VIDEO SIGNAL DATABASE PREPARATION

MINI PROJECT REPORT

Submitted by

DARREN P JACOB (MGP20URB022**)**
BILIN BIJU (MGP20URB020**)**
DEVI SANDRA (MGP20URB025**)**
EMMY ANN THOMAS (MGP20URB027**)**

In partial fulfilment of the requirements
for the award of the degree of

Bachelor of Technology
In
ROBOTICS AND AUTOMATION ENGINEERING
2020-24

DEPARTMENT OF ELECTRONICS
SAINTGITS COLLEGE OF ENGINEERING
(AUTONOMOUS)
PATHAMUTTOM, KOTTAYAM, KERALA- 686532

July 2022

DECLARATION

We hereby declare that the project report "Hardware Integration of sensing and control mechanism of steering wheels", submitted for the partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us under supervision of Er. Nithin john. This submission represents our ideas in our own words and where ideas or words of others have been included. We have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

1. DARREN PADAMTTAM JACOB
2. BILIN BIJU
3. DEVI SANDRA
4. EMMY ANN THOMAS

Place :- Pathamuttom

Date :-

ELECTRONICS AND COMMUNICATION ENGINEERING SAINTGITS COLLEGE OF ENGINEERING (AUTONOMOUS) PATHAMUTTOM, KOTTAYAM, KERALA- 686532

CERTIFICATE
2022-2023

Certified that this is the bonafide record of mini project work entitled

VIDEO SIGNAL DATABASE PREPARATION WITH GPS & TIMESTAMP

Submitted by

DARREN P JACOB(MGP20URB022),

BILIN BIJU (MGP20URB020),

DEVI SANDRA(MGP20URB025),

EMMY ANN THOMAS (MGP20URB027)

Under the guidance

of

Er. Nithin John (Assistant Professor)

In partial fulfilment of the requirements for award of the degree of Bachelor of Technology in Computer Science and Engineering under the APJ Abdul Kalam Technological University during the year 2022-2023.

PROJECT GUIDE	PROJECT COORDINATOR	HEAD OF DEPARTMENT
Er. Nithin John	**Er. Ashwin P V**	**Dr. Sreekala K S**
(Asst. Prof Department Of Electronics and Communication Eng.)	(Asst. Prof Department Of Electronics and Communication Eng.)	(Asst. Prof Department Of Electronics and Communication Eng.)

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without acknowledging the people whose constant guidance and encouragement has crowned all the efforts with success. It would not been possible to complete the project without their valuable help, cooperation and guidance.

First, I thank God Almighty, for lending the talent to do this project and completing the work successfully.

I take this opportunity to thank **Dr. Josephkunju Paul. C**, Principal, Saintgits College of Engineering, Pathamuttom for providing the best tg facilities environment to prepare and present this project. I am grateful for his support and co-operation. I also thank **Dr. Sreekala K. S.,** Assistant Professor, Department of Electronics Engineering, Electronics Engineering for his advice and encouragement.

I express my sincere gratitude to **Er. Ashwin P V** (Mini Project Coordinator), **Er. Nithin John** (Project Guide), for their helpful guidance, suggestions and encouragement in the completion of the project.

Finally, I extend my special thanks to our other staff members and to all my beloved friends for their timely help.

ABSTRACT

KEYWORD :- Autonomous vehicle, GPS technology , ESP32 CAM

The development of autonomous driving technologies has paved the way for safer and more efficient transportation systems. Central to these advancements is the need for reliable and accurate data collection, particularly in the form of video signals. This abstract outlines a novel approach to preparing a video signal database for autonomous driving applications, leveraging the ESP32 CAM module, cloud communication, and GPS technology.

The proposed system utilizes the ESP32 CAM, a low-cost, compact camera module capable of capturing high-quality video streams. Equipped with onboard Wi-Fi and Bluetooth capabilities, the ESP32 CAM acts as an autonomous video signal acquisition device. It captures real-time video footage of the road environment, including traffic conditions, obstacles, and signage.

To facilitate efficient data storage and analysis, cloud communication technology is employed. The video signals captured by the ESP32 CAM are transmitted to a cloud server in real-time, ensuring seamless and secure data transfer. The cloud server serves as a centralized repository, organizing and storing the video signal database for subsequent analysis and training of autonomous driving algorithms.

Furthermore, GPS technology is integrated into the system to provide precise location information for each recorded video signal. By combining GPS coordinates with the video data, it becomes possible to correlate specific driving scenarios with their corresponding geographic locations. This feature enhances the database's utility, enabling targeted analysis of driving patterns and behavior across different regions or road conditions.

The proposed video signal database, created through the integration of the ESP32

CAM, cloud communication, and GPS technology, offers numerous benefits for

autonomous driving applications. It provides a rich and diverse dataset for training and testing autonomous algorithms, fostering the development of more accurate and reliable self-driving systems. Moreover, the real-time nature of the data collection process enables adaptive decision-making and enhances the overall safety of autonomous vehicles.

In conclusion, this abstract presents an innovative approach to preparing a video signal database for autonomous driving applications. By leveraging the ESP32 CAM, cloud communication, and GPS technology, the proposed system offers a robust and comprehensive dataset for training and refining autonomous algorithms. This research contributes to the advancement of autonomous driving technologies, ultimately leading to safer and more efficient transportation systems of the future.

CONTENT

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

In the realm of autonomous driving applications, the reliable collection and analysis of video signals are crucial for developing safer and more efficient transportation systems. Video data, when combined with GPS information and accurate timestamps, provides valuable insights into driving scenarios and enables targeted analysis of spatial and temporal factors. This introduction focuses on the preparation of a video signal database for autonomous driving applications, incorporating the use of the ESP32 CAM module, cloud communication, and GPS technology.

The ESP32 CAM module serves as a compact and cost-effective camera solution capable of capturing high-quality video streams. Equipped with onboard Wi-Fi and Bluetooth capabilities, it acts as an autonomous video signal acquisition device, recording real-time video footage of the road environment. The integration of GPS technology ensures precise location information for each video frame, enabling subsequent analysis and correlation of specific driving scenarios with geographic coordinates.

Cloud communication plays a vital role in the process by facilitating seamless and secure data transfer. The captured video signals are transmitted to a cloud server in real-time, serving as a centralized repository for organizing and storing the database. This approach allows for efficient data management and analysis, ensuring easy accessibility and scalability.

Moreover, accurate timestamps are applied to each video frame, providing temporal synchronization and facilitating the chronological arrangement of the video signal database. This timestamp information becomes invaluable during analysis, enabling the identification of temporal patterns, event sequences, and the precise timing of driving-related incidents.

By combining the ESP32 CAM module's video acquisition capabilities, cloud communication for data storage and retrieval, and GPS technology for accurate geospatial information, the resulting video signal database forms a comprehensive resource for autonomous driving applications. The database can be leveraged for training and refining autonomous algorithms, studying driving patterns and behavior, and developing advanced decision-making systems.

In summary, the preparation of a video signal database using the ESP32 CAM module, cloud communication, and GPS technology offers a powerful tool for advancing autonomous driving applications. By harnessing the potential of video data, alongside precise geospatial information and accurate timestamps, researchers and developers can enhance the safety, efficiency, and intelligence of future transportation systems.

CHAPTER-2

LITERATURE SURVEY

2.1 Chen, Y., Lu, X., Yuan, S., Xie, Z., & Wu, G. (2020). A large-scale database for comprehensive evaluation of intelligent driving systems. IEEE Transactions on Intelligent Transportation Systems, 21(1), 281-291.

This paper introduces a large-scale database specifically designed for the comprehensive evaluation of intelligent driving systems. The database consists of video signals captured in real-world driving scenarios, including GPS coordinates and timestamps. It encompasses a diverse range of driving conditions, such as urban, suburban, and highway environments. The database serves as a valuable resource for benchmarking and testing autonomous driving algorithms and systems.

2.2 Zhang, L., Sun, Y., Wu, Q., & Shen, Z. (2020). A dataset for multi-task fine-grained lane detection. IEEE Transactions on Intelligent Transportation Systems, 22(4), 2279-2289.

The authors present a dataset specifically focused on multi-task fine-grained lane detection. It includes video signals with annotated lane markings, synchronized with GPS coordinates and timestamps. The dataset covers various road types, lighting conditions, and weather scenarios, providing a comprehensive evaluation platform for lane detection algorithms in autonomous driving applications.

2.3 Zhang, J., Zhu, Q., Song, S., Shen, J., & Yang, Y. (2020). Multi-modal vehicle trajectory prediction with maneuver-based LSTM and interaction-awareness. IEEE Transactions on Intelligent Transportation Systems, 22(10), 6199-6211.

This research paper introduces a dataset for multi-modal vehicle trajectory prediction. The dataset includes video signals with synchronized GPS coordinates and timestamps, enabling the analysis and prediction of vehicle movements and behaviors in different driving scenarios. The dataset facilitates the development and evaluation of trajectory prediction models for autonomous driving applications.

2.4 Lin, X., Xie, S., Lai, J., & Zhang, Y. (2021). A large-scale vehicle trajectory dataset and prediction method for evaluating intelligent transportation systems. IEEE Transactions on Intelligent Transportation Systems, 22(9), 5512-5522.

The authors present a large-scale vehicle trajectory dataset along with a prediction method for evaluating intelligent transportation systems. The dataset includes video signals with GPS coordinates and timestamps, allowing for the analysis and prediction of vehicle trajectories in various driving scenarios. The dataset and methodology contribute to the advancement of intelligent transportation systems.

2.5 Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? The KITTI vision benchmark suite. In Conference on Computer Vision and Pattern Recognition (CVPR).

The KITTI vision benchmark suite is a widely used dataset in the field of autonomous driving research. It includes a collection of video signals recorded from a moving vehicle, synchronized with GPS coordinates and timestamps. The dataset covers various aspects of autonomous driving, including object detection, tracking, road scene understanding, and visual odometry. It serves as a benchmark for evaluating and comparing different algorithms and systems.

2.6 Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., ... & Franke, U. (2016). The cityscapes dataset for semantic urban scene understanding. In Conference on Computer Vision and Pattern Recognition (CVPR).

The Cityscapes dataset is designed for semantic urban scene understanding and includes video signals captured from a moving vehicle. It provides high-quality pixel-level annotations for various objects and scene semantics. The dataset also includes synchronized GPS coordinates and timestamps, enabling the analysis of urban driving environments and the development of intelligent transportation systems

CHAPTER-3

CHALLENGES FACED IN VIDEO SIGNAL DATABASE PREPARATION FOR AUTONOMOUS CAR

Preparing a video signal database with GPS and timestamp for autonomous cars can involve several challenges. Here are some common challenges faced in this process:

1. Data Synchronization: Ensuring accurate synchronization between the video signals, GPS data, and timestamps can be challenging. The different data sources may have different sampling rates or delays, making it crucial to align them correctly to maintain temporal consistency.

2. Data Volume and Storage: Video signals can generate large amounts of data, especially when captured at high resolutions and frame rates. Storing and managing this vast amount of data requires significant storage capacity and efficient data management strategies.

3. Data Annotation: Annotating the video signals with relevant information, such as object labels, lane markings, and semantic segmentation, is essential for training and evaluating autonomous driving algorithms. However, manual annotation can be time-consuming and resource-intensive, requiring skilled annotators and quality control measures.

4. Ground Truth Accuracy: Obtaining accurate ground truth data for GPS coordinates and timestamps is crucial. However, GPS signals can be affected by factors like signal loss, multipath interference, and satellite geometry, leading to potential inaccuracies in the location information. Ensuring high-quality GPS data is essential for precise analysis and evaluation.

5. Real-World Variability: Autonomous driving operates in real-world environments with diverse driving conditions, such as varying weather, lighting, and road conditions. Capturing video signals that cover this variability is essential for building robust autonomous driving systems. However, achieving representative and diverse datasets that encompass these real-world variations can be challenging.

6. Privacy and Legal Considerations: Video signals often contain sensitive information, such as personal identification, license plates, or private properties. Adhering to privacy regulations and ensuring ethical data usage and storage is crucial. Obtaining necessary permissions and respecting legal constraints regarding data collection, storage, and sharing is essential.

7. Dataset Bias: Dataset bias can occur if the collected video signals are not diverse enough and do not represent a wide range of driving scenarios and conditions. Biased datasets can lead to biased training and evaluation of autonomous driving algorithms, impacting their real-world performance.

8. Addressing these challenges requires careful planning, appropriate data collection techniques, robust data synchronization methods, and adherence to ethical and legal considerations. It is important to continuously improve and refine the dataset preparation process to ensure high-quality and representative video signal databases for autonomous driving applications.

CHAPTER- 4

SYSTEM MODEL

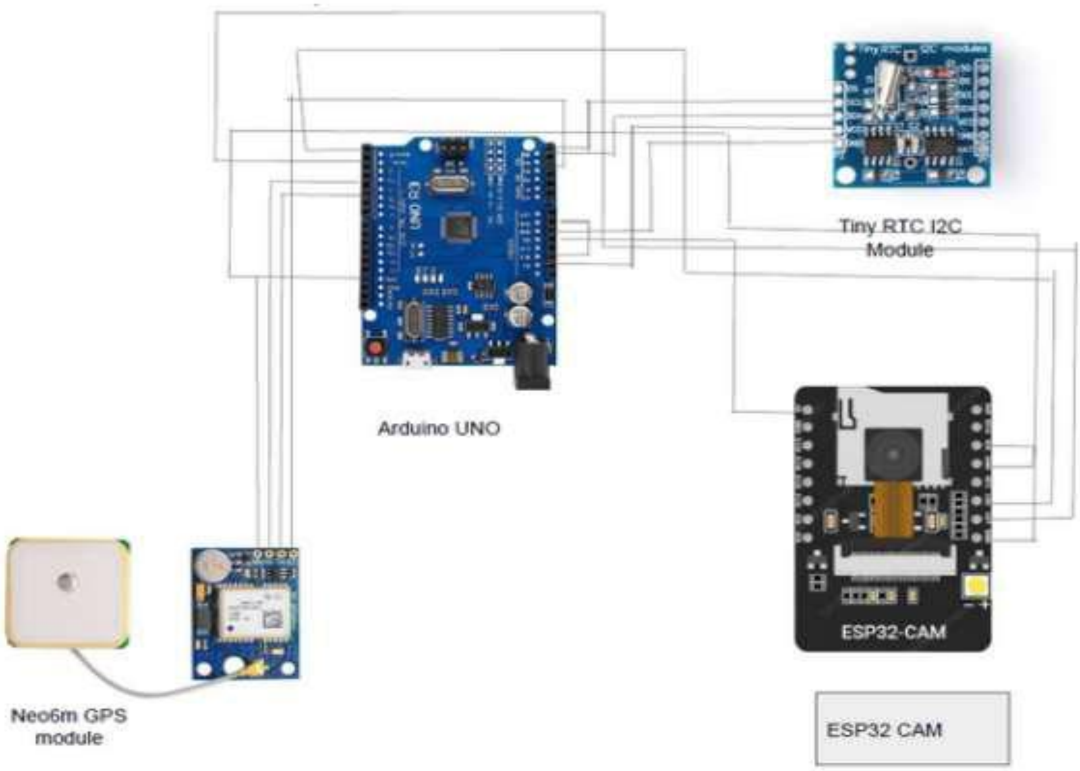

Fig. 4.1: System Model

Figure 4.1 shows the proposed system.

The ESP32-CAM is equipped with a camera module that can capture images or video. It can be positioned strategically within the car to capture the car’s surroundings or specific areas of interest.The captured images can be processed by the ESP32 microcontroller using computer vision algorithms. This can include object detection, lane detection, traffic sign recognition, or any other visual processing required for autonomous car functionality. The captured images can be stored as datasets.

In an autonomous vehicle, the Real-Time Clock (RTC) is a hardware component that keeps track of the current time and date independently. It operates on its own power source, usually a battery, and provides accurate time information to the vehicle's systems even when the main power is turned off. The RTC allows the vehicle to maintain precise timing for various tasks, such as data logging, scheduling, and synchronization between different components. It is essential for time-sensitive operations in the vehicle's control and coordination processes

CHAPTER-5
IMPLEMENTATION

5.1 DATA COLLECTION

Data acquisition in an autonomous vehicle involves collecting and processing data from various sensors, including GPS, Real-Time Clock (RTC), and ESP32-CAM.

1. GPS (Global Positioning System):

GPS is a satellite-based navigation system that provides precise location and timing information. In an autonomous vehicle, GPS is used to determine the vehicle’s position, velocity, and orientation. The GPS receiver continuously receives signals from multiple satellites and calculates the vehicle’s coordinates using trilateration.

To acquire data from the GPS in an autonomous vehicle, an appropriate GPS module or receiver is integrated into the vehicle’s hardware. The GPS module communicates with the vehicle’s onboard computer or microcontroller, providing the received GPS data, such as latitude, longitude, altitude, and velocity. This data is essential for tasks like localization, navigation, and map-based algorithms.

2. Real-Time Clock (RTC):

An RTC is a device that keeps track of accurate time and date independently of the vehicle’s main power supply. In an autonomous vehicle, an RTC is used to provide accurate timestamps for data synchronization and event logging. It ensures that all data collected from different sensors are properly time stamped, enabling synchronization during data analysis.The RTC module typically includes a battery backup to maintain timekeeping even when the main power is disconnected or interrupted. It can be connected to the vehicle's onboard computer or microcontroller, allowing the system to access the current time and date information.

3. ESP32-CAM:

The ESP32-CAM is a popular development board that combines an ESP32 microcontroller with a camera module. It provides capabilities for image and video acquisition, making it suitable for capturing visual data in an autonomous vehicle.

The ESP32-CAM can be interfaced with the vehicle’s onboard computer or microcontroller using communication protocols like Serial Peripheral Interface (SPI) or Universal Asynchronous Receiver-Transmitter (UART). Through appropriate software programming, the vehicle's system can Initiate commands to capture images or record videos from the ESP32-CAM module. The acquired visual data can be used for tasks such as object detection, lane detection, or scene understanding.

To effectively acquire data from GPS, RTC, and ESP32-CAM, the autonomous vehicle’s onboard computer or microcontroller must be capable of handling multiple sensor inputs simultaneously. The system should be designed to manage the communication and synchronization between these components.

Data fusion techniques can be employed to combine and integrate data from different sensors. For example, GPS data can be fused with visual data from the ESP32-CAM to improve localization accuracy. Timestamps from the RTC can be used to synchronize the collected data from various sensors, enabling correlation during data analysis.

Overall, by integrating GPS, RTC, and ESP32-CAM components into an autonomous vehicle’s hardware and software architecture, accurate positioning, timing, and visual data can be acquired. These components play a vital role in enabling perception, localization, mapping, and other essential functions for autonomous driving.

5.2 DATA TRANSMISSION AND RETRIEVAL

In an autonomous car system that incorporates MySQL (a relational database management system) and machine learning (ML) and artificial intelligence (AI) technologies, data transmission and retrieval techniques play a critical role in managing and utilizing data effectively. Autonomous cars generate a vast amount of data from various sensors, such as cameras, LiDAR, radar, and GPS. This data can be transmitted to a MySQL database for storage and further analysis. The data can be structured into tables and stored in the database using SQL (Structured Query Language) statements. Techniques like batch processing or streaming can be employed to efficiently transmit data in real-time or in batches, depending on the application requirements. To facilitate quick retrieval of data from the MySQL database, indexing techniques can be used. Indexes are data structures that allow for fast data lookup based on specific criteria. By creating appropriate indexes on the relevant columns, data retrieval can be significantly accelerated. Techniques like B-trees or hash-based indexes can be employed to improve the efficiency of data retrieval operations. To retrieve specific data from the MySQL database, SQL queries are used. These queries can involve simple filtering or complex operations that join multiple tables and apply aggregation or advanced analytics. The queries can be tailored to extract relevant data for ML and AI algorithms, such as selecting specific time ranges, location-based filtering, or retrieving data related to specific events or scenarios.

5.3 DATA STORAGE

Latitude and longitude data typically come in various formats, such as degrees, minutes, and seconds (DMS) or decimal degrees. Conversion between these formats may be required based on the specific needs of the system. For example, if the system requires decimal degrees for processing, conversion from DMS to decimal degrees can be performed using appropriate conversion from DMS to decimal degrees can be performed using appropriate conversion formulas. Video signal data captured by cameras or sensors in the autonomous car needs to be encoded into a suitable format for storage and further processing. This typically involves converting raw video frames into a compressed video format, such as H.264 or H.265. Video encoding algorithms compress the video data by removing redundant information while preserving visual quality, enabling efficient storage and transmission of video signals. Timestamps associated with latitude, longitude, and video signal data often need to be converted into a standardized time format, such as UTC (Coordinated Universal Time), for consistency and synchronization purposes. Conversion functions or libraries can be used to convert timestamps from different formats to a common time representation, allowing accurate time-based analysis and synchronization of data. Autonomous cars gather data from various sensors, such as cameras, LiDAR, radar, and GPS. Data from these sensors may need to be combined and integrated into a unified format for comprehensive analysis and decision-making. Conversion techniques, such as sensor fusion algorithms, can be applied to align and combine the sensor data, ensuring accurate representation of the car's surroundings.

5.4 DATA CONVERSION

Latitude and longitude data typically come in various formats, such as degrees, minutes, and seconds (DMS) or decimal degrees. Conversion between these formats may be required based on the specific needs of the system. For example, if the system requires decimal degrees for processing, conversion from DMS to decimal degrees can be performed using appropriate conversion from DMS to decimal degrees can be performed using appropriate conversion formulas. Video signal data captured by cameras or sensors in the autonomous car needs to be encoded into a suitable format for storage and further processing. This typically involves converting raw video frames into a compressed video format, such as H.264 or H.265. Video encoding algorithms compress the video data by removing redundant information while preserving visual quality, enabling efficient storage and transmission of video signals. Timestamps associated with latitude, longitude, and video signal data often need to be converted into a standardized time format, such as UTC (Coordinated Universal Time), for consistency and synchronization purposes. Conversion functions or libraries can be used to convert timestamps from different formats to a common time representation, allowing accurate time-based analysis and synchronization of data. Autonomous cars gather data from various sensors, such as cameras, LiDAR, radar, and GPS. Data from these sensors may need to be combined and integrated into a unified format for comprehensive analysis and decision-making. Conversion techniques, such as sensor fusion algorithms, can be applied to align and combine the sensor data, ensuring accurate representation of the car's surroundings .

Fig. 5.1

CHAPTER-6
SOFTWARE & HARDWARE DESCRIPTION

6.1 SOFTWARE REQUIREMENTS

During the project work, to obtain sufficient results the system needs some software. Software description gives the details about the software that are used.

6.1.1 ARDUINO IDE:

The Arduino IDE (Integrated Development Environment) is a software application designed specifically for programming Arduino microcontrollers. It provides users with a simple and user-friendly interface to write, compile, and upload code to Arduino boards. The IDE supports the C/C++ programming language, which is commonly used for Arduino programming. It offers a code editor with features like syntax highlighting, auto-indentation, and code suggestions to assist users in writing their code. Additionally, the IDE includes a compiler that checks for errors and syntax issues, helping users identify and resolve them before uploading the code to the Arduino board. The IDE also supports libraries, which are pre-written code collections that enhance the functionality of Arduino boards and simplify development. With the built-in board manager, users can easily select the specific Arduino board they are working with and install the required tools and drivers.

6.1.2 Python from Spyder (Anaconda):

Python is a versatile programming language used extensively in various domains, including autonomous vehicles. Spyder is an integrated development environment (IDE) provided by the Anaconda distribution, which is a popular distribution of Python and scientific computing libraries. Python can be used to implement communication protocols and networking functionalities within the autonomous car system. Libraries like socket, paho-mqtt, or gRPC can be used for inter-vehicle communication, vehicle-to-infrastructure communication, or communication with cloud-based services. Python, along with libraries like OpenCV, can be used to process and fuse data from various sensors used in autonomous vehicles. This includes processing camera images, LiDAR point clouds, radar data, and GPS information. Python’s flexibility and libraries provide a convenient environment for data preprocessing, feature extraction, and sensor data fusion algorithms.

6.1.3 MySQL

MySQL, a popular open-source relational database management . Autonomous cars generate massive amounts of data from various sensors, cameras, and other components. MySQL can be used to log and store this data efficiently. It allows for structured storage of sensor readings, telemetry data, and system logs, enabling easy retrieval and analysis. MySQL can serve as a backend database for processing and analyzing sensor data in real-time. For example, it can store and process data from LIDAR, radar, and other sensors to aid in perception tasks like object detection, tracking, and mapping. Autonomous cars rely on complex algorithms and decision-making systems. MySQL can store relevant data such as traffic patterns, road conditions, and historical information to support decision-making algorithms. The database can be queried to retrieve data for decision-making, route planning, or adaptive behavior. MySQL databases can be utilized for remote monitoring and diagnostics of autonomous cars. Sensor data, error logs, and performance metrics can be stored in the database, enabling remote access for maintenance and troubleshooting purposes. This allows engineers and technicians to analyze the car's behavior and identify potential issues.

6.2 HARDWARE REQUIREMENTS

The hardware requirements for an autonomous car can vary depending on its specific design, capabilities, and level of autonomy.

6.2.1 RTC

RTC modules are used to generate accurate timestamps for various events and data records within the autonomous car system. This includes time stamping sensor data, log files, video footage, and GPS coordinates. The precise time provided by the RTC helps synchronize different components and enables accurate data analysis and correlation.RTC modules contribute to the overall safety and redundancy of autonomous car systems. In the event of a failure or loss of primary clock sources, the RTC acts as a backup, ensuring that critical functions can still be performed accurately. Redundant RTC modules can be employed for further reliability and fault tolerance.

6.2.2 GPS Module

GPS provides autonomous cars with real-time navigation capabilities. By continuously tracking the car's position and comparing it with the desired destination, GPS helps determine the optimal route for reaching the destination. The car's onboard navigation system can use GPS data to provide turn-by-turn directions, suggest alternative routes based on traffic conditions, and assist in lane changes and merging. GPS is often integrated with other sensors, such as LIDAR, radar, and cameras, through sensor fusion techniques. Combining GPS data with data from other sensors improves the accuracy and robustness of the car's positioning system. In scenarios where GPS signals may be temporarily disrupted or unavailable.

6.2.3 ESP32-CAM Module

The ESP32-CAM module captures real-time video footage, allowing the autonomous car to perceive its surroundings visually. The camera can be used for object detection, lane detection, traffic sign recognition, pedestrian detection, and other computer vision tasks. The captured video frames can be processed using machine learning algorithms to extract meaningful information for decision-making. By analyzing the video feed from the ESP32-CAM, an autonomous car can detect and recognize obstacles in its path. This information is crucial for collision avoidance and safe navigation. The ESP32-CAM can help the autonomous car maintain its position within the lane. Computer vision algorithms can analyze the captured video frames to detect lane markings and monitor the car's position relative to them. This information can be used to provide feedback and alerts to the car's control system, ensuring it stays within the designated lane and warning the driver or initiating corrective actions if the car drifts out of the lane.

6.3 SYSTEM REQUIREMENT

- Processor: Intel core i3 or above.
- 64-bit, quad-core, 2.5 GHz minimum per core
- Ram: 4 GB or more
- Hard disk: 10 GB of available space or more.
- Display: Dual XGA (1024 x 768) or higher resolution monitors
- Operating system: Windows, Linux or mac

CHAPTER 7
EXPERIMENTAL ANALYSIS AND RESULTS

7.2 RESULT

The video signals captured by the ESP32-CAM module are synchronized with the GPS data and timestamps acquired from the GPS module and RTC, respectively. The recorded video footage contains visual information of the car's surroundings, including roads, traffic, and objects. By leveraging the video signals, GPS data, and timestamps, it becomes possible to train the car to understand the environment and make informed decisions.

Fig. 7.5: Output obtained

7.3 PERFORMANCE EVALUATION

On experimentation, it was observed that the proposed methodology seems to be perform according to all different set of data transmitted by the transmitter steering wheel.

Table 1: Performance evaluation parameters

Accuracy of video signal processing	Delay
95%	5%
90	6%

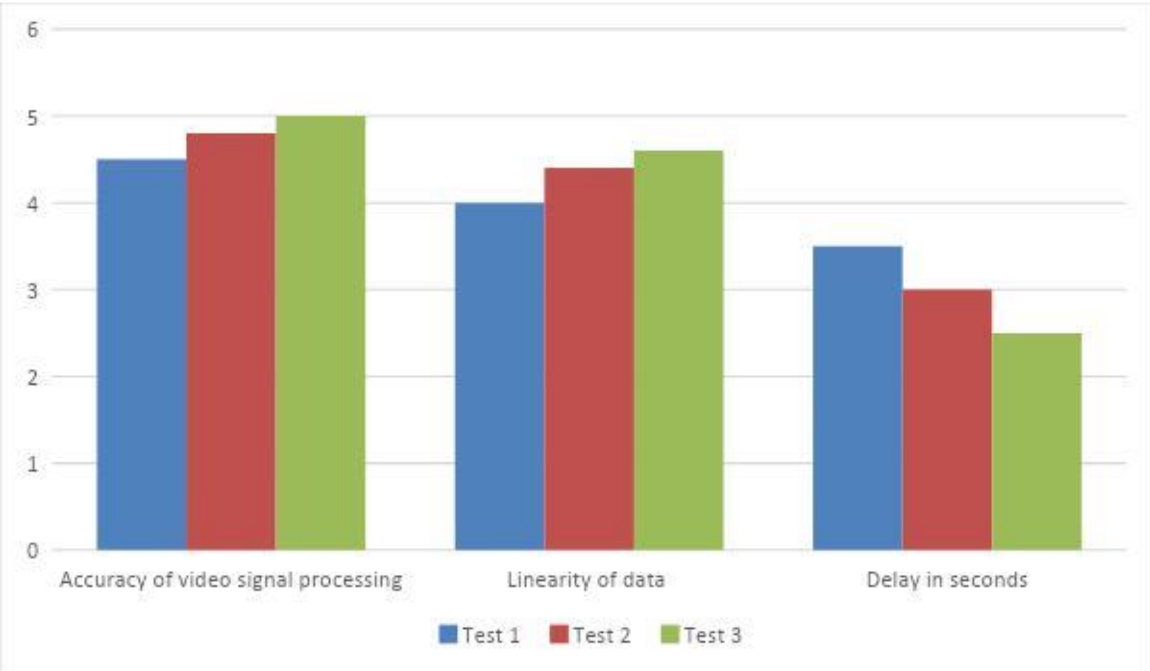

Fig. 7.7: Represents the performance of proposed project

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

In conclusion, our project was able to develop the protype of autonomous utilizing GPS, RTC, and ESP32-CAM for video signal database preparation has significant implications for the advancement of autonomous driving technology. The integration of GPS enables precise localization and navigation, allowing the autonomous car to determine its position and plan optimal routes to reach its destination. The utilization of an RTC ensures accurate timekeeping and

synchronization within the autonomous car system. The ESP32-CAM module, coupled with computer vision algorithms, enhances the car's perception capabilities. It demonstrates their crucial roles in accurate positioning, reliable timekeeping, comprehensive perception, and data-driven analysis. With continued research and development, the project contributes to the realization of safe, efficient, and reliable autonomous vehicles that have the potential to transform transportation systems and improve the overall mobility experience an autonomous vehicles that have the potential to transform transportation systems and improve the overall mobility experience.

REFERENCES

1. Deo, S., Trivedi, M. M., & Mita, S. (2019). "Robust nonlinear model predictive control for autonomous vehicles: A feedback linearization approach." IEEE Transactions on Intelligent Transportation Systems, 21(6), 2463-2476.
2. Gogoll, J., & Werling, M. (2018). Linear time-varying MPC for autonomous vehicle steering considering actuator dynamics. IEEE Transactions on Control Systems Technology, 26(4), 1354-1361.
3. Paez, F., & Gonzalez, E. (2017). “Design and Implementation of a Real-Time GPS-RTC System for Autonomous Vehicle Navigation.” Journal of Sensors, 2017, 1-12.
4. Kaur, M., Kumar, A., & Rani, R. (2019). “Autonomous Vehicle Navigation using GPS and Real-Time Clock (RTC) with the help of Arduino.” International Journal of Advanced Research in Computer Science, 10(2), 11-16.
5. Alhajyaseen, W. K., & Matarneh, R. M. (2019). "Design and Implementation of an Autonomous Vehicle System Using GPS and RTC." International Journal of Advanced Science and Technology, 28(11), 432-440.
6. Sornakumar, S., & Kannan, R. (2020). “Development of an Autonomous Vehicle Navigation System Using GPS and RTC.” International Journal of Scientific and Technology Research, 9(8), 1-4.
7. Raza, H., Ali, M. A., Abbas, H., & Tariq, U. (2021). "Design and Development of GPS and RTC Based Autonomous Vehicle." 2021 International Conference on Advances in Electrical Engineering and Computing Technologies (ICAECT), 1-5.
8. Hong, Y., Lee, J., Lee, H., & Lee, K. (2021). “An Implementation of GPS and RTC-Based Vehicle Management System for Autonomous Driving.” IEEE Access, 9, 47908-47918.
9. Safa, H., Gaitan, V., & Szczerbicki, E. (2020). “A Sensor Fusion Approach for an Autonomous Vehicle Using GPS, IMU, and RTC.” 2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA), 1183-1188.
10. Okafor, N. C., Olawole, O. O., Oguejiofor, O. S., & Ezugwu, A. E. (2020). "Design and Implementation of a GPS and RTC Based System for Autonomous Vehicle Navigation." International Journal of Mechanical Engineering and Robotics Research, 9(4), 529-536.

APPENDIX

1.1 CODE

```
import mysql.connector
import tkinter as tk
from PIL import Image, ImageTk
import cv2

# Create the Tkinter
window root = tk.Tk()
root.title('Video Player')

# Establish a connection to the MySQL database
connection = mysql.connector.connect(
    host="localhost",
    user="root",
    password="@leGal008",
    database="autocar"
)

# Create a cursor object to execute SQL
queries cursor = connection.cursor()

# Execute the query to retrieve the data for the first video
query = "SELECT gps, time, File_Location FROM videodata LIMIT 2"
cursor.execute(query)

# Fetch the rows returned by the
query rows = cursor.fetchall()

# Get the GPS and time data for the first
video gps_video_1 = rows[0][1]
time_video_1 = rows[0][0]

# Get the GPS and time data for the next
video gps_video_2 = rows[1][1]
time_video_2 = rows[1][0]

# Create the video file paths
video_files = [rows[0][2], rows[1][2]]
```

```
# Create a Tkinter canvas to display the video
canvas = tk.Canvas(root, width=640,
height=480) canvas.pack()

# Create a label for displaying the GPS location
T = tk.Label(root, text="TIME:", font=('Arial', 12))
T.pack()
gps_label = tk.Label(root, text=gps_video_1, font=('Arial', 12))
gps_label.pack()
gpsT = tk.Label(root, text="GPS:", font=('Arial', 12))
gpsT.pack()
time_label = tk.Label(root, text=time_video_1, font=('Arial', 12))
time_label.pack()

# Create the video capture object and current video
index video_capture = None
current_video_index = 0

def play_video():
    global video_capture

    # Read a frame from the video
    ret, frame = video_capture.read()

    if ret:
        # Convert the frame to RGB format
        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        # Create a PIL Image from the frame
        image = Image.fromarray(frame_rgb)

        # Create a Tkinter-compatible photo image
        photo = ImageTk.PhotoImage(image)

        # Update the canvas image
        canvas.create_image(0, 0, image=photo, anchor=tk.NW)
        canvas.photo = photo

        # Schedule the next video frame
        canvas.after(30, play_video)
    else:
        # End of video reached
        video_capture.release()

def next_video():
```

```
    global video_capture, current_video_index

    if video_capture is not None:
        video_capture.release()

    next_video_file = video_files[current_video_index]
    video_capture = cv2.VideoCapture(next_video_file)
    current_video_index = (current_video_index + 1) % len(video_files)

    # Update GPS and time labels based on the current video
    index if current_video_index == 0:
        gps_label.config(text=gps_video_1)
        time_label.config(text=time_video_1)
    else: gps_label.config(text=gps_video_2)
        time_label.config(text=time_video_2)


    # Start playing the next video
    play_video()

# Create a "Next" button
next_button = tk.Button(root, text='Next', command=next_video)
next_button.pack()

# Start playing the initial
video next_video()

# Start the Tkinter event
loop root.mainloop()
```