

ORIGINAL RESEARCH

Open access

Enhancing Electric Vehicle Navigation with 3D Vision-Based Object Localization and Classification

Ji-eun Choi^{1*}, Hyejin Park¹, Min-seo Kang², Seung-ho Lee¹

Abstract

Ongoing progress in intelligent systems and state-of-the-art technology has significantly shaped the evolution of intelligent vehicles. In recent years, the integration of multiple sensors into automobiles has enhanced and expanded advanced driver-assistance systems (ADAS), bringing the goal of full autonomy closer to reality. Despite existing self-driving techniques and implementations, autonomous driving remains an unresolved research challenge that must ensure driver safety and system dependability. This study uses images from two cameras and Global Positioning System (GPS) data to introduce a 3D vision-based approach for object localization and classification, aimed at supporting vehicle navigation. The experimental setup consists of a prototype two-seater electric vehicle, specifically developed and built for campus navigation under controlled driving conditions. At the same time, color and depth images from the main camera are fused to derive 2D features, which are then projected back into 3D space. Road detection combined with depth information isolates point clouds corresponding to objects, enabling the creation of an environmental occupancy map. A convolutional neural network was trained to categorize common urban objects within the color images. Experimental evaluations confirm the accuracy of vehicle and object positioning within the occupancy map across various scenarios, strengthening the visually derived vehicle position through integration with GPS measurements.

Keywords Visual navigation, Object classification, Driving assistance, Occupancy map, GPS pose

*Correspondence:

Ji-eun Choi

jjeun.choi@yonsei.ac.kr

¹ Department of Intelligent Power Systems, College of Engineering, Yonsei University, Seoul, South Korea

² Department of Energy Analytics and Control, Faculty of Engineering, Pusan National University, Busan, South Korea

Introduction

Over recent years, the autonomous vehicle (AV) has emerged as a major area of academic and industrial investigation, encompassing topics such as safety, electromobility, and self-driving capabilities. The rapid growth in computational capacity has accelerated the adoption of multi-sensor integration in vehicles to enable ADAS functionalities and progress toward full autonomy. Concurrently, machine learning techniques embed intelligence into ADAS, delivering timely and relevant environmental insights. Nevertheless, a vehicle's core purpose extends beyond mere transportation to encompass comfort, safety, and user convenience.

Technological innovations have revolutionized the conventional automotive sector, transforming outdated means of travel into advanced, intelligent systems that simplify mobility [1]. Autonomous vehicles enable capabilities including environmental perception, internet connectivity, compliance with traffic regulations, self-navigation, and rapid decision-making. Key drivers for AV research and development include improved road safety and the growth of multi-service delivery infrastructure [2].

Research institutions, universities, and corporations globally have engineered autonomous and driverless vehicles. The Society of Automotive Engineers (SAE) [3] established a classification system based on the degree of human involvement required for operation, serving as a standard for assessing the autonomy of self-driving vehicles. This scale spans from Level 0, offering only momentary driver assistance, to Level 5, requiring no human input whatsoever [4]. The framework of self-driving vehicles handles two primary functions: perception and decision-making. Perception encompasses localization, road mapping, tracking of dynamic obstacles, and traffic signal identification. Decision-making covers path planning, maneuver selection, obstacle avoidance, and vehicle control. Furthermore, the perception module determines the vehicle's state and constructs an internal environmental model from unprocessed sensor inputs.

Common sensors employed in AVs include Light Detection and Ranging (LiDAR), Radio Detection and Ranging (RADAR), depth and color cameras, GPS, Inertial Measurement Unit (IMU), and odometers. Consequently, diverse localization strategies have been developed, such as LiDAR-based, LiDAR-plus-camera-based, and purely camera-based approaches [5]. A prominent recent trend involves fusing data from multiple sensors to obtain robust, complementary measurements. For example, LiDAR and visual inputs are combined for object classification [6]. Similarly, in the framework proposed by Weon *et al.* [7], LiDAR data and images are used to analyze three-dimensional space. Following noise elimination, the Random Sample Consensus (RANSAC) algorithm separates ground surfaces from objects.

Interpolation refines the precision of object data. Image-based objects are aligned with corresponding 3D points; subsequently, the You Only Look Once (YOLO) detector [8] executes object identification.

In the study by Villaseñor-Aguilar *et al.* [9], vehicle localization relies on sensor fusion incorporating an odometer, GPS, and a visual sensor. Nonetheless, practical limitations must be addressed, including GPS signal degradation in regions obstructed by trees, tunnels, or tall structures. Cameras installed in vehicles have become widespread due to their safety benefits, affordability, and straightforward integration, offering richer data than many traditional sensors [10]. Generally, motion control for AVs is accomplished by processing images captured by onboard cameras and applying neural networks [11]. Semantic segmentation and object detection using Convolutional Neural Networks (CNNs) supply critical information for path-planning decisions [12].

In the work of Wu *et al.* [13], the authors split channels into grayscale versions to enhance visual object detection; subsequently, morphological preprocessing is applied to generate an improved binary image.

In the research by Muller *et al.* [14], the researchers employed deeper CNN architectures. They trained a sub-scale radio-controlled car to handle off-road navigation as part of the DARPA Autonomous Vehicle project. The training utilized images captured by two forward-facing cameras. Employing a six-layer CNN, the system learned to avoid obstacles while maintaining a speed of 2 m/s. Additionally, in the work of Rausch *et al.* [15], the authors applied supervised learning to develop an end-to-end lateral controller for steering an AV using a CNN. The training dataset consisted of steering angles paired with images from the front-facing camera, collected during human-driven operation in a CarSim simulation environment [16]. More recently, end-to-end learning approaches have been introduced to directly predict vehicle control signals from dashboard camera inputs [12]. The method outlined in Xu *et al.* [17] incorporates the temporal dynamics of dashboard camera sequences to develop a general-purpose driving model.

Since these methods rely on neural networks, effective parameter management is essential to improve their performance. Various quantization strategies help decrease memory usage and inference latency [18], while techniques to mitigate the impact of parameter choices include the class-aware trace ratio optimization (CATRO) approach [19] and the Sparse Connectivity Learning (SCL) technique [20], both adaptable to diverse network designs. Specifically for object classification tasks, the SqueezeNet architecture stands out as an effective option for substantially reducing model size, particularly when minimizing processing time is a priority [21].

The adoption of lightweight AI models offers a compelling avenue for advancing object classification techniques under constrained memory conditions, yielding encouraging results in real-world deployments [22]. MobileNetV2 is another suitable CNN choice for mobile and embedded platforms, effectively reducing memory requirements in real-time scenarios [23].

Multi-sensor fusion supports simultaneous localization and mapping (SLAM), a core capability required for autonomous vehicles. SLAM involves building a map of an unfamiliar environment while concurrently estimating the vehicle's position within that map [24]. The main sensors for SLAM include LiDAR and cameras. Furthermore, visual SLAM (vSLAM) approaches have gained prominence for feature initialization and mapping in autonomous driving contexts [25]. vSLAM is broadly categorized into direct SLAM and feature-based SLAM: direct SLAM tracks a new frame by directly optimizing pixel intensities, whereas feature-based SLAM estimates the pose of a new frame by extracting sparse features, matching them against a local map, and minimizing the reprojection errors. In the study by Xu *et al.* [26], an indoor vSLAM-based localization system generates an environmental map and determines the camera pose within it. In the study by Zhu *et al.* [24], an enhanced multi-sensor fusion positioning framework integrating GNSS, LiDAR, cameras, and IMU with graph optimization is introduced to achieve higher vehicle pose accuracy. Likewise, Cai *et al.* [27] present an innovative visual-inertial simultaneous localization and mapping (VI-SLAM) technique tailored for intelligent navigation systems, addressing challenges arising from vehicle dynamics and expansive outdoor environments.

Conversely, the swift evolution of the automotive industry demands substantial cuts in fossil fuel usage. Given that most gasoline vehicles emit exhaust gases, electric vehicles offer a viable substitute [28]. Designating dedicated lanes for AVs leverages interconnected multi-sensor systems to guide vehicles back onto the designated path when they deviate. This strategy has proven effective in alleviating traffic congestion and shortening travel durations within connected autonomous systems [29]. Furthermore, modern vehicles incorporate features such as autonomous parking, visual object detection and estimation, real-time global positioning, and other capabilities aimed at reaching the highest degree of autonomy. While dependable outcomes have been achieved in these areas, significant hurdles remain in performance and swift responses to critical situations. Vision-based methods have delivered robust solutions for real-time challenges, including visual odometry, object detection, and classification. Accordingly, this study examines visual and pose information through vSLAM and ORB techniques to identify and position objects, thereby generating an occupancy map for a controlled urban setting. This map supports the navigation of an electric vehicle, as depicted in **Figure 1**. The prototype vehicle operates across the university campus in an outdoor urban environment, under velocity-controlled conditions, with various obstacles such as safety cones, parked cars, trees, and traffic signs. The controlled setup allows reliance on a limited sensor suite onboard the vehicle, comprising two color-depth cameras, an IMU, and GPS. The key contributions of this research include: (1) developing a sparse scene representation from point clouds and mapping it to a 2D occupancy grid for displaying occupied areas while minimizing data storage and computational load; (2) supplying a dataset comprising visual data, global pose estimates, and inertial measurements collected from sensors on the electric vehicle; (3) configuring five distinct navigation scenarios to verify the constructed occupancy map against the baseline map and to classify objects along the route. This study constitutes an initial step toward realizing a fully autonomous electric vehicle.

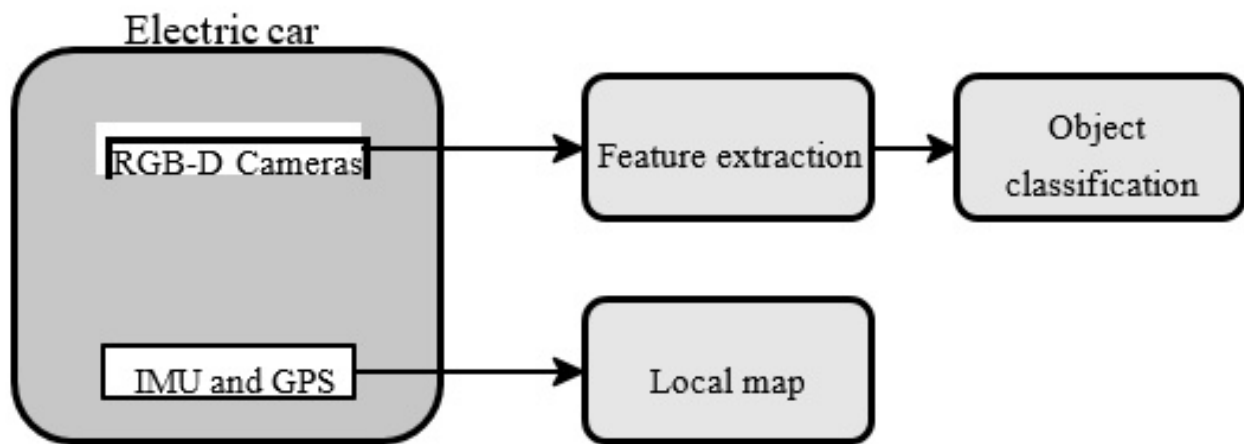


Figure 1. Block diagram of the proposed method for car driving assistance.

Materials and Methods

Overall navigation strategy

The proposed approach for vision-based object detection, 3D localization, and classification is illustrated in **Figure 2**. While the electric vehicle navigated the campus perimeter, a synchronized collection of images, IMU readings, and GPS signals took place. Positioning information came from the IMU and GPS units, and forward-facing images captured the immediate environment ahead. Color and depth images were recorded at 21 frames per second, whereas IMU and GPS samples were collected at 5 Hz. Key 2D feature points were identified in the color image and subsequently mapped to 3D coordinates using the aligned depth data. Estimation of the local ground plane enabled separation of point clouds corresponding exclusively to objects, allowing accurate definition of object bounding regions within the color image. These regions were then reprojected from 3D back to 2D space and incorporated into an occupancy grid, enabling cross-checking of object locations against the reference map and incremental refinement of the overall occupancy representation. Concurrently, a convolutional neural network inspired by the AlexNet architecture performed classification on segmented object patches extracted from color images. Throughout the operation, the prototype electric vehicle moved forward while performing real-time object detection, spatial localization, and semantic classification along its route, with GPS continuously logging the vehicle's pose to ensure consistency with the intended straight-line path. Using point clouds combined with a compact 2D occupancy grid to mark occupied cells significantly reduced both memory consumption and processing overhead during navigation.

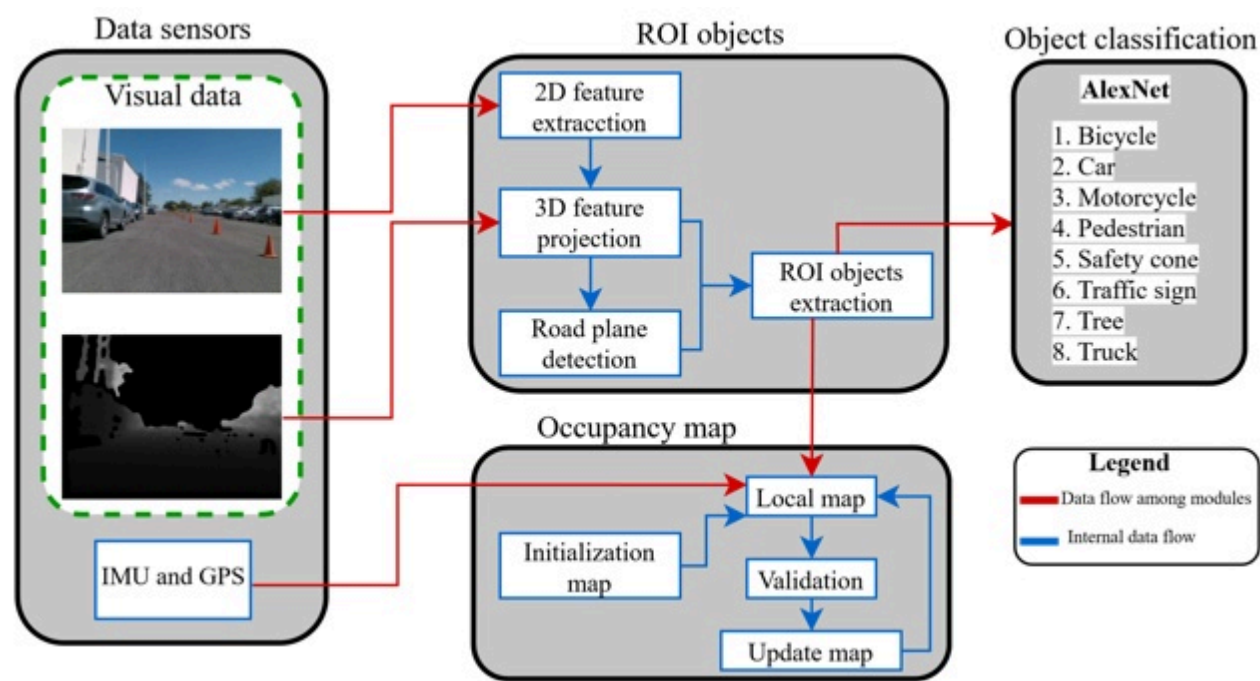


Figure 2. Global strategy of electric car driving assistance.

Figure 3a presents the experimental testbed developed to evaluate the proposed system. The platform is a campus-built electric vehicle prototype assembled from both newly sourced and reused parts. Although construction was only about 60% complete, the vehicle could reach speeds above 80 km/h, delivered roughly 90 km of range in standard driving scenarios, and exhibited stable ride quality with adequate suspension damping. The placement of the IMU and GPS modules at the rear of the vehicle is shown in **Figure 3b**. The prototype was further equipped with two RGB-D cameras, whose mounting positions are detailed in the schematic diagram in **Figure 3c**. Although the vehicle supported bidirectional motion, the experiments described here were restricted to forward driving.

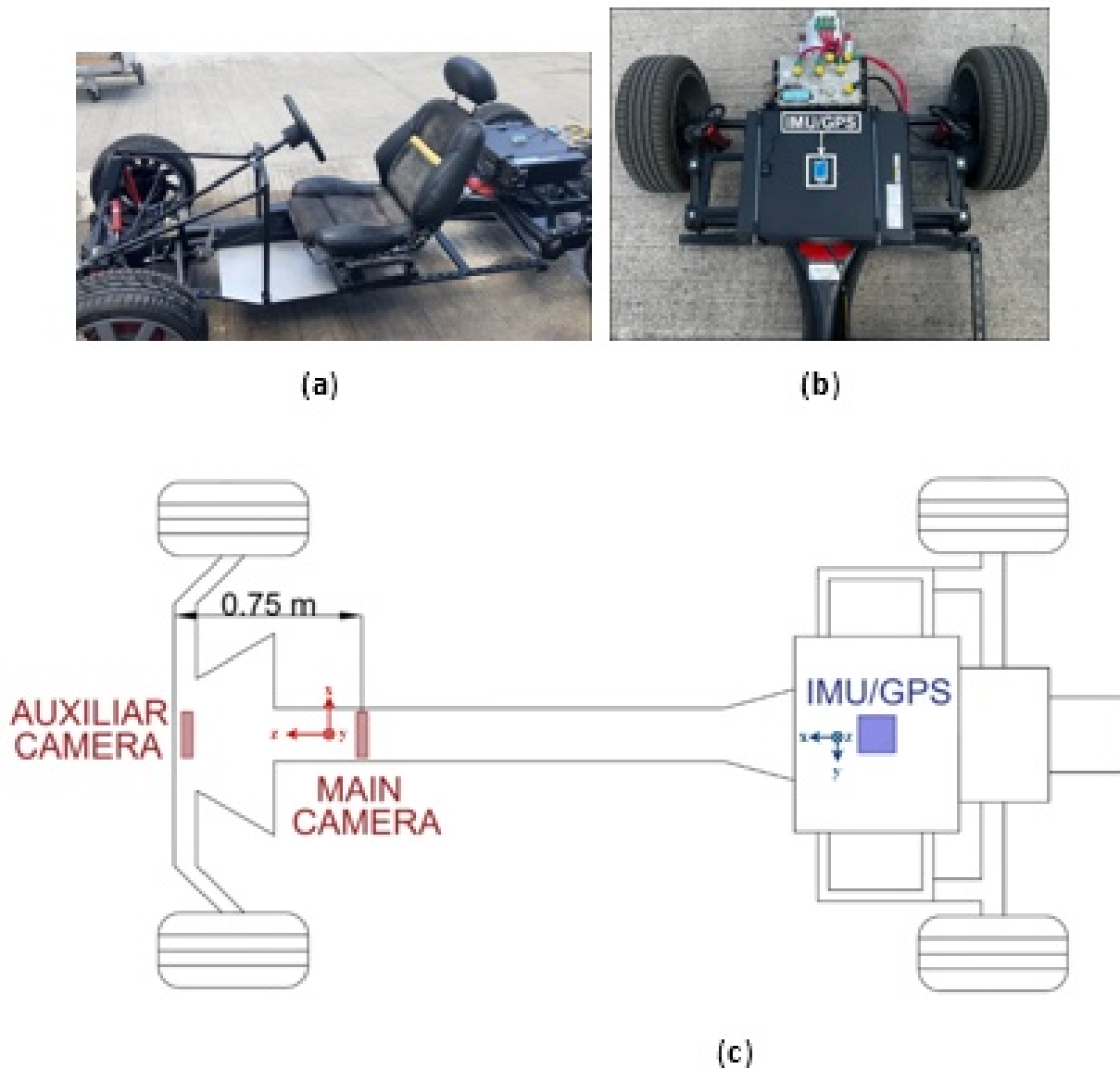


Figure 3. Prototype of the electric car: (a) Overall view, (b) IMU and GPS location, and (c) schematic diagram of the electric vehicle and sensors location.

Color and depth image acquisition

Both cameras on the electric vehicle were from the Intel RealSense 400 series RGB-D family [30]. According to the layout in **Figure 3c**, the main camera was fixed 0.75 m ahead of the vehicle's front end, aligned with the central chassis axis, and elevated on a 0.5 m support. Chosen as the primary sensor owing to its strategic positioning, it delivered a depth field of view of $65^\circ \times 40^\circ$ and a color field of view of $69^\circ \times 42^\circ$. This configuration reliably covered relevant objects situated between 0.6 m and 3 m in front of the vehicle. The area within the first 0.6 m remained obscured when relying solely on the primary camera; therefore, a secondary camera was mounted centrally on the front bumper at 0.3 m and slightly downward-tilted to eliminate the near-field blind spot.

Acquisition of RGB and depth streams was handled through the Open3D library [31] in a Python environment. The capture pipeline consisted of three principal stages: (1) detection and validation of camera connectivity, (2) configuration of acquisition settings and parameter retrieval, and (3) persistent storage of images together with supplementary metadata required for later stages. Stage one confirmed successful detection and communication with both cameras. Stage two involved applying the desired settings: image resolution was set to 640×480 pixels, and the capture rate was fixed at 21 fps for both RGB and depth channels. Intrinsic calibration parameters for the primary camera—essential for accurate 3D reprojection—were also determined at this step. These included focal lengths (scaled to pixel units in both horizontal and vertical directions) and the principal point offsets (measured as pixel distances from the image's left and top borders). The computed values appear in Table 1.

Table 1. Intrinsic parameters of the main camera.

Intrinsic parameter	Values
Focal length [f_x, f_y]	[611.71044, 611.11853]
Coordinates of the image's principal point [pp_x, pp_y]	[322.10314, 231.02186]

Stage three managed the saving of color and depth frames, with corresponding timestamps logged into an auxiliary text file. The overlapping fields of view for color and depth captures from the primary camera are demonstrated in Figure 4. Depth maps were stored in 16-bit precision, with each pixel encoding the measured distance to the scene point visible in the paired color image.

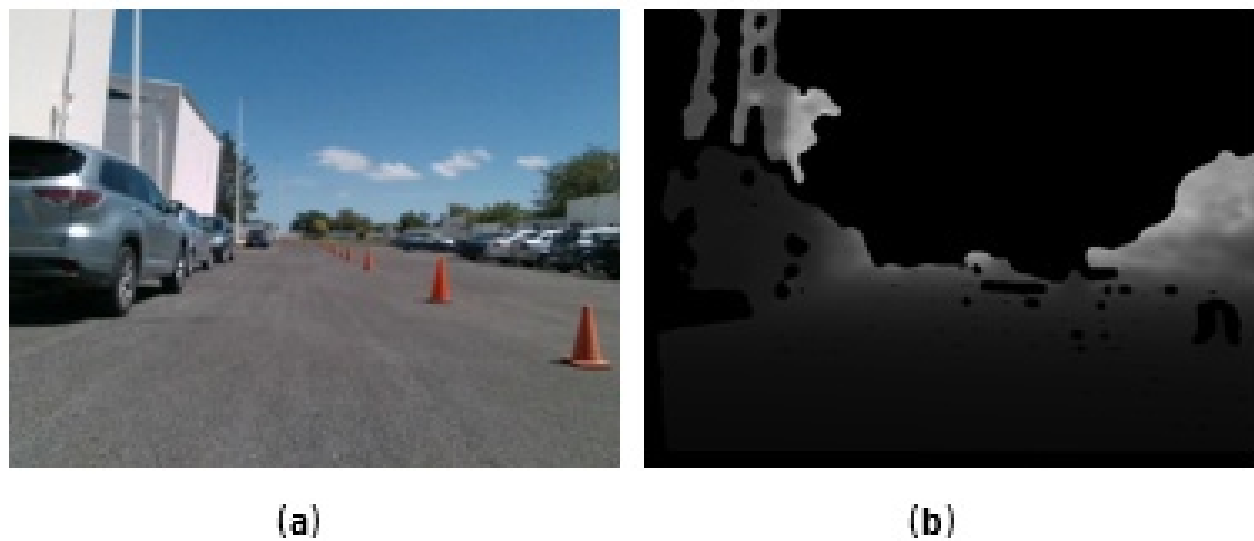


Figure 4. The FOV in the acquired images: (a) Color image, and (b) Depth image.

The electric vehicle completed five separate runs, collecting close to 2000 synchronized color-depth image pairs on each occasion. Every trajectory followed an approximately 100 m straight path, with parked vehicles lining the left and safety cones on the right. All recorded sequences represented static outdoor campus scenes. Analysis commenced with 2D processing of the color images, after which only the most salient feature points were elevated into 3D coordinates. This selective projection strategy kept computational expense modest and substantially decreased the amount of data requiring further handling. The next subsection explains the detailed image processing steps and the procedure for isolating object regions to support environment prediction and navigation verification.

ROI features and objects point clouds

The Oriented FAST and Rotated BRIEF (ORB) descriptor [32], an open-source feature detector, was used to detect 2D keypoints in the images. ORB integrates the FAST (Features from Accelerated Segment Test) corner detector with the BRIEF (Binary Robust Independent Elementary Features) binary descriptor. FAST excels at quickly locating prominent corners by comparing a central pixel's intensity with its immediate neighbors, making it highly efficient for real-time processing. Furthermore, ORB uses a multi-scale pyramid structure to identify features at different resolutions. In this implementation, an eight-level image pyramid was employed, with a scale factor of 1.2 applied between consecutive levels. BRIEF, in turn, produces a compact binary string for each keypoint by performing simple intensity comparisons between selected pairs of pixels within the local patch, thereby efficiently encoding the surrounding texture. After feature extraction, the ORB detector yielded a set of up to 1500 keypoints, each with pixel coordinates and orientation angles. It was observed that a consistent sky region appeared in the upper portion of the images and did not represent a relevant area of interest; consequently, this portion was excluded from the feature detection process. The predefined search zone (marked in blue) and the detected ORB points within it are shown in **Figure 5a**.

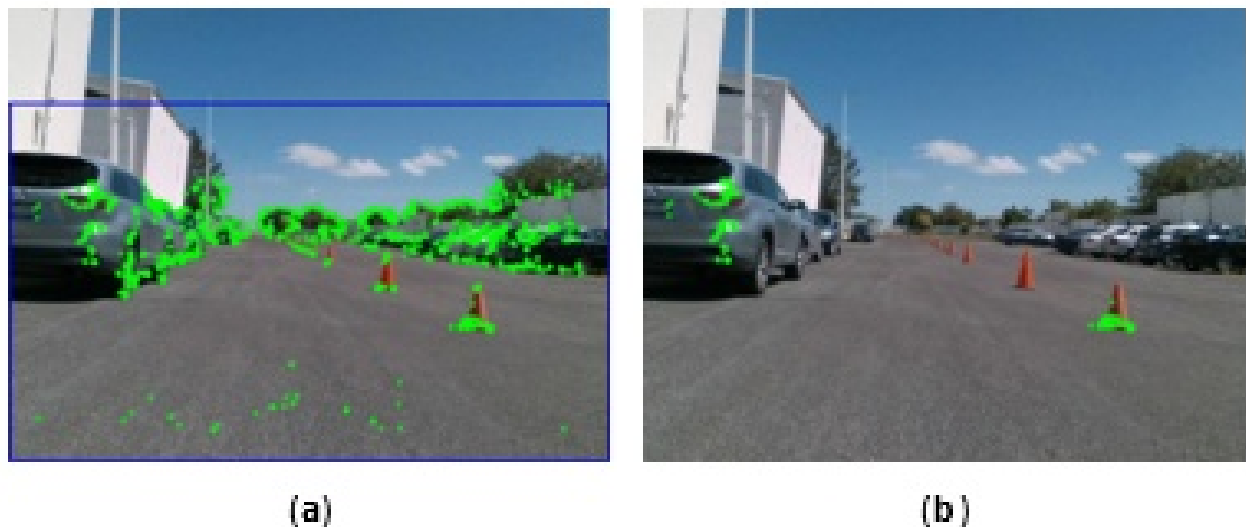


Figure 5. ORB points detected: (a) predefined area and detected points, and (b) meaningful points.

Projection of the 2D keypoints into 3D space was accomplished using Matlab's `pcfromdepth` function. This routine took the pixel locations of the ORB features, located their corresponding depth values in the aligned depth map, and derived the z-coordinate (depth) in real-world units. Regions in the depth image showing a value of zero indicated invalid measurements or areas beyond the infrared sensor's reliable range in the Intel camera; such erroneous points were therefore eliminated. As a result, only valid points falling within the camera's effective sensing range were retained for subsequent object detection, localization, and classification tasks. The nominal maximum depth range of the camera was 3 m. Although this limit could not be increased, the effective working interval could be shifted by adjusting the camera mounting height and orientation. In the current setup, the operational depth was restricted to 2–5 m ahead of the camera, necessitating the application of a depth-scaling factor. In the outdoor experimental conditions tested here, a depth factor of 1400 was applied. Leveraging the camera's intrinsic parameters, the pixel coordinates were converted into real-world x and y coordinates. Ultimately, this process generated a vector of valid 3D coordinates corresponding to the retained feature points.

As previously noted, the ORB keypoints were distributed fairly evenly across the image (excluding the sky region). However, some key points inevitably fell on the road surface. To isolate points belonging exclusively to objects, the Simple Morphological Filter (SMRF) [33] was applied to classify the point cloud into ground and non-ground categories.

SMRF partitions the point cloud into a user-defined grid and performs morphological operations to distinguish terrain from elevated objects. After enforcing the depth constraint (2–5 m) and executing ground segmentation, roughly 85% of the original points visible in **Figure 5a** were filtered out, leaving the distribution shown in **Figure 5b**. The majority of removals were due to the depth filter, with the surviving points clustering predominantly around actual objects.

A binary mask was then created based on the 2D locations of the remaining ORB points, as depicted in **Figure 6a**. A region-growing algorithm was applied to expand these isolated points into contiguous areas (**Figure 6b**). Vertical edges were detected in the color image using a Roberts cross kernel, producing the edge map shown in **Figure 6c**. These edges constrained and refined the region-growing process shown in **Figure 6b**, yielding the combined regions and boundaries illustrated in **Figure 6d**. Subsequent morphological dilation, combined with area-based filtering, was applied to the result shown in **Figure 6d**, producing the cleaned mask shown in **Figure 6e**. This refined binary mask facilitated merging of fragmented areas with the underlying objects in the color image, ultimately generating tight bounding boxes that accurately enclosed each detected object, as highlighted in red in **Figure 6f**.

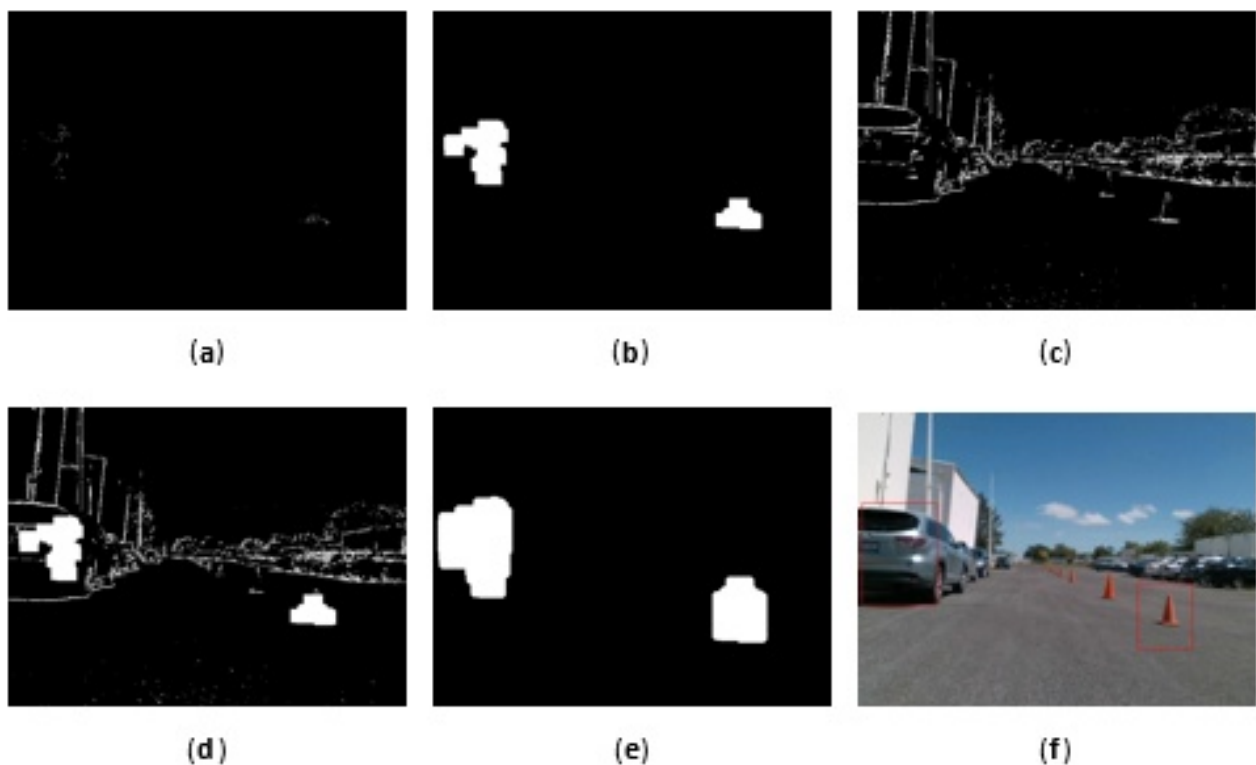


Figure 6. Image processing for bounding box extraction: (a) ORB points, (b) dilated points, (c) edges, (d) union of (b) and (c), (e) filtered areas, and (f) extracted bounding boxes.

To determine the distance from each object to the front of the vehicle, the centroid of its bounding box was computed. The corresponding depth value at this centroid location was then retrieved from the depth image to estimate the object's z-distance along the vehicle's forward axis. Rather than relying solely on the single centroid pixel, a 3×3 neighborhood centered on it was sampled from the depth image, and the average depth within that small matrix was used as the final distance measurement for greater robustness.

Occupancy map and car pose estimation

Figure 7 illustrates one of the vehicle's recorded paths, derived from its GPS sensor data. The trajectory begins at a green-marked starting point and concludes at a red-marked endpoint. The vehicle followed a 100 m route around

buildings A and B on the campus, averaging 8 km/h. While images were being captured, a CSV file recorded the index and timestamp of each acquired frame. Concurrently, IMU and GPS measurements were logged into a separate text file, capturing vehicle velocity, latitude/longitude coordinates, and corresponding timestamps. Velocity readings were filtered to retain only positive values. The initial non-zero velocity marked the moment the vehicle began moving, while the final positive velocity indicated the end of the navigation sequence.



Figure 7. Car trajectory around buildings A and B.

Although the experimental platform covered approximately 100 m, this study focuses solely on the detection, localization, and update of the global occupancy map for the initial 25 m of the path. **Figure 8** presents the reference (initial) occupancy map constructed from manual physical measurements of obstacle distances and sizes. The grid was discretized into 10×10 cm cells, with occupied zones marked in black. Any object that intersected even a fraction of a cell was classified as occupied, resulting in slightly enlarged representations of the actual obstacles on the map. The red dot denotes the vehicle's starting position. Throughout the navigation process, the vehicle's location was continuously provided by GPS readings and cross-verified against the vision-derived positions of detected objects, as detailed in the preceding section. Image timestamps enabled precise temporal alignment, allowing the fusion of visual pose estimates with GPS-derived pose information captured simultaneously. This synchronization supported simultaneous forward object detection via vision and real-time vehicle positioning via GPS.

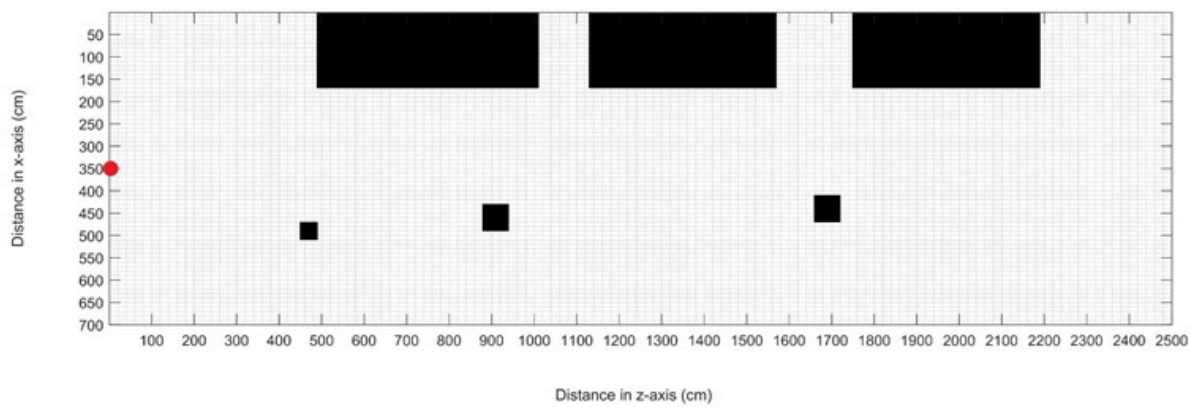


Figure 8. Initialization of the global occupancy map with predefined objects in the navigation scene.

The reference map was manually drafted based on precise measurements of the static environment. The vehicle commenced its route from the road centerline, corresponding to the grid coordinates (350, 0) shown in [Figure 8](#). All distance measurements referenced the front of the vehicle as the origin point. In contrast, the estimated occupancy map was generated by transforming the 3D real-world coordinates of detected points into grid indices using Matlab's `world2grid` function. These grid coordinates were then rendered into a binary occupancy map to produce the dynamically updated global representation.

Synchronization between image frames and GPS/IMU data relied on the timestamps recorded in the respective CSV and text files. Although the camera operated at 21 fps, only one frame per second was selected for processing to extract ORB features and perform their 3D reprojection. This downsampled sequence served to estimate vehicle pose at 1 Hz throughout the navigation task. Each processed frame (every 21st) served as a keyframe. Feature matching was conducted between consecutive keyframes, and the resulting displacement in feature locations was used to compute the camera's rotation and translation, thereby reconstructing the camera (and vehicle) trajectory. The 3D point positions were accordingly adjusted to account for the camera's motion between keyframes. Meanwhile, the 2D feature points from each processed frame were mapped from the local coordinate system to the global occupancy grid of the navigation environment.

To validate the vision-based pose estimates, GPS geographic coordinates were converted to Universal Transverse Mercator (UTM) x–y coordinates using the `deg2utm` function. Pairwise distances between successive pose measurements were computed, and their cumulative sum yielded the total traveled distance. The distance increment estimated from visual data at each one-second interval (21 frames) closely matched the distance derived from GPS measurements accumulated over the same interval. The GPS sensor delivered approximately 5 samples per second.

After computing bounding boxes around detected objects, these regions were simultaneously fed into a convolutional neural network (CNN)-based classifier for object identification. The architecture, training, and implementation details of this classifier are presented in the next section.

CNN architecture for object classification

The cropped regions identified in the color images were treated as candidate areas likely containing relevant environmental objects. To assign trustworthy category labels, advanced deep learning frameworks based on convolutional neural network (CNN) architectures were leveraged. To achieve strong classification performance, the CNN required a carefully prepared dataset with an adequate (but not excessive) number of labeled examples for both the training and validation phases. The complete classification pipeline developed in this research is outlined in [Figure 9](#). Eight representative object types commonly present in urban driving scenarios were included in the dataset. The refined

bounding boxes obtained after 3D validation were then fed into the finalized CNN model, yielding both the predicted class and an associated confidence score for each detected object.

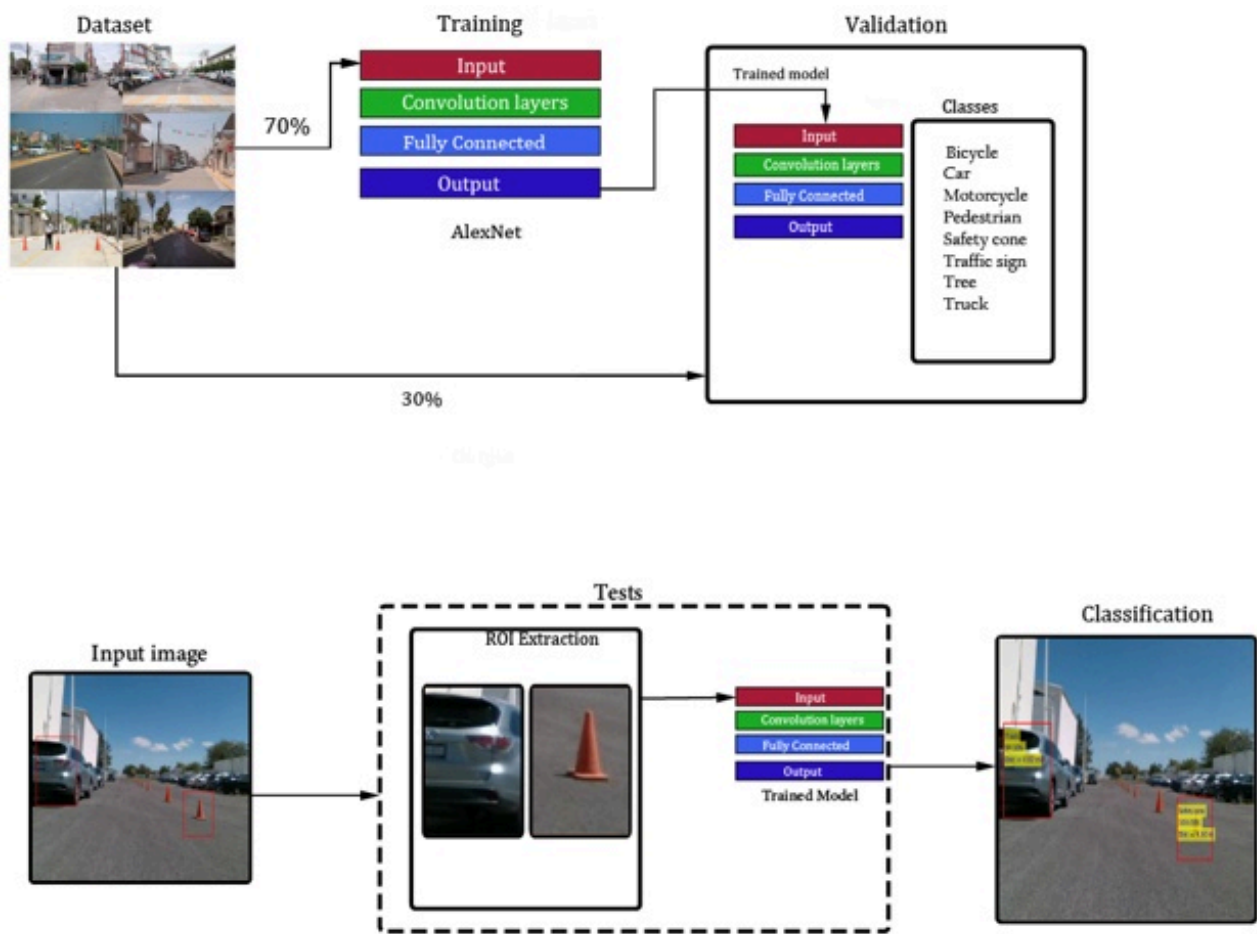


Figure 9. Global diagram for object classification based on the AlexNet model.

Dataset description and alexnet model

Training data were gathered by mounting a video camera in the rear-view mirror area of a car driven through the downtown areas of two cities near the university campus. In the first city, ten separate video sequences were recorded in the afternoon under clear-sky conditions to prevent glare and overexposure. The combined routes spanned roughly 11.5 km and amounted to 1.5 hours of continuous recording. The same recording duration and similar lighting conditions were replicated in the second city across a 13.5 km path. From these video streams, object instances matching the target classes were isolated using a dedicated segmentation CNN.

Table 2 displays representative object crops that exhibit illumination and appearance closely aligned with the images captured by the onboard Intel camera (**Figure 4a**). The chosen segmentation network was the Fast Convolutional Neural Network (F-CNN) [34], initialized with weights pretrained on the Common Objects in Context (COCO) dataset [35] and executed through the PixelLib framework [36]. This approach enabled efficient and accurate extraction of individual object regions from thousands of frames. As a result, the isolated patches had minimal background clutter, resulting in a clean, evenly distributed collection suitable for both training and validation. Overall, the dataset contained 23,436 object patches: 16,406 for training and 7,030 for validation.

For on-the-move classification during vehicle travel, the well-established AlexNet architecture was selected and trained across 30 epochs. All input images were uniformly resized to $227 \times 227 \times 3$ pixels. The network consists of eight principal layers: five convolutional blocks (layers 1, 3, 5, 6, and 7), each followed by ReLU activation, and three fully connected layers (layers 2, 4, and 8) trained with an adaptive gradient optimizer. A final softmax layer maps the activations to probabilities over the eight defined classes. The resulting AlexNet class predictions and confidence scores were superimposed directly on the bounding boxes to provide visual feedback within the proposed driver-assistance system.

Table 2. Categories trained with the CNN architecture.

Patches	Classes	# Samples
	Bicycle	2919
	Car	3464
	Motorcycle	2651

	Pedestrian	2584
	Safety cone	3001
	Traffic sign	2900
	Tree	2681

	Truck	3236
	Total	23,436

Results and Discussion

Here, the experimental findings are presented in detail. All image sequences were recorded under tightly controlled conditions along a specific campus roadway section. The model was trained to recognize eight object categories relevant to urban environments: bicycle, car, motorcycle, pedestrian, safety cone, traffic sign, tree, and truck.

The red line in **Figure 10** traces the vehicle's estimated trajectory based solely on onboard visual data. Thanks to the fine-grained camera-derived pose estimation, the path shows clear deviations and curvature rather than the idealized straight line produced by GPS data (**Figure 7**). Point clouds flanking the red trajectory correspond to the salient 3D features extracted from color images and reprojected into space. For each detected object cluster in this scene portion, the classification stage assigns an appropriate semantic label.

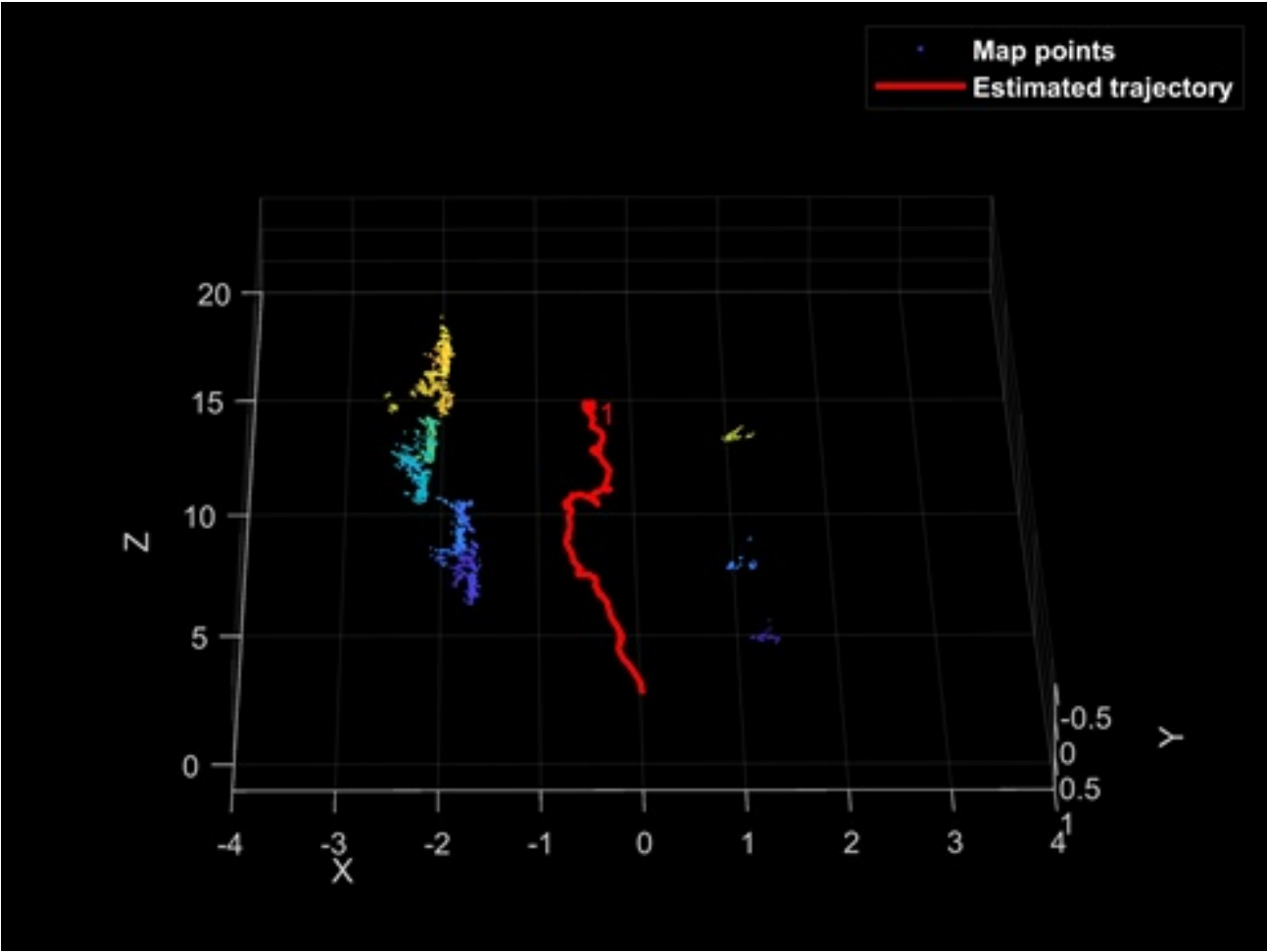


Figure 10. Plot of the point clouds and trajectory of the electric car.

Table 3 compares the forward (z-axis) distance estimates from the vision pipeline with the reference distances computed from synchronized IMU and GPS readings over a continuous 16-second segment of the drive.

Table 3. The difference between the visual-based estimated and the GPS car pose.

Time (s)	Z-Axis Position (m)			Velocity (km/h)
	Estimated	GPS	Error (m)	
1	0.3722	0.3891	0.0169	2.685
2	0.8706	0.8351	0.0355	3.018
3	1.4029	1.3362	0.0667	3.333
4	2.0728	1.8901	0.1827	3.852
5	2.7595	2.5002	0.2593	4.148
6	3.4908	3.7739	0.2831	4.407
7	4.5312	4.4386	0.0926	4.592
8	5.3728	5.1216	0.2512	4.833

9	6.1267	5.8237	0.3031	4.963
10	7.0877	7.2444	0.1568	5.093
11	8.0256	8.0008	0.0248	5.259
12	8.9706	8.7759	0.1947	5.463
13	9.8779	9.5701	0.3077	5.611
14	10.9351	10.3453	0.5898	5.630
15	11.7587	11.1393	0.6194	5.667
16	13.3592	12.7629	0.5964	5.907

The initial 25 m segment of the route captured roughly 350 images over 16 seconds.

Given the camera’s field of view, each frame was cropped to 640×100 pixels, eliminating the upper sky region while preserving critical near-field working range and depth information. The camera’s optical center served as the reference origin for all object detection and spatial localization procedures. The actual distance covered by the vehicle was derived from GPS recordings and compared with the trajectory reconstructed solely from visual cues. As detailed in **Table 3**, the smallest discrepancy in distance measurements was 0.0169 m, whereas the largest reached 0.6194 m. The table also reveals a clear pattern: errors tended to grow as vehicle speed increased. Notably, the majority of vision-based distance estimates exceeded the corresponding GPS-derived values. Even so, these discrepancies remained minor enough to be disregarded in the context of the primary objectives—namely, reliable object detection and localization to enable safe vehicle navigation. Objects were identified close to their true positions, providing sufficient warning to the vehicle control system about potential obstacles ahead.

Due to the sparse coverage of feature points across object surfaces, accurately delineating a tight bounding box around each target proved challenging. To refine the bounding boxes initially, additional preprocessing steps were applied to the color images. From the recovered 3D point set, a 2D projection was generated using only the X and Z coordinates, forming the basis for an estimated occupancy grid. **Figure 11** depicts the overlap between this estimated map and the manually initialized reference map. Green regions indicate occupied cells in the ground-truth initial map, pink regions mark occupied cells in the vision-derived map, and black zones highlight areas of agreement (intersection) between the two. Although the overlap was sometimes partial, even a small shared area was sufficient to confirm the presence of detected objects. The red marker in **Figure 11** denotes the vehicle’s starting position. In every tested configuration, safety cones were positioned along the right side of the vehicle’s path. The estimated cone locations consistently appeared slightly farther ahead than their actual positions. Another observed difference concerned object dimensions: the vision-based map tended to represent objects as somewhat larger than in the reference map. Nevertheless, all relevant obstacles along the route were successfully identified and marked as occupied cells.

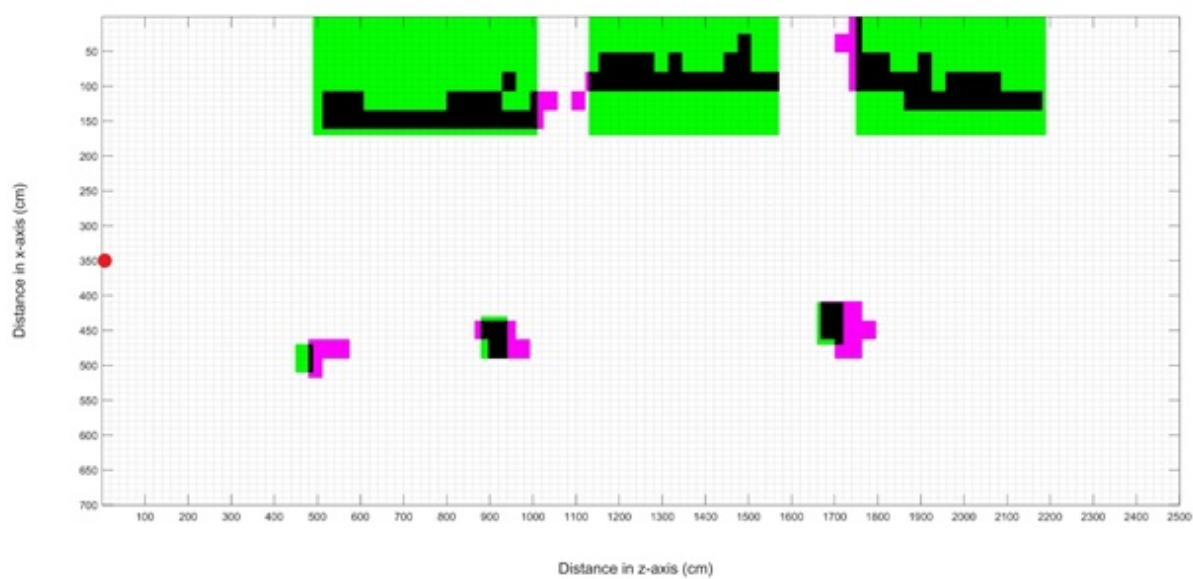


Figure 11. Intersection of the union between the initial occupancy map and the visually estimated map.

Classification results

The AlexNet model was trained using a 70/30 train-validation split over 30 epochs, with data drawn from the dataset summarized in Table 2. Training progress, shown in Figure 12, reached an accuracy of 96.87%. Starting around the tenth epoch, the model exhibited consistently stable performance, remaining above 90% thereafter.

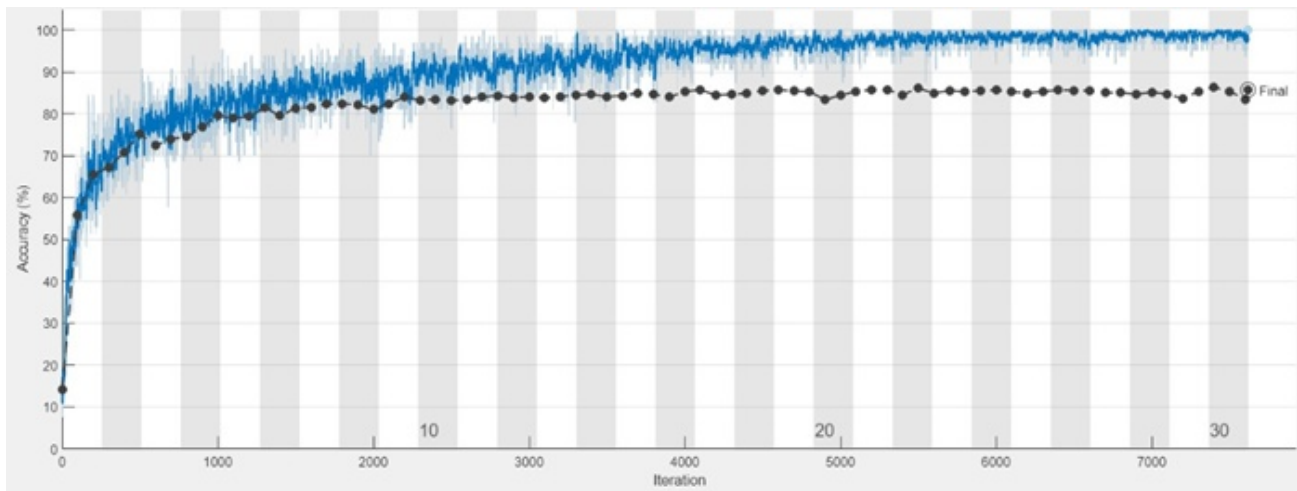


Figure 12. Accuracy results of the AlexNet architecture.

The confusion matrix presented in Figure 13 yielded an overall accuracy of 85.66% and provides a breakdown of classification performance across all categories. Categories sharing visual similarities—such as bicycle versus motorcycle, or car versus truck—exhibited the highest rates of misclassification. Comparable error patterns emerged across three additional datasets collected during outdoor drives with the electric vehicle, which included cars, trucks, trees, and safety cones. In these sequences, cars and trucks consistently showed the largest classification errors. Errors increased notably when the bounding box captured only a partial view of the object. Conversely, objects fully contained within their bounding boxes were classified with high confidence scores and near-perfect accuracy.

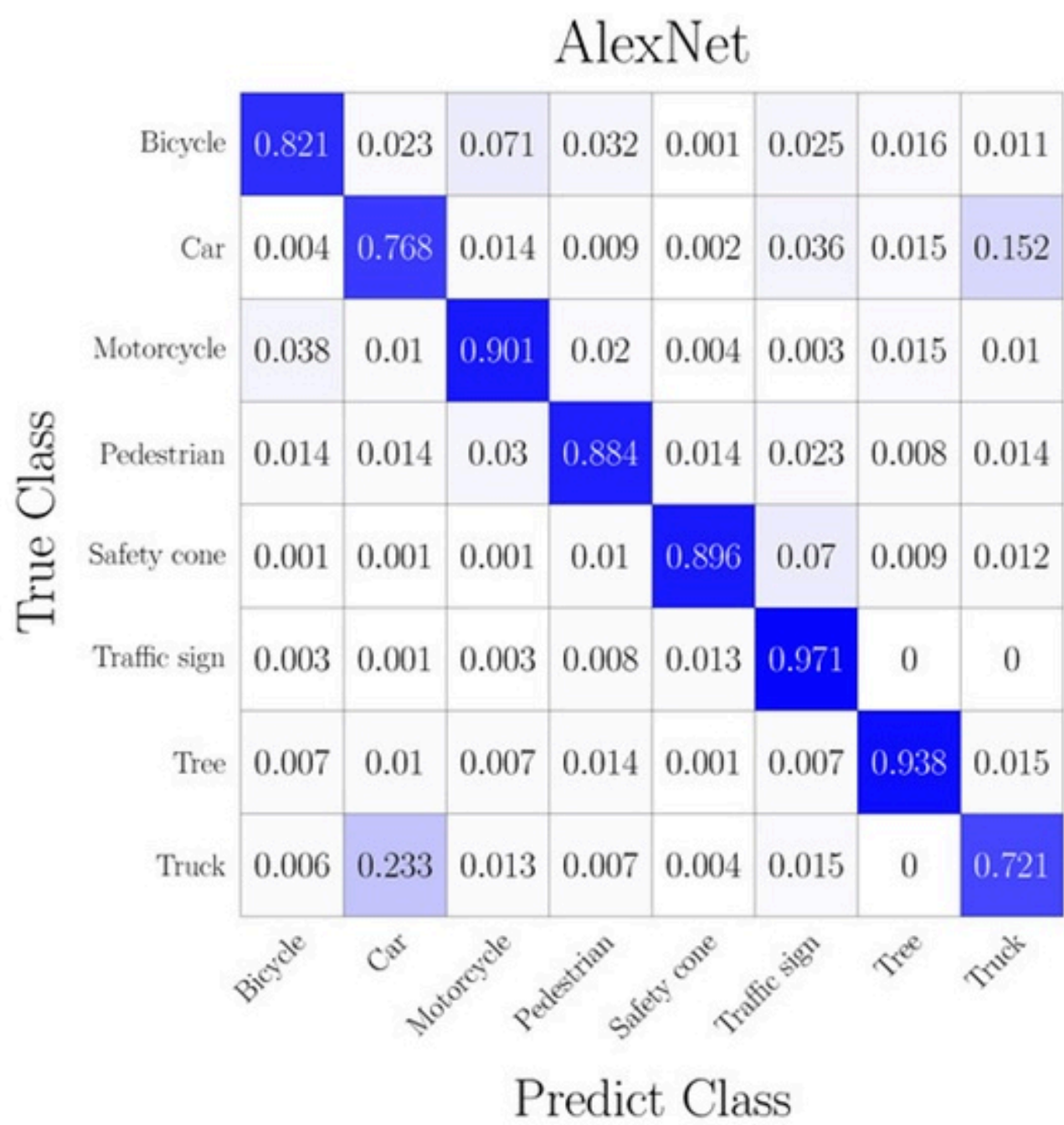


Figure 13. Confusion matrix for 8 categories.

Table 4 reports the Precision, Recall, and F1-score values for the dataset outlined in **Table 2**. Overall, the model demonstrated dependable performance across the eight classes. Increasing the sample count for underrepresented categories—bicycle, car, motorcycle, and truck—while removing ambiguous or conflicting examples could further mitigate bias and reduce classification errors [37].

Table 4. Evaluation metrics.

Class	Precision	Recall	F1-Score

Bicycle	92.29%	82.07%	86.88 %
Car	74.44%	76.80%	75.60 %
Motorcycle	85.44%	90.06%	87.69 %
Pedestrian	88.16%	88.40%	88.28 %
Safety cone	96.18%	89.55%	92.75 %
Traffic sign	83.91%	97.12%	90.03 %
Tree	93.08%	93.78%	93.43 %
Truck	77.00%	72.09%	74.46 %

Figure 14 displays eight representative frames extracted from the navigation sequence at timestamps $t = 0, 3, 5, 9, 13, 27, 32$, and 48 seconds. Each frame includes the computed bounding boxes, the assigned class label and confidence score from the classifier, and the estimated distance from the electric vehicle to the object. At $t = 3, 13$, and 32 seconds, insufficient feature points were detected within the safety cone regions, preventing the reliable generation of bounding boxes. In these cases, depth values near the image borders fell below 2 m, outside the camera's effective infrared sensing range, so objects in that near zone were overlooked. In contrast, the remaining frames in **Figure 14** contained adequate feature coverage and valid depth measurements within the operational range, enabling successful bounding box computation, object classification, and accurate pose estimation relative to the vehicle.



Figure 14. Detection, distance, and classification of objects in a sequence.

The presented vision-based navigation approach was specifically designed for an electric vehicle operating in a controlled outdoor setting, where the path was bordered by safety cones on the right and parked cars on the left. Starting from a predefined global map, the vehicle maintained a straight course while collecting visual data from two cameras and positional and inertial measurements from IMU and GPS sensors in a static environment. Five separate navigation runs were recorded, each with a different arrangement of objects along the route. Consequently, the experimental datasets used to evaluate and confirm the proposed method were collected at various points throughout a clear, sunny day. Within the camera's effective detection range, virtually all objects were successfully identified, positioned, and categorized with

good precision. For example, the majority of safety cones were reliably detected even as the vehicle moved at 8 km/h, and the computed object locations proved sufficient to alert the vehicle control system to impending obstacles. Furthermore, incorporating IMU-derived vehicle pose served as an additional reference to strengthen the visual estimates of both vehicle and object positions. The chosen resolution of the occupancy grid enabled representation of smaller objects, although parked cars remained the predominant obstacles encountered along the route.

Comparison of the GPS-based vehicle position with the vision-derived pose estimates, summarized in [Table 3](#), reveals that per-second discrepancies were negligible relative to the physical scale of the vehicle and surrounding objects. Errors of 20 to 30 cm were minor when evaluated against the vehicle's dimensions and the required safety buffers around obstacles during travel. Given that the smallest objects present (such as safety cones) measured approximately 45 cm in width, the observed errors remained well within acceptable safety margins. As illustrated in [Figure 10](#), the most prominent 3D feature points formed point clouds distributed to either side of the vehicle's trajectory, effectively delineating occupied regions in the surroundings. From a practical perspective, this method tended to place occupied areas slightly behind their true current locations. In the least favorable scenarios, the pose error would indicate to the vehicle that an object remained in a prior position—even if the object had already moved closer or passed nearby. Nonetheless, because objects in the environment were relatively large, achieving average positional accuracy was adequate to provide timely warnings about potential hazards ahead.

Similar to the approach introduced here, the study by Villaseñor-Aguilar *et al.* [9] employed multi-sensor fusion comprising an odometer, an IMU, and a monocular camera to estimate vehicle trajectory, reporting a low mean squared error in pose comparisons. However, systems that integrate GNSS, IMU, LiDAR, and visual sensors—as shown by Zhu *et al.* [24]—typically demand substantial computational resources and power consumption. By contrast, the current method relied solely on GPS combined with an RGB-D camera for localization within dedicated lanes, thereby reducing both processing complexity and synchronization challenges.

Conclusion

The difficulties encountered in developing fully driverless vehicles can be addressed by developing rapid, precise techniques that can handle uncertain or dynamic conditions. While extensive research has advanced the field of autonomous driving, significant opportunities remain for innovative, practical, embeddable, and replicable solutions tailored to experimental and real-world platforms. The present study introduced a vision-centric module for object detection and spatial localization to support navigation of an electric vehicle. By jointly processing color and depth images, 2D feature points were extracted and subsequently reprojected into 3D point clouds. Feature points lying on the road surface were identified and excluded, leaving only those associated with actual objects. Bounding boxes were then derived around these object-specific feature points in the color image and fed into a convolutional neural network for classification. This integrated pipeline enabled the extraction of object features, estimation of their positions relative to the camera, and construction of a vision-based occupancy map. The resulting map was cross-validated against the predefined global occupancy map of the route, revealing positional errors of 20–60 cm relative to the reference object locations.

Nevertheless, these deviations were modest given the objects' sizes and the essential safety clearances required during vehicle operation. In addition, the detected and localized objects were reliably classified using an AlexNet-based model, enabling semantic labels to be assigned to the occupied cells in the map. Although the proposed strategies were evaluated in static scenes, the generated visual occupancy map effectively supported path generation, serving as an initial assistance component for the experimental electric vehicle platform.

Future efforts will focus on broadening the applicability of this method across diverse navigation scenarios. Key priorities include incorporating dynamic elements such as pedestrians, moving vehicles, and cyclists into the environment. Onboard high-performance graphics processing units will be integrated to enable faster computation. Furthermore, an

embedded stereo camera system will be adopted to facilitate real-time visual processing and to provide quantitative benchmarks for assessing estimation accuracy.

Abbreviations

ADAS: Advanced drive-assistance systems

AV: Autonomous vehicle

BRIEF: Binary robust independent elementary features

CATRO: Class-aware trace ratio optimization

COCO: Common objects in context

CNN: Convolutional neural networks

DARPA: Defense Advanced Research Projects Agency

F-CNN: Fast convolutional neural network

FOV: Field of view

GPS: Global positioning system

IMU: Inertial measurement unit

LiDAR: Light detection and ranging

ORB: Oriented fast and rotated BRIEF

RADAR: Radio detection and ranging

RANSAC: Random sample consensus

ReLU: Rectified linear unit

SAE: Society of Automotive Engineers

SCL: Sparse connectivity learning

SLAM: Simultaneous localization and mapping

SMRF: Simple morphological filter

UTM: Universal Transverse Mercator

vSLAM: Visual simultaneous localization and mapping

VI-SLAM: Visual-inertial simultaneous localization and mapping

YOLO: You only look once

Acknowledgements

The authors Alfredo Medina-Garcia, Jonathan Duarte-Jasso, and Juan J. Cardenas-Cornejo and Yair A. Andrade-Ambriz are funded by the scholarships 829974, 830483, 809170, and 846226 granted by CONAHCYT, Mexico. We acknowledge the University of Guanajuato (CIIC 059/2023) for its sponsorship in realizing this work and the Fondo Mentefacturalo with the project "SOL-1024: Diseño y construcción de vehículo eléctrico biplaza para transporte de personas con discapacidad".

Conflict of interest

None

Financial support

This research was funded by the CIIC 059/2023 project. The University of Guanajuato partially funded the APC under POA 2024. Additionally, this research was partially funding by the Fondo Mentefacturalo with the project "SOL-1024: Diseño y construcción de vehículo eléctrico biplaza para transporte de personas con discapacidad".

Ethics statement

None

Received: 12 Mar 2024

Revised: 04 May 2024

Accepted: 16 Jun 2024

Published online: 20 September 2024

Rights and permissions

Open Access The author(s) retain copyright. This article is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. It may be shared and adapted for non-commercial purposes with appropriate attribution, an indication of changes, and distribution of adaptations under the same license. Third-party material may be subject to separate terms identified in its credit line. View the license at <https://creativecommons.org/licenses/by-nc-sa/4.0/>.

References

- Sharma O, Sahoo N, Puhan N. Recent advances in motion and behavior planning techniques for software architecture of autonomous vehicles: A state-of-the-art survey. *Eng Appl Artif Intell*. 2021;101:104211.
- Parekh D, Poddar N, Rajpurkar A, Chahal M, Kumar N, Joshi GP, et al. A Review on Autonomous Vehicles: Progress, Methods and Challenges. *Electronics*. 2022;11(14):2162.
- International S. Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. *SAE Int*. 2018;4970:1-5.
- Halin A, Verly JG, Droogenbroeck MV. Survey and Synthesis of State of the Art in Driver Monitoring. *Sensors*. 2021;21(16):5558.
- Badue C, Guidolini R, Carneiro RV, Azevedo P, Cardoso VB, Forechi A, et al. Self-driving cars: A survey. *Expert Syst Appl*. 2021;165:113816.

Khan MQ, Lee S. A Comprehensive Survey of Driving Monitoring and Assistance Systems. *Sensors*. 2019;19(11):2574.

Weon IS, Lee SG, Ryu JK. Object Recognition Based Interpolation with 3D LIDAR and Vision for Autonomous Driving of an Intelligent Vehicle. *IEEE Access*. 2020;8:65599-608.

Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*; 2016 Jun 27-30; Las Vegas, NV, USA. p. 779-88.

Villaseñor-Aguilar MJ, Peralta-López JE, Lázaro-Mata D, García-Alcalá CE, Padilla-Medina JA, Perez-Pinal FJ, et al. Fuzzy Fusion of Stereo Vision, Odometer, and GPS for Tracking Land Vehicles. *Mathematics*. 2022;10(12):2052.

Ondruš J, Kolla E, Vertal' P, Šarić Ž. How Do Autonomous Cars Work? *Transp Res Procedia*. 2020;44:226-33.

Kuutti S, Bowden R, Jin Y, Barber P, Fallah S. A Survey of Deep Learning Applications to Autonomous Vehicle Control. *IEEE Trans Intell Transp Syst*. 2021;22(2):712-33.

Fujiyoshi H, Hirakawa T, Yamashita T. Deep learning-based image recognition for autonomous driving. *IATSS Res*. 2019;43(4):244-52.

Wu F, Yang Z, Mo X, Wu Z, Tang W, Duan J, et al. Detection and counting of banana bunches by integrating deep learning and classic image-processing algorithms. *Comput Electron Agric*. 2023;209:107827.

Muller U, Ben J, Cosatto E, Flepp B, Cun Y. Off-road obstacle avoidance through end-to-end learning. *Adv Neural Inf Process Syst*. 2005;18:739-46.
<https://doi.org/10.5555/2976248.2976341>.

Rausch V, Hansen A, Solowjow E, Liu C, Kreuzer E, Hedrick JK. Learning a deep neural net policy for end-to-end control of autonomous vehicles. In: *Proceedings of the 2017 American Control Conference (ACC)*; 2017 May 24-26; Seattle, WA, USA. IEEE; 2017.

Simulation M. CarSim Mechanical Simulation [Internet]. 2023 [cited 2023 Oct 10]. Available from: <https://www.carsim.com/>.

Xu H, Gao Y, Yu F, Darrell T. End-To-End Learning of Driving Models From Large-Scale Video Datasets. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*; 2017 Jul 21-26; Honolulu, HI, USA. 2017.

Gholami A, Kim S, Dong Z, Yao Z, Mahoney MW, Keutzer K. A Survey of Quantization Methods for Efficient Neural Network Inference. *arXiv:2103.13630*. 2021.

Hu W, Che Z, Liu N, Li M, Tang J, Zhang C, et al. CATRO: Channel Pruning via Class-Aware Trace Ratio Optimization. *IEEE Trans Neural Netw Learn Syst*. 2023; in press.

Tang Z, Luo L, Xie B, Zhu Y, Zhao R, Bi L, et al. Automatic Sparse Connectivity Learning for Neural Networks. *arXiv:2201.05020*. 2022.

Huang Q. Weight-Quantized SqueezeNet for Resource-Constrained Robot Vacuums for Indoor Obstacle Classification. *AI*. 2022;3(1):180-93.

Huang Q. Towards Indoor Suctionable Object Classification and Recycling: Developing a Lightweight AI Model for Robot Vacuum Cleaners. *Appl Sci*. 2023;13(17):10031.

Wu Z, Tang Y, Hong B, Liang B, Liu Y. Enhanced Precision in Dam Crack Width Measurement: Leveraging Advanced Lightweight Network Identification for Pixel-Level Accuracy. *Int J Intell Syst*. 2023;2023:9940881.

Zhu J, Zhou H, Wang Z, Yang S. Improved Multi-Sensor Fusion Positioning System Based on GNSS/LiDAR/Vision/IMU with Semi-Tight Coupling and Graph Optimization in GNSS Challenging Environments. *IEEE Access*. 2023;11:95711-23.

Gao B, Lang H, Ren J. Stereo Visual SLAM for Autonomous Vehicles: A Review. In: *Proceedings of the 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*; 2020 Oct 11-14; Toronto, ON, Canada. IEEE; 2020.

Xu L, Feng C, Kamat VR, Menassa CC. An Occupancy Grid Mapping enhanced visual SLAM for real-time locating applications in indoor GPS-denied environments. *Autom Constr*. 2019;104:230-45.

Cai Y, Qian W, Zhao J, Dong J, Shen T. Visual-Inertial Navigation System Based on Virtual Inertial Sensors. *Appl Sci*. 2023;13(13):7248.

Bathla G, Bhadane K, Singh RK, Kumar R, Aluvalu R, Krishnamurthi R, et al. Autonomous Vehicles and Intelligent Automation: Applications, Challenges, and Opportunities. *Mob Inf Syst*. 2022;2022:7632892.

Ma K, Wang H. Influence of Exclusive Lanes for Connected and Autonomous Vehicles on Freeway Traffic Flow. *IEEE Access*. 2019;7:50168-78.

Intel. Depth Camera D415—Intel® RealSense™ Depth and Tracking Cameras [Internet]. 2023. Available from: <https://www.intelrealsense.com/depth-camera-d415/>.

Zhou QY, Park J, Koltun V. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*. 2018.

Rublee E, Rabaud V, Konolige K, Bradski G. ORB: An efficient alternative to SIFT or SURF. In: *Proceedings of the 2011 International Conference on Computer Vision*; 2011 Nov 6-13; Barcelona, Spain. IEEE; 2011.

Han J, Liu Y, Sun X. A scalable random forest algorithm based on MapReduce. In: *Proceedings of the 2013 IEEE 4th International Conference on Software Engineering and Service Science*; 2013 May 23-25; Beijing, China. IEEE; 2013.

Abdulla W. Mask R-CNN for Object Detection and Instance Segmentation on Keras and TensorFlow [Internet]. 2017. Available from: https://github.com/matterport/Mask_RCNN.

COCO Consortium. Common Objects in Context (COCO) [Internet]. 2021. Available from: <https://cocodataset.org/>.

Olafenwa A. PixelLib: A Library for Quickly Segmenting Objects in Images and Videos [Internet]. 2022. Available from: <https://github.com/ayoolaolafenwa/PixelLib>.

Ledesma S, Ibarra-Manzano MA, Almanza-Ojeda DL, Avina-Cervantes JG, Cabal-Yepez E. On removing conflicts for machine learning. *Expert Syst Appl*. 2022;206:117835.