

SEMANTIC SEGMENTATION OF ROAD REGION DETECTION FOR SELF- DRIVING CARS

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Technology degree in
Information Technology

By

HEMALATHA.V(Reg.No - 39120038)
PREETHI.S (Reg.No – 39120084)



DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600119

MAY- 2023



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

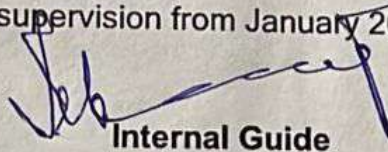
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

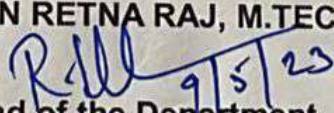
DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of Hemalatha.V (Reg.No - 39120038) and Preethi.S (Reg.No - 39120084) who carried out the Project Phase-2 entitled "SEMANTIC SEGMENTATION OF ROAD REGION DETECTION FOR SELF-DRIVING CARS" under my supervision from January 2023 to May 2023.

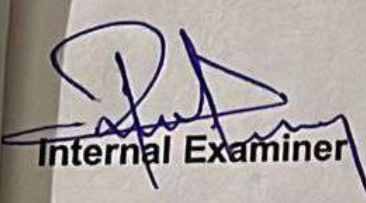

Internal Guide

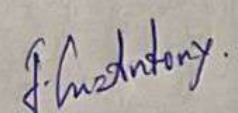
Dr. R. JEBERSON RETNA RAJ, M.TECH., Ph. D.,


Head of the Department

Dr. R. SUBHASHINI M.E., Ph.D.

Submitted for Viva voce Examination held on 10.05.2023

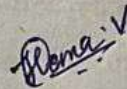

Internal Examiner


External Examiner

DECLARATION

I, HEMALATHA.V (Reg.No – 39120038), hereby declare that the Project Phase - 2 Report entitled **SEMANTIC SEGMENTATION OF ROAD REGION DETECTION FOR SELF-DRIVING CARS**” done by me under the guidance of Dr. JEBERSON RETNA RAJ., M.TECH., Ph. D is submitted in partial fulfillment of the requirements for the award Bachelor of Technology degree in Information Technology.

DATE: 10.06.2023



PLACE: Chennai

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA M.E., Ph. D, Dean**, School of Computing, **Dr. R. SUBHASHINI M.E., Ph.D.**, Head of the Department of Information Technology for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. R. JEBERSON RETNA RAJ M.TECH., Ph.D**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my phase - 2 project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

ABSTARCT

Any self-driving system must have a solid understanding of the situation. For scenario comprehension and subsequent acceptance of this new technology, accurate real-time visual signal processing for creating pixel-wise classed pictures, also known as semantic segmentation, is essential. Prior to recent advancements in deep learning algorithms, it was impossible to achieve such efficiency in terms of processing time and accuracy due to the intricate interaction between pixels in each frame of the received camera data. In this study, we present an efficient strategy for semantic segmentation for self-driving cars. Convolutional neural networks and other deep learning architectures are combined here. The cam video, which has had a lot of data added to. Real-time and precise processing of visual inputs is necessary for autonomous driving vehicles to understand their environment and implement this technology. For this job, semantic segmentation functions that produce pixel-labeled images are required. The complicated interplay between the components in each component of the data received from the camera made it impossible in the past to analyse data efficiently and accurately. Convolutional neural networks and other deep learning architectures are being employed in the creation of self-driving automobiles based on semantic segmentation since recent developments in deep learning algorithms have made it possible to achieve this efficiency. Self-driving car cameras record videos that are jam-packed with data that must be processed by deep learning algorithms in order to provide pixel-labeled images for precise scene interpretation. Objects, lanes, and barriers in the scene can be identified with the aid of semantic segmentation functions. Providing precise and timely information about the environment has the potential to increase the safety of autonomous driving cars. The development of self-driving cars still faces difficulties, though, including handling complicated and unforeseen situations and safeguarding the safety of passengers and other road users.

TABLE OF CONTENTS

Chapter No	TITLE	Page No.
	ABSTRACT	v
	LIST OF FIGURES	vii
1	INTRODUCTION	1
2	LITERATURE SURVEY	2
	2.1 Inferences from Literature Survey	5
	2.2 Open problems in Existing System	5
3	SYSTEM & REQUIREMENT ANALYSIS	11
	3.1 Existing System	11
	3.2 Proposed System	11
	3.3 Feasibility Studies/Risk Analysis of the Project	12
	3.4 Hardware & Software Requirements Specification Document	13
	3.5 System Use case	13
4	SYSTEM DESIGN	16
	4.1 Image Processing	16
5	DESCRIPTION OF PROPOSED SYSTEM	45
	5.1 Architecture / Overall Design of Proposed System	45
	5.2 Selected methodology or process model	46
	5.3 Description of Software for Implementation and Testing plan of the Proposed Model/System	47
	5.4 Project Management Plan	50
	5.5 Financial report on estimated costing	50
	5.6 Transition/ Software to Operations Plan	51

6	IMPLEMENTATION DETAILS	52
6.1	Development and Deployment Setup	52
6.2	Algorithm	54
6.3	Testing	56
6.4	Testing techniques / testing strategies	58
6.5	Software testing strategies:	59
7	RESULTS AND DISCUSSION	62
8	CONCLUSION	63
8.1	Future Work	63
8.2	Research Issues	64
8.3	Implementation Issues	64
	REFERENCES	65
	APPENDIX	68
	A. SOURCE CODE	68
	B. SCREENSHOT	71
	C. RESEARCH PAPER	73

FIGURE NO	FIGURE NAME	PAGE NO
1.1	Use Case Diagram	15
1.2	Image Processing	18
1.3	Gray level Image and corresponding surface	36
1.4	System Architecture	45

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION ABOUT SELF-DRIVING CARS

In recent years, the ability to move quickly and safely in driverless vehicles has been a hot topic of research, and numerous businesses and research centers are attempting to develop the first fully functional driverless car model. This is a field that looks very promising and has a lot of potential benefits, like more safety, lower costs, comfortable travel, more mobility, and less impact on the environment. The process of assigning each pixel of the received image to one of the predefined classes is known as semantic segmentation. The image's segment labels, such as roads, cars, signs, traffic lights, or pedestrians, are represented by these classes. As a result, "pixel wise classification" is a term used to describe semantic segmentation. The principal advantage of semantic division is circumstance understanding. As a result, as a first step toward achieving visual perception, it is utilized in numerous fields like autonomous driving, robotics, medical images, satellite images, precision agriculture, and facial images. To get a complete picture of the driving situation, autonomous driving relies on information from sensors in the surrounding environment. Semantic segmentation is essential for scene comprehension because the visual signal contains a lot of this information. The ego vehicle is better able to comprehend the environment around it and, as a result, to make the appropriate decision each and every time, the more we perform semantic segmentation in a short amount of time and with high accuracy. Semantic segmentation, on the other hand, is difficult due to the intricate relationship between pixels in each image frame and between frames. Performing accurate semantic segmentation in real time is still a hot topic in current research, as will be shown in detail later. This is the case despite the rapid development of new technologies like deep learning, which have made the mission of semantic segmentation more efficient

Advancement of technology offering solutions to the complex problems to a simple one. Automation in engineering and technology simplifies the human life which assures safety and security. Autonomous vehicle technology grows enormous nowadays for its demand in the market. In the field of constructing completely autonomous vehicles, many research laboratories and academic institutions are currently striving to achieve the best results. Autonomous self-driving vehicles work on the principles of artificial intelligence (AI), machine learning, and computer vision. These vehicles use a combination of sensors, cameras, and algorithms to perceive and interpret their surroundings, make decisions, and execute actions without human intervention.

1.2 ROLE OF AI AND MACHINE LEARNING ON SELF DRIVING CARS

The AI and machine learning algorithms that power autonomous vehicles rely on large amounts of data collected from various sources, such as LiDAR (Light Detection and Ranging) sensors, radar, and GPS. This data is used to build a detailed map of the vehicle's environment and to train the vehicle's algorithms to recognize and respond to different scenarios.

Computer vision technology is used to help autonomous vehicles perceive their surroundings in real-time. Cameras mounted on the vehicle capture images of the road, other vehicles, and pedestrians, which are then analyzed by the vehicle's AI algorithms to identify and track objects and obstacles. Overall, the principles behind autonomous self-driving vehicles involve complex technologies and algorithms that allow them to navigate roads and make decisions in a way that is safe and efficient.

Segmentation of the road in a road image is a key component for an autonomous driving vehicle. Semantic segmentation is a method in which each pixel in a picture is assigned a preset label, enabling the recognition of objects such as automobiles, signs, or pedestrians. This method has a wide range of applications in fields such as facial recognition, medical imaging, and autonomous driving. Semantic segmentation provides

situational understanding and a large amount of data required for scene recognition. However, due to the intricate relationships between the elements in each

image and the interactions between frames, this task is challenging. Although deep learning has significantly improved the efficiency of this task, achieving real-time and accurate semantic segmentation is still an area of active research. Semantic segmentation enables better environmental comprehension, and autonomous vehicles rely on data from sensors to make decisions. Therefore, real-time and precise semantic segmentation is essential for the safe operation of autonomous vehicles. However, due to the complexity of the task, it is still difficult to perform reliable semantic segmentation in real-time. Further research aims to enhance its effectiveness and accuracy.

CHAPTER 2

LITERATURE SURVEY

Ohgushi et . al introduced semantic segmentation of road in a complex image for road extraction for self driving car. This method employs with an autoencoder with semantic segmentation algorithm for detecting road that is trained on just typical road pictures. This feature offers a feasible option for advancing self-driving systems [1].

Deepika et al proposed a vision-based navigation system for autonomous driving that can classify and detect obstacles on the road. The system relies on a camera mounted on the vehicle to capture images of the road, which are processed to identify obstacles. The image processing techniques used include color segmentation, edge detection, and blob analysis. The system employs SegNet encoder-decoder architecture and it is tested with NVIDIA's Jetson TK1[2].

Cai et.al proposed a novel method for robust drivable road region detection for fixed-route autonomous vehicles using map-fusion images is proposed. The approach involves the fusion of the high-definition map with the visual images captured by the vehicle's camera to improve the accuracy and robustness of road detection. The map-fusion approach is used to improve the accuracy of road detection by incorporating prior knowledge about the road network[3].

Viswanath et.al proposed OFFSEG a framework for semantic segmentation in off-road driving scenarios using a deep learning approach. The framework consists of a convolutional neural network (CNN) architecture that can effectively distinguish between different types of terrain, such as grass, rocks, sand, and water. The approach is applied in real-time, allowing for effective navigation in off-road environments [4].

Saez et al. proposed a CNN-based semantic segmentation approach employs a fisheye camera pictures of urban traffic. The approach leverages with Efficient Residual

Factorized CNN (ERFNet) architecture used to handle distorted fish-eye pictures, and a new dataset of distorted fish-eye images is developed for training and assessment[5].

Chen et al proposes a semantic segmentation framework for urban scenes that can adapt to different lighting and weather conditions using a reality-oriented approach. The framework is based on a fully convolutional network (FCN) architecture that can accurately segment different objects in urban scenes, such as cars, buildings, and pedestrians [6].

Cheng et al. presented a framework for sequential semantic segmentation of road profiles for path and speed planning. The framework uses a combination of deep learning and rule-based approaches to segment the road profiles into different categories, such as straight, curve, intersection, and roundabout. One of the main advantages of this framework is its ability to provide accurate and timely information for path and speed planning, making it more effective for autonomous driving applications[7].

Asgarian et al. proposes a deep learning-based framework for detecting drivable areas for autonomous driving. The framework is designed to process images in real-time and accurately detect drivable areas under various environmental conditions [8].

Shim et al proposes a lightweight semantic segmentation framework for road-surface damage recognition . The framework uses a multi-scale learning approach to segment the road surface into different categories, such as cracks, potholes, and patches. One of the main advantages of this framework is its lightweight design, which allows for efficient processing of images on low-power devices[9].

Farag et. al proposes a real-time lane detection system for autonomous driving. The system uses computer vision techniques to detect and track lane markings on the road and provide feedback to the autonomous driving system[10].

Lee et.al proposes a deep learning-based semantic segmentation framework for road scenes. The framework uses a parallel complement network to perform semantic segmentation of road scenes in real-time[11].

Hofmarcher et al. presents a deep learning-based semantic segmentation approach for autonomous driving. The proposed approach utilizes a fully convolutional neural network (FCN) to segment images into various classes, such as road, sidewalk, vegetation, and vehicles. The approach also incorporates a post-processing step to refine the segmentation results and remove false positives[12].

Metzger et al. proposes a new dataset for semantic segmentation of unstructured driving scenarios. The authors created a dataset with fine-grained annotations, including 18 different object classes and 6 road classes, to improve the accuracy and robustness of the semantic segmentation models. The authors also propose an efficient segmentation model that utilizes a residual network with atrous spatial pyramid pooling (ASPP) and a decoder network for accurate and fast segmentation[13].

Rateke et al. proposes a system for detecting and differentiating road surfaces and surface damages using a deep learning approach. The system is designed for autonomous vehicles to improve their ability to navigate safely and efficiently in various road conditions[14].

Shaban et al. presents a semantic terrain classification framework for off-road autonomous driving. The proposed approach uses a deep neural network to segment terrain images into different categories, such as sand, grass, water, and rocks. The network architecture is based on a convolutional neural network (CNN) and uses a ResNet-18 backbone to extract high-level features from the input image[15].

Alkhorshid et al. presents a supervised classification approach for road detection, which involves training a machine learning algorithm using labeled examples of road and non-road pixels. The authors use a feature extraction method based on local binary patterns (LBP) to capture texture information in the image, and then apply a random forest classifier to classify each pixel as road or non-road[16].

John et al. (2016) proposed a method for road detection in images using supervised classification. They extracted features such as color, texture, and shape from the images and trained a Support Vector Machine (SVM) classifier to distinguish

between road and non-road pixels. Their approach achieved a high accuracy of 96.2%, but the limitation of their work is that it requires manual feature extraction[17].

Asgarian et al. (2021) proposed a fast drivable area detection method for autonomous driving using deep learning. They used a Fully Convolutional Network (FCN) to perform semantic segmentation of the input image and then applied a post-processing step to detect the drivable area. Their approach achieved high accuracy and fast processing speed, but the limitation of their work is that it requires a large amount of labeled data for training the FCN[18].

Boudissa et al. proposed a method for semantic segmentation of traffic landmarks using classical computer vision and a U-Net model. The classical computer vision algorithms include Canny edge detection, Hough transform, and watershed segmentation. They achieved promising results on a dataset containing traffic landmarks, including traffic signs, traffic lights, and lane markings. A limitation of this approach is that it relies on handcrafted features and may not generalize well to complex scenes with different lighting conditions or occlusions[19].

Huang et al. presented a real-time road curb and lane detection system using LiDAR point clouds. They used a combination of a ground plane estimation algorithm, a road curb detection algorithm, and a lane detection algorithm. The system achieved high accuracy and real-time performance on a dataset of urban roads. However, the limitation of this approach is that it relies on LiDAR sensors, which can be expensive and have limited range and resolution[20].

Wei et al. reviewed the use of mmWave radar and vision fusion for object detection in autonomous driving. They discussed the advantages and limitations of mmWave radar and vision sensors and proposed a fusion approach that can enhance the detection performance in challenging conditions such as low visibility, occlusions, and adverse weather conditions. A limitation of this approach is that mmWave radar has limited resolution and can produce many false alarms, which can lead to high computational overhead and may require additional processing steps to reduce false positives[21].

Rajaram et al. proposes a RefineNet architecture that can improve object detection performance in autonomous driving scenarios by refining and verifying the output of a base object detector. The RefineNet uses a multi-stage approach that learns to refine the object proposals at different scales and resolutions, leading to better localization and classification performance. The limitation of this approach is that it requires a pre-trained base object detector to function, which can be computationally expensive[22].

Teichmann et al. proposes a MultiNet architecture that can perform joint semantic reasoning for autonomous driving tasks. The MultiNet consists of multiple neural networks that specialize in different tasks such as object detection, semantic segmentation, and depth estimation. These networks work together to provide a more comprehensive understanding of the environment and improve overall performance. One limitation of this approach is that it requires multiple neural networks, which can be computationally expensive[23].

Petrovai and Nedevschi's proposes a Semantic Camera system for 360-degree environment perception in automated urban driving. The system uses a combination of fisheye cameras and deep neural networks to detect and classify objects in the environment. The system is designed to be low-cost and can be easily integrated into existing vehicles. A limitation of this approach is that fisheye cameras may suffer from distortion and other imaging issues, which can affect object detection performance[24].

2.1 INFERENCES FROM LITREATURE SURVEY

From the above literature survey we can see that they are using deep learning cnn algorithm used. By using that CNN algorithm what will happen the network is trained on perception data, where it learns what decision it should make. Because the CNNs are very good at extracting features of representations from the input, DRL algorithms can be trained on those representations

2.2 OPEN PROBLEMS IN EXISTING SYSTEM

The existing system are not done with many semantic segment Images and also the existing system has low accuracy and low efficiency in terms of loading time and implementation time. Also the testing and training is not done with the proper test-train split ratio.

CHAPTER 3

SYSTEM & REQUIREMENT ANALYSIS

3.1 EXISTING SYSTEM

In the existing system we used deep learning algorithm called CNN.

By mapping pixels from the camera input to the steering commands, a level 2 autonomous vehicle is implemented using a convolutional neural network (CNN). Since the camera input is used to teach the network the most variable features, human intervention is minimal

Disadvantages

Include the fact that the CNN cannot encode the position and orientation of objects and that a large amount of training data is required for its effectiveness. They are incapable of encoding an object's orientation and position. They struggle to classify images in various positions

3.2 PROPOSED SYSTEM

In the proposed system we are going to use R-CNN algorithm Regions with convolutional neural networks (R-CNN) is a method for deep learning that combines the features of convolutional neural networks with proposals for rectangular regions. R-CNN is a detection algorithm with two stages. R-CNN or RCNN is a type of machine learning model used for computer vision tasks, specifically self-driving semantic segmentation. R-CNN or RCNN stands for Region-Based Convolutional Neural Network.

Advantages

They are very good at recognizing and classifying images. R- CNNs also have a significant advantage in weight sharing. When compared to a conventional

neural network, convolutional neural networks also reduce computation.

3.3 FEASIBILITY STUDIES/RISK ANALYSIS OF THE PROJECT

3.3.1 *feasibility Study*

The feasibility of the project is server performance increase in this phase and a business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis, the feasibility study of the proposed system is to be carried out. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- Economical feasibility
- Technical feasibility
- Operational feasibility

Economical Feasibility:

This study is carried out to check the economic impact that the system will have on the organization. The amount of funds that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

Technical Feasibility:

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands being placed on the client.

The developed system must have modest requirements, as only minimal or null changes are required for implementing this system.

Operational Feasibility:

The aspect of the study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system

3.4 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

Hardware specifications:

- Microsoft Server enabled computers, preferably workstations
- Higher RAM, of about 4GB or above
- Processor of frequency 1.5GHz or above

Software specifications:

- Python 3.6 and higher
- Anaconda software

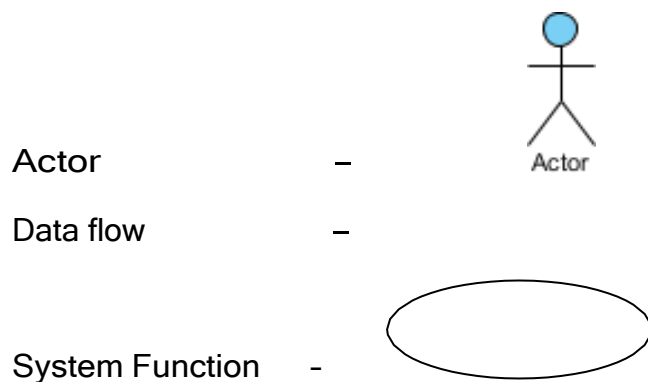
3.5 System Use case

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a

system/subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system. Use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. A use case diagram doesn't go into a lot of detail—for example, don't expect it to model the order in which steps are performed. Instead, a proper use case diagram depicts a high-level overview of the relationship between use cases, actors, and systems. Experts recommend that use case diagrams be used to supplement a more descriptive textual use case. An effective use case diagram can help your team discuss and represent:

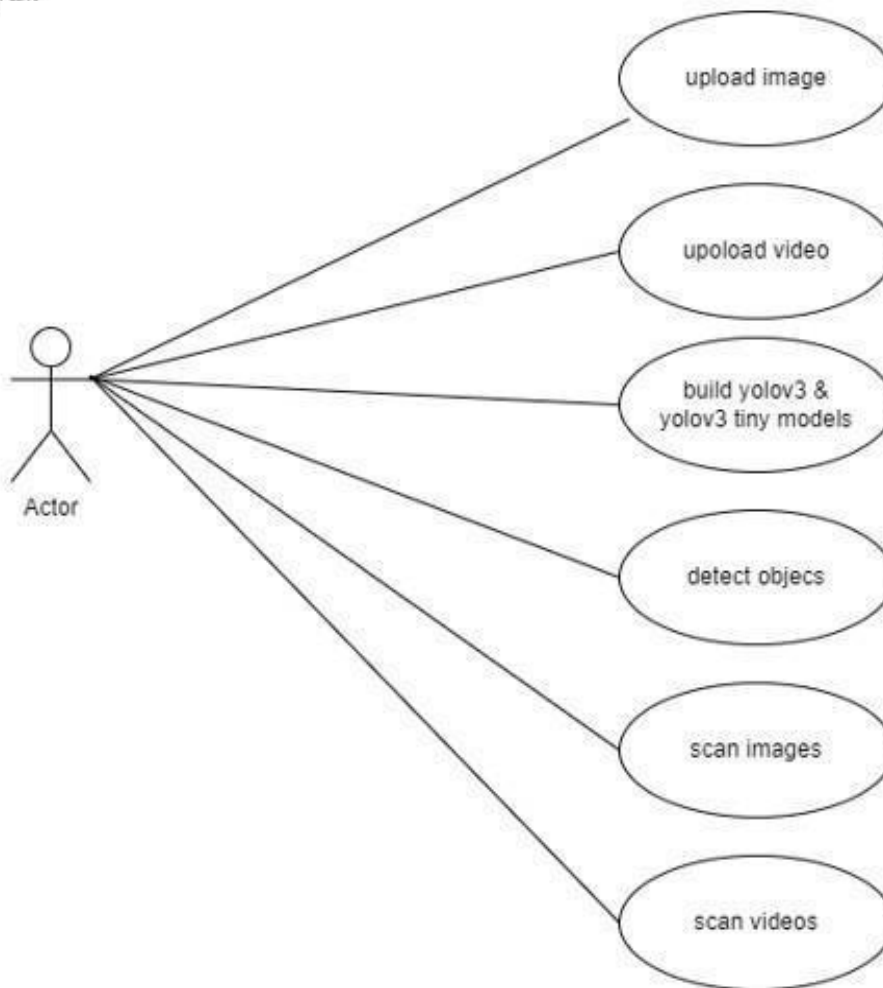
- Scenarios in which your system or application interacts with people, organizations, or external systems
- Goals that your system or application helps those entities (known as actors) achieve the scope of your system

➤ Symbols Used



In the below system user case diagram they mention that by using semantic segmentation we are going to segment the vehicle part radius of curvature. In the first step we are going to upload the video. Second step is all about convert that video into fps after that fps image conversion it will show and classify the each and every part of the video. Detect the objects which means it will classify the objects called vehicle.

Diagram



CHAPTER 4

SYSTEM DESIGN

R-CNN (Region-based Convolutional Neural Networks) is a popular algorithm used for object detection and semantic segmentation tasks in the field of self-driving cars. The algorithm works by first generating region proposals in an image using a selective search algorithm. These regions are then processed using a convolutional neural network (CNN) to extract features. The features are then fed into a set of support vector machines (SVMs) to classify the regions into object categories. In the case of semantic segmentation, a fully convolutional network (FCN) is used to generate dense pixel-wise predictions for each object class. The R-CNN algorithm has been shown to achieve state-of-the-art performance on benchmark datasets for object detection and semantic segmentation tasks in the field of self-driving cars.

A. System architecture process

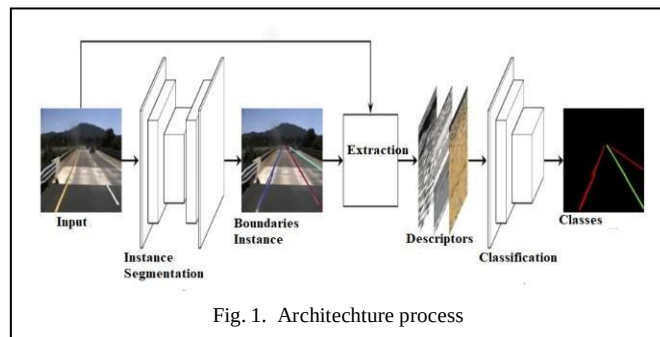


Fig. 1. **Architecture process**

The system architecture for a self-driving car involves a series of steps. Firstly, the system acquires a video. Then, the video is converted into frames per second (fps) to make it easier to process. After the fps image conversion, feature extraction and segmentation take place. During this process, the essential elements of the image are processed, and the image is divided into groups to categorize them using a mask or image. Once the segmentation is complete, the final output is presented. This architecture is designed to ensure that only the relevant elements of the image are processed, which helps to improve the accuracy and efficiency of the system.

The following is the procedure for segmenting road features from a image frame.

1. Load pre-trained Convolutional Neural Network (CNN) weights
2. Set the image input size and the size of the Region of Interest (RoI)
3. Load the image and generate a set of region proposals using selective search
4. For each region proposal, do the following:
 - a. Crop and resize the image to the input size of the CNN
 - b. Feed the cropped image to the CNN and obtain the convolutional feature maps
 - c. Pass the feature maps through a Region-based Convolutional Neural Network (R-CNN) head to obtain region-wise features
 - d. Feed the region-wise features to a multi-layer perceptron (MLP) to obtain class scores and bounding box offsets
 - e. Apply non-maximum suppression (NMS) to filter out overlapping regions
5. Generate the final segmentation map by assigning each pixel to the class label of its corresponding region proposal

B. Input data acquisition system

The process of retrieving an image or video for additional image processing is described in the paragraph as image acquisition. Since there is no processing available prior to image acquisition, it is usually the initial stage in the workflow.

C. Preprocessing

Image pre-processing is a crucial step that needs to be performed before using photos for model training and inference. It helps to increase or decrease the speed of pattern preparation. The video can be converted into frames per second, and the final image can be converted into grayscale during pre-processing. Some common image pre-processing techniques include color correction, reorientation, and scaling. Pre-processing can be considered as a pond-level module that can significantly affect the accuracy of the model's predictions.

D. Feature extraction

In the context of segmentation for a self-driving car, feature extraction is a crucial step that involves converting raw image data into numerical features that can be processed by machine learning algorithms. This process is performed by transforming the raw image data, which is typically in grayscale, into a set of numerical features that capture important information about the image. The main feature of interest, such as the region of interest, is extracted using this process. The Re-LU layer, which stands for Rectified Linear Unit layer, is used to perform segmentation and feature extraction. By performing feature extraction, the machine learning algorithm can better understand the important aspects of the image and make more accurate predictions about the environment in which the self-driving car is operating.

E. Segmentation

Semantic segmentation is a process that involves assigning a label to each pixel in an image. In the context of self-driving cars, semantic segmentation is used to identify

objects such as roads, cars, pedestrians, and traffic signs. This is achieved through the use of image segmentation techniques, which divide the image into distinct regions, followed by feature extraction, which identifies the key characteristics of each region. The Re-LU layer is often used for both of these processes. Once the features have been extracted, they are used to classify each pixel in the image, resulting in a map that shows the location of different objects on the road. This map can then be used by the self-driving car to make decisions about how to navigate through the environment.

F. Classification

The classification process for a self-driving car involves grouping images or vectors according to predefined rules. This process uses spectral or texture functions to produce an effective classification rule. The key criteria set during this process are the radius of curvature and the distance to the car. The radius of curvature determines the final circle that is drawn during turning, while the distance to the car is an essential factor for detecting objects in the car's environment. The final step of the classification process is carried out by a completely connected layer, which uses the extracted features from the previous stages to classify the input image into one of the predetermined categories.

4.1 IMAGE PROCESSING

Image processing is a method to convert an image into digital form and perform some operations on it, in order to get an enhanced image or to extract some useful information from it. It is a type of signal dispensation in which input is image, like video frame or photograph and output may be image or characteristics associated with that image. Usually Image Processing system includes treating images as two dimensional signals while applying already set signal processing methods to them.

It is among rapidly growing technologies today, with its applications in various

aspects of a business. Image Processing forms core research area within engineering and computer science disciplines too.

Image processing basically includes the following three steps.

Importing the image with optical scanner or by digital photography. Analyzing and Manipulating the image which includes data compression and image enhancement and spotting patterns that are not to human eyes like satellite photographs. Output is the last stage in which result can be altered image or report that is based on image analysis.

Purpose of Image processing

The purpose of image processing is divided into 5 groups. They are:

1. Visualization - Observe the objects that are not visible.
2. Image sharpening and restoration - To create a better image.
3. Image retrieval - Seek for the image of interest.
4. Measurement of pattern - Measures various objects in an image.
5. Image Recognition - Distinguish the objects in an image.

Types

The two types of methods used for Image Processing are Analog and Digital Image Processing. Analog or visual techniques of image processing can be used for the hard copies like printouts and photographs. Image analysts use various fundamentals of interpretation while using these visual techniques. The image processing is not just

confined to area that has to be studied but on knowledge of analyst. Association is another important tool in image processing through visual techniques. So analysts apply a combination of personal knowledge and collateral data to image processing.

Digital Processing techniques help in manipulation of the digital images by using computers. As raw data from imaging sensors from satellite platform contains deficiencies. To get over such flaws and to get originality of information, it has to undergo various phases of processing. The three general phases that all types of data have to undergo while using digital technique are Pre- processing, enhancement and display, information extraction.

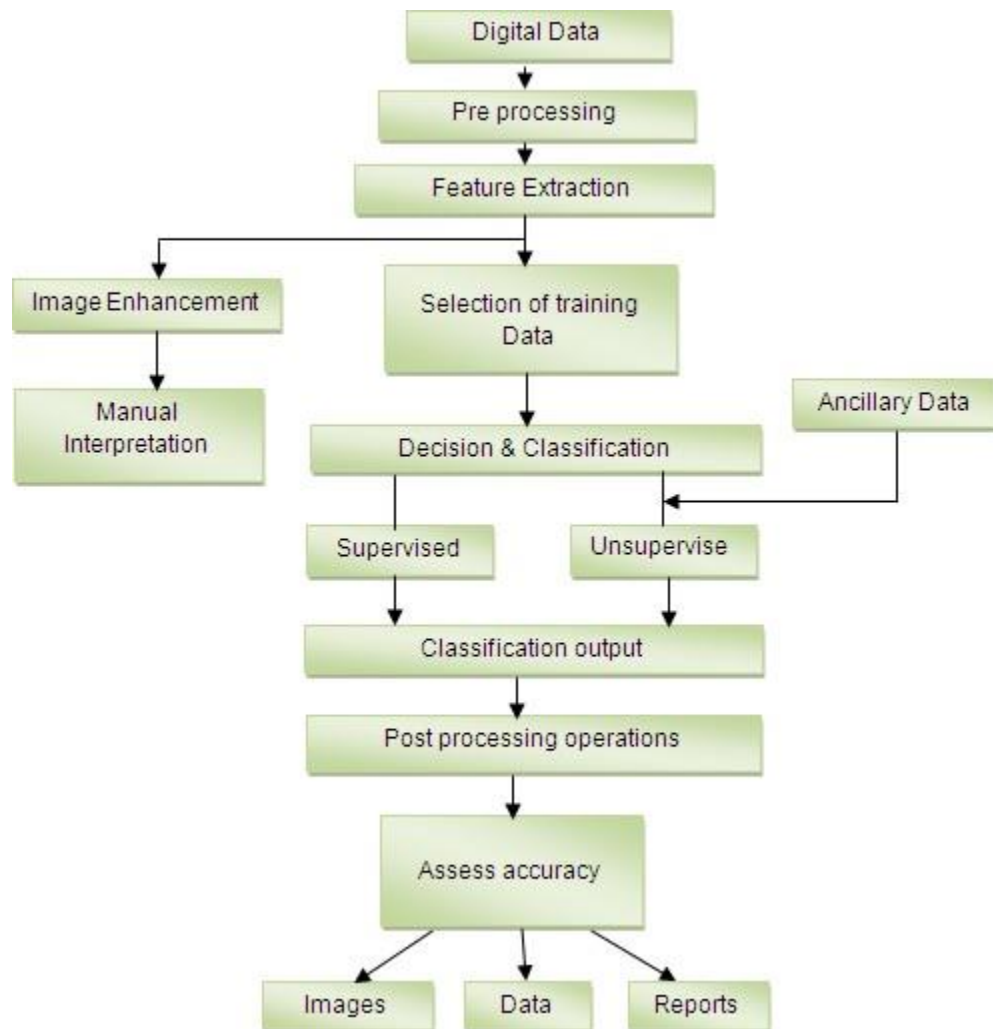


Fig:1.2 Detailed workflow of process a image

IMAGE PROCESSING CONCEPTS

Binary Images

Binary images are images whose pixels have only two possible intensity values. They are normally displayed as black and white. Numerically, the two values are often 0 for black, and either 1 or 255 for white. Binary images are often produced by thresholding a grayscale or color image, in order to separate an object in the image from the background. The color of the object (usually white) is referred to as the foreground color. The rest (usually black) is referred to as the background color. However, depending on the image which is to be thresholded, this polarity might be inverted, in which case the object is displayed with 0 and the background is with a non-zero value. Some morphological operators assume a certain polarity of the binary input image so that if we process an image with inverse polarity the operator will have the opposite effect. For example, if we apply a closing operator to a black text on white background, the text will be opened.

Pixel Values:

Each of the pixels that represents an image stored inside a computer has a pixel value which describes how bright that pixel is, and/or what color it should be. In the simplest case of binary images, the pixel value is a 1-bit number indicating either foreground or background. For a grayscale images, the pixel value is a single number that represents the brightness of the pixel. The most common pixel format is the byte image, where this number is stored as an 8-bit integer giving a range of possible values from 0 to 255. Typically zero is taken to be black, and 255 is taken to be white. Values in between make up the different shades of gray.

To represent color images, separate red, green and blue components must be specified for each pixel (assuming an RGB color space), and so the pixel 'value' is actually a vector of three numbers. Often the three different components are stored as three separate 'grayscale' images known as color planes (one for each of red, green and blue), which have to be recombined when displaying or processing. Multi-spectral images can contain even more than three components for each pixel, and by extension these are stored in the same kind of way, as a vector pixel value, or as separate color planes.

The actual grayscale or color component intensities for each pixel may not actually be stored explicitly. Often, all that is stored for each pixel is an index into a colormap in which the actual intensity or colors can be looked up.

Although simple 8-bit integers or vectors of 8-bit integers are the most common sorts of pixel values used, some image formats support different types of value, for instance 32-bit signed integers or floating point values. Such values are extremely useful in image processing as they allow processing to be carried out on the image where the resulting pixel values are not necessarily 8-bit integers. If this approach is used then it is usually necessary to set up a colormap which relates particular ranges of pixel values to particular displayed colors.

Primary Colors:

It is a useful fact that the huge variety of colors that can be perceived by humans can all be produced simply by adding together appropriate amounts of red, blue and green colors. These colors are known as the primary colors. Thus in most image processing applications, colors are represented by specifying separate intensity values for red, green and blue components. This representation is commonly referred to as

RGB.

The primary color phenomenon results from the fact that humans have three different sorts of color receptors in their retinas which are each most sensitive to different visible light wavelengths.

The primary colors used in painting (red, yellow and blue) are different. When paints are mixed, the 'addition' of a new color paint actually subtracts wavelengths from the reflected visible light.

RGB and Color spaces:

A color perceived by the human eye can be defined by a linear combination of the three primary colors red, green and blue. These three colors form the basis for the RGB- color space. Hence, each perceivable color can be defined by a vector in the three- dimensional color space. The intensity is given by the length of the vector, and the actual color by the two angles describing the orientation of the vector in the color space.

The RGB-space can also be transformed into other coordinate systems, which might be more useful for some applications. One common basis for the color space is IHS. In this coordinate system, a color is described by its intensity, hue (average wavelength) and saturation (the amount of white in the color). This color space makes it easier to directly derive the intensity and color of perceived light and is therefore more likely to be used by human beings.

CHAPTER 5

DESCRIPTION OF PROPOSED SYSTEM

5.1 ARCHITECTURE / OVERALL DESIGN OF PROPOSED SYSTEM

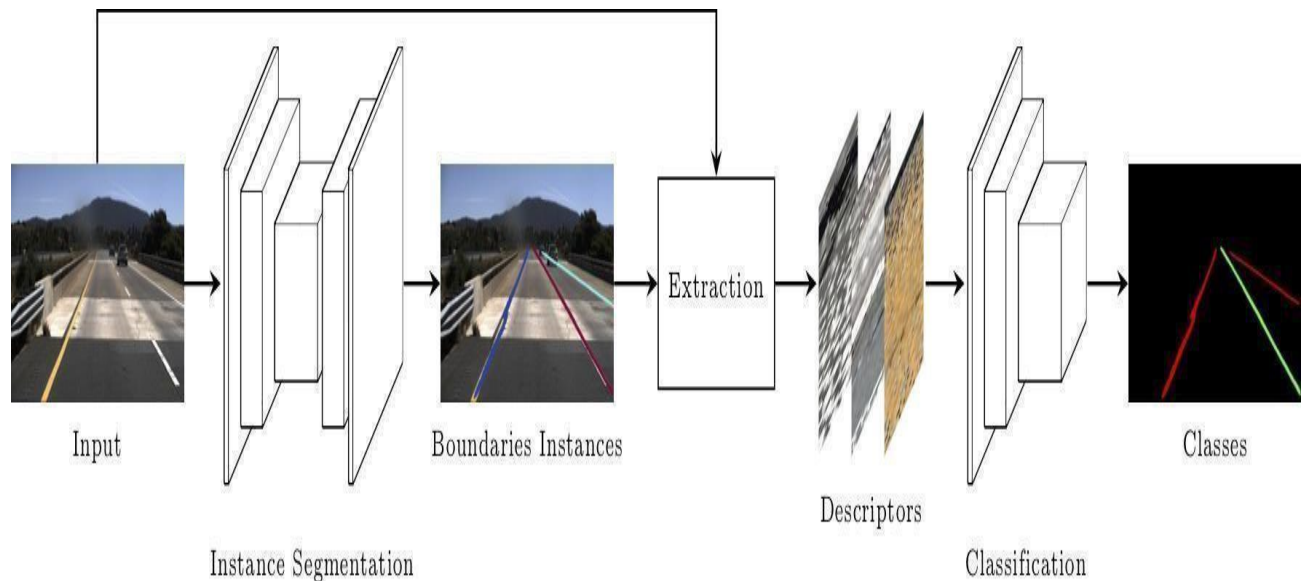


Fig:1.4 System Architecture

In the first step we are going to upload the image. Its called as input data acquisition system. Here we upload the video In the second step we are going to convert that video into fps image. That fps image getting convert into grey scale image. This scale image is getting converter into label image by using label technique. By using the threshold process that image from segmentation is getting converted into detection by using SOBEL method SOBEL method is one of the process in edge detection technique. Once the process is getting over then the output will be shown.

5.2 SELECTED METHODOLOGY OR PROCESS MODEL

We're going to implement the R-CNN algorithm in the suggested device. Regions Convolutional Neural Networks (R-CNN) is a deep learning method that combines convolutional neural network features with square area suggestions. Level PC Discovery uses the R-CN algorithm .A type of machine learning model called R-CN or RCN is utilise for computer vision tasks, particularly to aid in self-semantic segmentation. Region-Based Convolutional Neural Network is referred to as R-CN or RCN.

Module 1: Input data acquisition system

Module 2: Preprocessing

Module 3: Feature extraction

Module 4: Segmentation

Module 5: Classification

Module 1: Input data acquisition system

Image acquisition is the process of retrieving an image or video from an external source for further processing in image processing. Because there is no process available prior to obtaining an image, it is always the foundational step in the workflow. Here we are going to give input video in a mp4 format. Convolutional layer will do the module process called input data acquisition system

Module 2: Data Preprocessing

The steps taken to format images before they are used by model training and

inference are referred to as image preprocessing. Color corrections, reorientation, and resizing are all examples of this. Picture preprocessing may likewise diminish model preparation time and speed Up Preprocessing is all about it will converted the video into fps and that fps image will get Converted into grey scale image. Pooling layer will do the preprocessing process module

Module 3 : Featureextraction

The process of converting raw data into numerical features that can be processed while preserving the information in the original data set is known as feature extraction. When compared to applying machine learning directly to the raw data, it produces superior results. From the raw data image i.e grey scale image we are going to find out the main feature like region of interest. Re-LU layer will do the process module called feature extraction and segmentation.

Module 4: Segmentation

Converting an image into a group of pixels that are represented by a mask or a labeled image is the process of image segmentation. You can process only the important parts of an image rather than the entire image by breaking it up into segments. Segmentation is all about to find out the radius of curvature and vehicle distance. Re-LU layer will do the process module called feature extraction and segmentation.

Module 5 : Classification

The process of categorizing and labeling groups of pixels or vectors within an image based on particular rules is known as image classification. One or more

spectral or textural characteristics can be used to create the categorization law. At the end it will classify the main things like radius of curvature and vehicle distance. The radius of curvature is the radius of curvature of the road, the radius of the outermost circle among the concentric circles drawn when steering and turning. Fully connected layer will do the process called Classification.

5.3 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN OF THE PROPOSED MODEL/SYSTEM

5.3.1 PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This tutorial gives enough understanding on Python programming language. The finest programming language for machine learning is Python. According to research and surveys, Python is the sixth most significant language to date as the go-to language for information science and machine learning. Python has the following benefits because of

- Easy to be told and perceive- Because of Python's simplified structure, learning and understanding the language is quite simple, especially for beginners.
- Multi-purpose language – Because it offers both structured and object-oriented programming in addition to practical programming, Python may be a multi-purpose programming language.
- Support of open supply community – Python is maintained by an incredibly sizable developer community as an open source programming language. As a result, the Python community has been merely mounting the bugs. Due to this quality, Python is incredibly resilient and adaptable

APPLICATION OF PYTHON

- Easy-to-learn – Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- Easy-to-read – Python code is more clearly defined and visible to the eyes.
- Easy-to-maintain – Python's source code is fairly easy-to-maintain.
- A broad standard library – Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- Interactive Mode – Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- Portable – Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- Extendable – You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- Databases – Python provides interfaces to all major commercial databases.
- GUI Programming – Python supports GUI applications that can be created and ported to many system calls, libraries and window systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
 - Scalable – Python provides a better structure and support for large programs than shell scripting.

IMAGE PROCESSING IN PYTHON

Images define the world, each image has its own story, it contains a lot of crucial

information that can be useful in many ways. This information can be obtained with the help of the technique known as Image Processing. It is the core part of computer vision which plays a crucial role in many real-world examples like robotics, self-driving cars, and object detection. Image processing allows us to transform and manipulate thousands of images at a time and extract useful insights from them. It has a wide range of applications in almost every field. Python is one of the widely used programming languages for this purpose. Its amazing libraries and tools help in achieving the task of image processing very efficiently. Through this article, you will learn about classical algorithms, techniques, and tools to process the image and get the output.

5.4 PROJECT MANAGEMENT PLAN

Introduction:	September 1-30
Literature Survey:	October 1-31
System Design:	November 1-30
System Implementation:	December 1-31
Testing:	January 1-30

5.5 Financial report on estimated costing

Data preparation:

The R-CNN algorithm requires large amounts of image data to train the model. Therefore, data preparation, including data acquisition, labeling, cleaning, and

preprocessing, can be a significant cost factor.

COST FACTOR : RS.300

Hardware and software:

Training a R-CNN model can be computationally intensive and require specialized hardware, such as graphics processing units (GPUs), and software tools, such as TensorFlow or PyTorch. The cost of renting or buying these resources can add up quickly.

SOFTWARE AND HARDWARE COST : RS.280(Subscriptions)+ RS.1250 SSD

Testing and validation: Once the R-CNN model is trained, it needs to be tested and validated to ensure that it meets the project's requirements. This process can involve additional costs, such as data annotation and manual inspection of the output images.

COST:RS.280

The developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased

Overall cost : $300+280+1250+280 = RS. 2110$

5.6 Transition/ Software to Operations Plan

Develop a deployment strategy: This should include identifying the appropriate infrastructure and hardware needed to deploy the software in production. This may involve selecting a cloud provider or building an on-premise infrastructure, depending on the company's needs.

Set up an automated build process: To ensure that the software is built and deployed consistently, an automated build process should be set up. This will ensure that the latest version of the software is always available for deployment.

Create a testing strategy: A testing strategy should be developed to ensure that the software is functioning as expected. This should include unit tests, integration tests, and end-to-end tests.

Develop a monitoring plan: The software should be monitored to ensure that it is running as expected in production. This should include setting up monitoring tools such as logs, metrics, and alerts.

Establish a maintenance plan: A maintenance plan should be developed to ensure that the software is kept up to date and any issues are addressed promptly. This may involve creating a schedule for updating software dependencies or implementing regular security updates.

Train personnel: Personnel responsible for operating and maintaining the software should be trained on its usage and functionality. This will ensure that they are equipped to handle any issues that may arise.

Implement a feedback loop: Finally, a feedback loop should be implemented to ensure that any issues or user feedback are addressed promptly and effectively. This will help to ensure that the software remains relevant and meets the needs of its users.

CHAPTER 6

IMPLEMENTATION DETAILS

6.1 DEVELOPMENT AND DEPLOYMENT SETUP

CNN is one of the deep learning techniques with the best image recognition results at the moment. This is due to CNN's ability to process image information by imitating the visual cortex's image recognition system. However, CNN also has a weakness: it needs to train models, which takes a long time. Therefore we develop a new algorithm called RCNN- region convolutional neural network. Thus, Fast R-CNN was developed as an improvement of the R-CNN algorithm, Faster R-CNN was developed as an improvement of the Fast R-CNN algorithm, and the latest Mask R-CNN model was developed from Faster R-CNN and its algorithmic model architecture.

In the first step we are going to upload the image. Its called as input data acquisition system. Here we upload the video In the second step we are going to convert that video into fps image. that fps image getting convert into grey scale image. This scale image is getting converter into label image by using label technique By using the threshold process that image from segmentation is getting converted into detection by using SOBEL method SOBEL method is one of the process in edge detection technique. Once the process is getting over then the output will be shown.

- Module 1: Input data acquisition system
- Module 2: Preprocessing
- Module 3: Feature extraction

- Module 4: Segmentation
- Module 5: Classification

Module 1: Input data acquisition system

Image acquisition is the process of retrieving an image or video from an external source for Further processing in image processing. Because there is no process available prior to obtaining an image, it is always the foundational step in the workflow. Here we are going to give input video in a mp4 format. Convolutional layer will do the module process called input data acquisition system

Module 2: Preprocessing

The steps taken to format images before they are used by model training and inference are referred to as image preprocessing Color corrections, reorientation, and resizing are all examples of this. Picture preprocessing may likewise diminish model preparation time and speed up. Preprocessing is all about it will converted the video into fps and that fps image will get converted into grey scale image. Pooling layer will do the preprocessing process module

Module 3: Feature extraction

The process of converting raw data into numerical features that can be processed while Preserving the information in the original data set is known as feature extraction. When compared to applying machine learning directly to the raw data, it produces Superior result From the raw data image i.e grey scale image we are going to find out the main feature like Region of interest.

Module 4: Segmentation

Converting an image into a group of pixels that are represented by a mask or a labeled image is the process of image segmentation You can process only the important partsof an image rather than the entire image by breaking it up into segments . Segmentation is all about to find out the radius of curvature and vehicle distance .Re-LU layer will do the process module called feature extraction and segmentation.

Module 5: Classification

The process of categorizing and labeling groups of pixels or vectors within an image Based on particular rules is known as image classification. One or more spectral or textural characteristics can be used to create the categorization law. At the end it will classify the main things like radius of curvature and vehicle distance. The radius of curvature is the radius of curvature of the road, the radius of the outermost circle among the concentric circles drawn when steering and turning. Fully connected layer will do the process called Classification.

6.2 Algorithm

Region based convolutional neural networks (R-CNN) Object detection is the process of finding and classifying objects in an image. One deep learning approach, regions with convolutional neural networks (R-CNN), combines rectangular region proposals with convolutional neural network features. R-CNN is a two-stage detection algorithm. The first stage identifies a subset of regions in an image that might contain an object. The second stage classifies the object in each region.

Applications for R-CNN object detectors include:

- Autonomous driving
- Smart surveillance systems
- Facial recognition

COMPONENTS OF R-CNN

The R-CNN (Region-based Convolutional Neural Network) algorithm is a popular object detection method that consists of several components:

- Convolutional layer
- pooling layer
- RE- LU layer

- Fully layer

Convolutional layer:

The majority of a CNN's computation takes place in the convolutional layer, which is its fundamental component. A few parts are needed, including input data, a filter, and a feature map. Assume that a color image consisting of a 3D matrix of pixels will be the input. Convolutional layer will do the module process called input data acquisition system

Pooling layer:

The purpose of the pooling layers is to reduce the dimensions of the hidden layer by combining the outputs of neuron clusters at the previous layer into a single neuron in the next layer. Pooling layer will do the preprocessing process module

RE-LU layer:

Usage of RE-LU helps to prevent the exponential growth in the computation required to operate the neural network. Re-LU layer will do the process module called feature extraction and segmentation.

Fully connected layer:

A bias vector is added after a fully connected layer multiplies the input by a weight matrix. One or more fully connected layers appear after the convolutional and down-sampling layers. A fully connected layer, as the name suggests, has all of its neurons connected to all of the neurons in the layer before it. Fully connected layer will do the process called Classification

The accuracy is determined as the proportion between the quantity of Positive examples accurately grouped to the absolute number of tests delegated Positive (either accurately or erroneously). The accuracy estimates the model's exactness in grouping an example as sure.

Explain Recall computation and its formula

Recall is defined mathematically as follows: $\text{Recall} = \frac{TP}{TP + FN}$. A model that creates no misleading negatives has a review of 1.0.

TABLE I.

TABLE II. PERFORMANCE OF THE PROPOSED METHOD

Algorithm	Precision	Recall	Accuracy
CNN	0.6	0.65	80%
RCNN	0.8	0.87	92%

6.3 Testing

Testing Methodologies

There are many different types of testing methods or techniques used as part of the software testing methodology. Some of the important testing methodologies are:

SYSTEM TESTING

Testing is performed to identify errors. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing during phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. For example, the design must not have any logic faults in the design is detected before coding commences, otherwise, the cost of fixing the faults will be considerably higher as reflected. Detection of design faults can be achieved by means of inspection as well as a walkthrough. Testing is one of the important steps in the software development phase. Testing checks for the errors, as a whole of the project testing involves the following test cases:

- Static analysis is used to investigate the structural properties of the Source code.
- Dynamic testing is used to investigate the behavior of the source code by executing the program on the test data.

Unit Testing

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing.

Functional Tests

Functional test cases involved exercising the code with nominal input values for which the expected results are known, as well as boundary values and special values, such as logically related inputs, files of identical elements, and empty files.

Three types of tests in Functional test:

- Performance Test
- Stress Test
- Structure Test

Performance Test

It determines the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization by the program unit.

Stress Test

Stress Test is those test designed to intentionally break the unit. A Great deal

can be learned about the strength and limitations of a program by examining the manner in which a programmer in which a program unit breaks.

Structure Test

Structure Tests are concerned with exercising the internal logic of a program and traversing particular execution paths. The way in which White-Box test strategy was employed to ensure that the test cases could Guarantee that all independent paths within a module have been have been exercised at least once.

- Exercise all logical decisions on their true or false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Exercise internal data structures to assure their validity.
- Checking attributes for their correctness.
- Handling end of file conditions, I/O errors, buffer problems, and textual errors in the output information

Integration Testing

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules that makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach evolved from the unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated together and tested, to which

another module is added and tested in combination. And so on. The advantages of this approach are that interface dispenses can be easily found and corrected. The major error that was faced during the project is a linking error. When all the modules are combined the link is not set properly with all support files. Then we checked out for interconnection and the links. Errors are localized to the new module and its intercommunications. The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

6.4 TESTING TECHNIQUES / TESTING STRATEGIES

Testing is the process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet-undiscovered error. A successful test is one that uncovers an as-yet-undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hangs together. System testing requires a test consisting of several key activities and steps for running a program, string, and system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing. The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise, the program or the project is not said to be complete. Software testing is the critical element of software quality assurance and represents the ultimate review of specification design and coding. Testing is the process of executing the program with the intent of finding the error. A good test case design is one that has a probability of finding a yet undiscovered error. A successful test is one that uncovers a yet undiscovered error. Any engineering product can be tested in one of the two ways:

White-box testing:

This testing is also called Glass box testing. In this testing, by knowing the specific functions that a product has been designed to perform tests can be conducted that demonstrate each function is fully operational and at the same time search for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is white box testing. Basis path testing:

- Flow graph notation
- Cyclometric complexity
- Deriving test cases

Graph matrices ControlBlack box testing:

In this testing by knowing the internal operation of a product, a test can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software. The steps involved in black-box test case design are:

- Graph-based testing methods
- Equivalence partitioning
- Boundary value analysis
- Comparison testing

6.5 SOFTWARE TESTING STRATEGIES:

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically. For this reason, a template for software testing a set of steps into which we can place specific test case design methods should be strategy should have the following characteristics:

- Testing begins at the module level and works “outward” toward the integration of the entire computer-based system.
- Different testing techniques are appropriate at different points in time.
- The developer of the software and an independent test group conducts testing.
- Testing and Debugging are different activities but debugging must be accommodated in any testing strategy.

Integration Testing

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with it. Individual modules, which are highly prone to interface errors, should not be assumed to work instantly when we put them together. The problem, of course, is “putting them together”- interfacing. There may be the chances of data loss across another’s sub-functions when combined may not produce the desired major function; individually acceptable impressions may be magnified to unacceptable levels; global data structures can present problems.

Program Testing

The logical and syntax errors have been pointed out by program testing. A syntax error is an error in a program statement that violates one or more rules of the language in

which it is written. An improperly defined field dimension or omitted keywords are common syntax errors. These errors are shown through error messages generated by the computer. A logic error on the other hand deals with the incorrect data fields, out-of-range items, and invalid combinations. Since the compiler will not deduct logical errors, the programmer must examine the output. Condition testing exercises the logical conditions contained in a module. The possible types of elements in a condition include a Boolean operator, Boolean variable, a pair of Boolean parentheses, a relational operator or an arithmetic expression. The condition testing method focuses on testing each condition in the program the purpose of condition test is to deduct not only errors in the condition of a program but also other errors in the program.

Security Testing

Security testing attempts to verify the protection mechanisms built into a system well, in fact, protect it from improper penetration. The system security must be tested for invulnerability from frontal attacks must also be tested for invulnerability from rear attacks. During security, the tester places the role of the individual who desires to penetrate the system.

Validation Testing

At the culmination of integration testing, the software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test- validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in a manner that is reasonably expected

by the customer. Software validation is achieved through a series of black-box tests that demonstrate conformity with requirements. After the validation test has been conducted, one of two conditions exists. The function or performance characteristics conform to

specifications and are accepted. A validation from the specification is uncovered and a deficiency is created. Deviation or errors discovered at this step in this project are corrected prior to completion of the project with the help of the user by negotiating to establish a method for resolving deficiencies. Thus the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily. Though there were deficiencies in the system they were not catastrophic

User Acceptance Testing:

User acceptance of the system is a key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective systems and users at the time of developing and making changes whenever required. This is done in regard to the following points.

- Input screen design.
- Output screen design.

CHAPTER 7

RESULTS AND DISCUSSION

Semantic segmentation of road region detection is a critical task in the field of self-driving cars as it allows the car to navigate safely and efficiently on the road. In this task, the goal is to segment the image into different regions corresponding to different objects, such as road, buildings, pedestrians, and vehicles. In this discussion, we will explore the results of semantic segmentation for road region detection and their implications for self-driving cars. One of the key metrics used to evaluate the performance of semantic segmentation algorithms is the Intersection over Union (IoU) metric. IoU measures the overlap between the predicted and ground truth segmentation masks. Higher IoU values indicate better segmentation accuracy. Another important metric is the pixel accuracy, which measures the percentage of pixels that are correctly classified. Recent advances in deep learning have led to significant improvements in semantic segmentation accuracy for road region detection. State-of-the-art methods such as Deep Lab, PSPNet, and U-Net have achieved IoU scores of over 90% on benchmark datasets such as Cityscapes and KITTI. These methods use convolutional neural networks (CNNs) with skip connections, atrous convolutions, and pyramid pooling to capture both local and global context information. They also use data augmentation techniques such as random scaling, cropping, and flipping to improve the generalization performance of the models. In addition to accuracy, real-time performance is also critical for self-driving cars. Most state-of-the-art semantic segmentation methods use deep CNNs, which are computationally expensive and can cause delays in real-time applications. To address this issue, researchers have proposed various optimization techniques such as network pruning, quantization, and low-rank approximation. These techniques reduce the number of parameters and operations in the CNNs while maintaining high accuracy, thus enabling real-time inference on embedded systems. Despite the significant progress in semantic segmentation for road region detection, there are still some challenges that need to be

addressed. One of the main challenges is the generalization of the models to different weather and lighting conditions. For example, snow, rain, and fog can significantly reduce the visibility of the road, making it difficult for the models to accurately detect the road region. Another challenge is the robustness of the models to occlusions and dynamic objects such as pedestrians and vehicles. These objects can occlude the road region and cause the models to misclassify the pixels.

CHAPTER 8

CONCLUSION

The self-driving segmentation car detection algorithm is thoroughly examined in this project. A strategy in view of conventional picture handling and oneself driving segmentation location technique in light of profound learning are dissected and contrasted with take care of the issue of path identification under complex circumstances like shadows and hindrances. In the inverse perspective transformation, the utilization of a fixed transformation matrix will result in errors when the ground changes, which will cause the vanishing point projected to infinity to move either upward or downward. In order to increase the efficiency of central computing, a customized training network based on semantic segmentation and a two-branch training network have been proposed. The combination of vehicle edge calculation and actual engineering can also improve the effect. A custom loss function was used to train a neural network that can dynamically predict the transformable matrix's parameter values. Second- or third-order polynomials are used to fit the transformed lane points. The model has excellent stickiness because the prediction is based on the input image and the network can adjust the projection parameters when the ground plane changes. The final tests show that the model in our paper performs better in situations like lane line degradation and insufficient lighting

8.1 FUTURE WORK

Performance in object detection had remained stagnant in recent years. Complex ensembles that combine multiple low-level image features with high-level context from object detectors and scene classifiers were the systems with the highest performance. A simple and scalable object detection algorithm that outperforms the best previous results on PASCAL VOC 2012 by 30% is presented in this project. This performance was made possible by two insights. The first is to use high-capacity convolutional neural networks to localize and segment objects using proposals from the bottom up. When labeled training data is scarce, the second is a paradigm for training

large CNNs. Pre-training the network with supervision for a data-rich auxiliary task (image classification) and then fine-tuning it for the data-deficient target task (detection), as we demonstrate, is highly effective. We speculate that the "supervised pre-training/domain-specific finetuning" paradigm will be extremely useful for a wide range of vision issues involving limited data. We conclude by noting how significant it is that we utilized both conventional computer vision and deep learning techniques (bottom-up region proposals and convolutional neural networks) to achieve these results.

8.2 RESEARCH ISSUES

Training the network still takes a lot of time because you have to classify 2000 region suggestions per image. Because it takes approximately 47 seconds for each test image, it cannot be implemented in real time. A fixed algorithm is the selective search algorithm. As a result, no learning is taking place at that point. Bad candidate region proposals could result from this.

8.3 IMPLEMENTATION ISSUES

There are two primary drawbacks to this approach: difficulty in dealing with the data's variance. Images with partially hidden objects are difficult for CNN to process. The network has trouble classifying titled or rotated images in the same way it has trouble with image classification.

REFERENCE

- [1] Ruturaj Kulkarni, Shruti Dhavalikar, Sonal Bangar, "Traffic Light Detection and Recognition for Self Driving Cars Using Deep Learning", Fourth International Conference on Computing Communication Control and Automation (ICCUBE), 2019
- [2] Ze Liu, Yingfeng Cai, Hai Wang, Long Chen, Hongbo Gao, Yunyi Jia, Yicheng Li, "Robust Target Recognition and Tracking of Self-Driving Cars With Radar and Camera Information Fusion Under Severe Weather Conditions", IEEE Transactions on Intelligent Transportation Systems (Volume: 23, Issue: 7), 2022.",
- [3] Ramin Nabati, Hairong Qi, "RRPN: Radar Region Proposal Network for Object Detection in Autonomous Vehicles", IEEE International Conference on Image Processing (ICIP), 2019."
- [4] Zhenchao Ouyang, Jianwei Niu, Yu Liu, Mohsen Guizani, "Deep CNN-Based Real-Time Traffic Light Detector for Self-Driving Vehicles", IEEE Transactions on Mobile Computing (Volume: 19, Issue: 2), 2020.
- [5] RikuyaTakehara, Tad Gonsalves, "Autonomous Car Parking System using Deep Reinforcement Learning", 2nd International Conference on Innovative and Creative Information Technology (ICITech), 2021.
- [6] P. Arbelaez, J. Pont-Tuset, J. T. Barron, F. Marques, and ´ J. Malik. Multiscale combinatorial grouping. In CVPR, 2014. 2
- [7] A. Arnab and P. H. Torr. Pixelwise instance segmentation with a dynamically instantiated network. In CVPR, 2017. 3, 9
- [8] M. Bai and R. Urtasun. Deep watershed transform for instance segmentation. In CVPR, 2017. 3, 9

- [9] S. Bell, C. L. Zitnick, K. Bala, and R. Girshick. Insideoutside net: Detecting objects in context with skip pooling and recurrent neural networks. In CVPR, 2016. 5
- [10] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multiperson 2d pose estimation using part affinity fields. In CVPR, 2017. 7, 8
- [11] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes dataset for semantic urban scene understanding. In CVPR, 2016. 9
- [12] J. Dai, K. He, Y. Li, S. Ren, and J. Sun. Instance-sensitive fully convolutional networks. In ECCV, 2016. 2
- [13] J. Dai, K. He, and J. Sun. Convolutional feature masking for joint object and stuff segmentation. In CVPR, 2015. 2
- [14] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. In CVPR, 2016. 2, 3, 4, 5, 6
- [15] J. Dai, Y. Li, K. He, and J. Sun. R-FCN: Object detection via region-based fully convolutional networks. In NIPS, 2016. 2
- [16] R. Girshick. Fast R-CNN. In ICCV, 2015. 1, 2, 3, 4, 6
- [17] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, 2014. 2, 3
- [18] R. Girshick, F. Iandola, T. Darrell, and J. Malik. Deformable part models are convolutional neural networks. In CVPR, 2015. 4
- [19] B. Hariharan, P. Arbelaez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In ECCV. 2014. 2
- [20] B. Hariharan, P. Arbelaez, R. Girshick, and J. Malik. Hyper- columns for object segmentation and fine-grained localization. In CVPR, 2015. 2

- [21] Z. Hayder, X. He, and M. Salzmann. Shape-aware instance segmentation. In CVPR, 2021.9
- [22] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In ECCV. 2021. 1, 2
- [23] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In CVPR, 2020. 2, 4, 7, 10
- [24] Y. -T. Chen, M. Garbade and J. Gall, "3D Semantic Scene Completion from a Single Depth Image Using Adversarial Training,"2019
- [25] L. -C. Chen, G. Papandreou, I. Kokkinos, K. Murphy and A. L. Yuille, "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs
- [26] L. A. de Oliveira Junior, H. R. Medeiros, D. Macêdo, C. Zanchettin, A. L. I. Oliveira and T. Ludermir, "SegNetRes-CRF: A Deep Convolutional Encoder-Decoder Architecture for Semantic Image Segmentation.

CHAPTER 10

APPENDICES

10.1 SOURCE CODE

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import pickle
from combined_thresh import combined_thresh
from perspective_transform import perspective_transform
from Line import Line
from line_fit import line_fit, tune_fit, final_viz, calc_curve, calc_vehicle_offset
from moviepy.editor import VideoFileClip

# Global variables (just to make the moviepy video annotation work) with
open('calibrate_camera.p', 'rb') as f:
    save_dict = pickle.load(f)
    mtx = save_dict['mtx']
    dist = save_dict['dist']
    window_size = 5 # how many frames for line smoothing
    left_line = Line(n=window_size)
    right_line = Line(n=window_size)
    detected = False # did the fast line fit detect the lines?
    left_curve, right_curve = 0., 0. # radius of curvature for left and right lanes
    left_lane_inds, right_lane_inds = None, None # for calculating curvature
```

```

# MoviePy video annotation will call this function def annotate_image(img_in):
    """
    Annotate the input image with lane line markings Returns annotated image
    """

    global mtx, dist, left_line, right_line, detected
    global left_curve, right_curve, left_lane_inds, right_lane_inds

    # Undistort, threshold, perspective transform
    undist = cv2.undistort(img_in, mtx, dist, None, mtx)
    img, abs_bin, mag_bin, dir_bin, hls_bin = combined_thresh(undist) binary_warped, binary_unwarped,
    m, m_inv = perspective_transform(img)

    # Perform polynomial fit if not detected:
    # Slow line fit
    ret = line_fit(binary_warped) left_fit =
    ret['left_fit'] #convolutional layer
    right_fit = ret['right_fit'] nonzerrox = ret['nonzerrox'] nonzeroy = ret['nonzeroy']
    left_lane_inds = ret['left_lane_inds'] right_lane_inds = ret['right_lane_inds']

    # Get moving average of line fit coefficients left_fit = left_line.add_fit(left_fit)
    right_fit = right_line.add_fit(right_fit) #fully

    layer

    # Calculate curvature
    left_curve, right_curve = calc_curve(left_lane_inds, right_lane_inds, nonzerrox,
    nonzeroy)

    detected = True # slow line fit always detects the line else: # implies detected

```

```
== True

# Fast line fit

left_fit = left_line.get_fit() right_fit = right_line.get_fit()

ret = tune_fit(binary_warped, left_fit, right_fit) left_fit = ret['left_fit']

right_fit = ret['right_fit'] nonzeroy = ret['nonzeroy'] nonzeroy = ret['nonzeroy']

left_lane_inds = ret['left_lane_inds']
```

```

right_lane_inds = ret['right_lane_inds'] #pooling
layer

# Only make updates if we detected lines in current frame if ret is not None:

left_fit = ret['left_fit'] right_fit = ret['right_fit'] nonzerrox = ret['nonzerrox'] nonzeroy =
ret['nonzeroy']

left_lane_inds = ret['left_lane_inds']
right_lane_inds = ret['right_lane_inds'] #relu layer

left_fit = left_line.add_fit(left_fit) right_fit = right_line.add_fit(right_fit)

    left_curve, right_curve = calc_curve(left_lane_inds, right_lane_inds, nonzerrox,
nonzeroy)
else:
detected = False

vehicle_offset = calc_vehicle_offset(undist, left_fit, right_fit) # Perform final visualization on
top of original undistorted image

result = final_viz(undist, left_fit, right_fit, m_inv, left_curve, right_curve,
vehicle_offset) return result

def annotate_video(input_file, output_file):
    """ Given input_file video, save annotated video to output_file """ video =
VideoFileClip(input_file)

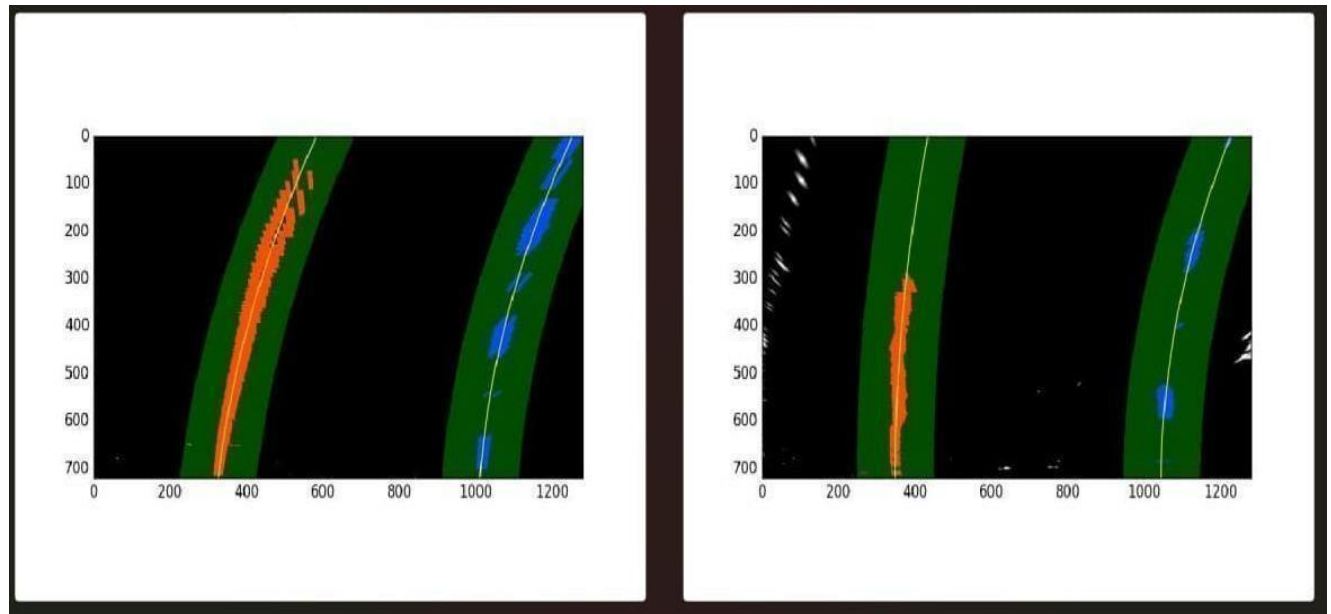
    annotated_video = video.fl_image(annotate_image) annotated_video.write_videofile(output_file,
audio=False)

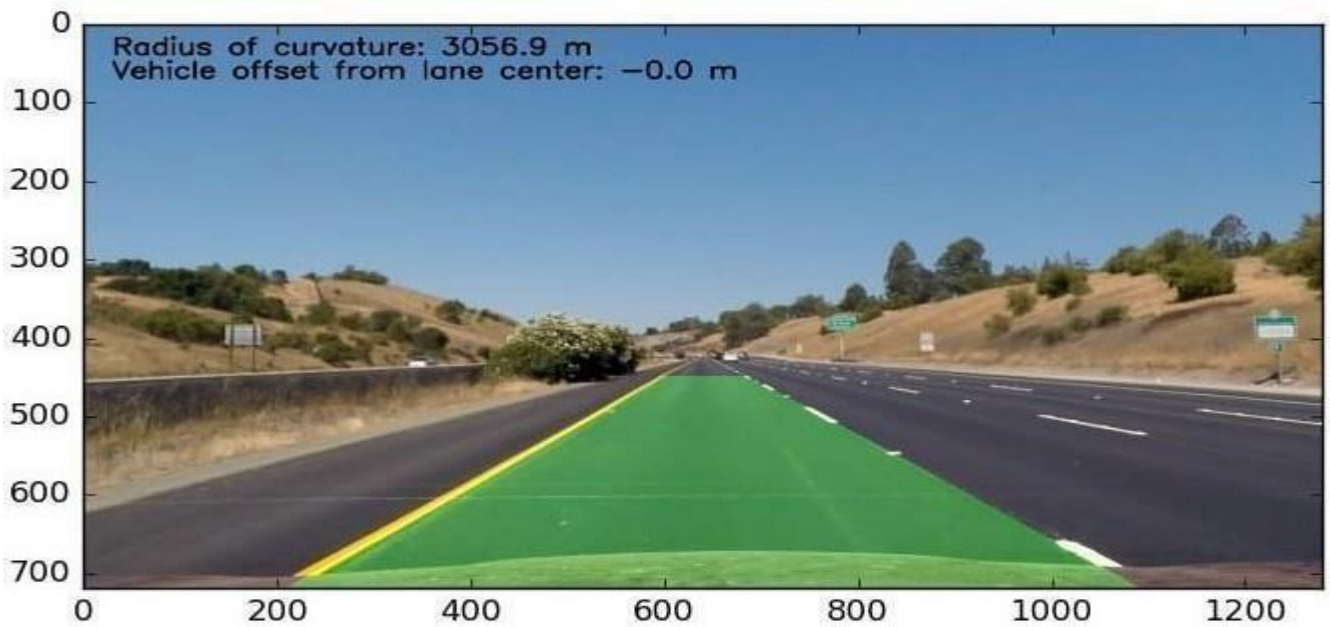
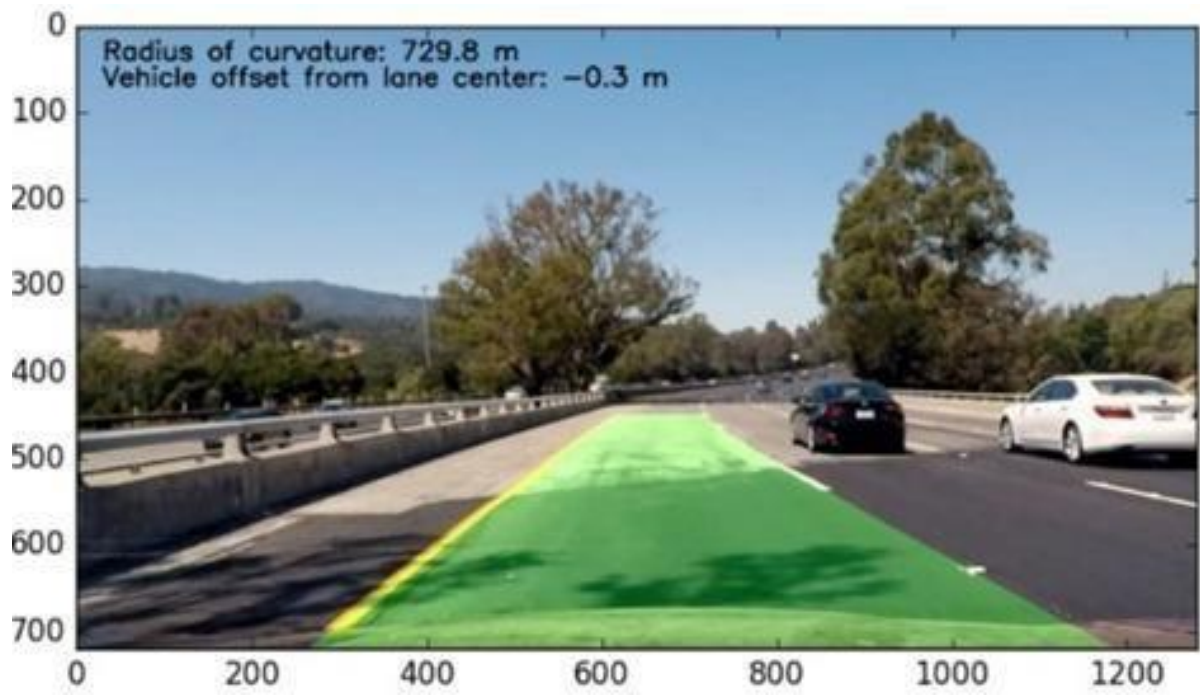
if __name__ == '__main__': #Annotate the video
    annotate_video('Input_video.mp4', 'out.mp4')

```

```
# Show example annotated image on screen for sanity check
img_file = 'test_images/test2.jpg'
img = mpimg.imread(img_file)
result = annotate_image(img)
result = annotate_image(img)
result = annotate_image(img)
plt.imshow(result)
plt.show()
```

10.2 SCREENSHOT





Detection of Road Regions and Surfaces for Autonomous Driving Using Semantic Segmentation Methods

Hemalatha V, Jeberson Retna Raj, Preethi S
Department of Information Technology, School of Computing
Sathyabama Institute of Science and Technology.
Chennai, India.
jebersonretnarajr@gmail.com

Abstract—The advancement of Artificial Intelligence has simplified complex problems, and it is now providing solutions for autonomous driving vehicles. Semantic segmentation of road images plays a vital role in this technology for understanding the environment of self-driving vehicles. However, existing solutions face challenges in processing data from multiple camera sensors, causing inaccurate results. To address these challenges, Convolutional neural networks and deep learning algorithms are utilized to improve accuracy. This paper focuses on using the R-CNN algorithm to segment road features accurately to enable the self-driving car to drive on a precise path. The proposed system processes camera-recorded videos using deep learning algorithms to provide pixel-labeled images for accurate scene interpretation. Semantic segmentation functions can identify objects, lanes, and barriers in the scene. The proposed system is tested using a data set, and the results show promising outcomes.

Keywords—DeepLearning, R-CNN algorithm, self driving cars

I INTRODUCTION

Advancement of technology offering solutions to the complex problems to a simple one. Automation in engineering and technology simplifies the human life which assures safety and security. Autonomous vehicle technology grows enormous nowadays for its demand in the market. In the field of constructing completely autonomous vehicles, many research laboratories and academic institutions are currently striving to achieve the best results. Autonomous self-driving vehicles work on the principles of artificial intelligence (AI), machine learning, and computer vision. These vehicles use a combination of sensors, cameras, and algorithms to perceive and interpret their surroundings, make decisions, and execute actions without human intervention.

The AI and machine learning algorithms that power autonomous vehicles rely on large amounts of data collected from various sources, such as LiDAR (Light Detection and Ranging) sensors, radar, and GPS. This data is used to build a detailed map of the vehicle's environment and to train the vehicle's algorithms to recognize and respond to different scenarios.

Computer vision technology is used to help autonomous vehicles perceive their surroundings in real-time. Cameras

mounted on the vehicle capture images of the road, other vehicles, and pedestrians, which are then analyzed by the vehicle's AI algorithms to identify and track objects and obstacles. Overall, the principles behind autonomous self-driving vehicles involve complex technologies and algorithms that allow them to navigate roads and make decisions in a way that is safe and efficient.

Segmentation of the road in a road image is a key component for an autonomous driving vehicle. Semantic segmentation is a method in which each pixel in a picture is assigned a preset label, enabling the recognition of objects such as automobiles, signs, or pedestrians. This method has a wide range of applications in fields such as facial recognition, medical imaging, and autonomous driving. Semantic segmentation provides situational understanding and a large amount of data required for scene recognition. However, due to the intricate relationships between the elements in each image and the interactions between frames, this task is challenging. Although deep learning has significantly improved the efficiency of this task, achieving real-time and accurate semantic segmentation is still an area of active research. Semantic segmentation enables better environmental comprehension, and autonomous vehicles rely on data from sensors to make decisions. Therefore, real-time and precise semantic segmentation is essential for the safe operation of autonomous vehicles. However, due to the complexity of the task, it is still difficult to perform reliable semantic segmentation in real-time. Further research aims to enhance its effectiveness and accuracy.

II LITERATURE SURVEY

In[1], semantic segmentation of road in a complex image is presented. This method employs with an autoencoder with semantic segmentation algorithm for detecting road that is trained on just typical road pictures. This features offers a feasible option for advancing self-driving systems.

Deepika et al proposed a vision-based navigation system for autonomous driving that can classify and detect obstacles on the road. The system relies on a camera mounted on the vehicle to capture images of the road, which are processed to identify obstacles. The image processing techniques used include color segmentation, edge detection, and blob analysis. The system employs SegNet encoder-decoder architecture and it is tested with NVIDIA's Jetson TK1[2].

In[3], a novel method for robust drivable road region detection for fixed-route autonomous vehicles using map-

fusion images is proposed. The approach involves the fusion of the high-definition map with the visual images captured by the vehicle's camera to improve the accuracy and robustness of road detection. The map-fusion approach is used to improve the accuracy of road detection by incorporating prior knowledge about the road network.

Viswanath et.al proposed OFFSEG a framework for semantic segmentation in off-road driving scenarios using a deep learning approach. The framework consists of a convolutional neural network (CNN) architecture that can effectively distinguish between different types of terrain, such as grass, rocks, sand, and water. The approach is applied in real-time, allowing for effective navigation in off-road environments [4].

In[5], a CNN-based semantic segmentation approach employs a fisheye camera pictures of urban traffic. The approach leverages with Efficient Residual Factorized CNN (ERFNet) architecture used to handle distorted fish-eye pictures, and a new dataset of distorted fish-eye images is developed for training and assessment. Chen et al proposes a semantic segmentation framework for urban scenes that can adapt to different lighting and weather conditions using a reality-oriented approach. The framework is based on a fully convolutional network (FCN) architecture that can accurately segment different objects in urban scenes, such as cars, buildings, and pedestrians [6]. Authors of [7] presented a framework for sequential semantic segmentation of road profiles for path and speed planning. The framework uses a combination of deep learning and rule-based approaches to segment the road profiles into different categories, such as straight, curve, intersection, and roundabout. One of the main advantages of this framework is its ability to provide accurate and timely information for path and speed planning, making it more effective for autonomous driving applications. Asgarian et al. proposes a deep learning-based framework for detecting drivable areas for autonomous driving. The framework is designed to process images in real-time and accurately detect drivable areas under various environmental conditions [8]. Shim et al proposes a lightweight semantic segmentation framework for road-surface damage recognition [9]. The framework uses a multi-scale learning approach to segment the road surface into different categories, such as cracks, potholes, and patches. One of the main advantages of this framework is its lightweight design, which allows for efficient processing of images on low-power devices.

Farag et. al proposes a real-time lane detection system for autonomous driving. The system uses computer vision techniques to detect and track lane markings on the road and provide feedback to the autonomous driving system[10]. In [11], Lee et.al proposes a deep learning-based semantic segmentation framework for road scenes. The framework uses a parallel complement network to perform semantic segmentation of road scenes in real-time. Hofmarcher et. al presents a deep learning-based semantic segmentation approach for autonomous driving. The proposed approach utilizes a fully convolutional neural network (FCN) to segment images into various classes, such as road, sidewalk, vegetation, and vehicles. The approach also incorporates a post-processing step to refine the segmentation results and remove false positives[12]. Authors of [13] proposes a new dataset for semantic segmentation of unstructured driving scenarios. The authors created a dataset with fine-grained

annotations, including 18 different object classes and 6 road classes, to improve the accuracy and robustness of the semantic segmentation models. The authors also propose an efficient segmentation model that utilizes a residual network with atrous spatial pyramid pooling (ASPP) and a decoder network for accurate and fast segmentation. Rateke et. al proposes a system for detecting and differentiating road surfaces and surface damages using a deep learning approach. The system is designed for autonomous vehicles to improve their ability to navigate safely and efficiently in various road conditions[14]. Shaban et. al presents a semantic terrain classification framework for off-road autonomous driving. The proposed approach uses a deep neural network to segment terrain images into different categories, such as sand, grass, water, and rocks. The network architecture is based on a convolutional neural network (CNN) and uses a ResNet-18 backbone to extract high-level features from the input image[15]. Authors of [16] presents a supervised classification approach for road detection, which involves training a machine learning algorithm using labeled examples of road and non-road pixels. The authors use a feature extraction method based on local binary patterns (LBP) to capture texture information in the image, and then apply a random forest classifier to classify each pixel as road or non-road.

Alkhorshid et al. (2016) proposed a method for road detection in images using supervised classification. They extracted features such as color, texture, and shape from the images and trained a Support Vector Machine (SVM) classifier to distinguish between road and non-road pixels. Their approach achieved a high accuracy of 96.2%, but the limitation of their work is that it requires manual feature extraction[17]. John et al. (2016) presented a method for fast road scene segmentation using deep learning and scene-based models. They used a Convolutional Neural Network (CNN) to learn features from the input image and trained a Conditional Random Field (CRF) model to refine the segmentation result. Their approach achieved high accuracy and fast processing speed, but the limitation of their work is that it requires a large amount of labeled data for training the CNN[18]. Asgarian et al. (2021) proposed a fast drivable area detection method for autonomous driving using deep learning. They used a Fully Convolutional Network (FCN) to perform semantic segmentation of the input image and then applied a post-processing step to detect the drivable area. Their approach achieved high accuracy and fast processing speed, but the limitation of their work is that it requires a large amount of labeled data for training the FCN[18]. Boudissa et al. proposed a method for semantic segmentation of traffic landmarks using classical computer vision and a U-Net model. The classical computer vision algorithms include Canny edge detection, Hough transform, and watershed segmentation. They achieved promising results on a dataset containing traffic landmarks, including traffic signs, traffic lights, and lane markings. A limitation of this approach is that it relies on handcrafted features and may not generalize well to complex scenes with different lighting conditions or occlusions[19]. Huang et al. presented a real-time road curb and lane detection system using LiDAR point clouds. They used a combination of a ground plane estimation algorithm, a road curb detection algorithm, and a lane detection algorithm. The system achieved high accuracy and real-time performance on a dataset of urban roads. However, the limitation of this approach is that it relies on LiDAR sensors,

which can be expensive and have limited range and resolution[20]. Wei et al. reviewed the use of mmWave radar and vision fusion for object detection in autonomous driving. They discussed the advantages and limitations of mmWave radar and vision sensors and proposed a fusion approach that can enhance the detection performance in challenging conditions such as low visibility, occlusions, and adverse weather conditions. A limitation of this approach is that mmWave radar has limited resolution and can produce many false alarms, which can lead to high computational overhead and may require additional processing steps to reduce false positives[21]. Rajaram et al. proposes a RefineNet architecture that can improve object detection performance in autonomous driving scenarios by refining and verifying the output of a base object detector. The RefineNet uses a multi-stage approach that learns to refine the object proposals at different scales and resolutions, leading to better localization and classification performance. The limitation of this approach is that it requires a pre-trained base object detector to function, which can be computationally expensive[22].

Teichmann et al. proposes a MultiNet architecture that can perform joint semantic reasoning for autonomous driving tasks. The MultiNet consists of multiple neural networks that specialize in different tasks such as object detection, semantic segmentation, and depth estimation. These networks work together to provide a more comprehensive understanding of the environment and improve overall performance. One limitation of this approach is that it requires multiple neural networks, which can be computationally expensive[23]. Petrovai and Nedeveschi's proposes a Semantic Camera system for 360-degree environment perception in automated urban driving. The system uses a combination of fisheye cameras and deep neural networks to detect and classify objects in the environment. The system is designed to be low-cost and can be easily integrated into existing vehicles. A limitation of this approach is that fisheye cameras may suffer from distortion and other imaging issues, which can affect object detection performance[24].

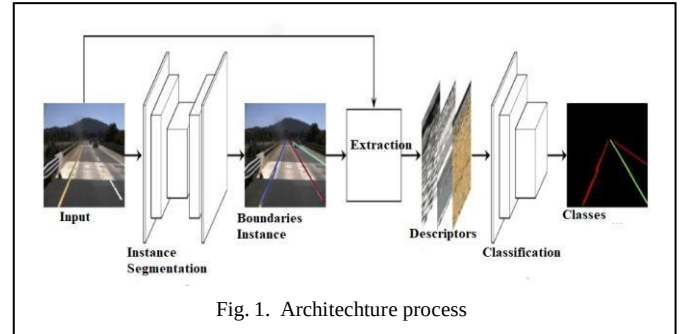
These 24 papers all focus on various aspects of detecting and recognizing obstacles or road conditions for autonomous driving using machine learning and computer vision techniques. Limitations of the papers include lack of real-world testing or validation, limited dataset size, lack of generalization to varying environments or conditions, and high computational complexity or resource requirements. Some papers focus only on specific types of road conditions or obstacles, limiting their applicability, while others do not address issues related to safety or risk management. Additionally, some papers may not be easily replicable or applicable to other research contexts, due to proprietary or non-public datasets or software.

III . PROPOSED ARCHITECTURE

R-CNN (Region-based Convolutional Neural Networks) is a popular algorithm used for object detection and semantic segmentation tasks in the field of self-driving cars. The algorithm works by first generating region proposals in an image using a selective search algorithm. These regions are then processed using a convolutional neural network (CNN) to extract features. The features are then fed into a set of support vector machines (SVMs) to classify the regions into object categories. In the case of semantic

segmentation, a fully convolutional network (FCN) is used to generate dense pixel-wise predictions for each object class. The R-CNN algorithm has been shown to achieve state-of-the-art performance on benchmark datasets for object detection and semantic segmentation tasks in the field of self-driving cars.

A. System architecture process



The system architecture for a self-driving car involves a series of steps. Firstly, the system acquires a video. Then, the video is converted into frames per second (fps) to make it easier to process. After the fps image conversion, feature extraction and segmentation take place. During this process, the essential elements of the image are processed, and the image is divided into groups to categorize them using a mask or image. Once the segmentation is complete, the final output is presented. This architecture is designed to ensure that only the relevant elements of the image are processed, which helps to improve the accuracy and efficiency of the system.

The following is the procedure for segmenting road features from a image frame.

- 1) Load pre-trained Convolutional Neural Network (CNN) weights
- 2) Set the image input size and the size of the Region of Interest (RoI)
- 3) Load the image and generate a set of region proposals using selective search
- 4) For each region proposal, do the following:
 - a. Crop and resize the image to the input size of the CNN
 - b. Feed the cropped image to the CNN and obtain the convolutional feature maps
 - c. Pass the feature maps through a Region-based Convolutional Neural Network (R-CNN) head to obtain region-wise features
 - d. Feed the region-wise features to a multi-layer perceptron (MLP) to obtain class scores and bounding box offsets
 - e. Apply non-maximum suppression (NMS) to filter out overlapping regions
- 5) Generate the final segmentation map by assigning each pixel to the class label of its corresponding region proposal

B. Input data acquisition system

The process of retrieving an image or video for additional image processing is described in the paragraph as image

acquisition. Since there is no processing available prior to image acquisition, it is usually the initial stage in the workflow.

C. Preprocessing

Image pre-processing is a crucial step that needs to be performed before using photos for model training and inference. It helps to increase or decrease the speed of pattern preparation. The video can be converted into frames per second, and the final image can be converted into grayscale during pre-processing. Some common image pre-processing techniques include color correction, reorientation, and scaling. Pre-processing can be considered as a pond-level module that can significantly affect the accuracy of the model's predictions.

D. Feature extraction

In the context of segmentation for a self-driving car, feature extraction is a crucial step that involves converting raw image data into numerical features that can be processed by machine learning algorithms. This process is performed by transforming the raw image data, which is typically in grayscale, into a set of numerical features that capture important information about the image. The main feature of interest, such as the region of interest, is extracted using this process. The Re-LU layer, which stands for Rectified Linear Unit layer, is used to perform segmentation and feature extraction. By performing feature extraction, the machine learning algorithm can better understand the important aspects of the image and make more accurate predictions about the environment in which the self-driving car is operating.

E. Segmentation

Semantic segmentation is a process that involves assigning a label to each pixel in an image. In the context of self-driving cars, semantic segmentation is used to identify objects such as roads, cars, pedestrians, and traffic signs. This is achieved through the use of image segmentation techniques, which divide the image into distinct regions, followed by feature extraction, which identifies the key characteristics of each region. The Re-LU layer is often used for both of these processes. Once the features have been extracted, they are used to classify each pixel in the image, resulting in a map that shows the location of different objects on the road. This map can then be used by the self-driving car to make decisions about how to navigate through the environment.

F. Classification

The classification process for a self-driving car involves grouping images or vectors according to predefined rules. This process uses spectral or texture functions to produce an effective classification rule. The key criteria set during this process are the radius of curvature and the distance to the car. The radius of curvature determines the final circle that is drawn during turning, while the distance to the car is an essential factor for detecting objects in the car's environment. The final step of the classification process is

carried out by a completely connected layer, which uses the extracted features from the previous stages to classify the input image into one of the predetermined categories.

IV

EXPERIMENTAL RESULT

We evaluate the R-CN mask with final instance segmentation strategies. Our version performs higher than preceding base variant fashions in all cases. Although it's far outside the scope of this work, we assume a lot of those enhancements to be proposed. Even in more difficult situations, the Mask R-CN gives satisfactory consequences. Our baseline Mask R-CNN and comparison to cover the remains of instances, assuming s from the urgent occasion to differentiate the test effects. R-CNN does no longer showcase such antiquities.



Fig. 2. Process of segmentaion

The process of segmentation for a self-driving car involves separating the lane path and car part. The first step of segmentation is the identification of the lane path, which is indicated in green color. This enables the car to understand its position within the lane and adjust its movements accordingly. The second segmentation is focused on car finding, which identifies the position of the vehicle and its offset from the center of the lane. This information is crucial for the car to maintain proper alignment and avoid collisions with other vehicles. The radius of curvature is a key parameter that is determined from the lane path segmentation. It provides information about the sharpness of the curve and is used to adjust the car's speed and steering angle accordingly. Overall, the segmentation process is critical for a self-driving car to navigate safely and effectively on the road.

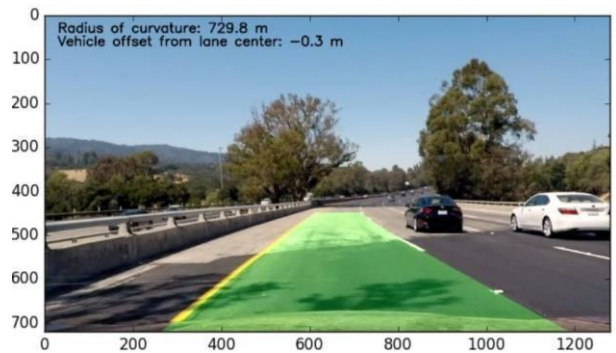


Fig. 3. Process of vehicle offset

The process of segmentation for a self-driving car involves separating the car portion from the lane path. This is achieved through two segments, with the first segment displaying the lane path in green and the second segment displaying the vehicle offset from the lane center. The radius of curvature is then used to display the lane path. By segmenting the image in this way, the self-driving car can more easily detect and navigate along the appropriate path, while also being able to accurately detect and position itself within that path. This is an important process in enabling the car to operate safely and effectively, and to ensure that it can navigate complex driving scenarios with ease.

A. Performance evaluation

accuracy, precision, and recall are three important evaluation metrics for a model's performance. Accuracy measures the proportion of correctly classified positive examples to the total number of positive examples, regardless of negative examples. In other words, accuracy indicates how often the model is correct when making a prediction.

Precision, on the other hand, measures the proportion of true positive examples to the total number of positive examples predicted by the model, including false positives. Precision measures the model's ability to correctly identify positive examples without wrongly labeling negative examples as positive.

Recall is the ratio of true positive examples to the total number of positive examples in the dataset, including false negatives. Recall measures the model's ability to identify all relevant positive examples in the dataset, without missing any. The formula to calculate recall is $TP / (TP + FN)$, where TP is the number of true positive examples, and FN is the number of false negative examples. A model with a recall score of 1 indicates that it has no false negatives, meaning it correctly identifies all positive examples in the dataset.

TABLE I. PERFORMANCE OF THE PROPOSED METHOD

Algorithm	Precision (%)	Recall(%)	Accuracy(%)
CNN	60	65	80
RCNN	80	87	92

Table 1 shows the performance comparison of RCNN with other algorithms. It shows that RCNN perform well with the test cases with increased accuracy.

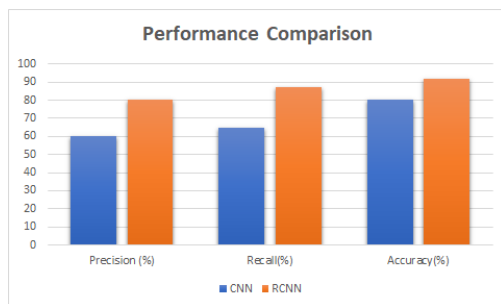


Fig.2 Performance comparison of RCNN with other methods

Figure 2 show the performance comparison of RCNN algorithm with other existing methods. The result show that the proposed algorithm outperforms well with the existing implementations

V

CONCLUSION

An auto-arc identification system for motors based on deep learning is presented in this research, with a focus on route recognition in intricate surroundings. To boost performance, a network with two branches and a specific learning network built on semantic segmentation were presented. Variable points were fitted with second or third order polynomials, and a bespoke loss function was utilise to forecast transformation matrix parameter values on the fly. The model performs effectively in extreme circumstances like lane exits and bad lighting because it adapts to network projection parameters.

REFERENCES

- [1] Ohgushi, Toshiaki, Kenji Horiguchi, and Masao Yamanaka. "Road obstacle detection method based on an autoencoder with semantic segmentation." Proceeding of the Asian Conference on Computer Vision. 2020.
- [2] Deepika, N., and VV Sajith Variyar. "Obstacle classification and detection for vision based navigation for autonomous driving." In 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 2092-2097. IEEE, 2017.
- [3] Cai, Yichao, Dachuan Li, Xiao Zhou, and Xingang Mou. "Robust drivable road region detection for fixed-route autonomous vehicles using map-fusion images." Sensors 18, no. 12 (2018): 4158.
- [4] Viswanath, Kasi, Kartikeya Singh, Peng Jiang, P. B. Sujit, and Srikanth Saripalli. "Offseg: A semantic segmentation framework for off-road driving." In 2021 IEEE 17th International Conference on Automation Science and Engineering (CASE), pp. 354-359. IEEE, 2021.
- [5] Sáez, Alvaro, Luis M. Bergasa, Eduardo Romeral, Elena López, Rafael Barea, and Rafael Sanz. "CNN-based fisheye image real-time semantic segmentation." In 2018 IEEE Intelligent Vehicles Symposium (IV), pp. 1039-1044. IEEE, 2018.
- [6] Chen, Yuhua, Wen Li, and Luc Van Gool. "Road: Reality oriented adaptation for semantic segmentation of urban scenes." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 7892-7901. 2018.
- [7] Cheng, Guo, and Jiang Yu Zheng. "Sequential Semantic Segmentation of Road Profiles for Path and Speed Planning." IEEE Transactions on Intelligent Transportation Systems 23, no. 12 (2022): 23869-23882.
- [8] Asgarian, Hojat, Abdollah Amirkhani, and Shahriar B. Shokouhi. "Fast drivable area detection for autonomous driving with deep learning." In 2021 5th International Conference on Pattern Recognition and Image Analysis (IPRIA), pp. 1-6. IEEE, 2021.
- [9] Shim, Seungbo, and Gye-Chun Cho. "Lightweight semantic segmentation for road-surface damage recognition based on multiscale learning." IEEE Access 8 (2020): 102680-102690.
- [10] Farag, Wael. "Real-time detection of road lane-lines for autonomous driving." Recent Advances in Computer Science and Communications (Formerly: Recent Patents on Computer Science) 13, no. 2 (2020): 265-274.
- [11] Lv, Qingxuan, Xin Sun, Changrui Chen, Junyu Dong, and Huiyu Zhou. "Parallel complement network for real-time semantic segmentation of road scenes." IEEE Transactions on Intelligent Transportation Systems 23, no. 5 (2021): 4432-4444.
- [12] Hofmarcher, Markus, Thomas Unterthiner, José Arjona Medina, Günter Klambauer, Sepp Hochreiter, and Bernhard Nessler. "Visual scene understanding for autonomous driving using semantic segmentation." Explainable AI: Interpreting, Explaining and Visualizing Deep Learning (2019): 285-296.
- [13] Metzger, Kai A., Peter Mortimer, and Hans-Joachim Wuensche. "A fine-grained dataset and its efficient semantic segmentation for unstructured driving scenarios." In 2020 25th International Conference on Pattern Recognition (ICPR), pp. 7892-7899. IEEE, 2021.

- [14] Rateke, Thiago, and Aldo Von Wangenheim. "Road surface detection and differentiation considering surface damages." *Autonomous Robots* 45 (2021): 299-312.
- [15] Shaban, Amirreza, Xiangyun Meng, JoonHo Lee, Byron Boots, and Dieter Fox. "Semantic terrain classification for off-road autonomous driving." In *Conference on Robot Learning*, pp. 619-629. PMLR, 2022.
- [16] Alkhorshid, Yasamin, Kamelia Aryafar, Sven Bauer, and Gerd Wanielik. "Road detection through supervised classification." In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 806-809. IEEE, 2016.
- [17] John, Vijay, Kiyosumi Kidono, Chunzhao Guo, Hossein Tehrani, Seiichi Mita, and Kasuhiza Ishimaru. "Fast road scene segmentation using deep learning and scene-based models." In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 3763-3768. IEEE, 2016.
- [18] Asgarian, Hojat, Abdollah Amirkhani, and Shahriar B. Shokouhi. "Fast drivable area detection for autonomous driving with deep learning." In *2021 5th International Conference on Pattern Recognition and Image Analysis (IPRIA)*, pp. 1-6. IEEE, 2021.
- [19] Boudissa, Mehieddine, Hiroharu Kawanaka, and Tetsushi Wakabayashi. "semantic segmentation of traffic landmarks using classical computer vision and U-Net model." In *Journal of Physics: Conference Series*, vol. 2319, no. 1, p. 012031. IOP Publishing, 2022.
- [20] Huang, Jing, Pallab K. Choudhury, Song Yin, and Lingyun Zhu. "Real-time road curb and lane detection for autonomous driving using LiDAR point clouds." *IEEE Access* 9 (2021): 144940-144951.
- [21] Wei, Zhiqing, Fengkai Zhang, Shuo Chang, Yangyang Liu, Huici Wu, and Zhiyong Feng. "Mmwave radar and vision fusion for object detection in autonomous driving: A review." *Sensors* 22, no. 7 (2022): 2542.
- [22] Rajaram, Rakesh Nattoji, Eshed Ohn-Bar, and Mohan Manubhai Trivedi. "Refinenet: Refining object detectors for autonomous driving." *IEEE Transactions on Intelligent Vehicles* 1, no. 4 (2016): 358-368.
- [23] Teichmann, Marvin, Michael Weber, Marius Zoellner, Roberto Cipolla, and Raquel Urtasun. "Multinet: Real-time joint semantic reasoning for autonomous driving." In *2018 IEEE intelligent vehicles symposium (IV)*, pp. 1013-1020. IEEE, 2018.
- [24] Petrovai, Andra, and Sergiu Nedevschi. "Semantic cameras for 360-degree environment perception in automated urban driving." *IEEE Transactions on Intelligent Transportation Systems* 23, no. 10 (2022): 17271-17283.