REAR END OBJECT DETECTION AND ALARM SYSTEM FOR INTELLIGENT TRANSPORTATION

Benila S

Assistant Professor, Department of computer Science and Engineering,SRM Valliammai Engineering College
SRM Nagar, Kattankulathur, Chengalpet- 603203, Tamil Nadu , India
benilas.cse@srmvalliammai.ac.in

Karan Kumar R

UG student, Department of computer Science and Engineering,SRM Valliammai Engineering College
SRM Nagar, Kattankulathur, Chengalpet- 603203, Tamil Nadu , India
rkarankumar2001@gmail.com

Abstract

With the rapid development of the economy, vehicles have become the primary mode of transportation in people's daily lives. Among the various types of car accidents, rear-end collisions are quite common. Installing a rear-facing camera on the back of a vehicle can provide valuable assistance to drivers, including collision warning systems. By incorporating rear-end detection, drivers no longer need to look behind them. This system can detect objects on the road when the car is traveling at speeds over 80 km/h on a highway. Once activated, the system pre-processes the camera image to identify objects within it. If another vehicle is less than ten feet away and traveling in the same lane, a beep will sound. This is achieved by determining the lane the vehicle is in, estimating the object's distance from the camera, and utilizing the YOLOv5 object detection algorithm. To address the issue of the YOLOv5 vehicle detection algorithm missing detections for small and dense objects in complicated situations, the YOLOv5 vehicle detection method has been developed. The third-order B-spline curve model and the canny edge detection method were employed to fit the lane lines. This method has strong flexibility and resilience, and can describe lane lines of various shapes. The distance can be approximated by considering the labeled region found in the video. An alarm will sound to alert the driver if the distance is less than 3 meters. This technology will eliminate the vehicle's rear blind spot, ensuring the driver's safety.

***Keywords*: vehicle detection, deep learning, safety, lane detection, distance estimation**

1. Introduction

Since the advent of industrialized transportation, there has been an increase in demand for vehicles. However, as the number of vehicles on the road has grown, there has been a corresponding rise in the number of fatal accidents. Between 2014 and 2022, there was a steady increase in fatalities, with the number rising from 3,931 deaths to 5,081 deaths, a 29% increase. Pedestrians, cyclists, and motorcyclists accounted for half of these fatalities.

"Smart transportation" is a term used to describe the integration of contemporary technology and effective management strategies within transportation systems. This integration encompasses various fundamental systems, including car navigation, traffic signal control systems, automatic number plate recognition systems, and speed cameras. These advancements are constantly evolving in response to the dynamic landscape of technology, offering the potential to enhance transportation in all its forms and improve its efficiency and effectiveness.

The potential impact of object detection technology is vast, and its implementation has the ability to transform multiple aspects of our world. One such area is road safety, where the detection of road objects plays a critical role. While on-vehicle sensors like radars and ultrasonic sensors are commonly used, cameras offer a cost-effective solution with rich information. Vehicle detection, in particular, is an essential aspect of intelligent transportation systems, providing support for higher-level tasks such as decision planning and behavior control.

In recent years, significant advancements in key technologies such as the Internet of Things, deep learning, cloud computing, and artificial intelligence have triggered a notable technological revolution. These advancements, facilitated by big data and internet connectivity, have had a profound impact on various industries. One area that has particularly gained attention is intelligent transportation systems, where computer vision-based applications have played a vital role [5]. The escalating number of vehicles on the roads has given rise to concerns regarding safety and traffic congestion [12]. As a result, there has been extensive research focused on vision-based approaches, which utilize onboard cameras mounted on the host vehicle, for the purpose of detecting and tracking surrounding vehicles. The accurate detection and analysis of vehicle statistics in highway monitoring video scenes are of utmost importance for intelligent traffic management and control on the highway [12].

Vehicle detection methods can be broadly categorized into traditional algorithms and deep learning-based algorithms. Traditional algorithms are known for their complex computations and high computational load, which often result in lower detection efficiency and an inability to meet real-time detection requirements. On the other hand, deep learning-based algorithms offer advantages such as faster processing, higher accuracy, and enhanced robustness.

Deep learning-based algorithms employ various techniques for candidate region extraction, including R-CNN [1], SPP-Net [11], Fast R-CNN [6], Faster R-CNN [4], Mask R-CNN [10], regression-based YOLO series algorithms [7-11], and SSD algorithms [12]. These methods enable precise detection by effectively identifying and localizing vehicles in the given images or video frames. Their integration of deep learning allows for improved performance in terms of accuracy and speed compared to traditional approaches.

Regenerate responseTo overcome the challenges of detecting dense targets with low accuracy, a new model called YOLOv5-GE has been proposed in this paper. This model combines the benefits of YOLO series algorithms with a novel feature extraction approach to improve detection accuracy while maintaining high efficiency. In addition, the study provides a technique for calculating the distance between the camera and the identified object and a lane detection mechanism to assist drivers.

2. Related Work

Numerous techniques have been proposed for vehicle tracking and detection using machine learning and deep learning. One notable approach is a two-part deep learning-based system [6] that combines the advantages of "blackening" photos instead of "cutting" them with the GOTURN and YOLOv2-tracker algorithms. Through testing, it has been shown that this framework offers higher tracking speeds.

Another method for long-distance vehicle dynamic detection and positioning [13] utilizes Gm-APD Lidar and LIDAR-YOLO. By enhancing the three-channel IIR lidar picture as input, this network provides more detailed information about objects in the image. Experimental results demonstrate the accurate localization of objects in three dimensions based on distance values.To address night-time vehicle detection and tracking, a multi-camera fusion framework [2] has been proposed. This approach involves grouping headlights and taillights in real traffic surveillance footage captured under various traffic densities and lighting conditions. The framework also includes a vehicle contour refinement technique to align the generated vehicle contours based on traffic dynamics.

For urban road lane detection, a real-time and robust algorithm [14] incorporates the YOLOv4-tiny neural network to filter vehicles and other targets on the road. It combines image thresholding, Sobel edge detection, and gradient calculation. While this research focuses on the detection of a single frame of road images, it explores both traditional methods and deep learning techniques such as CNN, FCN, and RNN. Future studies aim to develop a high-speed and efficient network that can be implemented in real-time. A practical and effective method [9] for multiple lane detection utilizes five complementary constraints in the Hough transform space to remove false alarm candidates. It employs a dynamic programming strategy to find the best solutions for multiple lanes, ensuring efficiency and robustness in the presence of interference.

Another approach [7] for multiple lanes considers temporal-spatial historical information and leverages slice images for lane marker detection, confidence map generation, and tracking. The method incorporates prior knowledge to generate clean binary images and employs an improved distance transform to calculate the confidence map. Results obtained in challenging night scenarios demonstrate the method's capability to achieve satisfactory outcomes with minimal post-processing. To estimate inter-vehicle distances accurately, a combination of vehicle 3D detection and ranging geometry modeling [12] is proposed. This method uses a vehicle-mounted monocular vision system to obtain the actual area and projected area of the target vehicle in the image. By applying ranging geometry principles, the absolute distance between vehicles can be recovered.

Furthermore, an intra-vehicular measurement method [5] is proposed using a single camera to detect the front car's license plate. This method includes license plate detection, digit segmentation, and distance estimation modules. Experimental results demonstrate improved performance compared to previous methods, particularly for shorter distances up to 10m.In a modified version of YOLOv3 [3], the loss function responsible for absolute distance estimation is adjusted, and the training data is modified accordingly. By utilizing the KITTI dataset for benchmarking, the researchers achieved promising results, with mean absolute distance error of 2.5m and mean relative error of 11%. These findings highlight the potential of integrating distance estimation into YOLO's functionality to achieve superior results.

Overall, the literature highlights the significance of rear-end vehicle detection in the context of smart transportation. Existing systems employ diverse detection and tracking algorithms, sensor types, and warning strategies. Future research should prioritize enhancing the accuracy and speed of these systems while addressing their limitations.

3. Proposed Methodology

3.1. *Architecture*

The layout described in Figure1, provides a general overview of the components involved in a rear end object detection and alarm system. The components included are,

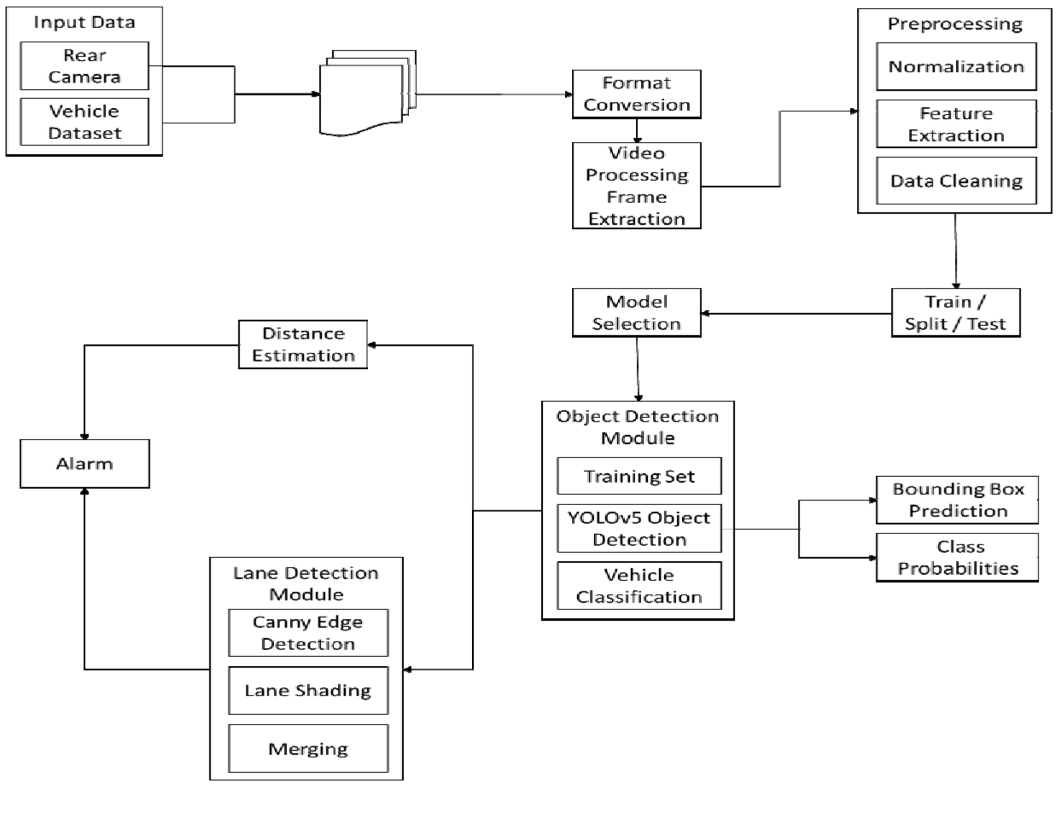

Fig. 1. System Architecture

Cameras: One or more cameras are used to capture video footage of the road and traffic behind the vehicle. These cameras act as the input source for the system.
Data Storage and Analysis: The system has the capability to store the captured video footage for further analysis and future use. The recorded data can be valuable for evaluating system performance, conducting post-incident analysis, or refining the algorithms used in the object detection and distance estimation modules.

Preprocessing Module: The preprocessing module plays a crucial role in preparing data for analysis and machine learning tasks. It involves several important steps, including normalization, feature extraction, and data cleaning.

Object Detection Module: The video footage captured by the cameras is processed by an image processing module. This module utilizes deep learning algorithms like YOLO (You Only Look Once) to detect and track objects, such as other vehicles or obstacles, in the footage. The object detection module analyzes the video frames in real-time to identify potential hazards.

Distance Estimation Module: Once the objects are detected and tracked, a distance estimation module comes into play. It uses the size or area of the detected objects in the video frames to estimate their distance from the vehicle. This information is crucial in determining if an object is dangerously close and requires an alert to the driver.

Lane Detection Module: The system incorporates a lane detection module that utilizes the Canny edge detection algorithm to identify and track the lane in which the vehicle is traveling. This information helps in assessing the relative position of detected objects with respect to the vehicle's lane.

Alerting System: When an object is detected at a dangerously close distance, the system triggers an alert to the driver. This can be done through various means such as sounding an alarm, displaying a visual warning, or providing haptic feedback. The alerting system aims to notify the driver of potential collisions or hazards to ensure timely response and avoidance.

User Interface: The system can incorporate a user interface that allows the driver to monitor the system's status and adjust settings. This user interface can be implemented as a dashboard display within the vehicle or as a dedicated device/interface accessible to the driver.

3.2. *Rear End Object Detection*

3.2.1 Data Set & Preprocessing

The dataset obtained from Kaggle, contains thousands of images of cars in various views. Working with a dataset of car images, collected from different angles and views, can be a common task in computer vision and object detection. By training an algorithm on this dataset, enables it to recognize and classify car images accurately. Techniques such as Computer vision Convolutional Neural Networks (CNNs), which have demonstrated significant success in image recognition tasks.

By training a CNN on car image dataset, the algorithm can learn to extract relevant features and patterns from the images, enabling it to detect and classify car images accurately. The algorithm's performance can be evaluated based on metrics such as accuracy, precision, recall, and F1 score. Data preprocessing and the splitting of the dataset for training and testing purposes are important for acuurate data analysis. The steps involved in preprocessing the data set are given below.

Normalization: Normalizing the values of the dataset involves transforming the numeric columns into a common scale. This step ensures that the values across different features are on a similar scale, preventing one feature from dominating the learning process due to its larger value range. Common normalization techniques include Min-Max scaling or Standardization.

Data Cleaning: Data cleaning involves removing unnecessary or irrelevant information from the dataset. This step helps in reducing noise and improving the efficiency of the training process. Unwanted columns or features that do not contribute to the task at hand, such as metadata or irrelevant labels, are typically removed during this process.

Dataset Splitting: The dataset was split into two subsets: a training set and a test set. The training set is used to train the model, while the test set is used to evaluate its performance. In this case, 70% (700 images) of the dataset was used for training, and the remaining 30% (300 images) were used for testing.

3.2.2 Model Building

The next stage is to develop the model after the data is prepared. Given that the input is an image, the Python model should also be a YOLO model

Model Development: It uses the YOLO model architecture in Python to build the model. YOLO consists of a convolutional neural network (CNN) with additional layers for bounding box predictions and class probabilities. The exact structure and configuration of the YOLO model can vary based on the specific implementation and requirements.

Activation Function: Apply the Rectified Linear Activation Function (ReLU) to the neurons in the model. ReLU is commonly used in deep learning models as an activation function to introduce non-linearity and improve the model's ability to learn complex patterns.

Model Compilation: After defining the model architecture, compile the model. Compilation involves specifying the optimizer, loss function, and metrics to be used during training. The optimizer determines how the model's weights are updated, the loss function quantifies the model's performance during training, and the metrics evaluate the model's performance during training and evaluation.

Training the Model: Train the compiled YOLO model using the prepared training dataset. The weights of the model are adjusted iteratively by feeding the training data through the model and backpropagating the gradients to minimize the loss.

Testing the Model: Once the model is trained successfully, the next step is to evaluate its performance on the test dataset. Feed the test data through the trained model and analyze the model's predictions, including its ability to detect and classify vehicles accurately.

3.2.3. Object detection

YOLO (You Only Look Once) is a real-time object detection system that uses a deep neural network to detect objects in an image. YOLO works by dividing the input image into a grid and predicting bounding boxes and class probabilities for each cell in the grid. The bounding boxes represent the predicted location of an object in the image, and the class probabilities represent the predicted class of the object.

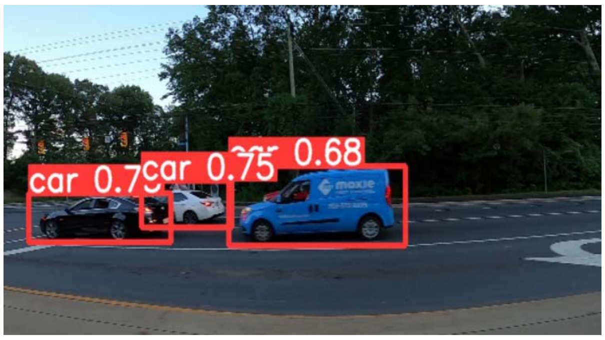

Fig . 2. Detected Classes

Dividing an image into a grid of cells is a key step in many object detection systems. Here's how an image is typically divided into a grid of cells:

- Define the grid size: The size of the grid can be determined by the size of the input image and the desired size of the output predictions.

- Divide the image into cells: The image is then divided into a fixed number of cells based on the grid size. Each cell is typically square in shape and is assigned a unique grid cell index.

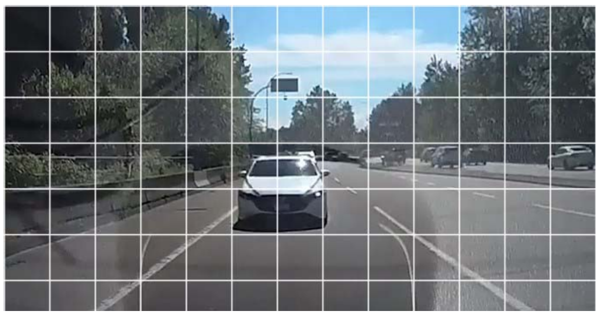

Fig . 3. Dividing into Grids

Objects in the image are associated with the grid cell that contains the center of the object. This is typically done by comparing the center point of the object with the boundaries of the grid cells. For each grid cell, the system predicts a fixed number of bounding boxes and their corresponding class probabilities. These predictions are made relative to the boundaries of the grid cell, allowing the system to make spatially accurate predictions at different scales.

Each bounding box is represented by four coordinates (x, y, w, h), where (x, y) represents the center of the box, and (w, h) represents the width and height of the box. For each cell in the grid, the predicted bounding boxes are adjusted based on the coordinates of the cell. This means that the predicted boxes are relative to the cell they are in. The class probabilities for each box are also predicted for each cell. The bounding boxes and class probabilities are combined into a single tensor and the boundary boxes are drawn.

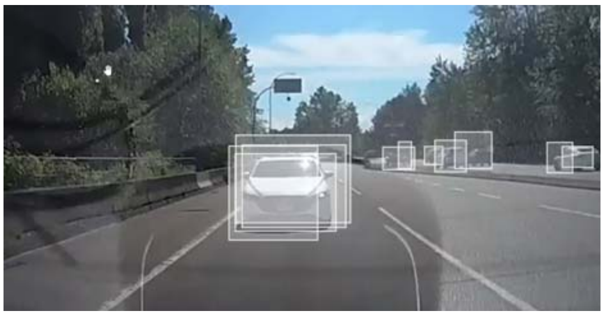

Fig. 4. Bounding Boxes

Pseudocode for object detection

Input:

I: The input image for object detection

G: Grid size

N: Number of bounding boxes

Output:

S: Coordinate set S{x, y, w, h}, where (x, y) represents the center of the box, and (w, h) represents the width and height of the box

Algorithm:

```
DivideImageIntoGrid(I, G)
For each object 'i' in I:
    CalculateObjectCenter(i)
    FindGridCellForObject(center)
    AssignObjectToGridCell(object, cell)
For each grid cell in Grid:
    PredictBoundingBoxes(gridCell)
    PredictClassProbabilities(gridCell)
For each cell in Grid:
    For each predicted bounding box in cell:
        AdjustBoundingBoxCoordinates(boundingBox, cell)
DrawBoundaryBoxesWithLabels()
Output the image with drawn boundary boxes
```

The IoU is calculated as follows:

IoU = (Area of Intersection) / (Area of Union)

In the context of bounding boxes, the area of intersection refers to the region where the predicted bounding box and the ground truth bounding box overlap. On the other hand, the area of union represents the total combined area covered by both bounding boxes

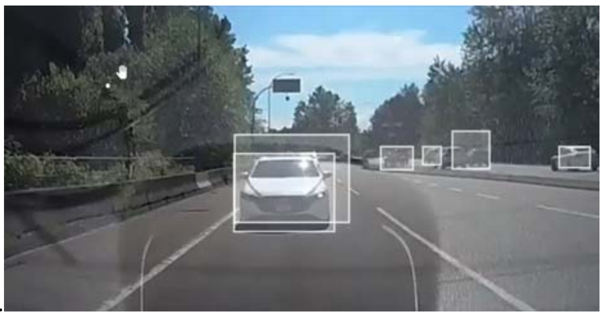
.

Fig. 5. After IOU

During the object detection process, an algorithm may generate multiple bounding box predictions that partially or fully cover the same object. Non-maximum suppression (NMS) is used to filter out these redundant predictions and select only the most accurate and confident bounding boxes.

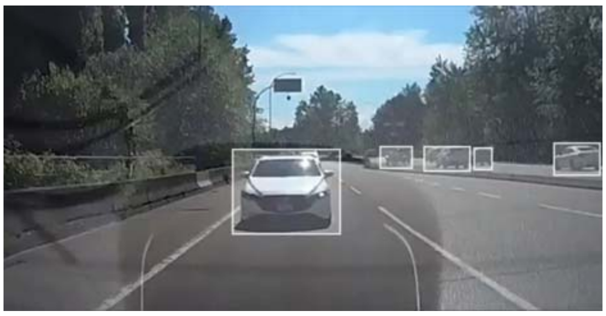

Fig. 6. After NMS

After boundary boxes the Intersection over Union (IoU) is calculated. The IoU is calculated as the ratio of the area of intersection between the two bounding boxes to the area of their union.

3.3 Lane Segmentation

The output from the object detection model is obtained and will be used for the next phase, which is lane detection. The process of lane detection is described below.

Lens Distortion Correction:
Lens distortion is a common issue in photography and computer vision caused by the characteristics of the camera lens.Camera calibration is performed using images of a known pattern (e.g., a checkerboard) to estimate the camera lens parameters.These parameters are then used to correct the lens distortion in subsequent images taken with the same camera. The lane segmentation includes

Canny Edge Detection:
Canny edge detection is a popular algorithm used to detect edges in an image.

1. Smoothing the image:
   - Apply a Gaussian filter to the input image to reduce noise and eliminate small details.
   - Let G(x, y) denote the smoothed image obtained by convolving the input image I(x, y) with a Gaussian kernel G(x, y; σ).
2. Finding image gradients:
   - Apply the Sobel operator to calculate the gradient magnitude and direction in the x and y directions.
   - Compute the gradient magnitude M(x, y) as $\sqrt{(Gx^2(x, y) + Gy^2(x, y))}$, where Gx and Gy are the gradients in the x and y directions, respectively.
   - Calculate the gradient angle θ(x, y) as arctan(Gy(x, y) / Gx(x, y)).
3. Non-maximum suppression:
   - For each pixel (x, y) in the gradient magnitude image M(x, y):
     - Determine the direction of the gradient angle θ(x, y).
     - Compare the pixel's gradient magnitude M(x, y) with the gradient magnitudes of the two neighboring pixels along the gradient direction.

        - Keep the pixel's gradient magnitude value only if it is the local maximum along the gradient direction; otherwise, set it to zero.

4. Thresholding:
    - Define a high threshold T_high and a low threshold T_low.
    - For each pixel (x, y) in the non-maximum suppression image:
        - If M(x, y) > T_high, mark it as a strong edge.
        - If M(x, y) < T_low, discard it as a weak edge.
        - If T_low ≤ M(x, y) ≤ T_high, consider it a weak edge only if it is connected to a strong edge.
5. Edge tracking by hysteresis:
    - Starting from the strong edges, trace along the connected weak edges to form longer edges.
    - Connect weak edges to strong edges if they are adjacent and form continuous paths.
    - Mark the connected weak edges as strong edges.
6. Output the detected edges:
    - Generate an output image or data structure that represents the detected edges.
    - The output can be a binary image where edge pixels are set to white, while non-edge pixels are set to black.

Highlighting Contrast Region:
The contrast region of the image, including the detected lane edges, is highlighted. This step enhances the visibility of the lane for the driver during driving.

Lane Shading:
To further aid the driver's recognition of the lane, the detected lane is shaded. By shading the lane, it becomes easier for the driver to quickly identify and follow the lane while driving.

Merging Layers:
The original layer of the image, the detected lane edges, and the shaded lane layer are merged together. This process combines the different layers into a single layer, resulting in the final representation of the lane detection.The output from the object detection model is fetched. These will be used for lane detection which is the next phase. The process of detecting the lane is discussed in below.

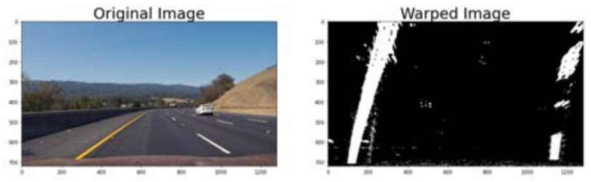

Fig. 8. Detected Edges

The contrast region of the image is highlighted. From the Figure 5.10 we can see that already the lanes have been detected. In order to make it easier for the driver while driving we can shade the lane. This will make easier for the driver to recognize the lane faster while driving. The original layer of the image, the detected edge of the lane and the shaded layer of the lane are merged together in order to make it as a single layer.

Algorithm:

```
# Lens Distortion Correction
perform_lens_distortion_correction(input_image)

# Canny Edge Detection
smoothed_image = apply_gaussian_filter(input_image)
gradient_x,gradient_y=calculate_gradients(smoothed_image)
magnitude,angle=calculate_magnitude_and_angle(gradient_x, gradient_y)
suppressed_edges=apply_non_maximum_suppression(magnitude, angle)
strong_edges = apply_thresholding(suppressed_edges, threshold)

# Highlight Contrast Region
contrast_region=highlight_contrast_region(input_image, strong_edges)
```

```
# Lane Shading
shaded_lane = shade_lane(contrast_region)

# Merge Layers
final_image= merge_layers(input_image, strong_edges, shaded_lane)

# Output the final image
output_image(final_image)
```

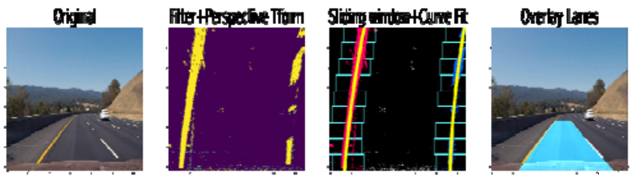

Fig. 9. Overlapping Layers

3.4 Distance Estimation

Distance estimation is an important aspect of rear-end object detection and alarm systems for smart transportation. It helps to determine the distance between the vehicle and the object in front of it, which is essential for detecting potential collisions and triggering an alarm.
The annotated data which will be created at the time of detecting the object is retrieved here. The annotated data consists of data such as the frame id, x co-ordinate, y co-ordinate, height and width of the bounding box.
The area of the bounding box is calculated using the Shoelace Theorem.

$$A = 0.5 * \begin{vmatrix} x1\ x2 \ldots xn & x1 \\ y1\ y2 \ldots yn & y1 \end{vmatrix} \tag{1}$$

where (x1, y1), (x2, y2), ..., (xn, yn) are the coordinates of the vertices of the polygon in order.

The distance estimated between the camera and the vehicle should be displayed along with the class of the vehicle. To do this the annotated data created at the time of object detection is again retrieved and few changes are made to the data. The value of the x co-ordinate is added by three pixels so that the cursor would come down without changing its horizontal position. Now the distance can be displayed on the screen and there will no overlapping of the text.

It alerts the driver when the distance between their vehicle and the detected object is less than a certain threshold. One way to implement this is to play an audio alert, such as a beep or a spoken warning, when the estimated distance falls below a certain value.
.

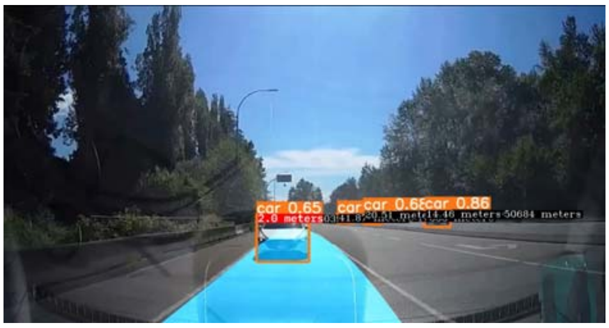

Fig. 10. Estimated Distance

3.5 Video Processing

Frame rate (expressed in frames per second or FPS) is the frequency (rate) at which consecutive images (frames) are captured or displayed. Frame rate, also known as frame frequency is the frequency at which an imaging device produces unique consecutive images called frames. Frame rate conversion is observed as a necessity in our system.

Images can be created by extracting frames from a video. These images are given as input to the model for the purpose of object detection. The model will detect the vehicles present in those images and save a copy of another image with labels of what vehicles are present which has been detected. The processing steps are given below.

(i) FrameRateModule:
- **calculateFrameRate(video)**: Calculates the frame rate of a given video.
- **performFrameRateUpconversion(video, targetFrameRate)**: Performs frame rate up-conversion by synthesizing intermediate frames.

(ii) ImageProcessingModule:
- **extractFramesFromVideo(video)**: Extracts frames from a video and returns a list of individual frames.
- **performObjectDetectionOnFrames(frames)**: Applies an object detection model to the frames and returns a list of labeled images.

(iii) FileHandlingModule:
- **fetchDetectedImages(pattern)**: Retrieves detected images that match a specific pattern and returns them as a list.
- **convertImagesToVideo(images, outputVideo)**: Converts a list of images into a video file and saves it to the specified location.
- **cleanUpMemory(images)**: Deletes the images from memory to free up resources.

4. Result & Discussion

In this study, we evaluated the performance of our model using low computational complexity theoretical indicators, specifically the number of billion floating-point operations (BFLOP), as a measure of the model's size. Additionally, we used the F1 score and average precision (AP) to assess the performance of the model trained by the loss function.

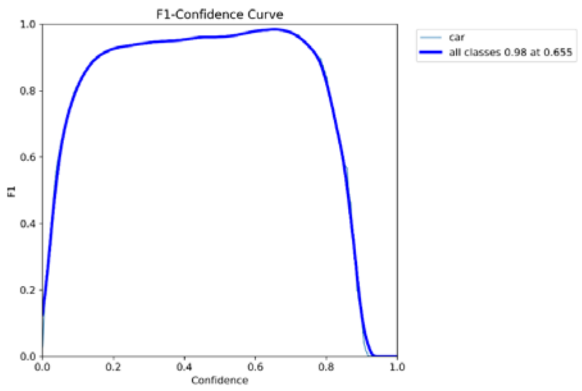

Fig. 11. F1 Vs Confidence

The F1 score and the confidence curve are two evaluation metrics commonly used in machine learning. The F1 score is a measure of the model's accuracy that takes into account both precision and recall. It ranges from 0 to 1, with higher values indicating better performance.

The F1 score is calculated using the formula:

$$F1 = 2 * \frac{Precision * Recall}{(Precision + Recall)} \quad (2)$$

On the other hand, the confidence curve is a plot of the model's accuracy as a function of confidence. It shows the percentage of correct predictions at different levels of confidence.

The confidence curve can be calculated using the formula:

$$Confidence\ Curve = \frac{Correct\ Predictions}{Total\ Predictions} * 100 \quad (3)$$

The F1 score and the confidence curve can be used together to evaluate the performance of a machine learning model. By comparing the F1 score and the confidence curve, we can gain a better understanding of how the model is performing and identify areas where it may need improvement. In general, a high F1 score and a smooth confidence curve are indications of a well-performing model.

Additionally, we assessed the model's performance using average precision (AP). Average precision is a measure of the precision-recall trade-off for a given set of predictions. It is calculated by computing the area under the precision-recall curve. The formula for average precision is:

$$Average\ Precision = \sum (Precision(i) * \Delta Recall\ (i)) \quad (4)$$

These evaluation metrics, including the F1 score, confidence curve, and average precision, help us quantify and analyze the performance of our model. They provide valuable insights into the accuracy, precision, recall, and reliability of the model's predictions.

By considering these metrics, we can make informed decisions to enhance the model's performance and optimize it for intelligent transportation applications. The F1 score and the confidence curve are two evaluation metrics commonly used in machine learning. The F1 score is a measure of the model's accuracy that takes into account both precision and recall. It ranges from 0 to 1, with higher values indicating better performance.

5. Conclusion and Future Scope

Our study focused on the development and evaluation of a rear-end object detection and alarm system for smart transportation. We presented the algorithmic approach and discussed its implementation in capturing consecutive frames, object detection, and video conversion. Through our analysis and evaluation, we have obtained valuable insights into the model's performance. In this system, we use YOLOv5 deep learning model to detect objects in real-time from a camera feed. We also apply various techniques such as canny edge detection for lane detection and estimating the distance between the camera and the detected object.

However, there are several areas where further improvement and future research can be explored. Firstly, the enhancement of the model's real-time capabilities would significantly benefit the system. By reducing processing and response times, we can improve the system's ability to detect and react to rear-end objects swiftly, thereby minimizing the risk of accidents. Moreover, collaborating with transportation authorities and industry stakeholders can facilitate the integration of the rear-end object detection and alarm system into existing transportation infrastructure. This would pave the way for its widespread adoption and contribute to making smart transportation safer and more efficient.

Funding

No funding is provided for the preparation of manuscript.

Conflicts of interest: "The authors have no conflicts of interest to declare".

References

[1] Chattopadhyay, D., Rasheed, S., Yan, L., Lopez, A. A., Farmer, J., & Brown, D. E. (2020). Machine Learning for Real-Time Vehicle Detection in All-Electronic Tolling System. Systems and Information Engineering Design Symposium (SIEDS) pp. 1-6.
[2] Chen, S., Huang, L., Chen, H., & Bai, J. (2022). Multi-Lane Detection and Tracking Using Temporal-Spatial Model and Particle Filtering. IEEE Transactions on Intelligent Transportation Systems, 23(3), pp. 2227-2245.
[3] Ding, Y., et al. (2022). Long-Distance Vehicle Dynamic Detection and Positioning Based on Gm-APD Lidar and LIDAR-YOLO. IEEE Sensors Journal, 22(17), pp. 17113-17125.
[4] Li, J. Y., Song, H. S., Zhang, Z. Y., et al. (2022). Multi-Object Vehicle Tracking and Trajectory Optimization Based on Video. In 2022 2nd International Conference on Physics. pp. 328-129.
[5] Luo, S., Zhang, X., Hu, J., & Xu, J. (2021). Multiple Lane Detection via Combining Complementary Structural Constraints. IEEE Transactions on Intelligent Transportation Systems, 22(12), pp. 7597-7606.
[6] Rong, Y., Jiang, J., Mutahira, H., & Muhammad, M. S. (2022). Monocular Intra-vehicular Distance Estimation using Front Vehicle License Plate. In 2022 16th International Conference on Ubiquitous Information Management and Communication (IMCOM) pp. 1-6.
[7] Song, H., Liang, H., Li, H., et al. (2019). Vision-based vehicle detection and counting system using deep learning in highway scenes. European Transportation Research Review, 11, 51.
[8] Vajgl, M., Hurtik, P., & Nejezchleba, T. (2022). Dist-YOLO: Fast Object Detection with Distance Estimation. Applied Sciences, 12, 1354.
[9] Wang, Y., et al. (2022). Lane Detection Based on Two-Stage Noise Features Filtering and Clustering. IEEE Sensors Journal, 22(15), pp. 15526-15536.
[10] Zakaria, N. J., Shapiai, M. I., Ghani, R. A., Yassin, M. N. M., Ibrahim, M. Z., & Wahid, N. (2023). Lane Detection in Autonomous Vehicles: A Systematic Review. IEEE Access, 11, pp. 3729-3765.
[11] Zhang, X., Story, B., & Rajan, D. (2022). Night Time Vehicle Detection and Tracking by Fusing Vehicle Parts From Multiple Cameras. IEEE Transactions on Intelligent Transportation Systems, 23(7), pp. 8136-8156.

[12] Zhe, T., Huang, L., Wu, Q., Zhang, J., Pei, C., & Li, L. (2020). Inter-Vehicle Distance Estimation Method Based on Monocular Vision Using 3D Detection. IEEE Transactions on Vehicular Technology, 69(5), 4907-4919.
[13] Zhou, S., Zhao, Y., & Guo, D. (2022). YOLOv5-GE Vehicle Detection Algorithm Integrating Global Attention Mechanism. In 2022 3rd International Conference on Information Science, Parallel and Distributed Systems (ISPDS) pp. 439-444.
[14] Zhou, Y., Zhou, J., & Liao, F. (2020). Research on Vehicle Tracking Algorithm Based on Deep Learning. Journal of Physics: Conference Series, 1621.

Authors Profile

Benila S, received her PhD in 2022 from Anna University, Chennai, India. She completed M.E in Network Engineering in 2008 and B.Tech, Information Technology in 2006 from Anna University. She is working as an Assistant Professor in the Department of Computer Science and Engineering at SRM Valliammai Engineering College, an autonomous institution affiliated to Anna University. Computer networks, smart healthcare, smart transportation, Internet of Things, and big data analysis are some of her research interests.

Karan Kumar R, completed his Bachelor's degree in Computer Science and Engineering from SRM Valliammai Engineering College an autonomous institution affiliated to Anna University. His research interests include data science, machine learning, deep learning, smart transportation, Internet of Things and Data Analytics.