CUDA, and TensorRT.

1.11.1 Arduino

Arduino is an open-source microcontroller platform known for its simplicity and ease of use. It is widely used in educational projects and prototyping due to its low cost and extensive community support. However, Arduino lacks the computational power required for complex deep learning tasks and is not typically used in autonomous driving applications where real-time processing is critical [36].

1.11.2 Raspberry Pi

Raspberry Pi [37] is a versatile single-board computer that offers greater computational capabilities than Arduino. It supports various operating systems and can run a range of applications, making it suitable for intermediate projects and some real-time processing tasks. While more powerful than Arduino, Raspberry Pi still falls short in handling intensive deep learning models due to its limited processing power and lack of specialized accelerators.

1.11.3 Nvidia Jetson Nano

Jetson Nano, developed by NVIDIA, is a single-board computer specifically designed for AI and deep learning applications . It features a powerful GPU that supports CUDA, making it capable of running complex neural networks and handling high-throughput data processing tasks in real time. Jetson Nano is highly suitable for autonomous driving applications due to its ability to efficiently execute deep learning algorithms and process multiple sensor inputs simultaneously [38].

1.12 Deep Learning Accelerators

- **CUDA:** CUDA (Compute Unified Device Architecture) [39] is a parallel computing platform and programming model developed by NVIDIA. It allows developers to leverage the power of NVIDIA GPUs for general-purpose computing, significantly accelerating deep learning model training and inference.

- **TensorRT:** TensorRT is an SDK from NVIDIA designed to optimize deep learning models for inference on NVIDIA hardware [40]. It provides mixed precision, layer fusion, and kernel auto-tuning to enhance performance and reduce latency.
- **DeepStream:** DeepStream is a streaming analytics toolkit from NVIDIA that enables real-time video and image processing using deep learning. It is optimized for NVIDIA GPUs and integrates well with CUDA and TensorRT, making it ideal for autonomous vehicle applications that require high-speed data processing and analysis [41].

Overall, for deep learning tasks in autonomous driving, Jetson Nano stands out as the best option among the discussed hardware platforms due to its specialized GPU and support for NVIDIA's deep learning accelerators. These accelerators, including CUDA, TensorRT, and DeepStream, provide the necessary tools to optimize and accelerate deep learning models, ensuring efficient and real-time performance.

1.13 RC Cars as Prototypes for Self-Driving Cars

Remote-controlled (RC) cars serve as valuable prototypes in the development and testing of autonomous driving technologies, figure 1.12. These scaled-down vehicles offer a cost-effective and manageable platform for exploring various algorithms and hardware configurations before scaling up to full-sized autonomous vehicles [42].

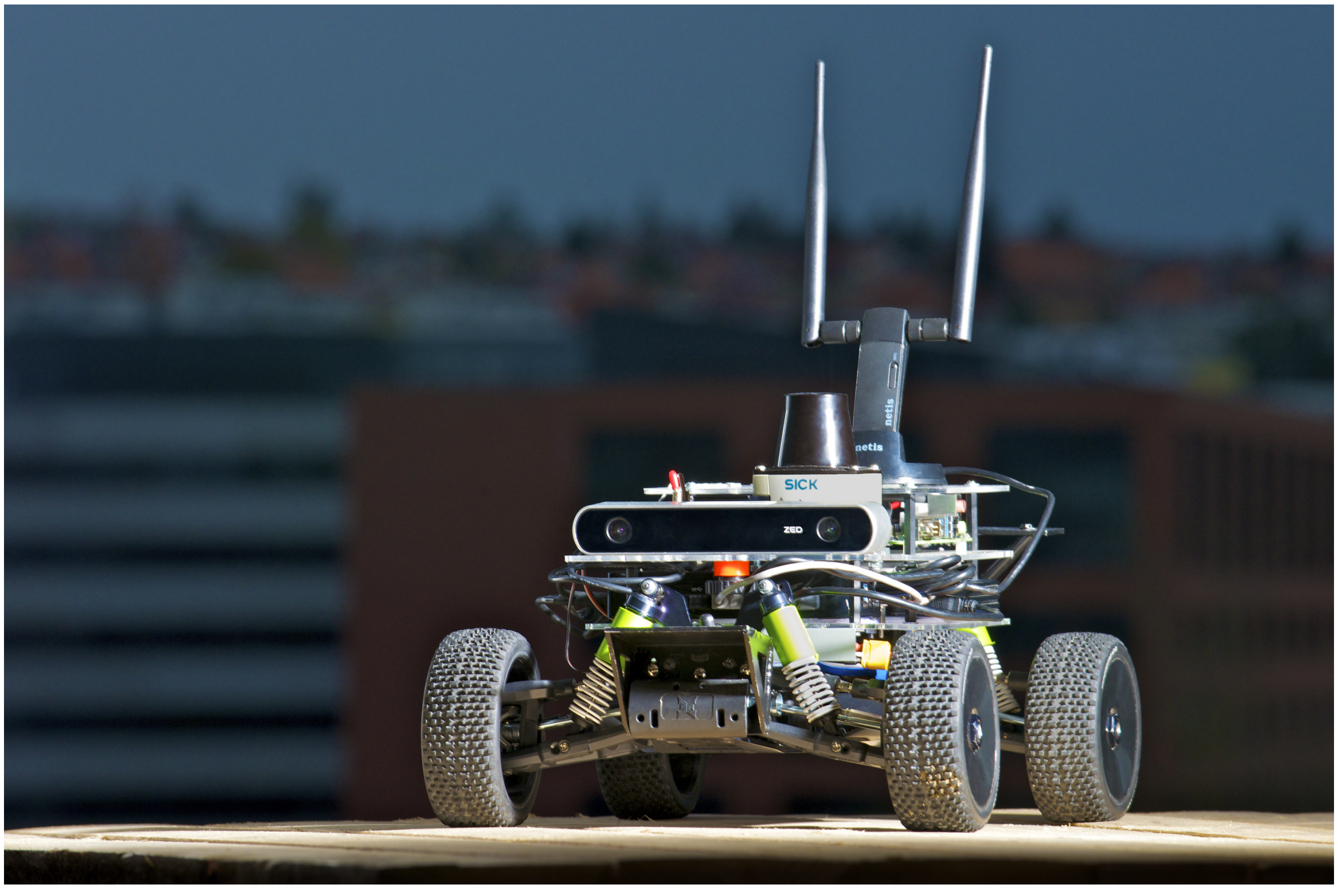


Figure 1.12: Example of an autonomous RC car

1.13.1 Implementation and Modifications

RC cars are typically equipped with miniature versions of sensors used in full-scale autonomous vehicles, such as cameras, LiDAR, and inertial measurement units (IMUs). These sensors enable perception of the vehicle's surroundings, crucial for navigation and obstacle avoidance. The on-board computational units, often microcontrollers or small embedded systems like Raspberry Pi or Arduino, process sensor data and execute control algorithms [43].

Modifications to RC cars include integrating actuators for steering, throttle, and braking, which are controlled autonomously based on sensor inputs. Additionally, GPS modules may be added to enhance localization accuracy, though this is less common due to the smaller scale and limited outdoor GPS reception quality.

1.13.2 Testing and Validation

RC cars provide a controlled environment for testing autonomous driving algorithms and hardware configurations. Researchers can simulate various driving scenarios, such as lane following, obstacle avoidance, and path planning, in a safe and repeatable manner. Real-time data logging allows for performance evaluation and debugging of algorithms, providing insights into their robustness and reliability.

Moreover, RC cars facilitate rapid prototyping and iterative development cycles. Researchers can quickly implement and test new algorithms, validate their effectiveness, and iterate on improvements before transitioning to larger-scale prototypes or real-world applications.

1.13.3 Relevance in Autonomous Driving Research

Despite their miniature size, RC cars play a significant role in advancing autonomous driving research. They serve as practical tools for exploring fundamental concepts in perception, control, and decision-making algorithms. Insights gained from RC car experiments inform the development of more sophisticated autonomous systems capable of operating in real-world environments.

In summary, RC cars provide a foundational platform for experimenting with and refining autonomous driving technologies. Their flexibility, cost-effectiveness, and scalability make them indispensable in the early stages of autonomous vehicle development, paving the way for innovations in the field of self-driving cars.

1.14 Self-Driving Simulators and Their Implementation

Self-driving simulators are essential tools in the development and testing of autonomous driving systems as shown in figure 1.13, offering a virtual environment to simulate real-world driving scenarios. These simulations provide a cost-effective and safe alternative to physical testing, allowing researchers to validate algorithms, assess vehicle performance, and train AI models under controlled conditions.

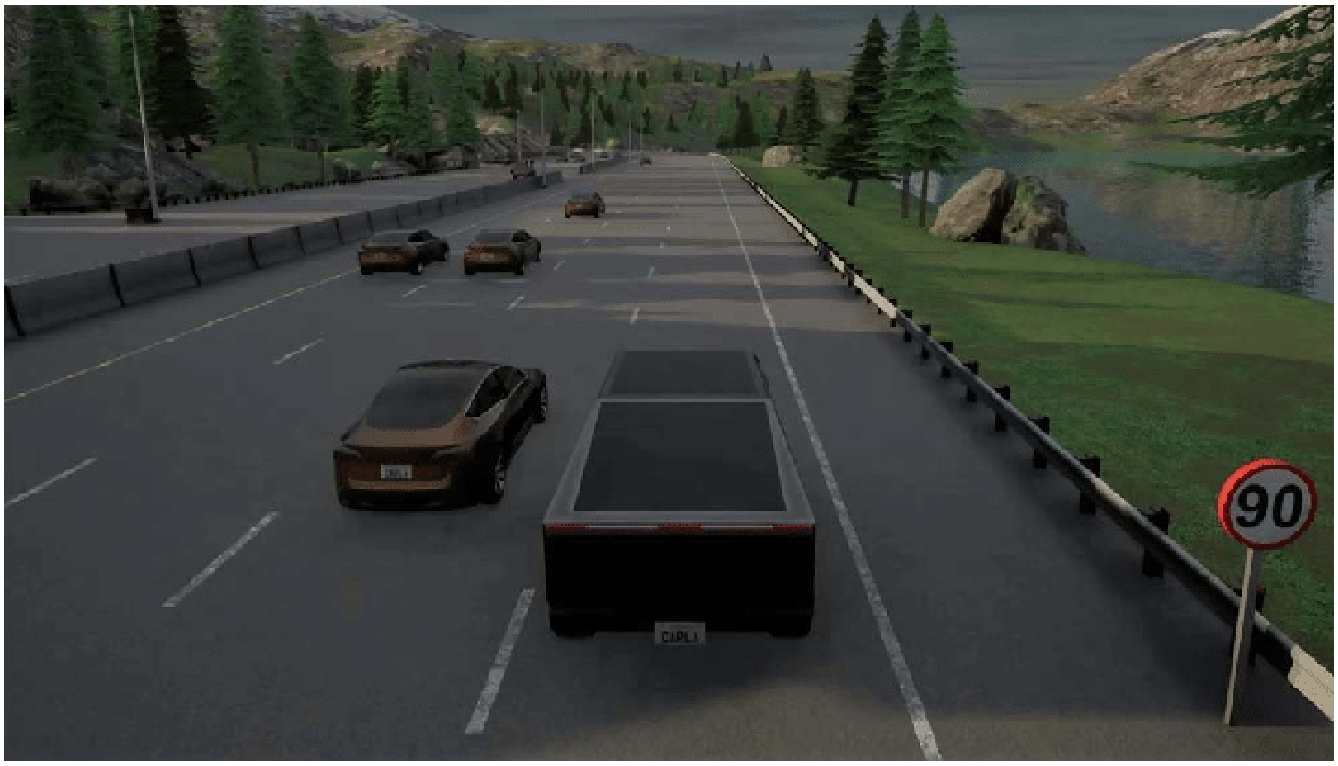


Figure 1.13: Simulation environment: CARLA simulator [6]

1.14.1 Implementation

Self-driving simulators replicate various aspects of the driving environment, including road layouts, traffic scenarios, weather conditions, and pedestrian behavior. They typically integrate realistic physics engines to simulate vehicle dynamics accurately. Sensor models such as cameras, LiDAR, and radar emulate sensor data processing in real-time, enabling perception algorithms to interpret virtual surroundings.

Simulation platforms often support scripting languages (e.g., Python, C++) for developing and customizing scenarios. This flexibility allows researchers to create complex driving scenarios, adjust environmental parameters, and modify vehicle dynamics to evaluate different aspects of autonomous driving systems.

1.14.2 Why Simulations?

Simulations offer several advantages over physical testing in autonomous driving research:

- **Cost-effectiveness:** Simulations reduce costs associated with vehicle procurement, maintenance, and testing infrastructure. Researchers can conduct extensive testing without the constraints of physical resources.
- **Safety:** Virtual environments eliminate risks to personnel and property associated with real-world testing, especially during early development stages.
- **Scalability and Reproducibility:** Simulations allow for rapid iteration and scalability. Researchers can reproduce scenarios consistently, facilitating benchmarking and comparison of algorithm performance.
- **Scenario Diversity:** Simulators provide the flexibility to test a wide range of driving scenarios, including rare or hazardous conditions that are impractical to replicate in physical tests.
- **Algorithm Development:** Researchers can iterate algorithms quickly in simulations, fine-tuning parameters and evaluating performance metrics under diverse conditions before deployment in real-world settings.

1.14.3 Examples of Simulators

Several prominent simulators used in autonomous driving research include:

- **CARLA**: CARLA [6] is an open-source autonomous driving simulator. It was built from scratch to serve as a modular and flexible API to address a range of tasks involved in the problem of autonomous driving. One of the main goals of CARLA is to help democratize autonomous driving R&D, serving as a tool that can be easily accessed and customized by users. To do so, the simulator has to meet the requirements of different use cases within the general problem of driving (e.g. learning driving policies, training perception algorithms, etc.).
- **LGSVL Simulator**: The LGSVL Simulator [44] specializes in providing a robust environment for testing autonomous vehicle algorithms within realistic urban settings. It leverages

the Unity game engine to offer advanced visualizations and accurate physics simulations.This simulator is tailored for researchers and developers working on autonomous driving systems, offering customizable urban environments with dynamic traffic scenarios. It supports a variety of sensors, including cameras, LiDAR, and radar, allowing for comprehensive testing and validation of perception and navigation algorithms.

Overall, self-driving simulators are indispensable tools for accelerating the development and validation of autonomous driving technologies. Their ability to replicate complex driving scenarios safely and efficiently makes them a cornerstone in advancing the capabilities and reliability of autonomous vehicles.

1.15 Conclusion

The future of autonomous vehicles promises to transform transportation through advancements in technology. Achieving Level 5 autonomy depends on progress in machine learning, deep learning, and hardware capabilities.

Enhanced object detection and tracking technologies, powered by CNNs and sensor fusion, have improved autonomous systems' ability to navigate their environment. Hardware platforms like NVIDIA Jetson Nano, along with accelerators such as NVIDIA DeepStream, CUDA, and TensorRT, provide the computational power needed for real-time decision-making.

Self-driving simulators like CARLA and LGSVL are essential for developing and validating autonomous driving systems, allowing for the testing of complex scenarios before real-world application. Using RC cars as prototypes offers cost-effective experimentation and innovation, aiding the development of full-scale autonomous vehicles.

As advanced algorithms, powerful hardware, and realistic simulations converge, the evolution toward fully autonomous vehicles will continue. This promises significant societal benefits, including enhanced safety and increased accessibility.

In the next chapter, we will explore state-of-the-art methods for object detection and tracking in autonomous driving. These methods are crucial for enabling accurate perception and interaction with the environment, ensuring safe and efficient navigation.

Chapter 2

State of the Art

Chapter 2

State of the Art

2.1 Introduction

In recent years, the field of autonomous driving has seen significant progress, highlighting the critical importance of multi-sensor fusion for enhanced object detection and tracking. Decision-making remains a undamental pillar for ensuring the safety and efficiency of autonomous vehicles. In this chapter, we will explore state-of-the-art of multi-sensor fusion methods, which play an essential role in the object detection and tracking process in autonomous cars. We will analyze these methods by highlighting their differences and conduct a comprehensive comparison based on established criteria

2.2 Problematic

How to develop an optimal multi-sensor fusion system for object detection and tracking in complex autonomous driving environments, taking into account the real-time and accuracy constraints required to ensure the safety and efficiency of autonomous vehicles? Specifically, How can data fusion from multiple sensors, such as cameras and LiDAR, be optimized to improve object detection and tracking in autonomous driving systems, contributing to more reliable and secure decision-making to ensure efficient autonomous driving?

2.3 Why This Research ?

The field of autonomous driving is rapidly expanding, and it is imperative to address a crucial aspect such as sensor fusion. Autonomous cars depend on a combination of cameras, LiDAR, and other sensors [19], each offering a unique perspective of the environment. The fusion of these diverse data sets allows for the creation of a holistic perception, leading to more precise and reliable decisions.

Our research theme on multi-sensor fusion aims to tackle this challenge by exploring advanced methods to enhance object detection and tracking. We place particular emphasis on integrating data from cameras and LiDAR, as this approach holds the promise of opening up new possibilities for more efficient and safer autonomous driving. We are confident that this research will significantly contribute to the advancement of autonomous driving technology and its extensive implementation, thereby creating reliable and secure driving systems for the future.

Autonomous vehicles depend on additional sensors with complementary measurement principles to enhance robustness and reliability through sensor fusion. We will study the LiDAR sensor, its various types, as well as the relevant criteria for sensor selection. Furthermore, we will learn how to detect objects in a 3D LiDAR point cloud using a deep learning approach, and then evaluate detection performance using a set of metrics.

Next, we will address the fusion of detections from the camera and LiDAR, as well as object tracking over time using an extended Kalman filter.

2.4 Study of Existing Work in Autonomous Driving

In recent years, the autonomous driving sector has seen significant advancements, highlighting the critical role of object detection and the integration of various technologies for enhanced tracking and decision-making, vital for the safe and efficient functioning of self-driving cars. This chapter delves into the latest developments in object detection, 3D object detection, multi-object tracking (MOT), sensor fusion, and lane tracking. We will explore a wide range of techniques, emphasizing their advantages, limitations, and practical applications in real-world autonomous driving scenarios.

Our research will focus on improving object and obstacle detection and tracking. Additionally, we will explore complementary methods such as sensor fusion and lane tracking. These techniques will aid us in conducting experiments directly on hardware, facilitating practical validation of our research findings.

2.4.1 Study of existing work in object detection in autonomous driving

1. A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS (2023)

This paper [45] presents an extensive examination of the evolution and advancements of the YOLO (You Only Look Once) object detection models. The authors analyze various versions of YOLO, detailing their improvements and impact on the field of computer vision, particularly in real-time object detection applications. the figure 2.1 below shows the Yolo official architecture:

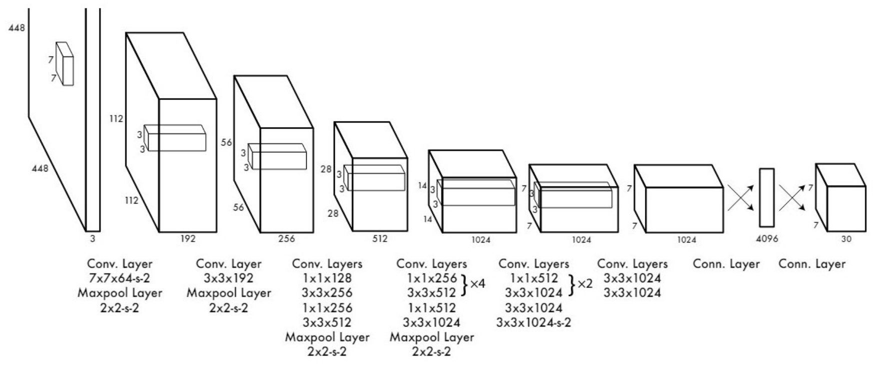


Figure 2.1: Yolo Architecture

Object detection models are evaluated using key metrics such as accuracy, speed, and efficiency. The paper highlights the strengths and limitations of each YOLO version, from YOLOv1 to YOLOv8 and YOLO-NAS, as follows:

- **YOLOv1 to YOLOv3:** YOLOv1 introduced a single regression problem approach for object detection, predicting bounding boxes and class probabilities directly from full images. YOLOv2 improved with batch normalization, high-resolution classifiers, and anchor boxes, while YOLOv3 further enhanced detection with a deeper network and feature pyramid networks for multi-scale detection.

- **YOLOv4 and YOLOv5:** YOLOv4 incorporated advanced techniques like the Cross-Stage Partial (CSP) network, new data augmentation methods, and other optimizations, resulting in more efficient and accurate models. YOLOv5, while not officially released as a research paper, brought practical improvements in usability, speed optimization, and ease of deployment.
- **YOLOv6 to YOLOv8:** These versions integrated state-of-the-art methodologies, including attention mechanisms and improved loss functions, progressively reducing model size while enhancing performance, making them suitable for resource-constrained environments.
- **YOLO-NAS:** The latest development utilizes Neural Architecture Search to discover optimal architectures for object detection tasks, balancing performance and efficiency better than manually designed models.

The YOLO family of models has set benchmarks in speed and accuracy as shown in figure 2.2, making them invaluable for real-time applications such as autonomous driving, surveillance, and robotics. The continuous evolution of these models highlights their profound influence on computer vision and underscores the importance of ongoing research to achieve higher levels of autonomy and precision in various technological domains.

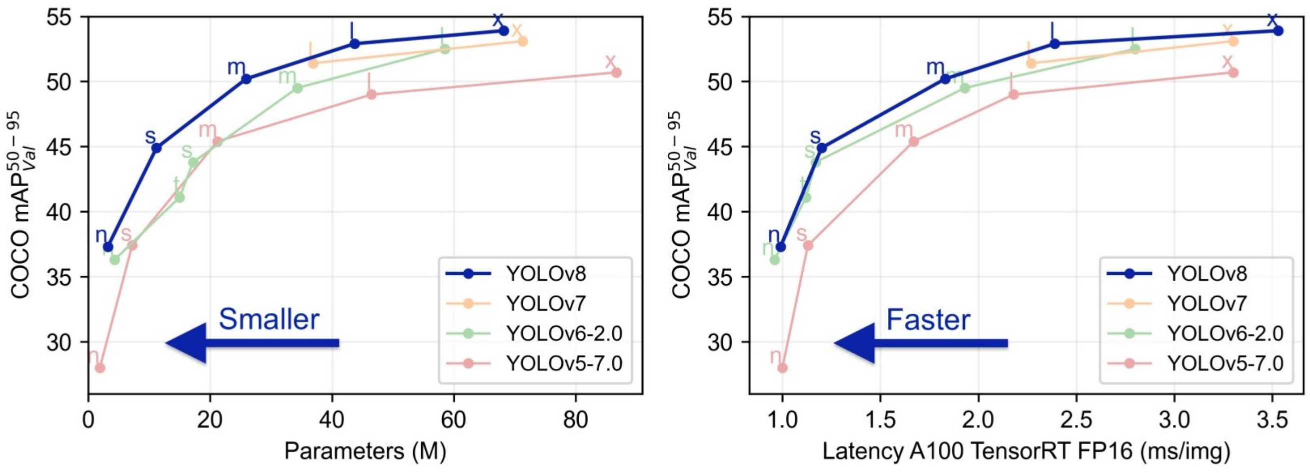


Figure 2.2: Performance comparison of YOLO object detection models

2. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks (2016)

This seminal paper [30] introduces Faster R-CNN, a groundbreaking framework that advances real-time object detection through the integration of Region Proposal Networks (RPN) with Convolutional Neural Networks (CNN). The authors propose a method that significantly accelerates object detection by generating region proposals directly within the network, eliminating the need for a separate region proposal step used in previous approaches as shown in the figure 2.3 below.

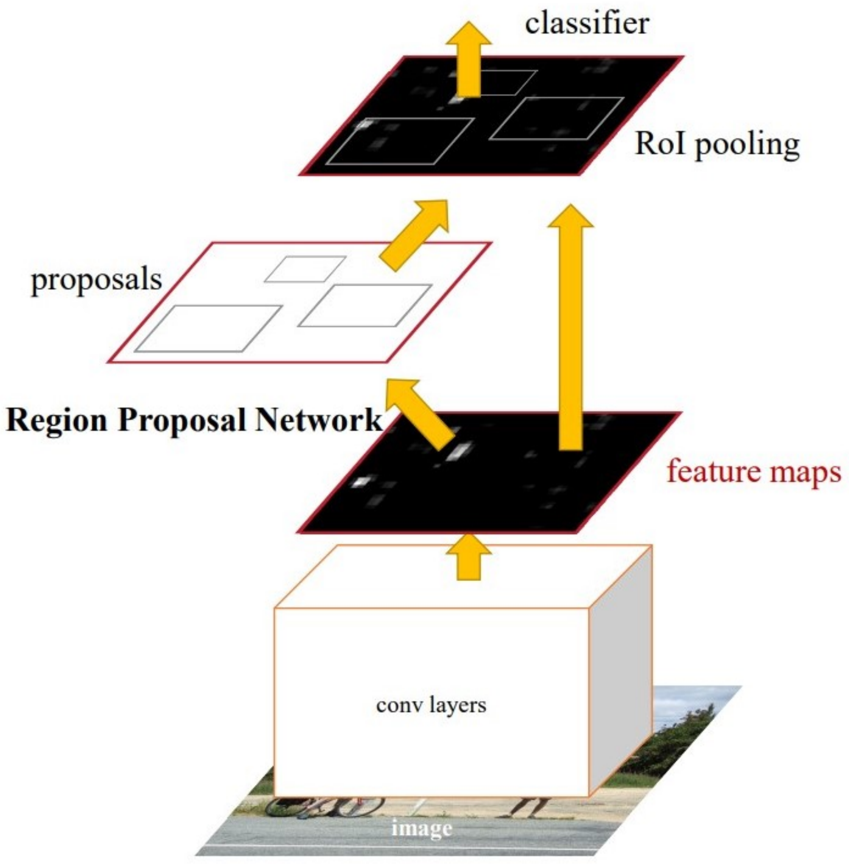


Figure 2.3: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

Key contributions and findings of this paper include:

- **Region Proposal Network (RPN):** The RPN shares convolutional layers with the detection network, allowing nearly cost-free region proposals. It simultaneously predicts object bounds and scores, achieving high efficiency and accuracy.
- **Integration with CNNs:** By combining RPN with Fast R-CNN, the authors create a unified network that enables end-to-end training and testing, significantly improving the speed and accuracy of object detection systems.
- **Performance Improvements:** Faster R-CNN demonstrates state-of-the-art performance on multiple benchmarks, including KITTI Dataset [20], and is capable of processing images at near real-time frame rates while maintaining high detection accuracy.
- **Scalability and Flexibility:** The architecture is scalable to different network depths and flexible in adapting to various tasks, making it a versatile choice for real-time object detection applications.

Faster R-CNN's innovation in combining region proposal and detection within a single network has set a new standard in the field of object detection. Its impact extends across numerous applications, from autonomous driving to surveillance, showcasing the potential of deep learning techniques in achieving real-time, accurate object detection.

3. Comparative analysis of deep learning image detection algorithms (2021)

In this research paper [46], the authors conduct a comparative analysis of three prominent image processing algorithms—Single Shot Detection (SSD), Faster Region-based Convolutional Neural Networks (Faster R-CNN), and You Only Look Once (YOLO). Using the Microsoft COCO dataset, they assess these algorithms' performance based on accuracy, precision, and F1 score to determine the fastest and most efficient approach for object detection. This study aims to identify strengths and limitations within these methodologies, offering insights into their suitability for practical applications in image analysis and processing.

Object detection models are evaluated using key metrics like 'Average Precision,' F1 score, and COCO metrics API. These metrics allow for a comprehensive comparison of the performance among the three algorithms as follow:

- SSD underperforms compared to Faster R-CNN due to its reliance on higher resolution layers for small object detection, hindering classification. Additionally, SSD's complexity in data augmentation demands substantial training data, potentially increasing cost and time for training
- Faster R-CNN offers high accuracy but suffers from time complexity, notably slower than YOLO. Despite advancements, its multi-pass approach and components like ROI pooling and RPN can create bottlenecks within the algorithm
- YOLOv3 showcased improvements but faced accuracy limitations in smaller image analysis; YOLOv4 addressed shortcomings with optimized models like CSPDarknet-53, offering enhanced speed and accuracy while analyzing different object sizes, as indicated by precision-recall curves using the COCO [49] metric API.

Finally, the authors underscore the pivotal roles of Yolo-v3, SSD, and Faster RCNN in CNN-based object detection. Yolo-v3 emerges as the fastest and most efficient among the three, notably displaying minimal false detections, while SSD balances speed and accuracy. Conversely, Faster RCNN showcases higher accuracy but lacks the efficiency for real-time processing, highlighting the distinct strengths of each algorithm in object detection tasks.

4. SSD: Single Shot MultiBox Detector (2016)

This paper [47] introduces the Single Shot MultiBox Detector (SSD), an object detection framework that combines the best aspects of previous approaches. SSD eliminates proposal generation and subsequent pixel or feature resampling stages, simplifying the pipeline and improving speed while maintaining high accuracy as indicated by the SSD network architecture in the figure 2.4 below.

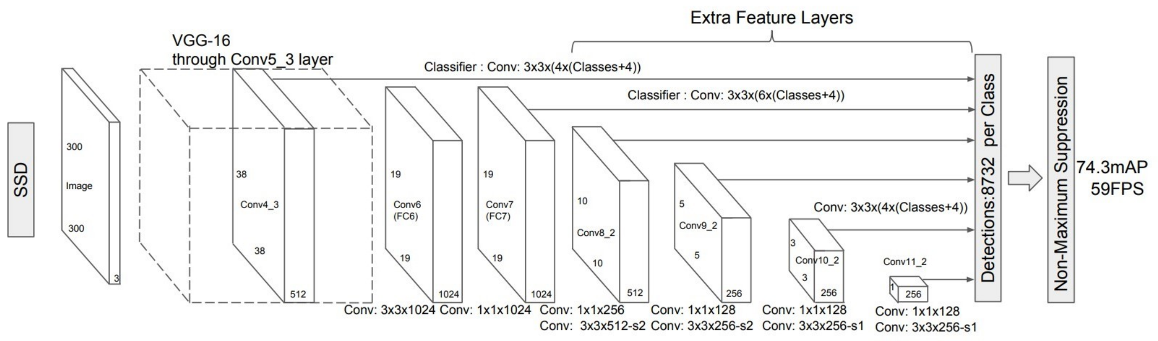


Figure 2.4: SSD network architecture

Key features include:

- **Single Shot Detection:** Detects objects in a single forward pass of the network, making it significantly faster.
- **Multi-scale Feature Maps:** Uses multiple feature maps at different scales to detect objects of various sizes.
- **Default Boxes:** Employs a set of default boxes of different aspect ratios at each feature map location, enhancing the detection of objects with varying shapes.

SSD achieves impressive performance on benchmarks like PASCAL VOC [50] , COCO [49], and ILSVRC [51], demonstrating a good balance between speed and accuracy.

5. FPN: Feature Pyramid Networks for Object Detection (2017)

In this influential paper [48], the authors propose the Feature Pyramid Network (FPN), a framework designed to enhance object detection performance by leveraging the inherent multi-scale, pyramidal hierarchy of deep convolutional networks. The FPN introduces a top-down architecture with lateral connections for building high-level semantic feature maps at various scales, significantly improving detection accuracy across different object sizes as shown in the figure 2.5 below.

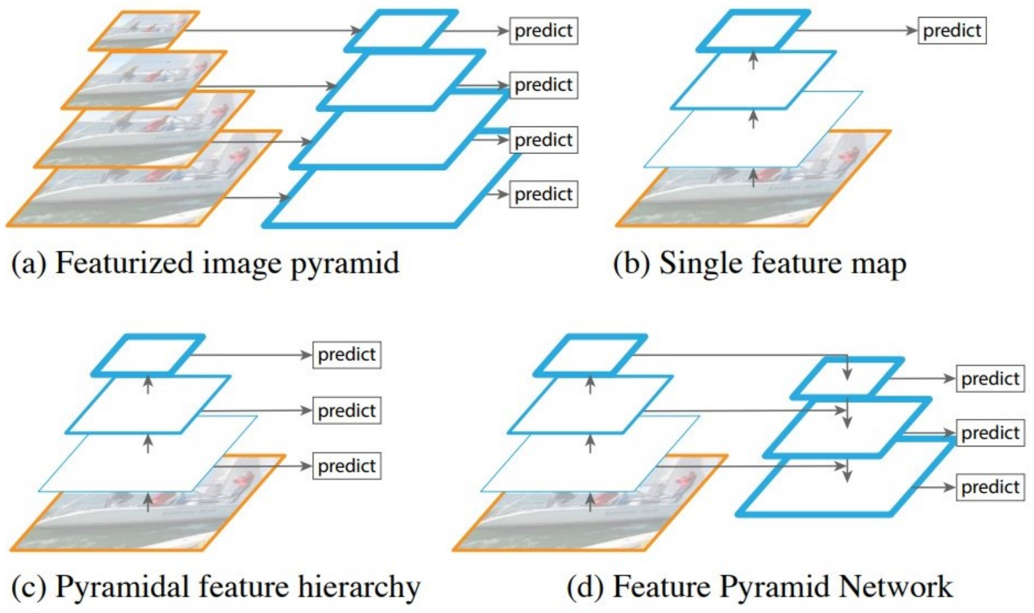

predict
(a) Featurized image pyramid
(b) Single feature map
(c) Pyramidal feature hierarchy
(d) Feature Pyramid Network

Figure 2.5: SSD network architecture

Key contributions and findings from the paper include:

- **Top-Down Pathway:** The FPN employs a top-down pathway to create higher resolution feature maps with strong semantics, which enhances the detection of smaller objects.
- **Lateral Connections:** Lateral connections are used to merge higher resolution features with semantically stronger, lower resolution features, ensuring that the feature maps at all pyramid levels are semantically rich.
- **Improved Accuracy:** Experiments demonstrate that FPNs, when integrated with baseline detectors like Faster R-CNN, significantly improve accuracy, especially for small and medium-sized objects.
- **Efficiency:** Despite the added complexity of the top-down pathway and lateral connections, the FPN architecture does not introduce a significant computational overhead, making it efficient for practical applications.

6. Comparative analysis of multiple YOLO-based target detectors and trackers for ADAS in edge devices (2023)

In this research paper [49], the authors presents a comprehensive analysis of several YOLO-based object detectors and trackers for ADAS applications on edge devices. The authors explore the trade-offs between accuracy, inference speed, and resource consumption, and they provide valuable insights into the performance of different models on real-world dataset.

Here are some of the key takeaways from the paper:

- **YOLOR-CSP emerges as a promising detector for ADAS on edge devices.** It balances accuracy and speed effectively, achieving high mAP achieving a mAP@50 of 69.70%

scores while maintaining real-time inference rates on the NVIDIA Jetson AGX Xavier platform.

- **DeepSORT performs well as a tracker.** It demonstrates robust performance in various scenarios, including fast-moving targets and occlusions. However, it is sensitive to longer detector intervals, where NvDCF might be a better choice.

- **Multi-camera inference can enhance tracking performance.** By utilizing multiple cameras with longer detector intervals, improved tracking accuracy can be achieved, especially with trackers like NvDCF that are less sensitive to these intervals.

- **Dataset quality significantly impacts model performance.** The study highlights the importance of using diverse and challenging datasets for training object detectors and trackers.

The paper also sheds light on several limitations and challenges in this area:

- **Limited standardization in tracking metrics.** Currently, there is no universally accepted standard for evaluating tracking performance, which makes it difficult to compare different models directly.

- **Need for more comprehensive datasets.** Existing datasets often lack diversity in challenging scenarios, which can limit the generalization ability of models.

- **Hardware limitations can hinder real-time performance.** While edge devices are becoming more powerful, resource constraints can still pose challenges for complex models.

Overall, this paper provides valuable insights into the use of YOLO-based object detectors and trackers for ADAS applications on edge devices. The findings can guide researchers and developers in their efforts to create robust and efficient systems for autonomous driving and other safety-critical applications.

7. EnsembleNet: a hybrid approach for vehicle detection and estimation of traffic density based on faster R-CNN and YOLO models (2023)

In this research paper [50], the authors focus on traffic congestion due to static regulations and emphasize the importance of estimating traffic density for efficient management. They explore vehicle recognition methods like motion analysis, handcrafted features, and CNN-based approaches, leveraging datasets such as MB7500, KITTI, and FLIR. This leads to the development of a hybrid Faster R-CNN and YOLO model evaluated against base estimators for traffic density estimation.

The paper introduces an ensemble model, merging Faster R-CNN and YOLOv5, aiming to improve overall detection accuracy. Faster R-CNN's tendency for multiple detections and YOLO's struggle with crowded or small objects prompt this approach. Ensemble learning, combining model results, is implemented via a majority voting approach. Predictions from Faster R-CNN and YOLOv5, represented by P(faster) and P(yolo) respectively, are compared based on bounding box coordinates; if predictions align within a threshold, indicating agreement on a vehicle, confidence scores determine the final bounding box. The difference between the two confidence scores assists in retaining a single bounding box, refining the ensemble model's detection output.

Finally, the research introduces EnsembleNet, an ensemble-based deep learning model merging Faster R-CNN and YOLOv5 for vehicle detection. Utilizing majority voting, this hybrid system improves overall predictions, compensating for YOLOv5's limitations in dense traffic scenarios and Faster R-CNN's efficiency in dense images. Despite increased computational time, the hybrid model enhances accuracy and includes traffic density estimation. However, addressing the high computational or inference time remains pivotal for practical real-time applications.

2.4.2 Study of existing work in Lane Detection and Trakcing in autonomous driving

Lane detection plays a crucial role in autonomous driving, providing essential information about road structure and vehicle positioning. Over the years, researchers have proposed various approaches to tackle this challenge, each with its own strengths and weaknesses [19].

Several notable CNN architectures have emerged for lane detection, each offering unique advantages:

- **Segmentation-based:** Models like DeepLab and U-Net treat lane detection as a semantic segmentation problem, directly predicting pixel-wise lane class labels.

- **Regression-based:** Approaches like LaneNet and BASNet utilize an encoder-decoder architecture to extract features and regress lane line coordinates directly.

- **Anchor-based:** Models like LaneATT employ anchors and learnable offsets to predict refined lane line locations, achieving impressive efficiency and accuracy.

Choosing the right approach and the optimal choice for our specific application depends on various factors, including:

- **Performance requirements:** Accuracy, speed, and robustness are crucial considerations.

- **Computational resources:** Some models are resource-intensive, while others are tailored for edge devices.

- **Data availability:** Training deep learning models often requires significant amounts of data.

This section critically reviews several state-of-the-art papers exploring diverse lane detection methodologies.

1. LaneNet: Real-Time Lane Detection Networks for Autonomous Driving(2018)

Published in 2018, the ”LaneNet” paper [4] by Wang and Ren introduced a groundbreaking real-time lane detection network tailored for autonomous driving applications. This network employs a two-stage architecture designed to maximize accuracy and efficiency. In the first stage, spatial and semantic features are extracted from input images using an encoder-decoder network. The second stage utilizes a pixel-wise prediction approach, enhanced by learnable anchors and offsets, to precisely localize lane pixels. This design choice, complemented by multi-level feature fusion and a customized loss function combining binary cross-entropy and L1 loss, contributes to LaneNet's high-performance capabilities.

LaneNet excels in real-time processing, achieving robust lane detection across diverse environmental conditions. However, its effectiveness hinges on substantial amounts of training data, which may limit its deployment on resource-constrained hardware platforms. Despite these considerations, LaneNet has significantly influenced the field of lane detection, serving as a foundational model that has inspired numerous subsequent advancements. Its modular architecture and emphasis on real-time performance continue to drive innovation in autonomous driving research and development.

2. Keep your Eyes on the Lane: Real-time Attention-guided Lane Detection(2021)

The paper ”Keep your Eyes on the Lane: Real-time Attention-guided Lane Detection” [34] introduces LaneATT, a model recognized for its efficiency and real-time performance, particularly suited for edge devices. This model employs an encoder-decoder architecture to extract features from input images and predict lane probabilities. What sets LaneATT apart is its integration of an attention mechanism within the decoder, enhancing the localization of lane lines by focusing on pertinent image regions. Its lightweight design, achieved through efficient building blocks and knowledge distillation techniques, enables real-time operation on resource-constrained devices.

LaneATT further distinguishes itself with an adaptive loss function that adjusts based on lane marking confidence predictions, enhancing robustness in challenging scenarios. However, compared to more complex models, LaneATT may not achieve the highest accuracy without additional preprocessing for lane marking confidence estimation. Despite this, LaneATT has made significant contributions by providing an effective solution for real-time lane detection, particularly where efficiency and accuracy are paramount. Its attention mechanism and lightweight design have spurred further advancements in efficient and robust lane detection methodologies.

2.4.3 Study of existing work in 3D object detection in autonomous driving

1. RTM3D: Real-time Monocular 3D Detection from Object Keypoints for Autonomous Driving (2020)

The paper [51] "RTM3D" presents an innovative approach to real-time monocular 3D object detection for autonomous driving. This method focuses on detecting 3D bounding boxes from a single camera image by leveraging object keypoints. Figure 2.6 shows an overview of the proposed keypoint detection architecture.

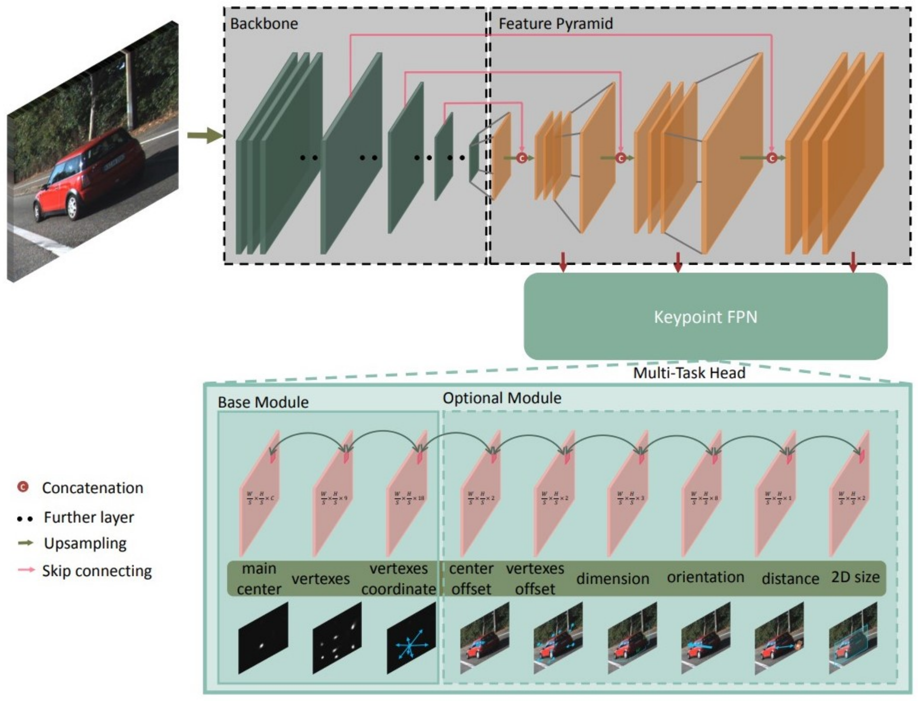


Figure 2.6: An overview of proposed keypoint detection architecture

Key contributions and findings from the paper include:

- **Monocular 3D Detection:** RTM3D uses a single RGB image to predict 3D bounding boxes, avoiding the complexity and cost associated with LiDAR and stereo camera setups.
- **Object Keypoints:** The approach identifies nine keypoints for each object, corresponding to the eight corners of the 3D bounding box and the center. These keypoints are used to derive the 3D dimensions, location, and orientation of the object.
- **Geometric Constraints:** The paper introduces a novel geometric constraint for keypoint predictions. The 2D-3D geometric relationship ensures accurate and consistent keypoint localization.
- **Loss Function:** The loss function in RTM3D is designed to minimize the reprojection error of the 3D bounding box corners and the center keypoint. It combines the keypoint heatmap loss, the offset loss, and the dimension loss.

- **Backbone Network:** RTM3D utilizes FPNResNet18 as its backbone network. FPN (Feature Pyramid Network) enhances feature extraction at multiple scales, which is crucial for detecting objects of varying sizes in the image. ResNet18, with its relatively lightweight architecture, provides a good balance between accuracy and computational efficiency, making it suitable for real-time applications.

- **Real-time Performance:** The RTM3D model is lightweight and optimized for real-time inference, achieving a balance between accuracy and speed, making it suitable for deployment in autonomous driving systems.

- **Experimental Results:** The method is evaluated on the KITTI 3D object detection benchmark, demonstrating competitive performance with a significant speed advantage over existing monocular 3D detection methods.

This approach underscores the potential of monocular vision systems in autonomous driving, offering a cost-effective and efficient solution for 3D object detection in real-time applications.

2. Complex-YOLO: An Euler-Region-Proposal for Real-time 3D Object Detection on Point Clouds (2018)

The paper "Complex-YOLO" [52] introduces a novel method for real-time 3D object detection on point clouds using an Euler-region-proposal technique. This approach enhances the efficiency and accuracy of detecting objects in 3D space, specifically targeting applications like autonomous driving where real-time processing is crucial.

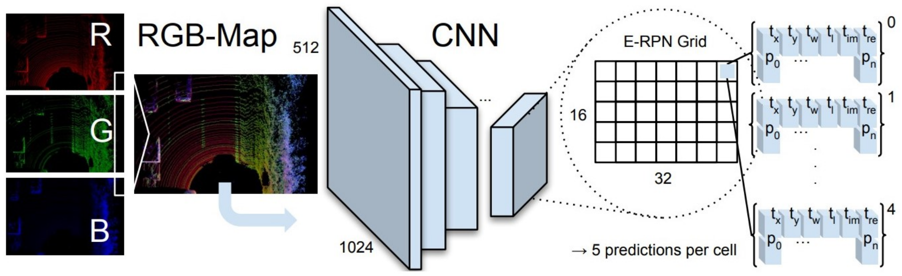


Figure 2.7: Overview of the Complex-YOLO architecture

Key contributions and findings from the paper include:

- **Euler-Region-Proposal:** Complex-YOLO introduces an Euler-region-proposal network that significantly reduces the computational load by focusing on regions of interest derived from Euler angles, thereby enhancing real-time performance.

- **3D Bounding Boxes:** The method accurately predicts 3D bounding boxes from point clouds, providing precise localization and dimensions of objects in the scene.

- **Real-time Performance:** The model is designed for real-time inference, achieving a balance between speed and accuracy, making it suitable for applications requiring immediate processing, such as autonomous driving.

- **Architecture:** Complex-YOLO employs a modified YOLO (You Only Look Once) framework tailored for 3D object detection, leveraging the strengths of YOLO in rapid object detection while adapting it to handle the complexities of 3D data.
- **Experimental Results:** The method demonstrates competitive performance on benchmark datasets, showing significant improvements in speed and accuracy over traditional 3D object detection techniques.
- **Applications:** While the primary application is autonomous driving, the technique is versatile enough for other domains requiring real-time 3D object detection, such as robotics and augmented reality.

This approach highlights the potential of using Euler-region-proposals in conjunction with deep learning frameworks like YOLO to achieve high-performance 3D object detection on point clouds in real-time scenarios as shown in the figure 2.8 below.

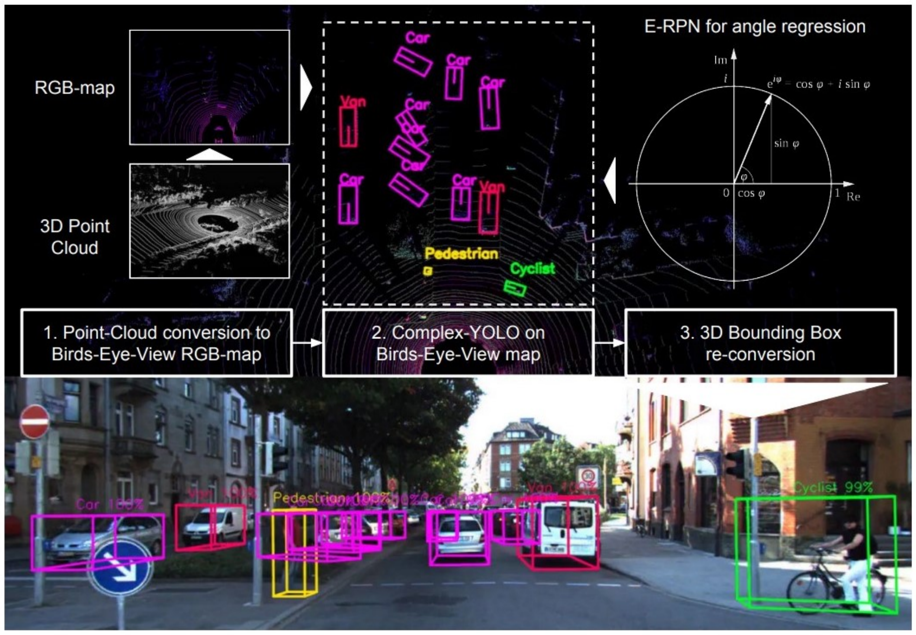

RGB-map
3D Point Cloud
E-RPN for angle regression
Car
Van
Pedestrian
Cyclist
1. Point-Cloud conversion to Birds-Eye-View RGB-map
2. Complex-YOLO on Birds-Eye-View map
3. 3D Bounding Box re-conversion
Im
Re
sin φ
cos φ
Cyclist 99%

Figure 2.8: Complex-YOLO is a very efficient model that directly operates on Lidar only based birds-eye-view RGB-maps to estimate and localize accurate 3D multiclass bounding boxes

2.4.4 Study of existing work in Object Tracking in autonomous driving

1. DeepSORT: Simple Online and Realtime Tracking with a Deep Association Metric (2017)

This influential paper presents DeepSORT [32], an advanced framework for multi-object tracking that integrates deep learning-based appearance descriptors with the SORT (Simple Online and Realtime Tracking) algorithm. DeepSORT enhances the standard SORT algorithm by incorporating a deep association metric, which significantly improves tracking performance in scenarios with occlusions and similar-looking objects.

Key contributions and findings of this paper include:

- **Deep Appearance Descriptor:** The integration of a deep appearance descriptor enables the tracker to effectively distinguish between objects with similar motion patterns but different appearances, enhancing robustness against occlusions and re-identification.
- **Kalman Filter and Hungarian Algorithm:** By utilizing a Kalman filter for motion prediction and the Hungarian algorithm for data association, DeepSORT maintains the efficiency and simplicity of the original SORT algorithm while significantly improving accuracy.
- **Improved Tracking Accuracy:** DeepSORT demonstrates superior performance on standard tracking benchmarks such as MOT16, achieving state-of-the-art accuracy and robustness in challenging multi-object tracking scenarios.
- **Real-Time Capability:** Despite the enhancements, DeepSORT maintains real-time processing capabilities, making it suitable for practical applications in surveillance, autonomous driving, and sports analytics.

DeepSORT's contribution to multi-object tracking lies in its ability to combine deep learning techniques with traditional tracking methods, resulting in a system that balances accuracy and computational efficiency. This approach has set a new benchmark in the field of object tracking, enabling more reliable and precise tracking in dynamic and crowded environments.

2. An Introduction to the Kalman Filter (1995)

This foundational paper [31] by Greg Welch and Gary Bishop provides a comprehensive introduction to the Kalman Filter, a powerful mathematical tool for estimating the state of a dynamic system from a series of noisy measurements. The Kalman Filter is widely used in object tracking applications due to its effectiveness in handling uncertainties in motion and measurement.

Key contributions and findings of this paper include:

- **Recursive Estimation:** The Kalman Filter provides a recursive solution to the discrete-data linear filtering problem, making it efficient for real-time applications.
- **State and Measurement Models:** The filter uses a state transition model and a measurement model to predict the state of the system and update estimates based on incoming measurements.
- **Error Covariance Matrix:** The Kalman Filter maintains an error covariance matrix to quantify the uncertainty in its estimates, which is crucial for adjusting the filter's sensitivity to new measurements.
- **Applications in Object Tracking:** The paper discusses the application of the Kalman Filter in tracking objects, such as estimating the position and velocity of moving objects in radar and computer vision systems.

The Kalman Filter's ability to provide optimal estimates in the presence of noise and uncertainty has made it a cornerstone technique in various fields, including navigation, control systems, and signal processing. Its real-time processing capability and robustness in dynamic environments have established it as a critical component in object tracking systems.

2.4.5 Study of existing work in Multi-Object Tracking with Camera-LiDAR Fusion in autonomous driving

1. Joint Multi-Object Detection and Tracking with Camera-LiDAR Fusion for Autonomous Driving (2021)

This paper [53] presents a comprehensive framework for joint multi-object detection and tracking using camera-LiDAR fusion specifically tailored for autonomous driving. The proposed system integrates detection and tracking in a unified pipeline to leverage the complementary strengths of camera and LiDAR sensors, as shown in Figure 2.9 below.

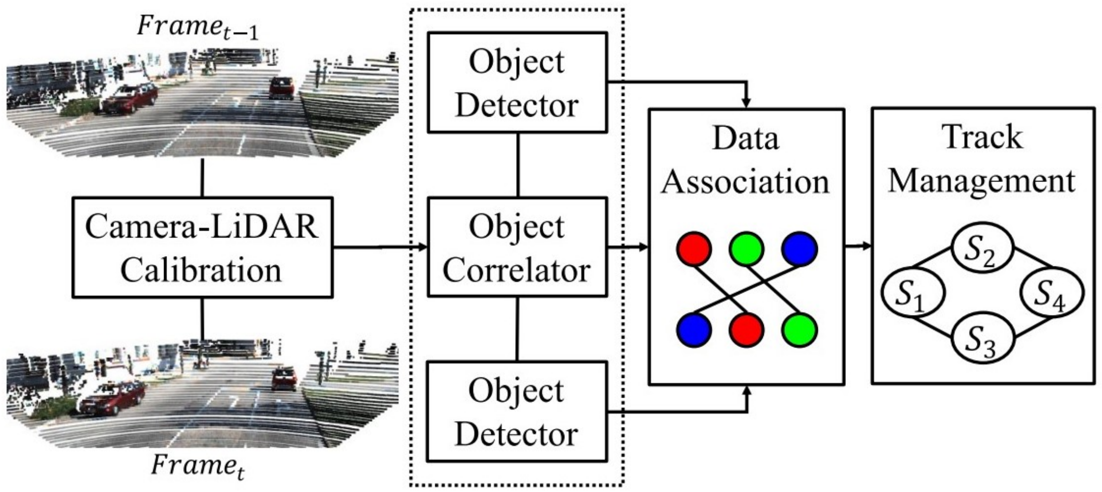


Figure 2.9: System architecture for joint multi-object detection and tracking using camera-LiDAR fusion.

Key contributions and findings of this paper include:

- **Region Proposal Network (RPN):** The system utilizes an RPN to generate regions of interest (RoIs) from the sensor data, which are then used to extract multi-modal features.
- **Parallel Detection and Correlation Networks:** The framework employs parallel networks for object detection and correlation, generating detection results and re-identification (Re-ID) affinities that help maintain object identity across frames.
- **Kalman Filter for Motion Prediction:** A Kalman filter is used for motion prediction, incorporating geometric similarities and measurement updates to refine object tracking.
- **Data Association and Track Management:** The mixed-integer programming module handles comprehensive data association based on detection results and computed affinities, ensuring continuous object tracking even in the presence of occlusions.
- **Robustness and Accuracy:** The system achieves state-of-the-art performance on standard benchmarks, demonstrating robustness and accuracy in complex multi-object tracking scenarios typical in autonomous driving.

This paper highlights the importance of combining appearance and motion cues through camera-LiDAR fusion to enhance the reliability and accuracy of multi-object tracking in dynamic environments.

2. DeepFusionMOT: A 3D Multi-Object Tracking Framework Based on Camera-LiDAR Fusion with Deep Association (2022)

The DeepFusionMOT framework [54] introduces an innovative approach to 3D multi-object tracking that balances tracking accuracy and computational efficiency through effective camera-LiDAR fusion and deep association techniques. This framework is particularly suited for real-time applications in autonomous driving, as illustrated in Figure 2.10 below.

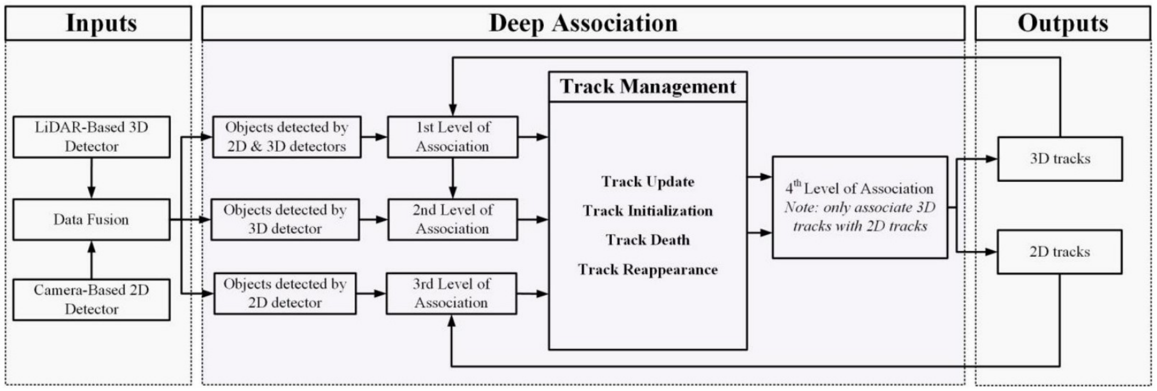


Figure 2.10: DeepFusionMOT architecture highlighting the deep association mechanism for camera-LiDAR fusion.

Key contributions and findings of this paper include:

- **Deep Association Mechanism:** The framework employs a deep association mechanism that integrates 2D camera data and 3D LiDAR data, allowing seamless fusion and accurate tracking of objects detected by either sensor.
- **Kalman Filter Integration:** A Kalman filter is utilized to maintain continuous tracking and update object trajectories, ensuring smooth transitions between 2D and 3D domains.
- **Real-Time Processing:** Despite its comprehensive fusion and association processes, DeepFusionMOT maintains real-time processing capabilities, making it suitable for practical applications.
- **Robust Performance:** The system demonstrates superior tracking performance on standard datasets, achieving a balance between accuracy and speed, and showing significant improvements over existing multi-object tracking methods.
- **Publicly Available Code:** The authors have made their code publicly available, promoting further research and development in the field of camera-LiDAR fusion-based multi-object tracking.

DeepFusionMOT's contribution lies in its ability to effectively fuse multi-modal data, leveraging deep learning techniques to achieve robust and efficient multi-object tracking for autonomous driving applications.

2.5 Comparative study

2.5.1 Classification of works reviewed

In the realm of computer vision for autonomous driving, research and development are centered around key categories: object detection, lane detection, object tracking, and sensor fusion.

Object detection algorithms, such as Faster R-CNN, SSD, FPN and the YOLO series, are extensively studied for their ability to accurately and efficiently identify objects on the road.

Lane detection research focuses on precise boundary segmentation using deep learning techniques and attention-guided methods to ensure vehicles stay within their designated lanes.

Object tracking algorithms, like DeepSORT and Kalman Filters, are essential for continuously monitoring moving objects, enabling safe navigation and decision-making.

Sensor fusion techniques integrate data from various sensors, such as cameras, LiDAR, and radar, to provide a comprehensive understanding of the vehicle's surroundings. Models like Joint Multi-Object Detection and Tracking leverage sensor fusion to enhance object detection and tracking accuracy, crucial for real-world deployment of autonomous driving systems.

The figure 2.11 below illustrates the classification of articles studied in this section:

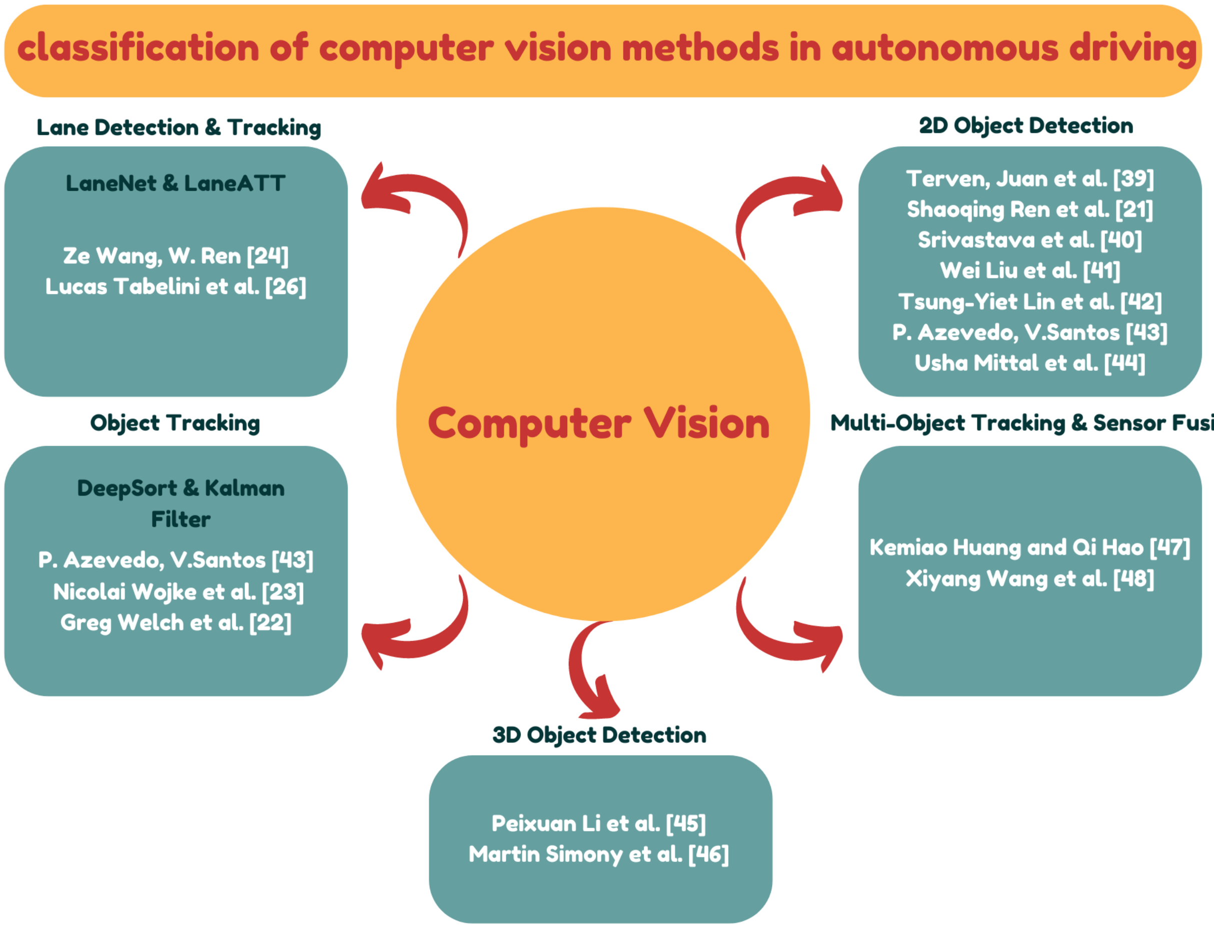


Figure 2.11: Computer Vision methods Classification in Autonomous Driving.

The Table 2.1 Below illustrates a comparative study of the works reviewed.

Paper	**Method**	**Type**	**Accuracy (mAP)**	**Complexity**	**Real-time**	**Strengths and Weak-nesses**
Object Detection						
[45]	YOLOv1 to YOLOv8, YOLO-NAS	CNN	High (YOLOv4-NAS)	Varies (low-high)	Yes	Fast, accurate, scalable, adaptable, resource-efficient
[30]	Faster R-CNN	CNN + RPN	High	High	No	Accurate, scalable, flexible, slower
[46]	SSD, Faster R-CNN & YOLO	CNN	High (Faster R-CNN)	High (Faster R-CNN)	Yes (YOLO & SSD)	Faster R-CNN is accurate but slower than YOLO and SSD for real-time use
[47]	SSD	CNN	Moderate	Moderate	Yes	Fast, simple, less accurate
[48]	FPN	CNN	High	Moderate	No	Accurate, multi-scale, complex
[49]	Different versions of YOLO	CNN	High	Varies (low-high)	Yes	The best performing object detector was YOLOR-CSP achieving a mAP@50 of 69.70 %

[50]	Faster R-CNN & YOLOv5	CNN	High	High	NO	Enhances prediction accuracy but also leads to an increase in computational time
3D object detection						
[51]	RTM3D	Keypoint Detection	High	Moderate	Yes	Accurate, cost-effective, real-time, single camera
[52]	Complex-YOLO	Euler-region-proposal	High	Moderate	Yes	Efficient, real-time, accurate, point cloud
Object Tracking						
[32]	DeepSORT	Deep Learning + Kalman Filter	High	Moderate	Yes	Accurate, robust, efficient, occlusion handling
[31]	Kalman Filter	Recursive Estimation	High	Low	Yes	Optimal, noise handling, efficient, foundational
Multi-Object Tracking with Camera-LiDAR Fusion						
[53]	JMODT	RPN + Kalman Filter	High	High	Yes	Robust, accurate, complex
[54]	DeepFusion-MOT	Kalman Filter + deep association mechanism	High	High	Yes	Accurate, efficient, robust
Lane Detection and Tracking						

[4]	LaneNet	Two-stage architecture, Semantic Segmentation	High	Moderate	Yes	Accurate, robust, high data needs
[34]	LaneATT	Attention-guided	High	Low	Yes	Efficient, real-time, lower accuracy

Table 2.1: A Comparison of Reviewed Articles

2.6 Discussion

In the realm of computer vision for autonomous driving, solutions are diverse, leveraging Convolutional Neural Networks (CNNs) and multi-sensor fusion. Object detection methods like YOLO series and Faster R-CNN enhance real-time performance. SSD eliminates region proposal step, achieving high-speed detection. Feature Pyramid Networks (FPNs) enable object detection at various scales. Comparative studies favor YOLO-based detectors for their speed and accuracy in edge devices. EnsembleNet combines Faster R-CNN and YOLO for enhanced vehicle detection.

Lane detection relies on LaneNet for precise segmentation, while attention-guided methods improve focus on relevant lane markers.

For 3D object detection, methods like RTM3D and Complex-YOLO focus on real-time detection using object keypoints and point clouds respectively.

Multi-object tracking employs DeepSORT, enhancing SORT with deep association metrics, and Kalman Filter for state estimation. Frameworks like Joint Multi-Object Detection and Tracking integrate camera and LiDAR data, using Extended Kalman Filter for motion prediction. DeepFusionMOT proposes a 3D multi-object tracking framework through effective sensor fusion and deep association techniques.

These advancements enhance reliability and efficiency in autonomous driving systems, each contributing unique strengths in detection, lane detection, and multi-object tracking.

2.7 Conclusion

This chapter has provided an extensive overview of the state-of-the-art methods and techniques used in computer vision for autonomous driving, specifically focusing on 2D/3D object detection, and object tracking. By comparing these diverse methodologies, we observed that each approach offers distinct advantages and presents certain limitations. For instance, while YOLO-based detectors excel in speed and adaptability, Faster R-CNN provides higher accuracy but at the cost of increased computational complexity. Similarly, in 3D object detection, methods like RTM3D and Complex-YOLO offer real-time capabilities using object keypoints and point clouds, respectively, enhancing detection accuracy and efficiency.

In the next chapter, we will introduce our approach to multi-object tracking with sensor fusion. This approach aims to integrate the strengths of the 2D/3D object detection, and tracking algorithms reviewed in this chapter. This system will utilize advanced sensor fusion methodologies to enhance detection and tracking performance, ensuring optimal results for autonomous driving applications.

Chapter 3

Proposed Approach in Multi-Object Tracking Based on Camera and LiDAR Fusion

Chapter 3

Proposed Approach in Multi-Object Tracking Based on Camera and LiDAR Fusion

3.1 Introduction

The previous chapter comprehensively reviewed existing research on object detection, tracking, and sensor fusion techniques for autonomous driving. We identified the critical role of sensor fusion in overcoming the limitations of individual sensors like cameras and LiDAR [27]. While each sensor provides valuable information about the environment, their strengths and weaknesses are complementary. Cameras excel at capturing rich visual details but struggle in low-light conditions. LiDAR excels at providing accurate 3D information but lacks the detailed texture data from cameras. By fusing data from these sensors, we can create a more robust and holistic perception of the surrounding environment, leading to more accurate object detection and tracking.

This chapter delves into our proposed sensor fusion approach specifically designed for Multi object detection and tracking in autonomous driving applications. We propose an architecture that leverages the strengths of both camera and LiDAR data. We will detail the individual processing pipelines for camera and LiDAR data, including the chosen algorithms. We will then explain the core of our approach, which is the fusion strategy used to combine the processed data from both sensors. Finally, we will describe how the fused data will be utilized for object detection and tracking.

By effectively fusing camera and LiDAR data, we aim to achieve significant improvements in object detection and tracking performance compared to using individual sensors alone. This enhanced perception system will be crucial for accurate decision-making and safe navigation in complex autonomous driving scenarios.

3.2 Project Context and Issue Focus: Smart City Surveillance with Enhanced Autonomy

The ever-growing complexity of urban environments necessitates smarter solutions for public safety and situational awareness. This project delves into the application of autonomous cars for smart city surveillance. While autonomous cars offer the potential for continuous monitoring, their effectiveness hinges on accurate object detection, tracking, and pose estimation. However, current limitations in these areas restrict autnomous cars ability to fully comprehend the surrounding environment and interpret human behavior.

3.2.1 Challenges in Smart City Surveillance with self driving cars

- **Accurate Pose Estimation for Behavior Understanding:** Effective public safety surveillance relies on comprehending human activity. Pose estimation, the process of determining a person's posture and body orientation, is crucial for interpreting behavior [55]. Current pose estimation algorithms often struggle with occlusions (people being partially hidden), complex backgrounds, and varying lighting conditions, leading to inaccurate assessments. This hinders robots' ability to distinguish between harmless actions and potential threats.

- **Robust Sensor Fusion for Enhanced Object Detection and Tracking:** Cameras provide rich visual details but struggle in low light. LiDAR excels at 3D information but lacks texture data. Sensor fusion combines data from these sensors to create a more complete picture of the environment [5]. However, existing fusion methods can be computationally expensive or prone to errors in dynamic environments with moving objects. This limits robots' ability to reliably track individuals and objects of interest, hindering their ability to effectively monitor large areas.

- **Advanced Decision-Making for Human-Level Response:** Autonomous cars in surveillance applications need to make critical decisions in real-time [17]. Current decision-making algorithms may struggle with situations requiring nuanced judgment or fail to adapt to unforeseen circumstances. These limitations can lead to unnecessary interventions or missed threats. To ensure public safety and trust, autonomous cars need to make decisions with a level of sophistication comparable to trained professionals.

3.3 Multi-Object Tracking in Autonomous Driving

3.3.1 Overview

Multi-object tracking (MOT) is a critical component in autonomous driving systems, enabling vehicles to perceive and navigate complex environments by identifying and tracking multiple objects such as vehicles, pedestrians, and cyclists. Effective MOT systems enhance situational awareness and decision-making, contributing to safer and more reliable autonomous driving.

3.3.2 Challenges in Multi-Object Tracking

Multi-object tracking [56] in autonomous driving faces several significant challenges. Occlusions, where objects become partially or fully obscured by other objects or environmental features, make maintaining continuous tracking difficult. The dynamic nature of the environments in which autonomous vehicles operate, with objects moving unpredictably, necessitates robust tracking algorithms capable of adapting to sudden changes. Sensor limitations also pose a challenge; cameras may struggle in low-light conditions, and LiDAR, though providing 3D data, often has lower resolution and can be adversely affected by weather conditions. Efficiently associating detections across frames to maintain object identities, known as data association, is another complex task, particularly in crowded scenes with similar-looking objects. These challenges underscore the need for advanced techniques and robust systems in multi-object tracking for autonomous vehicles.

3.3.3 Techniques for Multi-Object Tracking

- **Kalman Filter:** A widely used algorithm that predicts the future state of objects based on their previous states, helping to smooth out noise and improve tracking accuracy [31].
- **Hungarian Algorithm:** Employed for data association, it matches detected objects across consecutive frames to maintain consistent tracking identities [57].
- **DeepSORT:** An advanced MOT framework that integrates deep learning-based appearance descriptors with the SORT algorithm, enhancing performance in scenarios with occlusions and similar-looking objects [32].
- **Sensor Fusion:** Combining data from multiple sensors, such as cameras and LiDAR, to leverage their complementary strengths and improve overall tracking performance. This approach mitigates the limitations of individual sensors and provides more reliable and accurate object tracking [5].

3.3.4 Applications in Autonomous Driving

- **Obstacle Avoidance:** MOT systems enable autonomous vehicles to detect and track obstacles, allowing for timely evasive maneuvers.
- **Path Planning:** Accurate tracking of surrounding objects informs the vehicle's path planning algorithms, ensuring safe and efficient navigation.
- **Traffic Sign and Signal Recognition:** Tracking traffic signs and signals helps autonomous vehicles comply with traffic laws and respond appropriately to changing road conditions.

- **Pedestrian and Cyclist Safety:** By continuously monitoring pedestrians and cyclists, MOT systems enhance the safety of vulnerable road users.

In the following sections, we will delve into our proposed approach for multi-object tracking based on camera and LiDAR fusion, integrating the advancements in object detection, 3D object detection, and tracking algorithms to achieve superior results. Our goal is to combine the benefits of these techniques to enhance the accuracy and reliability of multi-object tracking in autonomous driving.

3.4 Description of the Global Proposition of our autonomous car

This project focuses on a crucial aspect of autonomous cars: sensor fusion for enhanced object detection and tracking. While our core contribution lies in this specific area, it's important to acknowledge the broader ecosystem of technologies required for a fully functional autonomous system. These include:

- **Sensors and Perception Module:** autonomous driving rely on a suite of sensors like cameras, LiDAR, radar, IMU, and ultrasonic sensors to gather data about the surrounding environment. The perception module processes this raw sensory data to create a real-time understanding of the world.
- **SLAM (Simultaneous Localization and Mapping):** SLAM [58] allows the robot to build a map of its surroundings while simultaneously keeping track of its own location within that map. This is essential for navigation and decision-making.
- **World Model:** The world model represents the car's understanding of the environment, including objects, obstacles, and their relationships. Sensor fusion plays a critical role in continuously updating and refining this world model.
- **Decision Making:** Based on the information from perception and the world model, the self driving car needs to make critical decisions about navigation, path planning, and interaction with the environment. This may involve techniques like rule-based systems or reinforcement learning.
- **Control:** The decisions made by the car translate into control actions that steer, accelerate, or brake the vehicle. Precise control is crucial for safe and efficient navigation.

While this project focuses on sensor fusion for object detection and tracking, it acknowledges the importance of these interconnected technologies for achieving full autonomy. Our proposed approach aims to contribute to a more robust and reliable perception system, paving the way for advancements in the broader field of autonomous driving and self-driving cars.

The figure 3.1 below presents the global schema of the proposed solution.

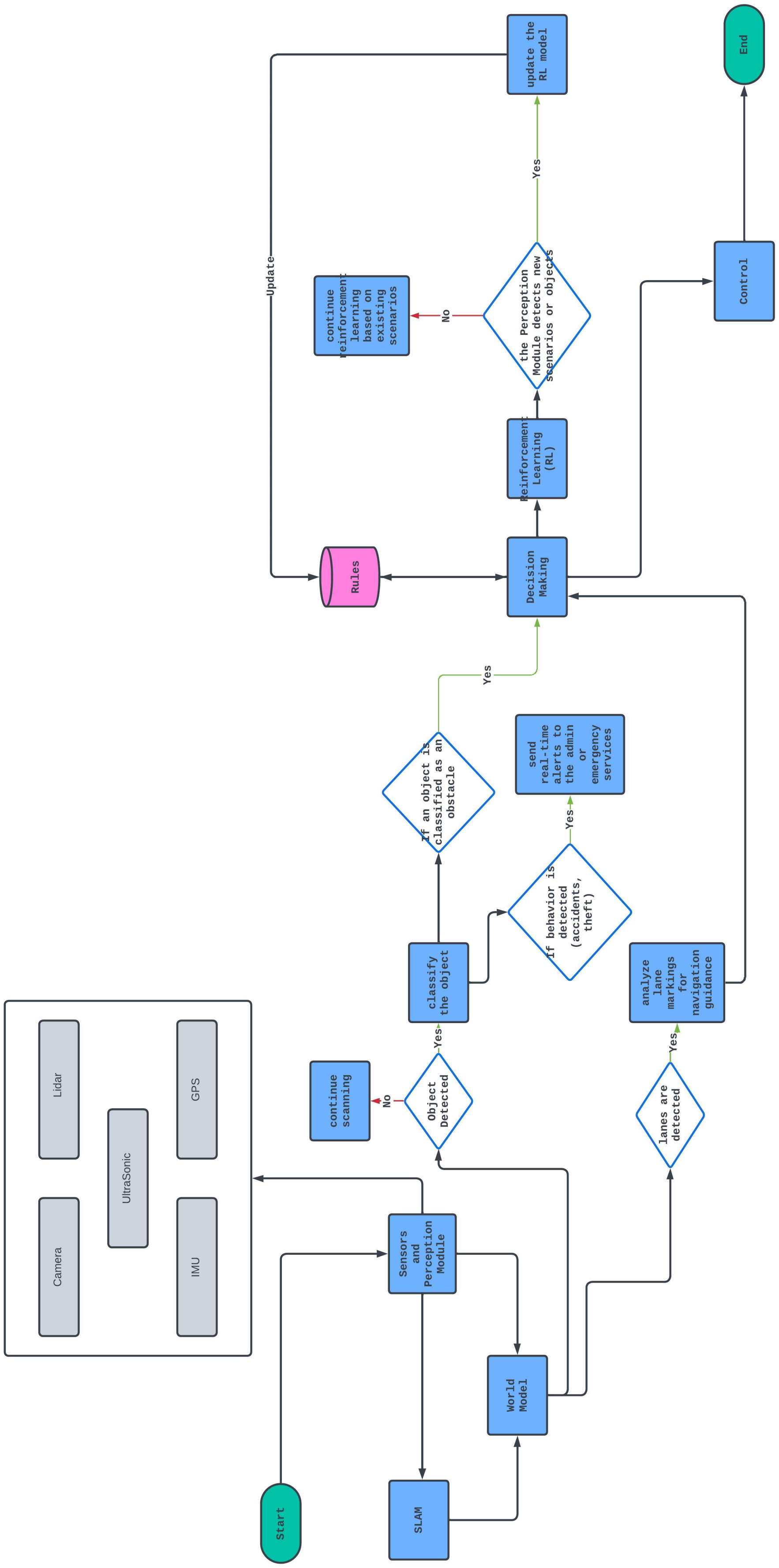

Start
Sensors and Perception Module
Camera
Lidar
UltraSonic
IMU
GPS
SLAM
World Model
Object Detected
No
continue scanning
Yes
classify the object
If an object is classified as an obstacle
Yes
If behavior is detected (accidents, theft)
Yes
send real-time alerts to the admin or emergency services
lanes are detected
Yes
analyze lane markings for navigation guidance
Decision Making
Rules
Update
Reinforcement Learning (RL)
the Perception Module detects new scenarios or objects
No
continue reinforcement learning based on existing scenarios
Yes
update the RL model
Control
End

Figure 3.1: Global flowchart of the proposed autonomous car

3.5 Description of the Proposition of our Multi object Tracking with sensor fusion Approach

The previous section outlined the various components that work in concert to enable autonomous navigation. This project delves deeper into one critical module: sensor fusion. While individual sensors like cameras and LiDAR provide valuable information, they have limitations [59]. Sensor fusion addresses this by intelligently combining data from multiple sensors, creating a richer and more robust perception of the environment. This enhanced understanding is crucial for accurate object detection and tracking, which are fundamental to safe and reliable navigation for autonomous systems and self-driving cars.

The figure 3.2below presents the flowchart of the proposed sensor fusion module for Multi Object Detection and Tracking.

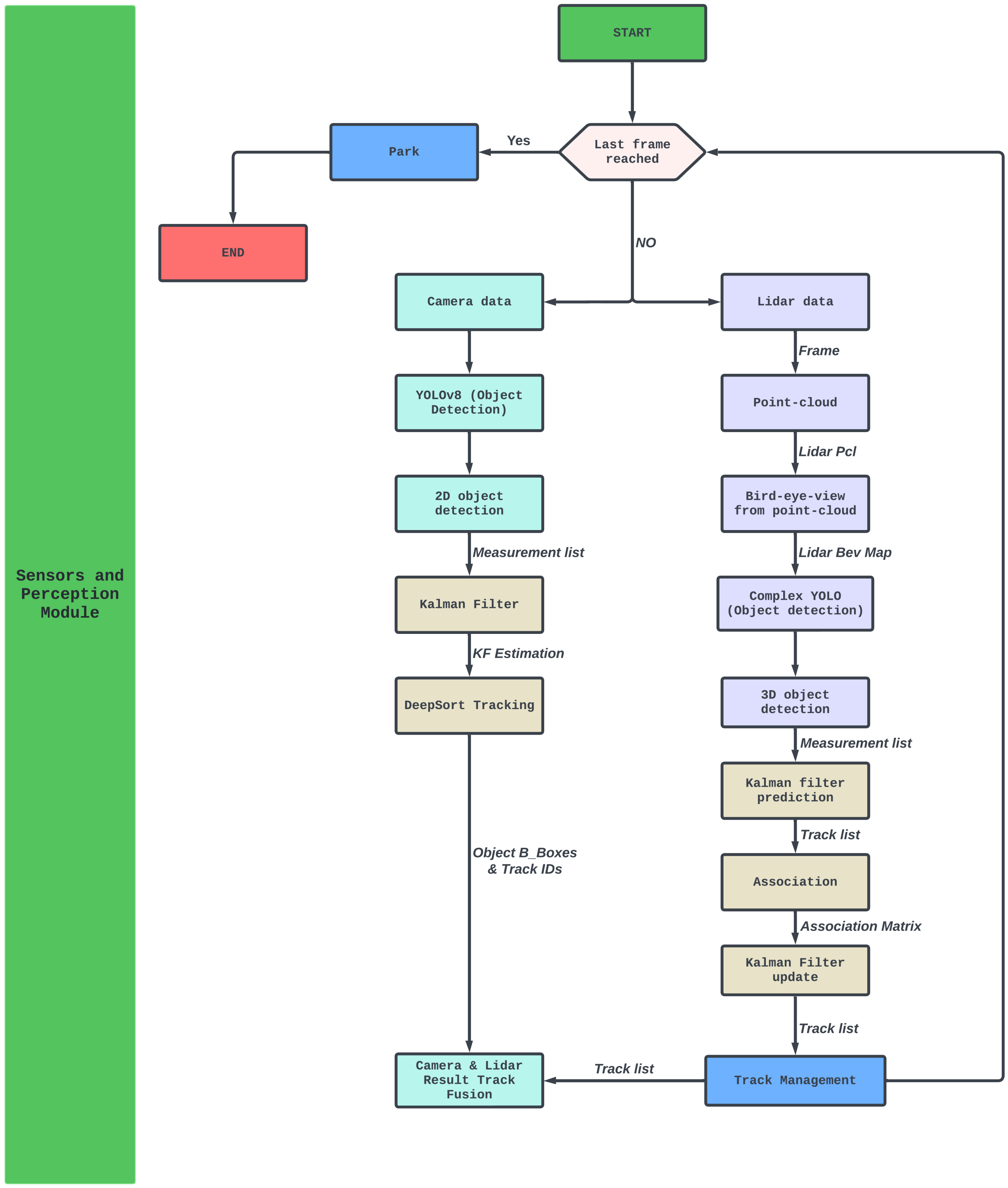


Figure 3.2: Flowchart of the proposed MOT with Sensor Fusion

- **YOLOv8 (Object Detection):** The camera captures the scene, and YOLOv8 [45], a state-of-the-art object detection algorithm, is applied to detect 2D objects within the frame. The output is a list of detected objects, including their bounding boxes and confidence scores as shown in the figure 3.3 below.

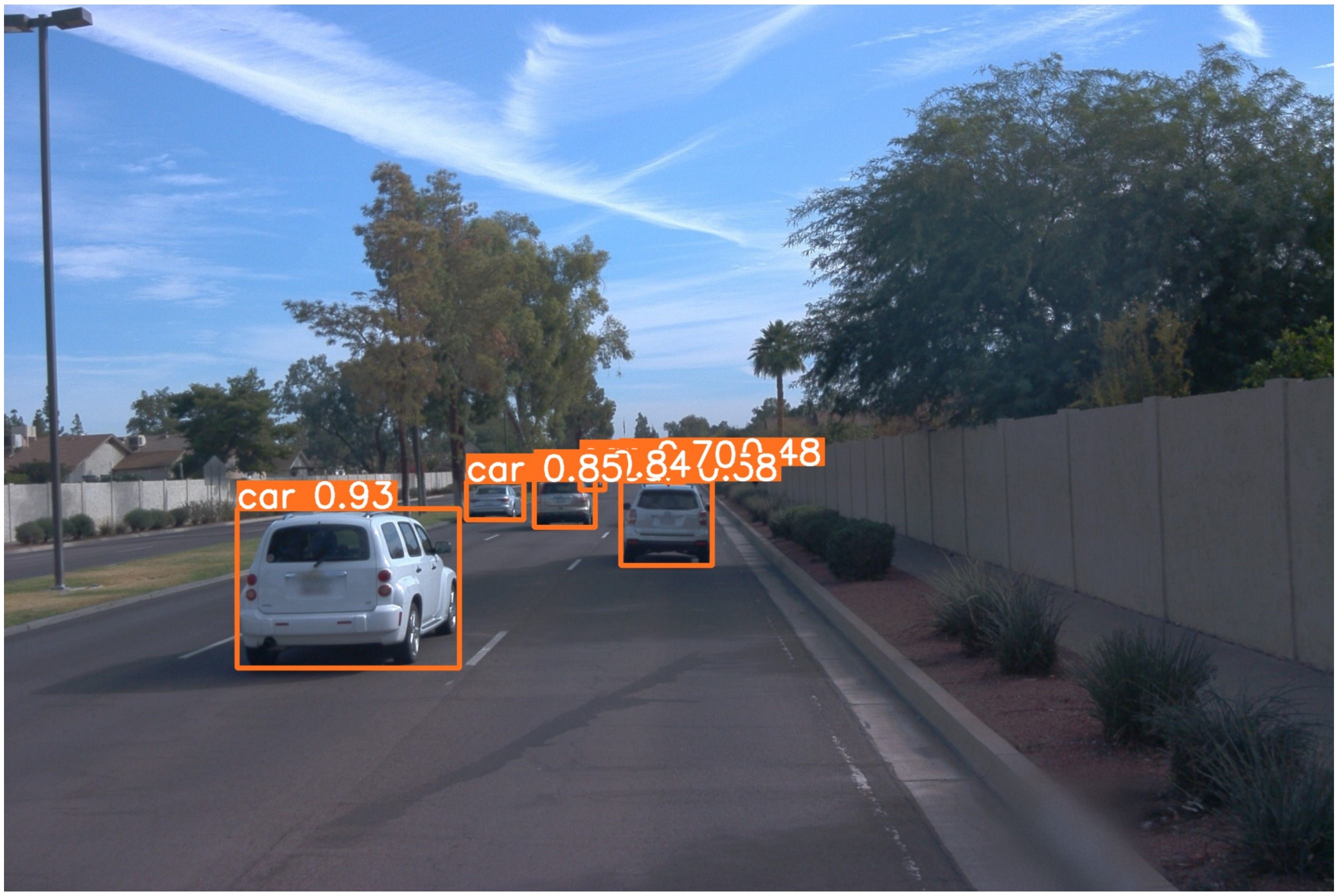


Figure 3.3: YoloV8 detection on A Camera Image

- **Kalman Filter (KF) Estimation:** The Kalman Filter takes the measurement list and predicts the state of each detected object. This includes position, velocity, and possibly other dynamics. The Kalman Filter algorithm will go through the following steps to track an object over time:
  - **First measurement:** The filter will receive initial measurements of the object's position relative to the car. These measurements will come from a camera or LiDAR sensor.
  - **Initialize state and covariance matrices:** The filter will initialize the object's position

    $$\begin{pmatrix} p_x \\ p_y \end{pmatrix}$$

    and velocity:

    $$\begin{pmatrix} v_x \\ v_y \end{pmatrix}$$

    based on the first measurement.
  - **Predict:** The algorithm will predict where the object will be after a time period Δt.
  - **Update:** The filter compares the "predicted" location with what the sensor measurement says. The predicted location and the measured location are combined to give an

updated location. The Kalman Filter will put more weight on either the predicted location or the measured location depending on the uncertainty of each value. The update step is often also referred to as the innovation or correction step.

- **Iterate:** The car will receive another sensor measurement after a time period Δt. The algorithm then does another predict and update step.

The figure 3.4 below demonstrates how the Kalman Filter predicts and updates based on the new measurements.

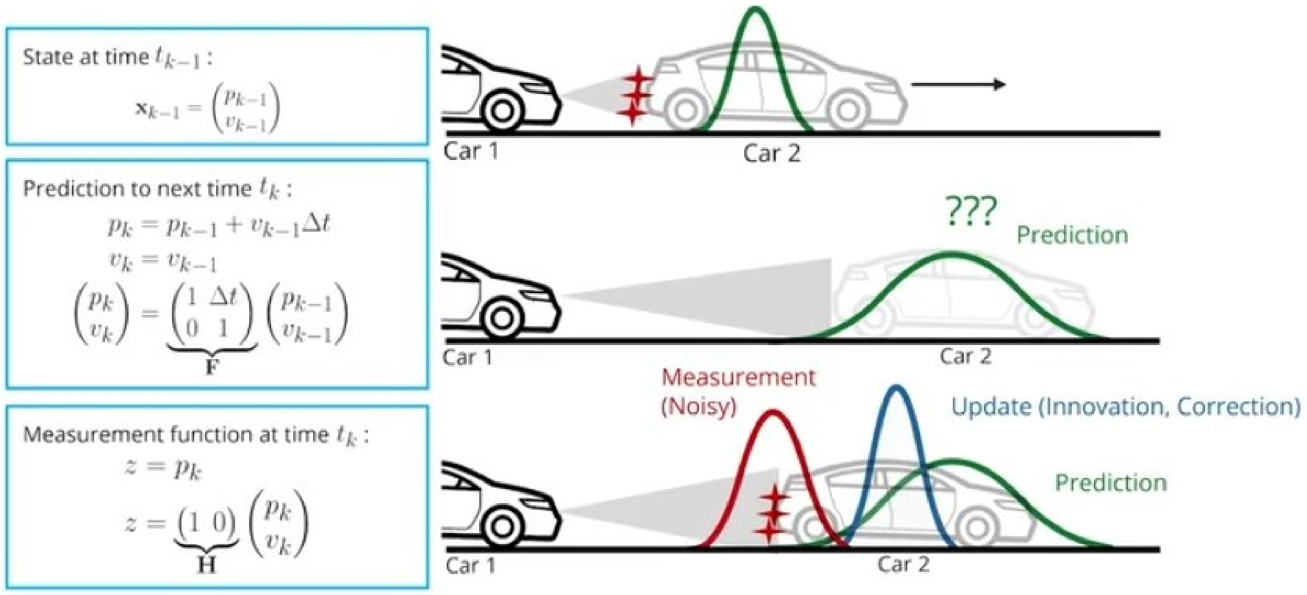

State at time t_{k-1} :
$\mathbf{x}_{k-1} = \begin{pmatrix} p_{k-1} \\ v_{k-1} \end{pmatrix}$
Prediction to next time t_k :
$p_k = p_{k-1} + v_{k-1}\Delta t$
$v_k = v_{k-1}$
$\begin{pmatrix} p_k \\ v_k \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix}}_{\mathbf{F}} \begin{pmatrix} p_{k-1} \\ v_{k-1} \end{pmatrix}$
Measurement function at time t_k :
$z = p_k$
$z = \underbrace{(1 \ 0)}_{\mathbf{H}} \begin{pmatrix} p_k \\ v_k \end{pmatrix}$
Car 1
Car 2
???
Prediction
Car 1
Car 2
Measurement (Noisy)
Update (Innovation, Correction)
Prediction
Car 1
Car 2

Figure 3.4: Kalman Filter Prediction and Update[7]

- **DeepSORT Tracking::** DeepSORT (Simple Online and Realtime Tracking) uses the KF predictions along with a deep appearance descriptor to track objects across frames. The output is a list of tracked objects, each with an assigned track ID and updated bounding boxes like the figure 3.5 below.

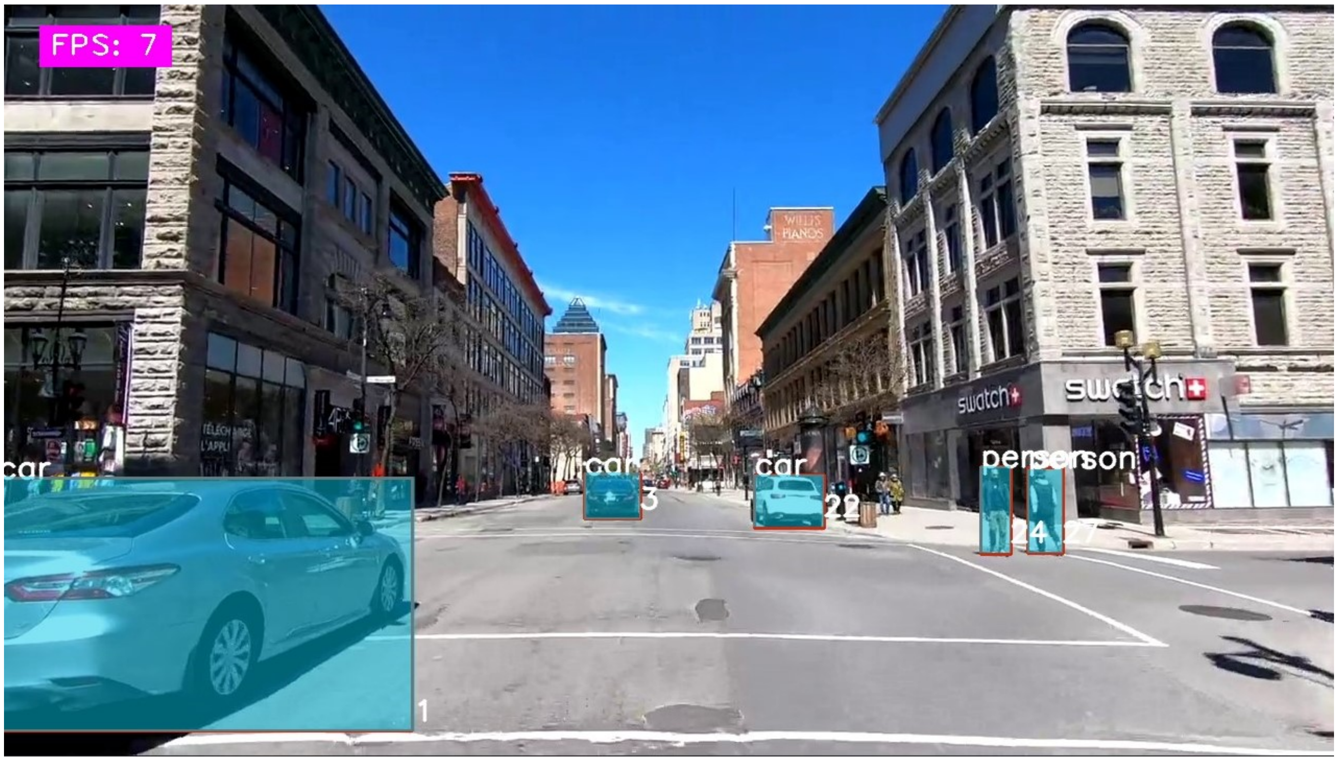


Figure 3.5: DeepSort assigned track ID and updated bounding boxes

- **LiDAR Data Processing:** The LiDAR sensor captures the environment, producing a point cloud representing the 3D space around the vehicle, The figure 3.6 below shows how the 3D point cloud looks.

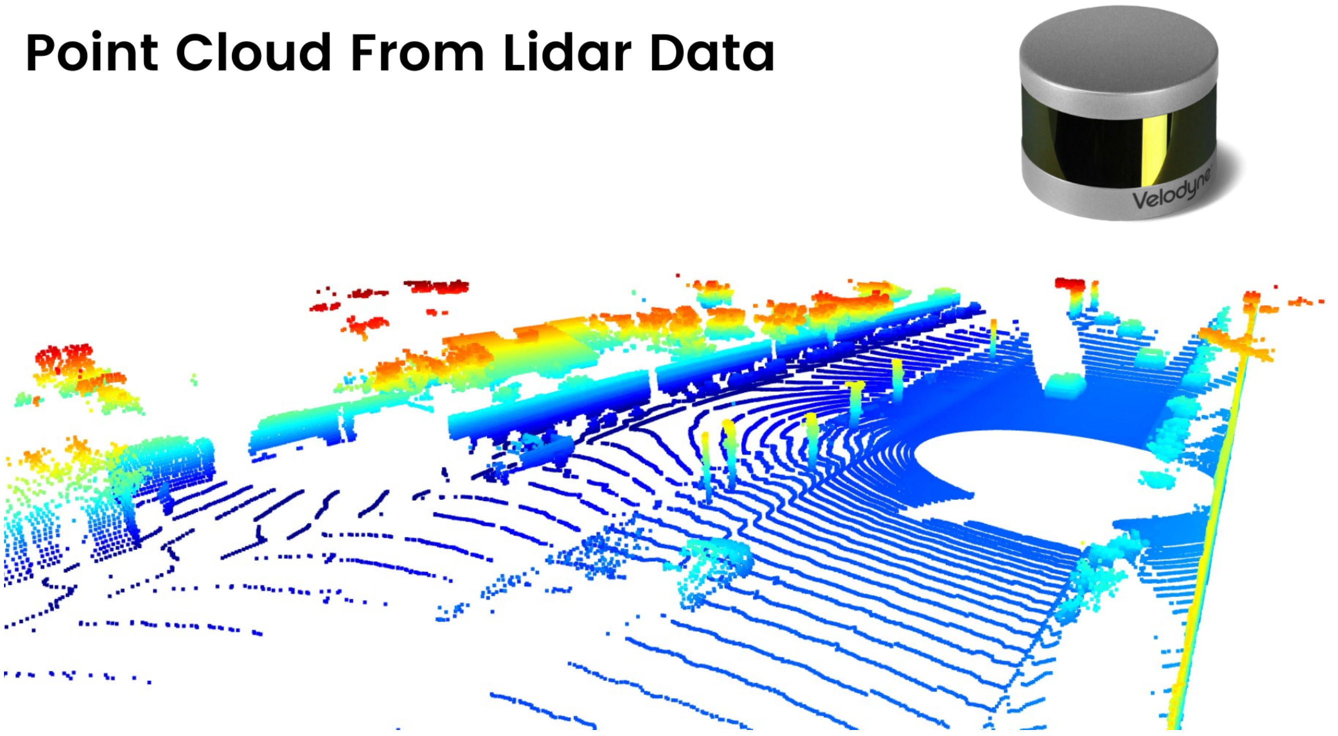


Figure 3.6: velodyne Lidar 3D point cloud

- **Bird-Eye-View (BEV) from Point-Cloud:** The point cloud data as illustrated in figure 3.6 is transformed into a bird's-eye-view (BEV) map [60], which simplifies the 3D data into a 2D representation for easier processing. The transformation involves the following steps:
  - **Projection:** Project the 3D point cloud (x, y, z) onto the BEV plane (x, y) by ignoring the height z.
  - **Discretization:** Discretize the continuous BEV coordinates into grid cells. Each cell (i, j) in the BEV map corresponds to a small area in the real world.
  - **Feature Encoding:** Encode the features within each grid cell, such as the density of points, the maximum height, and the intensity of the LiDAR returns.

    $$\text{BEV}(i,j) = f\left(\{(x_k, y_k, z_k, I_k)\}_{k=1}^{N}\right)$$

    where f is the feature encoding function, (x_k, y_k, z_k, I_k) are the point cloud coordinates and intensity, and N is the number of points within the grid cell.
  - **Normalization:** Normalize the encoded features to ensure they are within a suitable range for input to the neural network.

- **Complex YOLO (Object Detection):** Complex YOLO [52] is applied to the BEV map to detect 3D objects, outputting their bounding boxes and confidence scores. The process of generating the BEV map is as follows:

  First, we define the area to encompass for detection, setting the longitudinal range to 0...50m and the lateral range to -25...+25m. This choice is based on considerations from the original paper and existing implementations of Complex YOLO.

Next, we divide this area into a grid, configuring the BEV image size to 608 x 608 pixels, which corresponds to a spatial resolution of approximately 8 cm.

Each grid cell in the BEV map, identified by coordinates (i, j), contains a set of points P_{ij}. The number of points in each cell, denoted as N_{ij}, determines the content of the three channels assigned to the cell:

$$\text{Height } H_{ij} = \max(\mathbf{P}_{ij} \cdot [0, 0, 1]^T)$$

$$\text{Intensity } I_{ij} = \max(\mathbf{I}(\mathbf{P}_{ij}))$$

$$\text{Density } D_{ij} = \min\left(1.0, \frac{\log(N_{ij} + 1)}{64}\right)$$

where $\mathbf{P}_{ij}$ represents the points in the cell, $\mathbf{I}(\mathbf{P}_{ij})$ denotes the intensity of these points, and N_{ij} is the count of points. These parameters are derived from the original paper's recommendations to optimize object detection accuracy in the BEV map. As you can see, $H_{i,j}$ encodes the maximum height in a cell, $I_{i,j}$ the maximum intensity, and $D_{i,j}$ the normalized density of all points mapped into the cell. The resulting BEV image looks like the following figure 3.7:

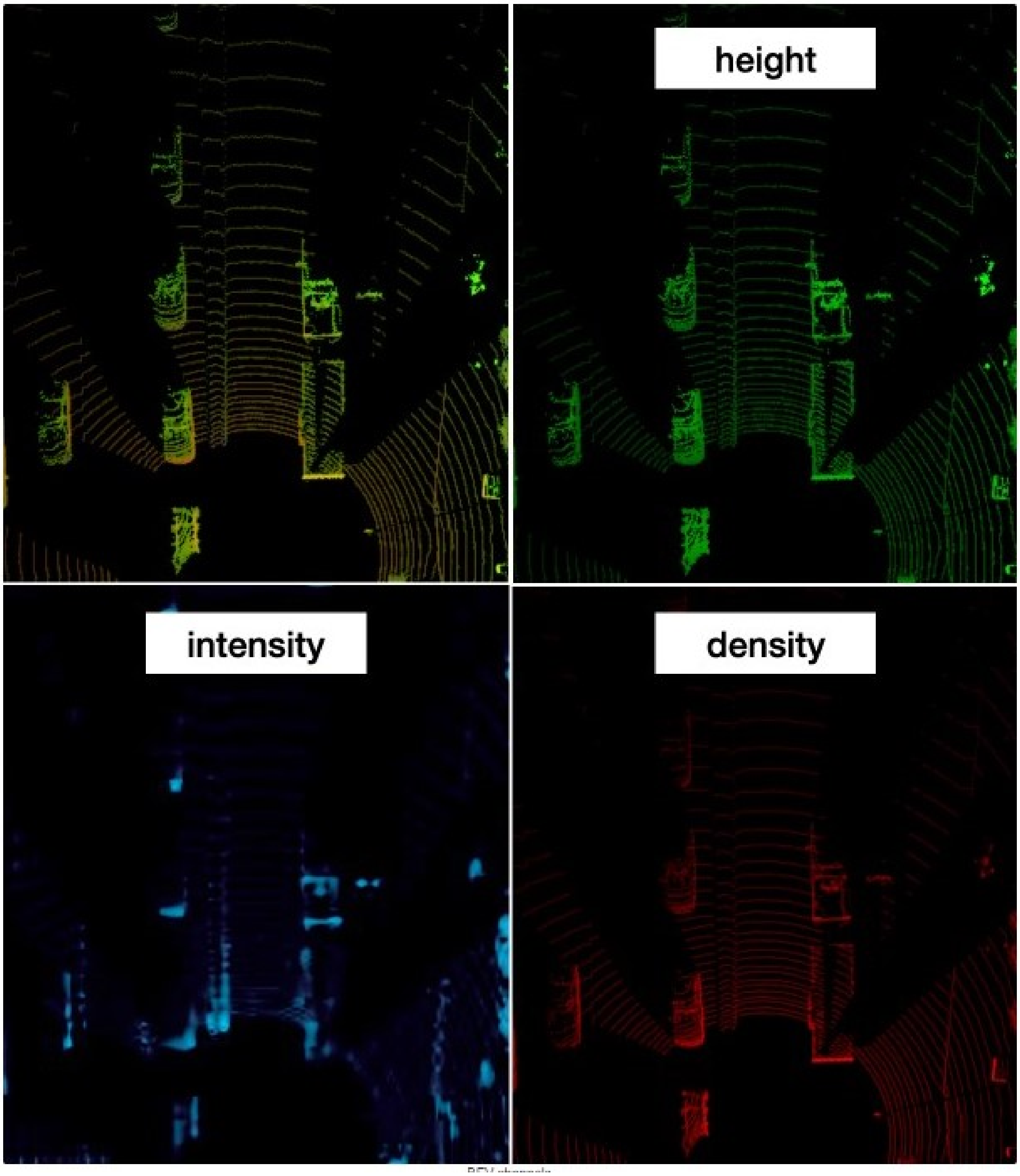


Figure 3.7: BEV map Height,Intensity and Density[7]

- **Measurement List and Kalman Filter Prediction:** Similar to the camera data, the detected 3D objects are compiled into a measurement list, The Kalman Filter predicts the states of the detected 3D objects based on the measurement list.
- **Association and Mahalanobis Distance:** An association matrix (A) is constructed to match

detected objects to existing tracks, ensuring continuity of tracking across frames. The data association assigns measurements to tracks and decides which track to update with which measurement. The Mahalanobis distance ($d(x, z)$) [61] is used as a measure of similarity between predicted track states (x) and measurements (z), defined by:

$$d(x, z) = (z - h(x))^T S^{-1}(z - h(x))$$

where $h(x)$ is the measurement prediction from the track state, S is the measurement covariance matrix, and z is the actual measurement.

Say we have N tracks and M measurements. The association matrix A is an $N \times M$ matrix that contains the Mahalanobis distances between each track and each measurement:

$$A = \begin{pmatrix} d(x_1, z_1) & d(x_1, z_2) & \cdots & d(x_1, z_M) \\ d(x_2, z_1) & d(x_2, z_2) & \cdots & d(x_2, z_M) \\ \vdots & \vdots & \ddots & \vdots \\ d(x_N, z_1) & d(x_N, z_2) & \cdots & d(x_N, z_M) \end{pmatrix}$$

We also maintain lists for unassigned tracks and unassigned measurements. To perform association, we select the smallest entry in A to determine which track to update with which measurement, then remove this row and column from A and update the respective track and measurement lists. This process repeats until A is empty as shown in the figure 3.8 below:

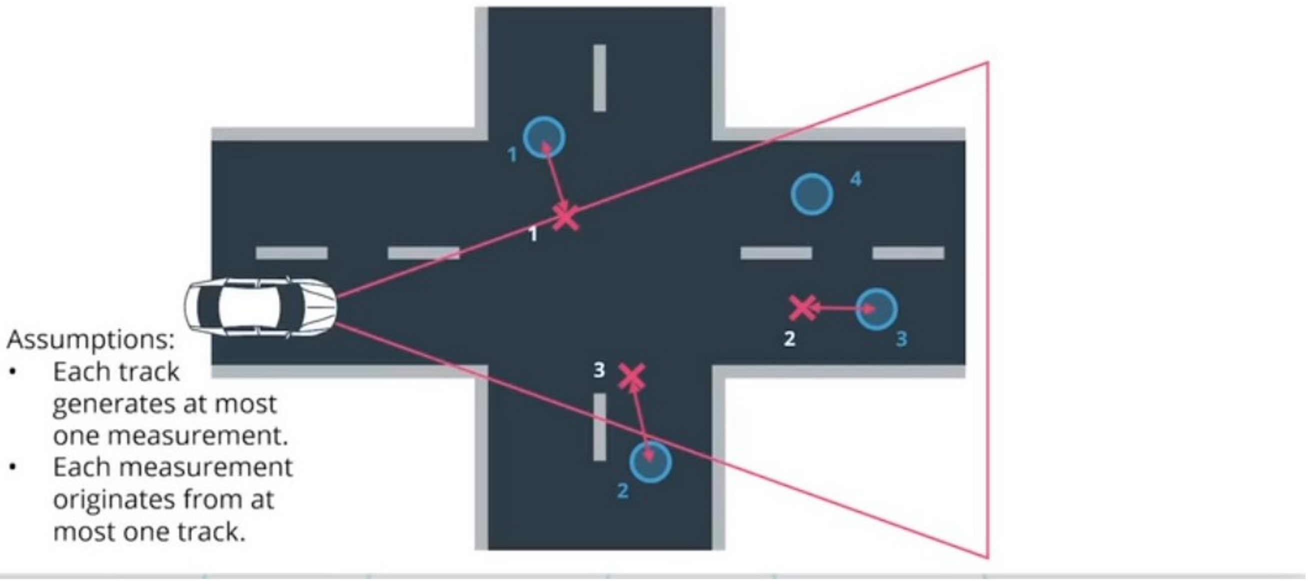

Assumptions:
• Each track generates at most one measurement.
• Each measurement originates from at most one track.
1
2
3
4
1
2
3

Figure 3.8: Data Association Problem [7]

- **Gating:** Gating [62] reduces association complexity by excluding unlikely track-measurement pairs. Since the residual (difference between predicted and measured values) is typically Gaussian, the Mahalanobis distance follows a χ^2 distribution. A measurement is considered inside a track's gate if the Mahalanobis distance is smaller than a threshold calculated from the inverse cumulative χ^2 distribution, given by:

$$d(x, z) \leq F_{\chi^2}^{-1}(0.995 \mid \dim(z))$$

If a measurement lies outside a track's gate, the distance in A for that pair can be set to infinity to denote it as an invalid association, for example:

$$A = \begin{pmatrix} d(1,1) & \infty & \infty \\ \infty & d(2,2) & d(2,3) \\ d(3,1) & d(3,2) & d(3,3) \\ \infty & d(4,2) & d(4,3) \end{pmatrix}$$

- **Track Management:** The system manages the track list, handling the creation, maintenance, and deletion of tracks as objects enter and exit the scene. A simple way to define a track score is by setting it as the ratio of detections in the last n frames to n, i.e., score $= \frac{\text{detections in last } n \text{ frames}}{n}$. However, there are various methods to define a track score, such as confidence or existence probability, allowing for custom definitions.

  Based on the track score, different track states can be defined, such as "initialized," "tentative," or "confirmed." For instance, thresholds can be set for transitioning between these states. An example of thresholding could be:

  - **Example Track Deletion Criteria:**
    - **Confirmed tracks:** Delete if score < 0.6. This applies only to confirmed tracks that previously had a score above 0.8 and has dropped below 0.6.
    - **Tentative or initialized tracks:** Delete if score < 0.17. This lower threshold accommodates newly initialized or tentative tracks, ensuring they stabilize before deletion.

  The figure 3.9 below illustrates different track states:

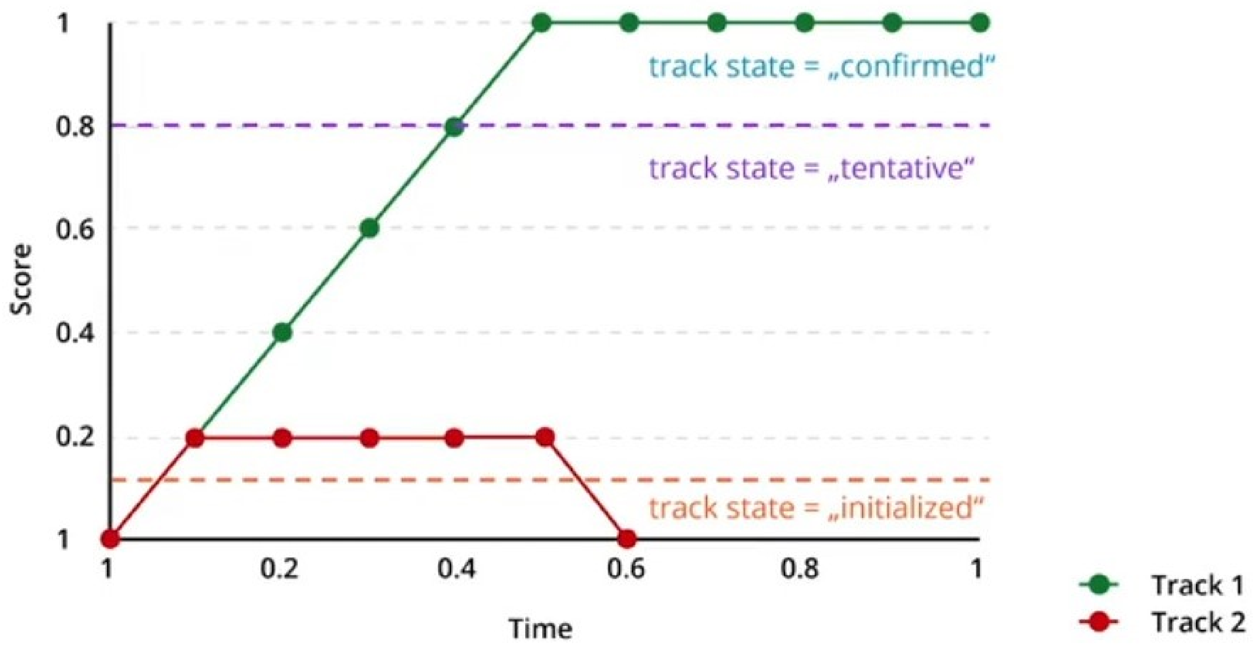


Figure 3.9: Track Management

- **Fusion of Camera and LiDAR Results:** The final step involves fusing the tracked objects from both the camera and LiDAR data. By combining the results from both sensor modalities, the system leverages the strengths of each sensor, achieving more robust and accurate multi-object tracking.

  To integrate objects detected in the BEV map and tracked through Kalman filtering and association into the camera frame for DeepSort, the following transformation process is applied:

  - **Transformation from BEV 3D Bounding Boxes to Camera Bounding Boxes:** Objects detected and tracked in the BEV map have 3D bounding box coordinates $(x_{BEV}, y_{BEV}, z_{BEV}, l_{BEV}, w_{BEV}, h_{BEV}, \theta_{BEV})$, where $(x_{BEV}, y_{BEV}, z_{BEV})$ denote the center coordinates, l_{BEV}, w_{BEV}, and h_{BEV} represent the length, width, and height of the bounding box in BEV space, and θ_{BEV} is the orientation angle relative to the BEV frame.

    To transform these coordinates into the camera frame:

    $$\begin{pmatrix} x_{cam} \\ y_{cam} \\ z_{cam} \end{pmatrix} = \mathbf{R}_{cam-BEV} \begin{pmatrix} x_{BEV} \\ y_{BEV} \\ z_{BEV} \end{pmatrix} + \begin{pmatrix} t_{x,cam} \\ t_{y,cam} \\ t_{z,cam} \end{pmatrix}$$

    where $\mathbf{R}_{cam-BEV}$ is the rotation matrix and $\begin{pmatrix} t_{x,cam} \\ t_{y,cam} \\ t_{z,cam} \end{pmatrix}$ is the translation vector from BEV to camera coordinates.

  - **Bounding Box Projection:**
    Once in the camera frame, project the 3D bounding box onto the 2D image plane using the camera intrinsic matrix $\mathbf{K}$ to obtain the 2D bounding box coordinates $(u_{cam}, v_{cam}, l_{cam}, w_{cam}, \theta_{cam})$:

    $$\begin{pmatrix} u_{cam} \\ v_{cam} \end{pmatrix} = \mathbf{K} \begin{pmatrix} x_{cam}/z_{cam} \\ y_{cam}/z_{cam} \\ 1 \end{pmatrix}$$

  - **Feed to DeepSort:**
    Finally, the projected 2D bounding box coordinates along with the associated track ID and confidence score are fed into DeepSort for integrated multi-object tracking across both camera and LiDAR data. DeepSort utilizes these inputs to refine track associations and maintain consistent object identities over time, leveraging the complementary strengths of both sensor modalities.

The figure 3.10 below demonstrates how the Kalman Filter predicts and updates based on new measurements, and how the fusion process integrates detections from camera and LiDAR into a unified tracking framework.

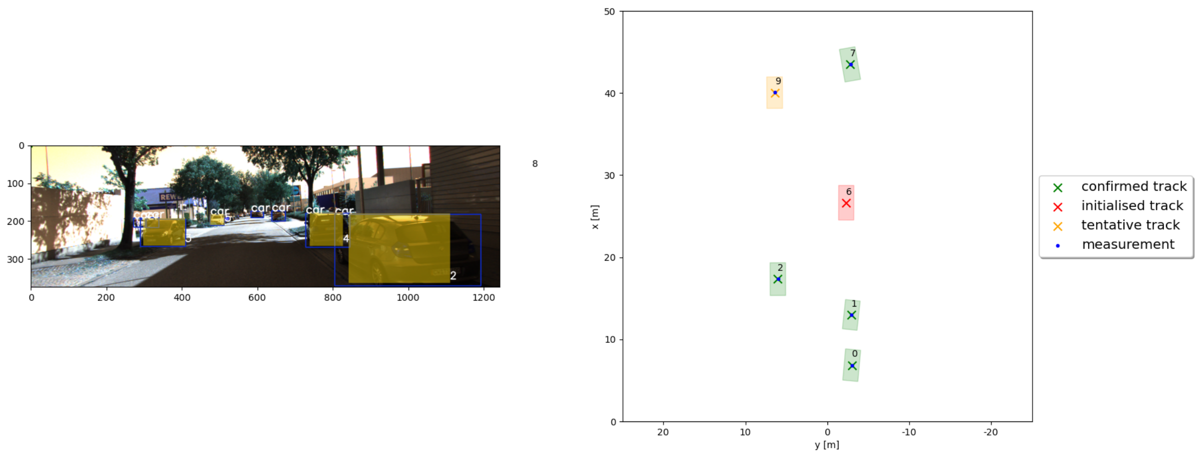


Figure 3.10: integrates detections from camera and LiDAR into a unified tracking framework

3.6 Algorithm of the Proposed Approach

The proposed approach for multi-object tracking integrates camera and LiDAR data to leverage the strengths of each sensor, achieving robust and accurate tracking. This section outlines the methodology used in our approach, which includes data processing from both camera and LiDAR sensors, object detection, Kalman Filter-based tracking, and the fusion of results from both sensors.

The process begins with the initialization of sensors and the perception module. The system continuously processes each frame until the last frame is reached. The camera captures the scene, and YOLOv8, a state-of-the-art object detection algorithm, is applied to detect 2D objects. These detected objects are compiled into a measurement list and processed through the Kalman Filter for state prediction. DeepSORT is then used to track these 2D objects across frames.

Simultaneously, the LiDAR sensor captures the environment to produce a point cloud, which is transformed into a Bird-Eye-View (BEV) map for easier processing. Complex YOLO is applied to this BEV map to detect 3D objects. Similar to the camera data, these detected objects are compiled into a measurement list and processed through the Kalman Filter for state prediction and tracking.

An association matrix is constructed to match detected objects to existing tracks, ensuring continuity of tracking across frames. The Kalman Filter updates the state of each track based on new measurements, and the system manages the track list by handling the creation, maintenance, and deletion of tracks as objects enter and exit the scene.

The final step involves fusing the tracked objects from both the camera and LiDAR data. By combining the results from both sensor modalities, the system achieves a more robust and accurate multi-object tracking solution. The detailed steps of the algorithm are as follows:

Algorithm 1 Multi-Object Tracking Based on Camera and LiDAR Fusion

```
**Input:**
    Camera images $I_t$
    LiDAR point clouds $P_t$
**Output:**
    Fused object tracks with 3D bounding boxes
**Step 1: Sensors and Perception Module Initialization**
Initialize sensors and perception module.
**while** last frame not reached **do**
    **Step 2: Camera Data Processing**
    Capture scene with the camera.
    $B_c \leftarrow$ YOLOv8($I_t$) {Detect 2D objects in camera images}
    Compile detected objects into a measurement list $M_c$.
    **Step 3: Kalman Filter (KF) Estimation for Camera**
    Predict the state of each detected object using the Kalman Filter.
    Initialize state and covariance matrices based on the first measurement.
    Predict where the object will be after time $\Delta t$.
    Compare the predicted location with the new measurement.
    Update the location based on the predicted and measured values.
    $T_c \leftarrow$ DeepSORT($M_c$) {Track 2D objects}
    **Step 4: LiDAR Data Processing**
    Capture the environment with the LiDAR sensor to produce a point cloud $P_t$.
    Transform the point cloud data into a Bird-Eye-View (BEV) map:
        $P_{bev} \leftarrow \begin{pmatrix} \cos\theta & -\sin\theta & 0 & t_x \\ \sin\theta & \cos\theta & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} P_t$
    $B_l \leftarrow$ Complex YOLO($P_{bev}$) {Detect 3D objects in BEV map}
    Compile detected 3D objects into a measurement list $M_l$.
    **Step 5: Kalman Filter Prediction for LiDAR**
    Predict the states of detected 3D objects based on the measurement list $M_l$.
    Create a list of predicted tracks $T_l$.
    **Step 6: Association**
    Construct an association matrix to match detected objects to existing tracks.
    Update the state of each track based on association results and new measurements.
    **Step 7: Track Management**
    Manage the track list, handling creation, maintenance, and deletion of tracks.
    **Step 8: Fusion of Camera and LiDAR Results**
    $T_{fused} \leftarrow$ Fuse($T_c$, $T_l$) {Combine 2D and 3D tracks for final fusion}
**end while**
Transition the vehicle to a parked state and end the process.
**Return:** Fused object tracks with 3D bounding boxes =0
```

3.7 Lane Tracking Using LaneNet for Enhanced Multi-Object Tracking

Lane tracking is a crucial component in autonomous driving, providing valuable information for object localization, obstacle detection, and avoidance. LaneNet [4], a real-time lane detection network, can significantly enhance our sensor fusion algorithm by integrating precise lane information into the object detection and tracking pipeline. This integration enables more accurate and reliable multi-object tracking and enhances the system's ability to perform obstacle detection and avoidance.

Integration of LaneNet with Sensor Fusion:

1. Lane Detection with LaneNet: LaneNet processes the camera images to detect lane lines in real-time. It utilizes a two-stage architecture, where the first stage extracts features from the image and the second stage localizes lane pixels using learnable anchors and offsets. The output is a set of lane boundaries and their corresponding confidence scores.

2. Object Detection and Tracking Enhancement: By combining lane information from LaneNet with object detections from YOLOv8 and Complex YOLO, the system can more accurately localize objects relative to the detected lanes. This spatial context improves the accuracy of object tracking, especially in dynamic environments where objects may temporarily occlude each other or move unpredictably.

3. Obstacle Detection and Avoidance: With precise lane information, the system can better predict potential collisions and plan avoidance maneuvers. The fusion of lane data with object tracks helps the vehicle understand its surroundings more comprehensively, leading to safer and more efficient navigation.

The detailed steps of the algorithm are as follows:

Algorithm 2 LaneNet Integration for Enhanced Multi-Object Tracking

```
Input:
    Camera images $I_t$
    LiDAR point clouds $P_t$
    Sensor fusion results $T_{fused}$
Output:
    Fused object tracks with 3D bounding boxes
    Lane boundaries
Step 1: Sensors and Perception Module Initialization
Initialize sensors and perception module.
while last frame not reached do
    Step 2: Camera Data Processing
    Capture scene with the camera.
    $B_c \leftarrow$ YOLOv8($I_t$) {Detect 2D objects in camera images}
    Compile detected objects into a measurement list $M_c$.
    Step 3: Lane Detection with LaneNet
    $Lanes \leftarrow$ LaneNet($I_t$) {Detect lane boundaries}
    Step 4: Fusion with Sensor Data
    Integrate $Lanes$ with sensor fusion results $T_{fused}$.
    Step 5: Enhanced Object Tracking
    Utilize integrated lane and object data for more accurate tracking.
    Step 6: Obstacle Detection and Avoidance
    Utilize fused data to predict potential collisions and plan avoidance maneuvers.
    Update the world model with the fused object tracks and lane boundaries.
end while
Return: Fused object tracks with 3D bounding boxes and lane boundaries =0
```

3.8 Conclusion

In this chapter, we have presented our proposition of sensor fusion for enhanced multi-object tracking in autonomous driving systems. The proposed approach integrates data from both camera and LiDAR sensors, leveraging their complementary strengths to achieve robust and accurate object tracking. Our methodology involves the use of advanced object detection algorithms such as YOLOv8 for camera data and Complex YOLO for LiDAR data, coupled with Kalman Filter-based tracking and DeepSORT for maintaining track continuity across frames. The fusion of camera and LiDAR data results in more reliable object localization and tracking, significantly improving the system's performance in dynamic environments.

Furthermore, we have discussed the integration of LaneNet for lane tracking, which provides additional contextual information for object detection and tracking. By incorporating precise lane boundaries into our sensor fusion algorithm, we enhance the system's ability to perform obstacle detection and avoidance. This integration allows for more accurate object localization relative to the detected lanes, improving safety and efficiency in navigation.

Chapter 4

Results and Evaluation

Chapter 4

Results and Evaluation

4.1 Introduction

In this chapter, we focus on the testing and evaluation of our sensor fusion algorithm. This phase is crucial for measuring the performance and effectiveness of our approach in multi-object tracking using camera and LiDAR data. The evaluation process ensures that the algorithm meets the desired accuracy and robustness required for real-world applications in autonomous driving. We will discuss the tools and frameworks used for development, followed by detailed evaluation metrics and datasets employed to assess the algorithm's performance.

4.2 Tools and Frameworks Used

The development and evaluation of our algorithm were supported by several tools and frameworks:

- **Visual Studio Code:** Used for coding and debugging the algorithm. Its extensive plugin ecosystem and integrated terminal made it an ideal choice for development [63].
- **PyCharm IDE:** Provided an integrated environment for development and testing, especially useful for its advanced code analysis, graphical debugger, and test runner [64].
- **Jupyter Notebook:** Used for interactive data analysis and visualization. Its ability to combine code execution with rich text annotations made it ideal for exploratory analysis and visualization tasks [65].

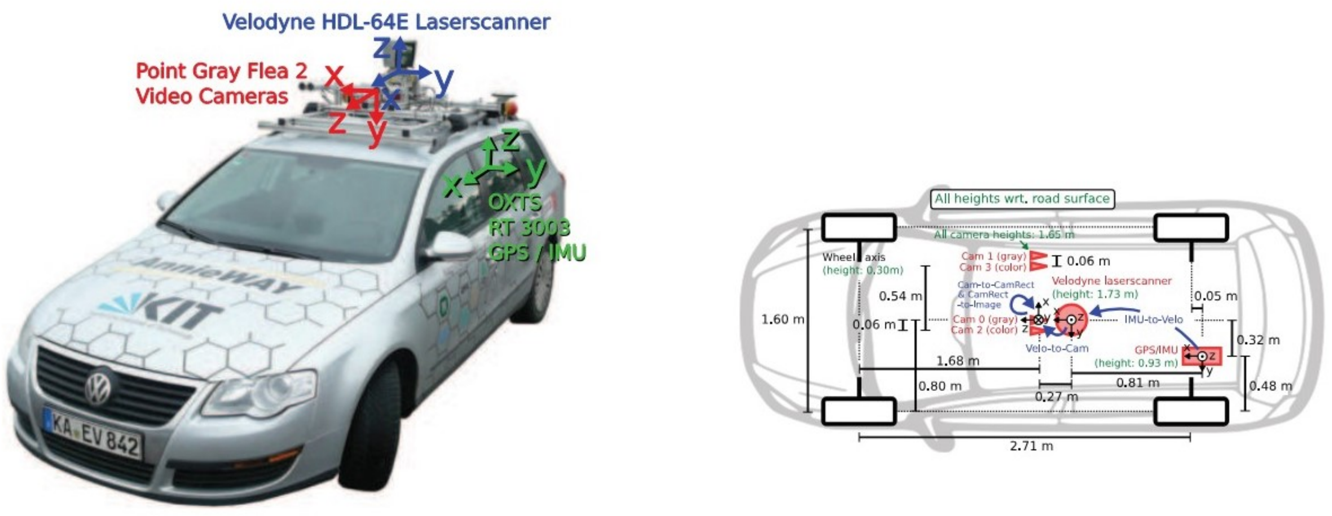
- **Python:** Programming language used for algorithm implementation, chosen for its simplicity and extensive libraries for machine learning and data processing [66].
- **NumPy:** Fundamental package for numerical computations in Python. It was used for handling arrays and performing mathematical operations required in our algorithm [67].
- **TensorFlow:** Deep learning framework used for training neural networks. Its comprehensive ecosystem allowed for efficient model development and deployment [68].
- **PyTorch:** Deep learning framework used for building and training models. Its dynamic computation graph and ease of use made it suitable for rapid prototyping and research [69].

These tools and frameworks facilitated efficient development and rigorous testing of our sensor fusion algorithm.

4.3 Metrics of Evaluation

To evaluate the performance of our algorithm, we utilized the following metrics and datasets:

1. **KITTI Dataset:** The KITTI Vision Benchmark Suite [20] offers an excellent set of datasets for automotive applications such as car and pedestrian recognition. These datasets include optical flow, stereo vision, visual odometry, SLAM, and object detection. KITTI object detection dataset is one of the most common datasets for driving scenes collected in the daytime and under favorable weather conditions. This dataset contains 7,481 training samples and 7,518 testing samples for both images (with a resolution of 1242×375) and point cloud. Like most other studies, the training dataset is divided into a training part (3,712 samples) and a validation part (3,769 samples). In addition to the easy-to-use dataset formats, Geiger, Lenz, and Urtasun provided a vast amount of information on every labeled object and provided high-quality images. They used the annotation files to apply a filter based on distance, car type, and orientation to retrieve a total of 1500 rear-view car images. As shown in 4.1 (a), the camera and the LiDAR utilize two distinct coordinate systems (red for the camera and blue for the LiDAR). The directions (X,Y, Z) are set as (rightward, downward, forward) and (forward, leftward, upward) from the camera and LiDAR, respectively. All ground-truth data is delivered in camera coordinates, but it is not difficult to convert from camera to LiDAR coordinates and vice versa by using the calibration information. As can be seen in Figure 4.1b (b), the sensors (red) are positioned with respect to the vehicle body according to their dimensions. The measurements are carried out according to the road surface and the height above the ground is indicated in green. The blue indicates transformations among the sensors.
2. **KITTI MOT Benchmark:** Specifically, the Multiple Object Tracking (MOT) benchmark within the KITTI dataset, which includes standardized evaluation metrics for tracking performance. This benchmark provides a standardized set of metrics to compare different tracking algorithms objectively.
3. **Evaluation Metrics:**


Velodyne HDL-64E Laserscanner
Point Gray Flea 2 Video Cameras
OXTS RT 3003 GPS / IMU
All heights wrt. road surface
All camera heights: 1.65 m
Wheel axis (height: 0.30m)
Cam 1 (gray) Cam 3 (color)
0.06 m
Velodyne laserscanner (height: 1.73 m)
Cam-to-CamRect & CamRect -to-Image
Cam 0 (gray) Cam 2 (color)
IMU-to-Velo
Velo-to-Cam
GPS/IMU (height: 0.93 m)
1.60 m
0.54 m
0.06 m
1.68 m
0.80 m
0.27 m
0.81 m
0.05 m
0.32 m
0.48 m
2.71 m

(a) Fully equipped vehicle (Volkswagen Passat B6)

(b) Setup bird's eye view (BEV)

Figure 4.1: KITTI vehicle setup. (a) Fully equipped vehicle (Volkswagen Passat B6); (b) Setup bird's eye view (BEV).

- **HOTA (Higher Order Tracking Accuracy):** HOTA [70] combines detection and association accuracy to provide a single unified metric for tracking performance. It balances the trade-off between detection accuracy and association accuracy.

$$\text{HOTA} = \sqrt{\text{DetA} \times \text{AssA}}$$

(Higher HOTA indicates better performance)

- **MOTA (Multiple Object Tracking Accuracy):** MOTA [71] measures the ratio of correctly tracked objects, considering false positives, false negatives, and identity switches.

$$\text{MOTA} = 1 - \frac{\sum_t (FN_t + FP_t + ID\_sw_t)}{\sum_t GT_t}$$

(Higher MOTA indicates better performance)

where FN_t is the number of false negatives, FP_t is the number of false positives, ID_sw_t is the number of identity switches, and GT_t is the number of ground truth objects at time t.

- **MOTP (Multiple Object Tracking Precision):** MOTP [72] indicates the localization precision of tracked objects by measuring the average distance between the predicted and the ground truth positions.

$$\text{MOTP} = \frac{\sum_{i,t} d_{i,t}}{\sum_t c_t}$$

(Higher MOTP indicates better precision)

where $d_{i,t}$ is the distance between the predicted and ground truth positions for object i at time t, and c_t is the number of matches at time t.

- **DetA (Detection Accuracy):** Detection accuracy assesses the accuracy of object detection, considering both true positives and false negatives.

$$\text{DetA} = \frac{TP}{TP + FN}$$

(Higher DetA indicates better detection accuracy)

where TP is the number of true positives and FN is the number of false negatives.

- **AssA (Association Accuracy):** Association accuracy evaluates the accuracy of object associations across frames.

$$\text{AssA} = \frac{TP}{TP + ID\_sw}$$

(Higher AssA indicates better association accuracy)

where ID_sw is the number of identity switches.

- **DetRe (Detection Recall):** Detection recall measures the proportion of correctly detected objects out of the total number of ground truth objects.

$$\text{DetRe} = \frac{TP}{TP + FN}$$

(Higher DetRe indicates better recall)

- **DetPr (Detection Precision):** Detection precision indicates the accuracy of detected object positions by measuring the proportion of true positives out of the total detected objects.

$$\text{DetPr} = \frac{TP}{TP + FP}$$

(Higher DetPr indicates better precision)

where FP is the number of false positives.

- **AssRe (Association Recall):** Association recall measures the proportion of correctly associated objects out of the total number of ground truth associations.

$$\text{AssRe} = \frac{TP_{assoc}}{TP_{assoc} + FN_{assoc}}$$

(Higher AssRe indicates better recall)

- **AssPr (Association Precision):** Association precision evaluates the precision of object associations by measuring the proportion of true positive associations out of the total detected associations.

$$\text{AssPr} = \frac{TP_{assoc}}{TP_{assoc} + FP_{assoc}}$$

(Higher AssPr indicates better precision)

- **LocA (Localization Accuracy):** Localization accuracy assesses the accuracy of object localization, measuring the average distance between predicted and ground truth object positions.

$$\text{LocA} = \frac{\sum_{i,t} d_{i,t}}{\sum_{i,t} 1}$$

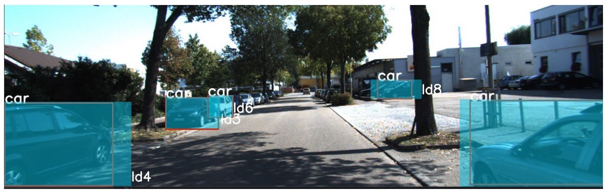
(Higher LocA indicates better localization accuracy)

These metrics provide comprehensive insights into the algorithm's performance in different aspects of object detection and tracking. Evaluating our algorithm against these metrics helps identify its strengths and areas for improvement, ensuring robust performance in real-world scenarios.

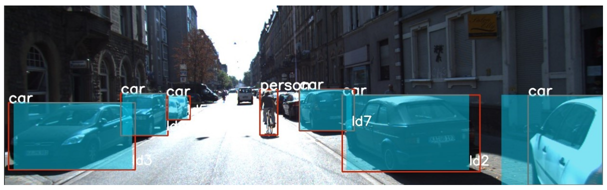
This chapter will present the results of our algorithm's evaluation based on these metrics and datasets, highlighting its effectiveness and reliability in multi-object tracking using camera and LiDAR data.

4.4 Results

In this section, we present the results of our proposed approach using the KITTI Tracking dataset. The process begins with acquiring the left camera images and LiDAR Velodyne binary files from the dataset. We detect objects in the camera images using YOLOv8 and track them with the DeepSORT tracker. The figure 4.2 below shows the results of detection and tracking on the left camera image data:



(a) Left camera image data results: object detection and tracking in sequence data 0



(b) Left camera image data results: object detection and tracking in sequence data 1

Figure 4.2: Left camera image data results: object detection and tracking in sequence data 0 And 1.

Next, we perform point cloud extraction as shown in Figure 4.3:

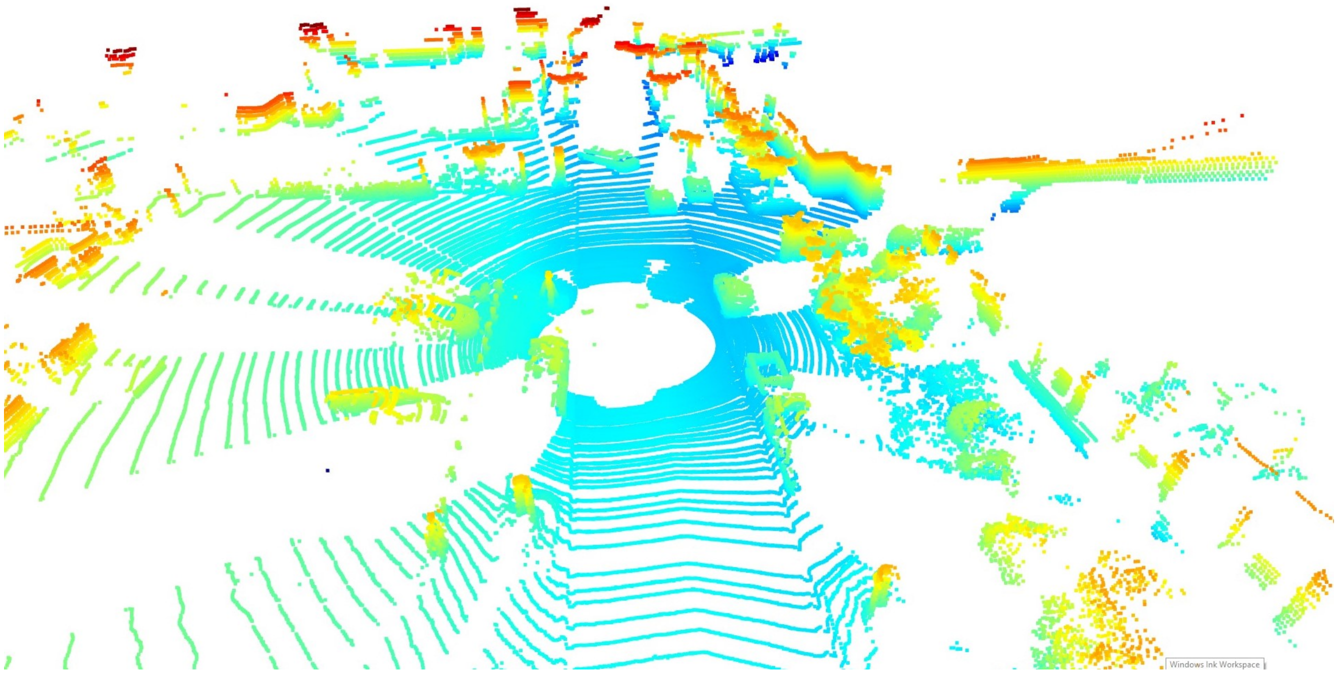


Figure 4.3: Point Cloud extraction

The point cloud is then converted into a BEV map with three channels: intensity, height, and density, resulting in an RGB BEV map as illustrated in the figure 4.4below:

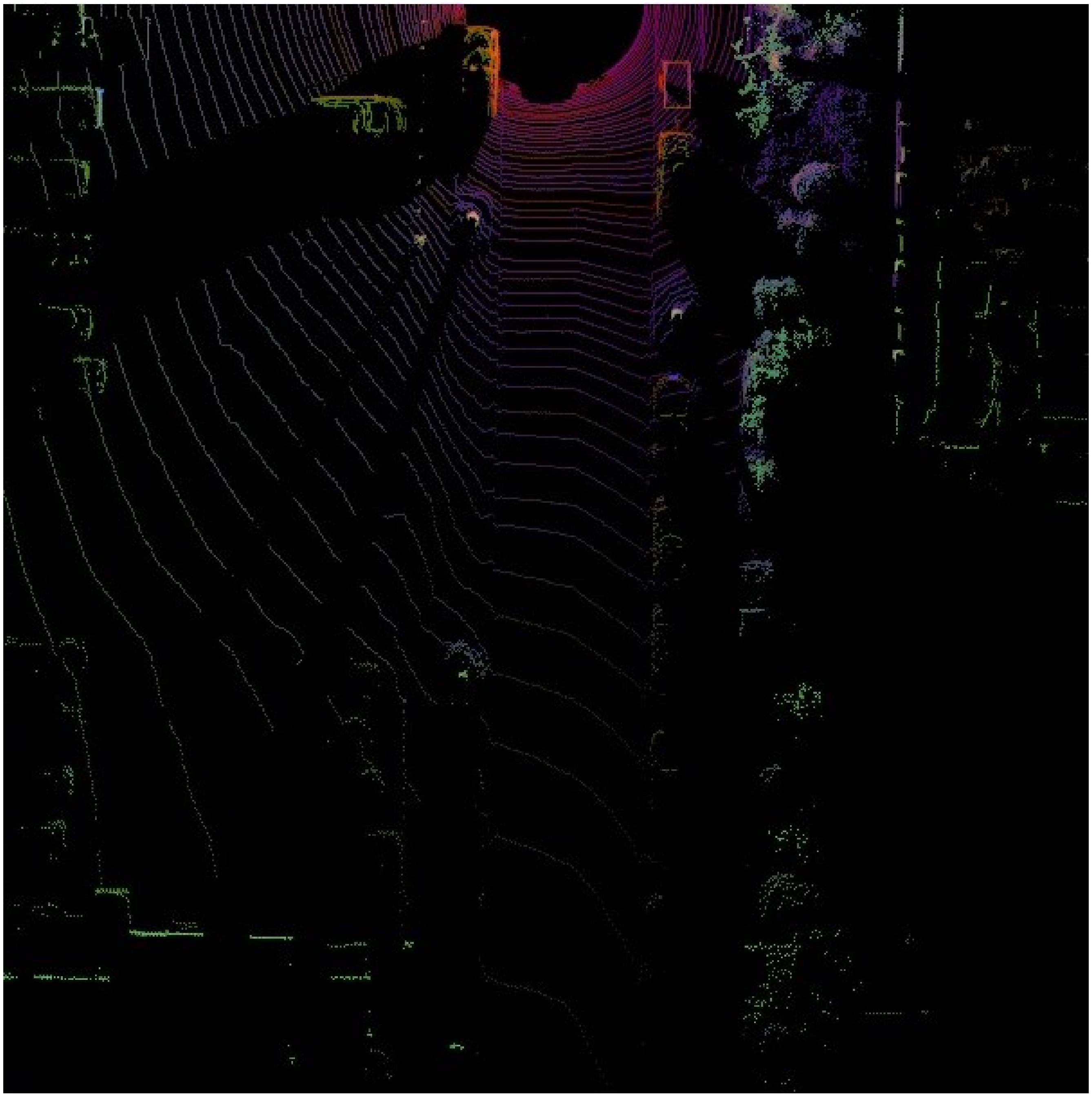


Figure 4.4: Point cloud conversion to BEV map with RGB channels (intensity, height, and density).

Following the BEV map creation, we perform 3D object detection using Complex YOLO. The results of 3D object detection are shown in the following figure 4.5:

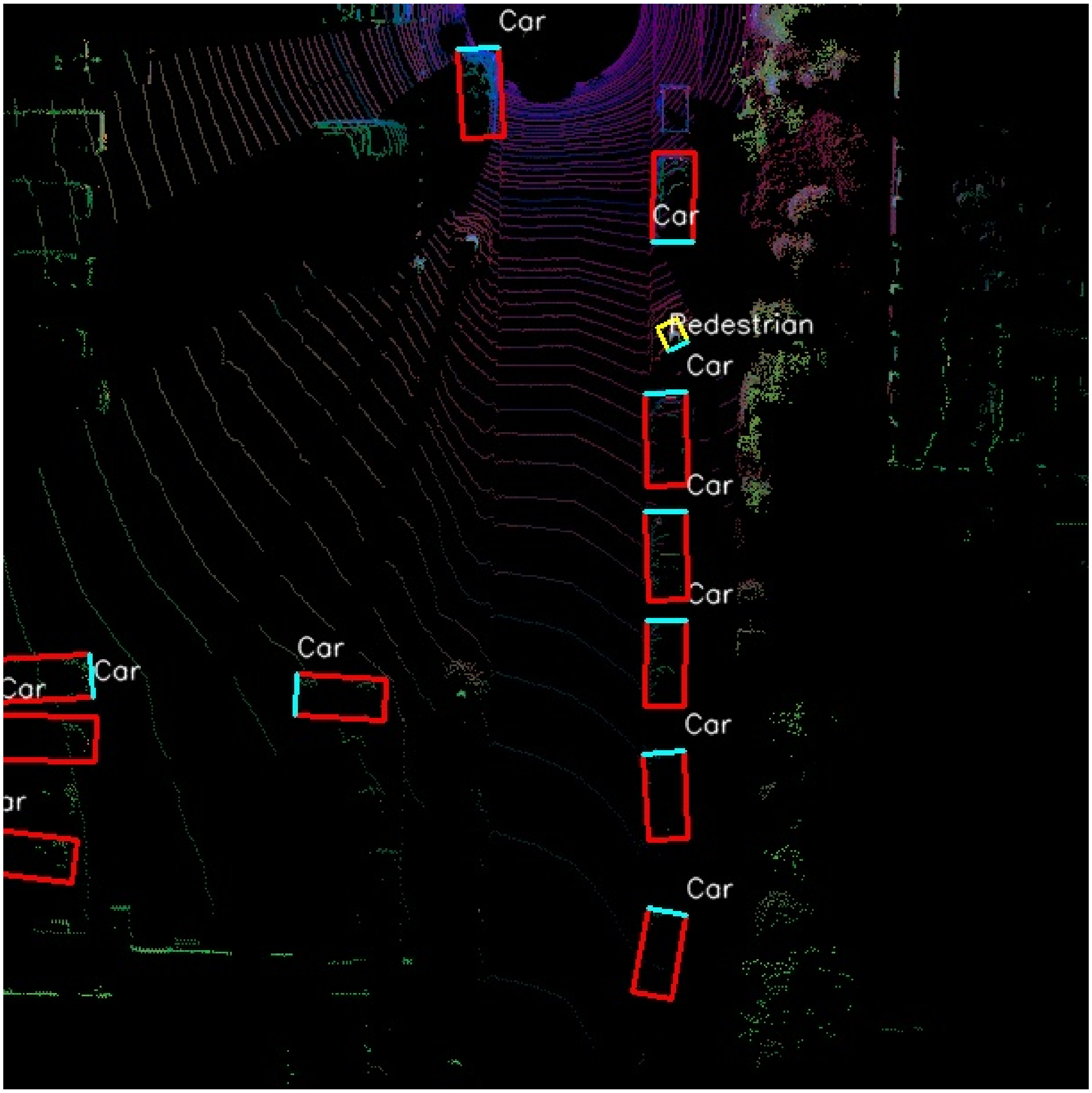


Figure 4.5: 3D object detection results

After obtaining the 3D detection results, we initialize our LiDAR sensor instance with calibration matrices and initialize our Kalman filter with the TrackManager. We then generate our measurement list using the Measurement class, which includes measurement values, covariance, timestamp, and sensor information. The Kalman filter prediction is performed for each track, followed by the association and update of the predicted state with the new measurement. Finally, we plot the results, as illustrated in the figure below, which demonstrates the results of our 3D object tracking:

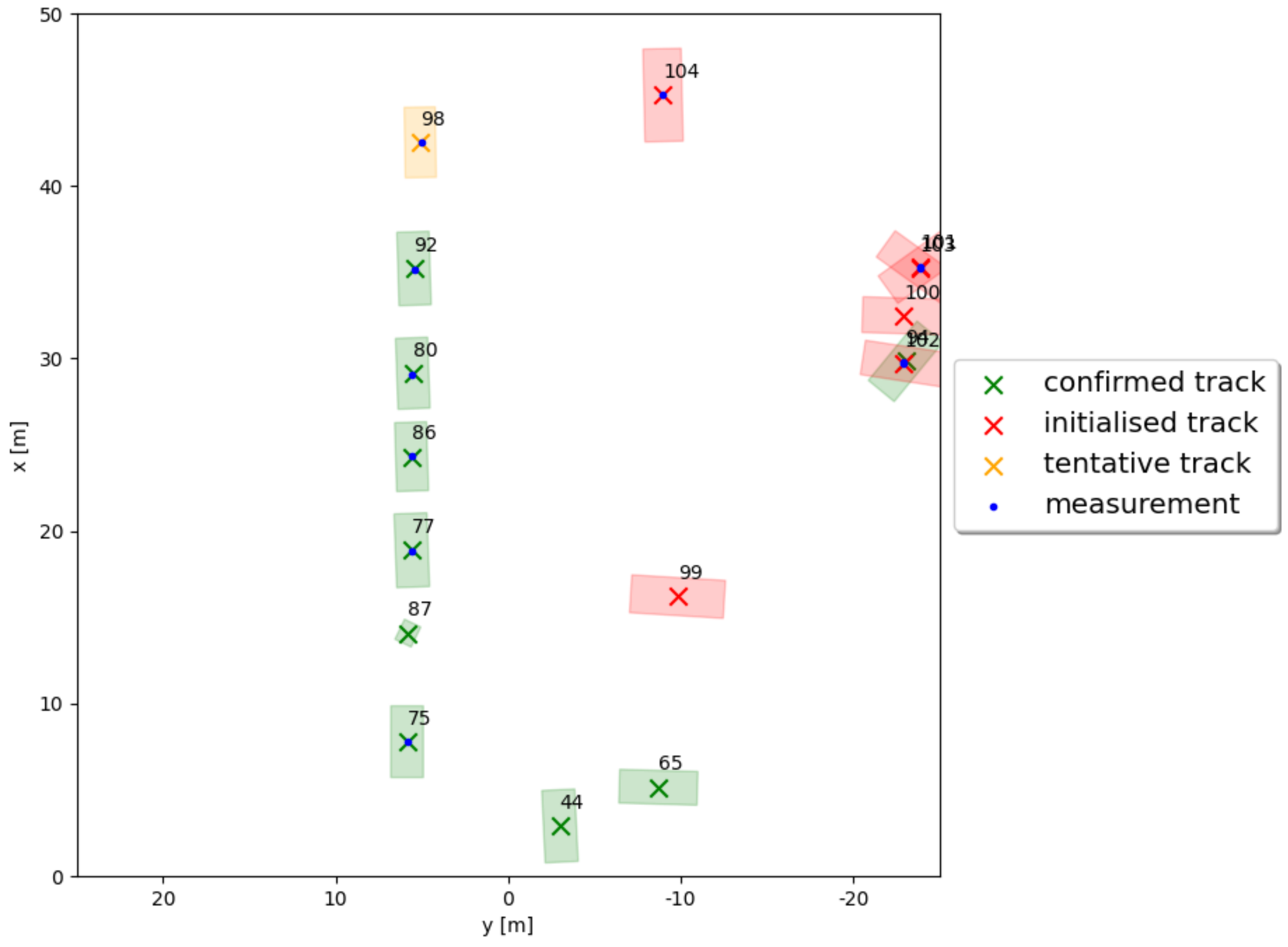
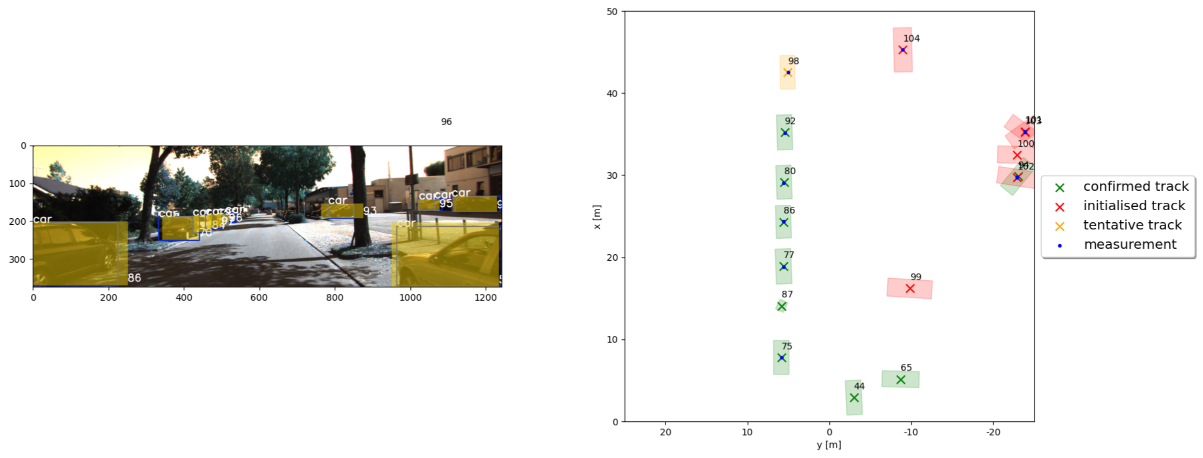
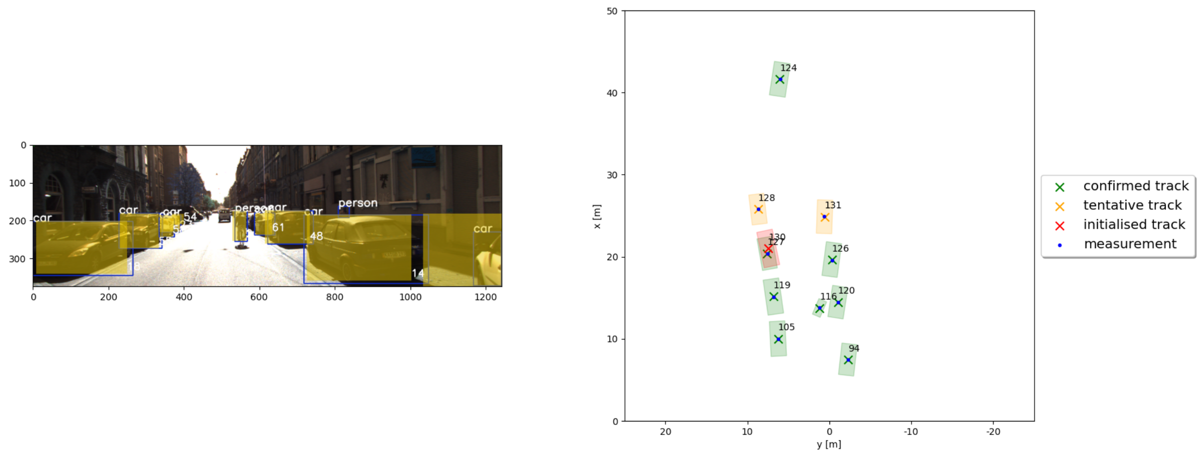


Figure 4.6: Results of 3D object tracking

After obtaining the tracking results, we perform a transformation from 3D LiDAR coordinates to 2D coordinates to feed the missed tracking of DeepSORT with the confirmed tracks of LiDAR detection. The figure 4.7 below illustrates the fusion results of the two trackers:



(a) Fusion results of camera and LiDAR trackers in sequence data 0



(b) Fusion results of camera and LiDAR trackers in sequence data 1

Figure 4.7: Fusion results of camera and LiDAR trackers.

As observed in the results, the fused tracking outputs shown in the images are significantly enhanced compared to the initial results depicted in Figure 4.2. The issues with missed or incorrect tracking have been rectified, resulting in more accurate and consistent tracking. This fusion process effectively enhances the DeepSORT tracker, providing superior performance compared to using only camera-based tracking.

4.5 Results Evaluation in KITTI MOT Evaluation

With the detection, tracking, and fusion processes completed, we proceed to evaluate our results using the KITTI Multiple Object Tracking (MOT) Evaluation. This evaluation will provide a comprehensive assessment of the performance and accuracy of our multi-object tracking algorithm.

From all 20 test sequences, our benchmark computes the HOTA tracking metrics (HOTA, DetA, AssA, DetRe, DetPr, AssRe, AssPr, LocA, MOTA, MOTP). The tables below show all of these metrics for cars and pedestrians.

4.5.1 Car Benchmark

Table 4.1: Evaluation Metrics for Car Benchmark

Benchmark	**HOTA**	**DetA**	**AssA**	**DetRe**	**DetPr**	**AssRe**	**AssPr**	**LocA**	**MOTA**	**MOTP**
CAR	67.566	64.103	71.415	66.337	88.451	73.394	90.858	88.154	74.923	86.533

- **HOTA (67.566)**: The Higher Order Tracking Accuracy (HOTA) score is relatively high, indicating good overall tracking performance.
- **DetA (64.103)**: The detection accuracy suggests that the system is reasonably accurate in identifying cars.
- **AssA (71.415)**: The association accuracy shows strong performance in maintaining the identity of detected cars across frames.
- **DetRe (66.337)**: The detection recall is moderate, meaning that the system successfully detects a significant portion of cars.
- **DetPr (88.451)**: The detection precision is high, implying that most of the detected objects are actually cars.
- **AssRe (73.394) & AssPr (90.858)**: The recall and precision for association indicate that the system effectively tracks cars once they are detected.
- **LocA (88.154)**: The localization accuracy is high, suggesting that the detected cars' positions are accurately estimated.
- **MOTA (74.923)**: The Multiple Object Tracking Accuracy (MOTA) score reflects a solid tracking performance, considering false positives, false negatives, and identity switches.
- **MOTP (86.533)**: The Multiple Object Tracking Precision (MOTP) is high, indicating precise location estimates for the tracked cars.

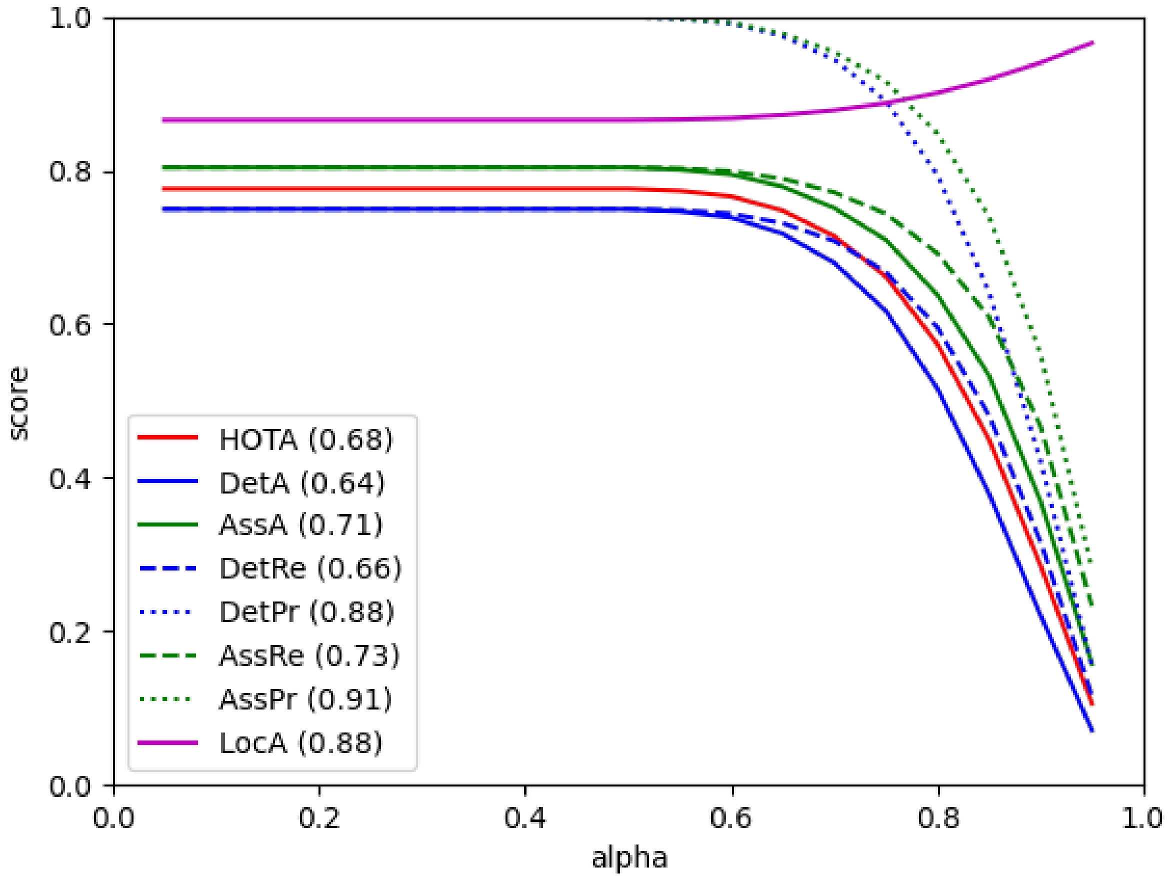


Figure 4.8: Car Plot Results

4.5.2 Pedestrian Benchmark

Table 4.2: Evaluation Metrics for Pedestrian Benchmark

Benchmark	**HOTA**	**DetA**	**AssA**	**DetRe**	**DetPr**	**AssRe**	**AssPr**	**LocA**	**MOTA**	**MOTP**
Pedestrian	70.617	67.843	73.650	70.740	86.585	76.209	88.739	86.869	81.042	84.784

- **HOTA (70.617)**: The HOTA score for pedestrians is slightly higher than for cars, indicating very good overall tracking performance.
- **DetA (67.843)**: The detection accuracy for pedestrians is better than for cars, showing reliable identification of pedestrians.
- **AssA (73.650)**: The association accuracy is robust, demonstrating effective tracking of pedestrian identities over time.
- **DetRe (70.740)**: The detection recall is higher for pedestrians, suggesting the system detects a larger portion of pedestrians.
- **DetPr (86.585)**: The detection precision for pedestrians is slightly lower than for cars but still indicates high precision.

- **AssRe (76.209) & AssPr (88.739)**: Both recall and precision for association are high, ensuring consistent tracking of pedestrians.
- **LocA (86.869)**: The localization accuracy is high, similar to the results for cars, showing precise positioning of detected pedestrians.
- **MOTA (81.042)**: The MOTA score for pedestrians is higher than for cars, indicating better overall tracking performance when considering detection and association errors.
- **MOTP (84.784)**: The MOTP score is slightly lower for pedestrians but still reflects good location precision for the tracked objects.

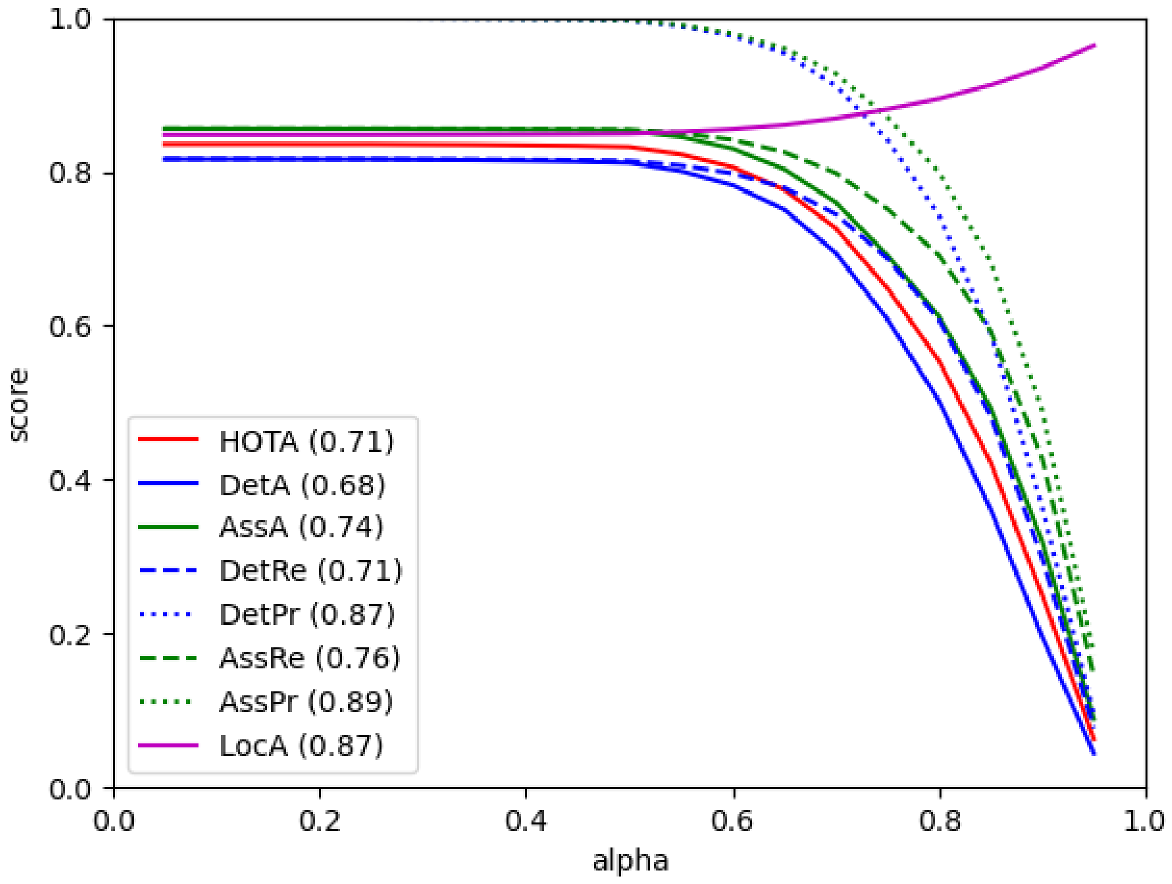


Figure 4.9: Pedestrian Plot Results

4.5.3 Discussion

The evaluation metrics indicate that the sensor fusion approach performs well for both cars and pedestrians, with particularly strong performance in tracking pedestrians. The high precision in detection and association shows that the system is effective in distinguishing between different objects and maintaining their identities over time. The slightly lower detection recall suggests room for improvement in detecting all relevant objects, especially for cars. Overall, the results demonstrate the capability of the sensor fusion system to provide reliable and precise tracking in dynamic environments.

Table 4.3: Comparative Evaluation Metrics for Car Benchmark

Benchmark	**HOTA**	**DetA**	**AssA**	**DetRe**	**DetPr**	**AssRe**	**AssPr**	**LocA**
Our Results	**67.566**	**64.103**	**71.415**	**66.337**	**88.451**	**73.394**	**90.858**	**88.154**
DeepFusionMOT[54]	75.46%	71.54%	80.05%	75.34%	85.25%	82.63%	89.77%	86.70%
JMTOD[53]	70.73%	73.45%	68.76%	78.67%	84.02%	72.46%	88.02%	86.95%
pnasMOT[73]	67.32%	77.69%	58.99%	81.58%	85.81%	64.70%	80.74%	86.94%

4.6 Hardware Implementation

Despite not having a 3D LiDAR for real-world evaluation of our proposed model and assessing its real-time capabilities, we can still conduct experiments and develop the first prototype of our self-driving car using the hardware currently available to us.

4.6.1 NVIDIA Jetson Nano

Figure 4.10: NVIDIA Jetson Nano [8]

The NVIDIA Jetson Nano is a small, powerful computer designed for AI and edge computing applications. It is ideal for running deep neural networks and processing data from various sensors in real-time.

4.6.2 Raspberry Pi 4

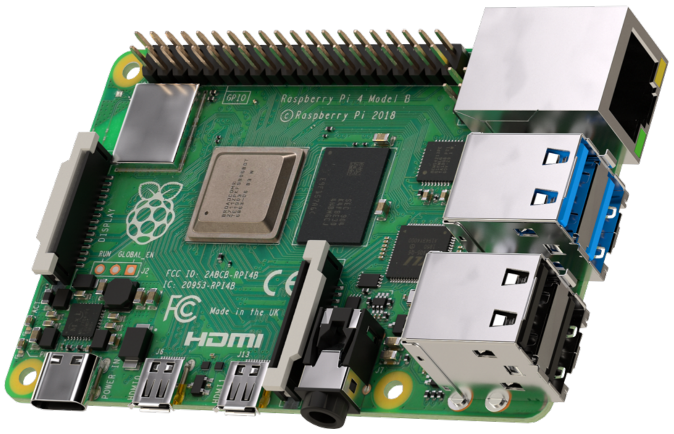


Figure 4.11: Raspberry Pi 4 [9]

The Raspberry Pi 4 is another popular single-board computer that supports a variety of applications, including robotics and IoT projects. It offers sufficient computing power for our self-driving car prototype.

4.6.3 Logitech C170 5MP Camera

Figure 4.12: Logitech C170 5MP Camera [10]

The Logitech C170 is a high-resolution camera capable of capturing detailed images and video, essential for vision-based tasks such as lane detection and object recognition.

4.6.4 Ultrasonic HC-SR04 Sensor

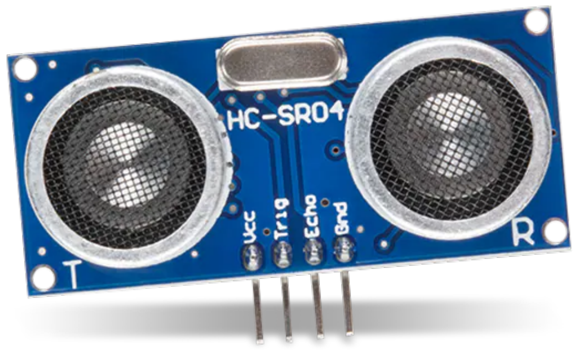


Figure 4.13: Ultrasonic HC-SR04 Sensor [11]

The HC-SR04 ultrasonic sensor provides distance measurements based on sound waves, suitable for detecting obstacles and assisting in navigation and collision avoidance.

4.6.5 DC-DC Converter Step-Down 5A

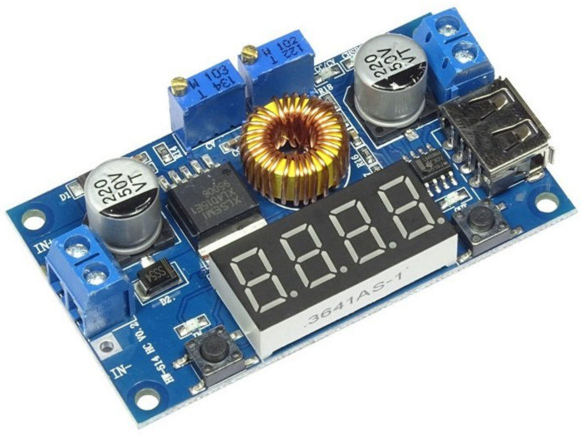


Figure 4.14: DC-DC Converter Step-Down 5A [12]

The DC-DC converter steps down voltage efficiently, providing stable power to various components of our self-driving car prototype.

4.6.6 Motor Driver L298N

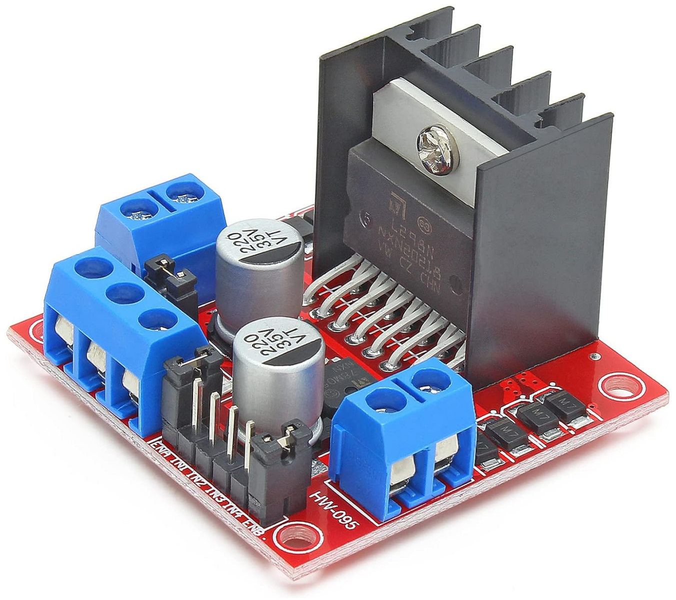


Figure 4.15: Motor Driver L298N

The L298N motor driver controls the speed and direction of DC motors in our prototype, enabling precise movement and maneuverability.

4.6.7 1/16 Scale RC Car Conquer

Figure 4.16: 1/16 Scale RC Car Conquer

The 1/16 scale RC car serves as the chassis for our self-driving car prototype, providing a platform for integrating all components and testing functionalities.

4.6.8 First Prototype V1

Combining all hardware components together, we have successfully developed the first prototype of our self-driving car. The prototype integrates the following key components:

- **NVIDIA Jetson Nano:** for onboard computation and processing of sensor data.
- **Raspberry Pi 4;** for coordinating overall decision-making and control.
- **Logitech C170 5MP Camera:** for visual perception tasks such as object detection and lane tracking.
- **Ultrasonic HC-SR04 Sensor:**for close-range obstacle detection and avoidance.
- **DC-DC Converter Step-Down 5A:** to ensure stable power supply to all components.
- **Motor Driver L298N:** for controlling the speed and direction of the DC motors.
- **1/16 Scale RC Car Conquer:** serving as the physical platform for the prototype.

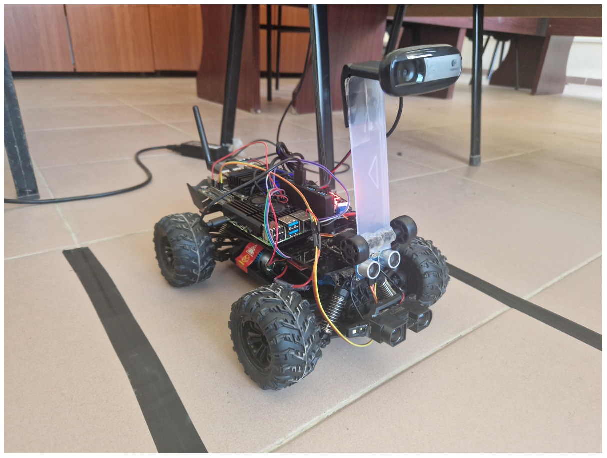
The software stack running on the NVIDIA Jetson Nano includes advanced algorithms for:

- **Object detection using YOLOv8:** for camera images and Complex YOLO for LiDAR point clouds.
- **Lane tracking and detection:** using deep learning-based models "LaneNet".
- **Obstacle avoidance strategies:** based on sensor fusion and real-time environmental perception.

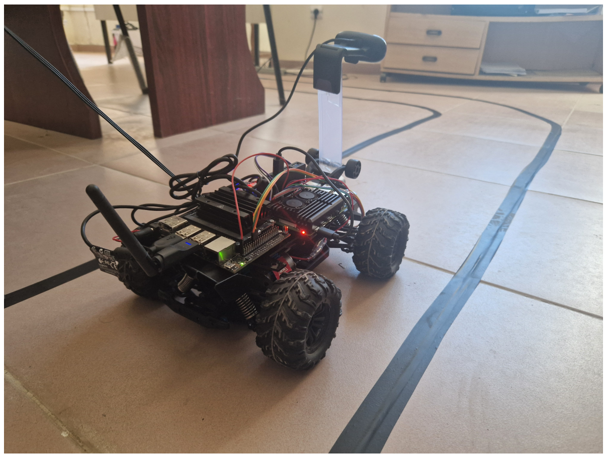
The NVIDIA Jetson Nano processes sensor data from cameras and ultrasonic sensors, combining them to create a comprehensive understanding of the vehicle's surroundings. It then computes the optimal steering angle and speed based on these inputs.

The decision outputs from the Jetson Nano, including steering angle and speed commands, are transmitted to the Raspberry Pi 4. The Raspberry Pi 4 handles high-level decision-making processes, coordinating navigation, route planning, and real-time adjustments based on the environmental data received from the Jetson Nano.

This integration allows our prototype to autonomously navigate environments, make decisions in real-time, and demonstrate fundamental capabilities of autonomous driving technology.



(a) Front side of our self-driving car prototype



(b) Back side of our self-driving car prototype

Figure 4.17: Our self-driving car prototype. (a) Front side view. (b) Back side view.

4.7 Conclusion

In this chapter, we evaluated the performance of our proposed sensor fusion algorithm using the KITTI dataset. The results demonstrate that our approach significantly enhances object detection and tracking accuracy by effectively integrating data from both camera and LiDAR sensors. By employing advanced object detection methods such as YOLOv8 and Complex YOLO, and combining them with the DeepSORT tracking framework and Extended Kalman Filters, we achieved robust and reliable tracking performance.

Our evaluation metrics showed substantial improvements in key performance indicators, including HOTA, DetA, AssA, and LocA, confirming the efficacy of our fusion approach. This comprehensive evaluation underscores the potential of our sensor fusion solution to improve the safety and reliability of autonomous vehicles, paving the way for future advancements in autonomous driving technology.

General Conclusion and Perspectives

The integration of sensor fusion in autonomous vehicles is crucial for ensuring safety and reliability, which are fundamental for the future of autonomous driving. Sensor fusion combines data from various sensors to provide a comprehensive understanding of the vehicle's surroundings, enabling precise and accurate object detection, tracking, and decision-making processes. This technology is vital for the development of autonomous vehicles that can operate safely in diverse and dynamic environments.

In our project, we proposed a novel sensor fusion algorithm that leverages both camera and LiDAR data to enhance object detection and tracking accuracy. Our approach integrates state-of-the-art object detection algorithms, YOLOv8 for camera images and Complex YOLO for LiDAR point clouds, with a tracking framework using DeepSORT and Kalman Filters. The evaluation of our approach on the KITTI dataset demonstrated significant improvements in tracking performance, showcasing the effectiveness of our sensor fusion method. The comprehensive evaluation metrics, including HOTA, DetA, AssA, DetRe, DetPr, AssRe, AssPr, LocA, MOTA, and MOTP, validate the robustness and accuracy of our proposed solution.

Looking ahead, there are several perspectives for enhancing our algorithm. One avenue is to incorporate more advanced object detection methods to achieve even more robust tracking, as accurate detection is the cornerstone of reliable tracking systems. Additionally, integrating our fusion algorithm with Simultaneous Localization and Mapping (SLAM) techniques could further improve the system's ability to navigate and understand complex environments.

However, we faced several difficulties during this project, particularly related to hardware limitations. The lack of high-performance hardware for training and validating our models posed significant challenges, as did the absence of a 3D LiDAR sensor for real-world testing and real-time application. Addressing these hardware constraints is essential for advancing our research and implementing our solutions in practical autonomous vehicle systems.

In conclusion, our research highlights the critical role of sensor fusion in autonomous vehicles and presents a promising solution for improving object detection and tracking. By continuing to refine our methods and addressing hardware limitations, we can contribute to the development of safer and more reliable autonomous driving technologies.

Bibliography

[1] Siddhant Jain. Six levels of automation: Fully manual to fully autonomous. Medium, 2021. https://medium.com/@siddahantjain50/six-levels-of-automation-fully-manual-to-fully-autonomous-2d528ea Accessed: 2024-06-23.

[2] De Jong Yeong, Gustavo Velasco-Hernandez, John Barry, and Joseph Walsh. Sensor and sensor fusion technology in autonomous vehicles: A review, 02 2021.

[3] Ning Lu, Nan Cheng, Ning Zhang, Xuemin Shen, and Jon Mark. Connected vehicles: Solutions and challenges. *Internet of Things Journal, IEEE*, 1:289–299, 08 2014.

[4] Ze Wang, Weiqiang Ren, and Qiang Qiu. Lanenet: Real-time lane detection networks for autonomous driving. *arXiv preprint arXiv:1807.01726*, 2018.

[5] G Ajay Kumar, Jin-Hee Lee, Jongrak Hwang, Jaehyeong Park, Sung-Hoon Youn, and Soon Kwon. Lidar and camera fusion approach for object distance estimation in self-driving vehicles. *Symmetry*, 12:324, 02 2020.

[6] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[7] Udacity. Self-driving car engineer nanodegree. https://www.udacity.com/course/self-driving-car-engineer-nanodegree--nd0013, 2024. Accessed: 2024-06-09.

[8] Nerves system: Nvidia jetson nano. https://hexdocs.pm/nerves_system_jetson_nano/readme.html. Accessed: 2024-07-09.

[9] Getting started with raspberry pi 4 model b. https://www.okdo.com/getting-started/get-started-with-raspberry-pi-4-model-b/. Accessed: 2024-07-09.

[10] Logitech. Logitech c170 webcam quickstart guide. https://www.logitech.com/assets/46920/2/webcam-c170-quickstart-guide.pdf, 2024. Accessed: 2024-07-09.

[11] SparkFun Electronics. Hc-sr04 ultrasonic distance sensor. https://www.mouser.fr/new/sparkfun/sparkfun-hcsr04-distance-sensor/. Accessed: 2024-07-09.

[12] Dc-dc converter step-down 1.25-32v 5a with led display (xl4015). https://grobotronics.com/dc-dc-converter-step-down-1.25-32v-5a-with-led-display-xl4015.html?sl=en. Accessed: 2024-07-09.

[13] Keshav Bimbraw. Autonomous cars: Past, present and future - a review of the developments in the last century, the present scenario and the expected future of autonomous vehicle technology. *ICINCO 2015 - 12th International Conference on Informatics in Control, Automation and Robotics, Proceedings*, 1:191–198, 01 2015.

[14] Pengcheng Chen. Surface vehicle recommended practice (r) taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. 11 2021.

[15] Bahador Khaleghi, Alaa Khamis, Fakhreddine O Karray, and Saiedeh N Razavi. Multisensor data fusion: A review of the state-of-the-art. *Information fusion*, 14(1):28–44, 2013.

[16] Qi Zhu, Bo Yu, Ziran Wang, Jie Tang, Qi Alfred Chen, Zihao Li, Xiangguo Liu, Yunpeng Luo, and Lingzi Tu. *Cloud and Edge Computing for Connected and Automated Vehicles*. 01 2023.

[17] Bangalore Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad Sallab, Senthil Yogamani, and Patrick Perez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, PP:1–18, 02 2021.

[18] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. pages 2722–2730, 12 2015.

[19] Jingyuan Zhao, Wenyi Zhao, Bo Deng, Zhenghong Wang, Feng Zhang, Wenxiang Zheng, Wanke Cao, Jinrui Nan, Yubo Lian, and Andrew Burke. Autonomous driving system: A comprehensive survey. *Expert Systems with Applications*, 242:122836, 12 2023.

[20] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. pages 3354–3361, 05 2012.

[21] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020.

[22] Inc. Tesla. Autopilot. `https://www.tesla.com/support/autopilot`. Accessed: 2024-06-18.

[23] Waymo LLC. Waymo. `https://waymo.com/`. Accessed: 2024-06-18.

[24] Inc. Zoox. Autonomy. `https://zoox.com/autonomy`. Accessed: 2024-06-18.

[25] Mercedes-Benz Group. Autonomous driving. `https://group.mercedes-benz.com/innovation/product-innovation/autonomous-driving/`. Accessed: 2024-06-18.

[26] Romain Boch. Tesla vs waymo: Two opposite visions. `https://www.thinkautonomous.ai/blog/tesla-vs-waymo-two-opposite-visions/`, 2021. Accessed: 2024-06-18.

[27] Felipe Alonso and Miguel Clavijo. Lidar point clouds analysis computer tools for teaching autonomous vehicles perception algorithms. *Computer Applications in Engineering Education*, 32, 02 2024.

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012.

[29] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. pages 779–788, 06 2016.

[30] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. pages 1–10, 01 2016.

[31] Greg Welch, Gary Bishop, et al. An introduction to the kalman filter. 1995.

[32] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[33] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020.

[34] Lucas Tabelini, Rodrigo Berriel, Thiago M Paixao, Claudine Badue, Alberto F De Souza, and Thiago Oliveira-Santos. Keep your eyes on the lane: Real-time attention-guided lane detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 294–302, 2021.

[35] Yongjun Zhang, Pengcheng Shi, and Jiayuan Li. 3d lidar slam : A survey. *The Photogrammetric Record*, 39:457–517, 05 2024.

[36] Yusuf Abdullahi Badamasi. The working principle of an arduino. In *2014 11th international conference on electronics, computer and computation (ICECCO)*, pages 1–4. IEEE, 2014.

[37] Eben Upton and Gareth Halfacree. *Raspberry Pi user guide*. John Wiley & Sons, 2016.

[38] Agus Kurniawan and Agus Kurniawan. Introduction to nvidia jetson nano. *IoT Projects with NVIDIA Jetson Nano: AI-Enabled Internet of Things Projects for Beginners*, pages 1–6, 2021.

[39] David Kirk et al. Nvidia cuda software and gpu parallel computing architecture. In *ISMM*, volume 7, pages 103–104, 2007.

[40] NVIDIA. *TensorRT Developer Guide*, 2023. `https://docs.nvidia.com/deeplearning/tensorrt/pdf/TensorRT-Developer-Guide.pdf`, Accessed: 2023-06-22.

[41] NVIDIA. Deepstream: Video analytics for smart cities. `https://developer.nvidia.com/blog/deepstream-video-analytics-smart-cities/`, 2023. Accessed: 2023-06-22.

[42] Richard W Wall, Jerry Bennett, and Greg Eis. Creating a low-cost autonomous vehicle. In *IEEE 2002 28th Annual Conference of the Industrial Electronics Society. IECON 02*, volume 4, pages 3112–3116. IEEE, 2002.

[43] Li Dang, Nishanth Sriramoju, Girma Tewolde, Jaerock Kwon, and Xiaoyuan Zhang. Designing a cost-effective autonomous vehicle control system kit (avcs kit). In *2017 IEEE AFRICON*, pages 1453–1458. IEEE, 2017.

[44] Guodong Rong, Byung Hyun Shin, Hadi Tabatabaee, Qiang Lu, Steve Lemke, Mārtiņš Možeiko, Eric Boise, Geehoon Uhm, Mark Gerow, Shalin Mehta, et al. Lgsvl simulator: A high fidelity simulator for autonomous driving. *arXiv preprint arXiv:2005.03778*, 2020.

[45] Juan Terven and Diana Cordova-Esparza. A comprehensive review of yolo: From yolov1 to yolov8 and beyond. *arXiv preprint arXiv:2304.00501*, 2023.

[46] Shrey Srivastava, Amit Vishvas Divekar, Chandu Anilkumar, Ishika Naik, Ved Kulkarni, and V Pattabiraman. Comparative analysis of deep learning image detection algorithms. *Journal of Big data*, 8(1):66, 2021.

[47] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016.

[48] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[49] Pedro Azevedo and Vítor Santos. Comparative analysis of multiple yolo-based target detectors and trackers for adas in edge devices. *Robotics and Autonomous Systems*, 171:104558, 2024.

[50] Usha Mittal, Priyanka Chawla, and Rajeev Tiwari. Ensemblenet: A hybrid approach for vehicle detection and estimation of traffic density based on faster r-cnn and yolo models. *Neural Computing and Applications*, 35(6):4755–4774, 2023.

[51] Peixuan Li, Huaici Zhao, Pengfei Liu, and Feidao Cao. Rtm3d: Real-time monocular 3d detection from object keypoints for autonomous driving. In *European Conference on Computer Vision*, pages 644–660. Springer, 2020.

[52] Martin Simony, Stefan Milzy, Karl Amendey, and Horst-Michael Gross. Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds. In *Proceedings of the European conference on computer vision (ECCV) workshops*, pages 0–0, 2018.

[53] Kemiao Huang and Qi Hao. Joint multi-object detection and tracking with camera-lidar fusion for autonomous driving. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6983–6989. IEEE, 2021.

[54] Xiyang Wang, Chunyun Fu, Zhankun Li, Ying Lai, and Jiawei He. Deepfusionmot: A 3d multi-object tracking framework based on camera-lidar fusion with deep association. *IEEE Robotics and Automation Letters*, 7(3):8260–8267, 2022.

[55] Jiaman Li, Karen Liu, and Jiajun Wu. Ego-body pose estimation via ego-head pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17142–17151, 2023.

[56] Temitope Ibrahim Amosa, Patrick Sebastian, Lila Iznita Izhar, Oladimeji Ibrahim, Lukman Shehu Ayinla, Abdulrahman Abdullah Bahashwan, Abubakar Bala, and Yau Alhaji Samaila. Multi-camera multi-object tracking: a review of current trends and future advances. *Neurocomputing*, 552:126558, 2023.

[57] Bima Sahbani and Widyawardana Adiprawita. Kalman filter and iterative-hungarian algorithm implementation for low complexity point tracking as part of fast multiple object tracking system. In *2016 6th international conference on system engineering and technology (ICSET)*, pages 109–115. IEEE, 2016.

[58] Andréa Macario Barros, Maugan Michel, Yoann Moline, Gwenolé Corre, and Frédérick Carrel. A comprehensive survey of visual slam algorithms. *Robotics*, 11(1):24, 2022.

[59] Hesai Technology. What you need to know about lidar: The strengths and limitations of camera, radar, and lidar. `https://www.hesaitech.com/what-you-need-to-know-about-lidar-the-strengths-and-limitations-of-camera-radar-and-li`, 2023. Accessed: 2024-06-01.

[60] Hongyu Zhou, Zheng Ge, Zeming Li, and Xiangyu Zhang. Matrixvt: Efficient multi-camera to bev transformation for 3d perception. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8548–8557, 2023.

[61] Raquel R Pinho and João Manuel RS Tavares. Tracking features in image sequences with kalman filtering, global optimization, mahalanobis distance and a management model. 2009.

[62] Tim Bailey, Ben Upcroft, and Hugh Durrant-Whyte. Validation gating for non-linear non-gaussian target tracking. In *2006 9th International Conference on Information Fusion*, pages 1–6. IEEE, 2006.

[63] Microsoft. Visual studio code. `https://code.visualstudio.com/`, 2015. Accessed: 2024-07-01.

[64] JetBrains. Pycharm. `https://www.jetbrains.com/pycharm/`, 2010. Accessed: 2024-07-01.

[65] Project Jupyter. Jupyter notebook. `https://jupyter.org/`, 2015. Accessed: 2024-07-01.

[66] Python Software Foundation. Python programming language. `https://www.python.org/`, 1991. Accessed: 2024-07-01.

[67] NumPy Contributors. Numpy. `https://numpy.org/`, 2006. Accessed: 2024-07-01.

[68] Google. Tensorflow. `https://www.tensorflow.org/`, 2015. Accessed: 2024-07-01.

[69] PyTorch Contributors. Pytorch. `https://pytorch.org/`, 2016. Accessed: 2024-07-01.

[70] Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. Hota: A higher order metric for evaluating multi-object tracking. *International journal of computer vision*, 129:548–578, 2021.

[71] Andrii Maksai and Pascal Fua. Eliminating exposure bias and metric mismatch in multiple object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4639–4648, 2019.

[72] Georg Volk, Jörg Gamerdinger, Alexander von Bernuth, and Oliver Bringmann. A comprehensive safety metric to evaluate perception in autonomous systems. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8. IEEE, 2020.

[73] Chensheng Peng, Zhaoyu Zeng, Jinling Gao, Jundong Zhou, Masayoshi Tomizuka, Xinbing Wang, Chenghu Zhou, and Nanyang Ye. Pnas-mot: Multi-modal object tracking with pareto neural architecture search. *IEEE Robotics and Automation Letters*, 2024.

[74] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[75] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88:303–338, 2010.

[76] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

Abstract

The development of autonomous vehicles is heavily reliant on the advancements in artificial intelligence (AI) and sensor fusion technologies. This project, a collaboration between the Cerist Research Center and the University of Bejaia, addresses the challenge of integrating data from multiple sensors to improve object detection and tracking accuracy. By leveraging state-of-the-art algorithms such as YOLOv8 for camera images and Complex YOLO for LiDAR point clouds, combined with the DeepSORT tracker and Kalman Filters, we propose a robust sensor fusion algorithm. Our solution has been evaluated using the KITTI dataset, showing significant improvements in tracking performance. The results demonstrate the potential of our approach to enhance the safety and reliability of autonomous vehicles.

Keywords: *Autonomous vehicles, artificial intelligence, sensor fusion, object detection, tracking, YOLOv8, Complex YOLO, DeepSORT, Kalman Filters, KITTI dataset.*

Résumé

Le développement des véhicules autonomes repose fortement sur les avancées de l'intelligence artificielle (IA) et des technologies de fusion de capteurs. Ce projet, une collaboration entre le Centre de Recherche Cerist et l'Université de Bejaia, traite du défi de l'intégration des données provenant de plusieurs capteurs pour améliorer la précision de la détection et du suivi des objets. En exploitant des algorithmes de pointe tels que YOLOv8 pour les images de caméra et Complex YOLO pour les nuages de points LiDAR, combinés avec le tracker DeepSORT et les filtres de Kalman, nous proposons un algorithme de fusion de capteurs robuste. Notre solution a été évaluée à l'aide du jeu de données KITTI, montrant des améliorations significatives des performances de suivi. Les résultats démontrent le potentiel de notre approche pour améliorer la sécurité et la fiabilité des véhicules autonomes.

Mots-clés: *Véhicules autonomes, intelligence artificielle, fusion de capteurs, détection d'objets, suivi, YOLOv8, Complex YOLO, DeepSORT, filtres de Kalman, jeu de données KITTI.*