Research Article

International Journal of Advanced Technology and Engineering Exploration, Vol 10(105)
ISSN (Print): 2394-5443 ISSN (Online): 2394-7454
http://dx.doi.org/10.19101/IJATEE.2023.10101118

Localization for self-driving vehicles based on deep learning networks and RGB cameras

Shahad S. Ghintab* **and Mohammed Y. Hassan**
Department of Control Engineering, University of Technology-Iraq, Alsinaa Street, Baghdad, Iraq

Received: 07-February-2023; Revised: 16-August-2023; Accepted: 20-August-2023

Abstract
Autonomous vehicles (AVs) have emerged as captivating engineering ventures in the twenty-first century, capturing the interest of numerous academics and engineers across multiple generations. The world looks forward to leveraging AVs for reducing accidents caused by human errors and optimizing parking space utilization, particularly in urban areas. Accurate localization is pivotal for effective AV navigation, enabling the vehicle to pinpoint its precise position. While global positioning system (GPS) coordinates are widely used, their inherent errors and limitations can render them inadequate for determining precise location information, particularly in urban settings. Furthermore, drifting errors can undermine the efficacy of simultaneous localization and mapping (SLAM) algorithms. The proposed approach involves the utilization of a deep neural network, specifically a modified AlexNet architecture, which is a convolutional neural network (CNN), for localizing AVs in well-lit urban driving environments. The CNN enhances accuracy while reducing computational complexity and training time. Instead of relying on costly light detection and ranging (LiDAR) or radar sensors, a more affordable red green blue (RGB) camera sensor is employed. During testing, depth images are combined with RGB images using the intensity hue saturation (IHS) algorithm to enhance precision. Simulation results demonstrate an impressive accuracy rate of 95.49%, affirming the effectiveness of the proposed strategy. This study introduces a lightweight, precise, and reliable CNN architecture that significantly improves the accuracy of AV localization, simultaneously reducing predicted position errors by a considerable margin. The network's superiority is evidenced by mean square error (MSE) values of 0.039, 0.0099, and 0.0047 for position x, y, and orientation predictions, respectively. To validate real-time performance, the trained CNN was implemented in Python and integrated into the car learning to act (CARLA) simulator, enabling the online localization of a self-driving vehicle. This application successfully showcases the feasibility and efficacy of the proposed method.

Keywords
Autonomous vehicle, Localization, Deep learning, Convolutional neural networks CNN, Intensity hue saturation IHS, K-mean algorithm.

1.Introduction
Localization is a crucial aspect of the operation and safety of autonomous vehicles (AV). It pertains to the AV's ability to precisely determine and uphold its position and orientation within its environment. Accurate localization is vital for the vehicle's navigation, obstacle avoidance, route planning, and overall dependable functionality. The navigation process for an AV encompasses establishing its starting point, identifying the destination, and determining the route to move from one point to another (known as the path planning issue). The AV's success in this process heavily relies on its localization capabilities, ensuring it can accurately locate itself in its surroundings [1].

AVs utilize a blend of sensor inputs and advanced algorithms to determine their position relative to the surrounding environment, as shown in *Figure 1*. These sensors may consist of the global positioning system (GPS), the inertial measurement unit (IMU), light detection and ranging (LiDAR), radio detection and ranging (RADAR), and cameras. Each sensor offers unique data types, and their seamless integration is crucial for achieving precise localization. By combining the information from these diverse sensors, AVs can accurately navigate and operate in complex and dynamic environments

*Author for correspondence

[2]. The main challenges in AV navigation revolve around four key aspects: perception, localization, path planning, and motion control [3]. The main objective of localization is to determine the vehicle's position relative to its surroundings by using a reference coordinate system. Relying solely on GPS for localization can lead to inaccuracies and inconsistencies in the coordinates, posing challenges for self-driving vehicles to achieve precise positioning. Additionally, the utilization of simultaneous localization and mapping (SLAM) algorithms can be affected by drifting errors over time, leading to a decline in performance and accuracy. Some older techniques heavily depended on expensive LiDAR or RADAR sensors, making them impractical and limited in widespread adoption.

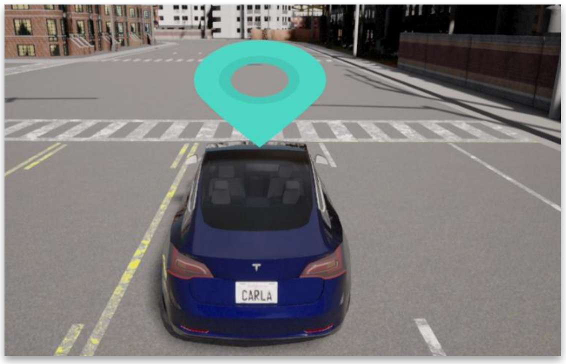


Figure 1 One of AV functions: localization

Furthermore, earlier efforts may have struggled to maintain consistent localization accuracy in various weather conditions, including bright sunny, wet, or cloudy environments [4]. Conventional localization techniques employ feature detectors such as oriented fast and rotated briefly (ORB) [5], scale-invariant feature transform (SIFT) [6], and speeded up robust features (SURF) [7] to mirror visual elements from previous frames. Triangulation estimates 6-degrees of freedom (DOF) position from matches, which is optimized using powerful fitting algorithms such as random sample consensus (RANSAC) [8]. SIFT and ORB-SLAM2 [9] can real-time localize indoors with a few millimeter error but fails in low-texture settings. Direct methods minimize photometric and geometric errors to determine camera position between images [10]. To avoid drift, these approaches compare the current image to a database of recorded frames to locate loops. After loop validation, camera postures are globally modified to minimize drift. probabilistic approaches like random forests [11] and neural networks may tackle the localization problem. Trainable systems can interpret high-dimensional input sequences. Unlike feature-based techniques, neural networks have continual temporal complexity. PoseNet, the first end-to-end neural network for 6-DoF camera pose estimation, was based on GoogLeNet [5]. Two fully-connected regression layers were altered to generate a vector with position and orientation estimates instead of the softmax classification layer from the original GoogLeNet model. Training data is greatly reduced by transfer learning using object recognition weights. Networks enhanced post-PoseNet. Hyper parameter modification was avoided by PoseNet's trainable geometric loss function [12]. No method used input data depth. Red green blue-depth (RGB-D) data capture is harder without affordable high-resolution sensors. Training time and computing complexity made prior approaches unsuitable for real-time deployment. These issues highlight the need for new self-driving car localization solutions that address GPS accuracy, SLAM drifting errors, sensor accessibility, weather performance, and training effectiveness. Prior studies using outmoded and insufficient AV methods showed various drawbacks when implementing modern approaches like deep learning. These constraints ignored weather and time of day, requiring expensive sensors to obtain detailed information about the driving environment. In conclusion, outmoded self-driving vehicles have sensor cost, weather adaptability, training efficiency, and data availability issues [13]. The objectives of this paper are as follows: Reducing the number of layers in AlexNet, a popular convolutional neural network (CNN), to precisely locate AV using an RGB camera sensor, thereby eliminating the need for expensive sensors such as LiDAR and RADAR. This research aims to accomplish high precision in AV localization and minimize errors in estimated positions by utilizing a lightweight, dependable, and robust network for efficient training. In addition, the implementation of a CNN-based visual localization system for AV in different urban driving scenarios is investigated. The CNN algorithm classifies and interprets texture with greater precision than other detection sensors, such as RADAR and LiDAR, by utilizing modest sensors. The CNN algorithm's training procedure utilizes RGB camera images as inputs and generates the projected position as output. The objective of this study was to develop a precise and reliable technique for AV localization. To accomplish this, implemented an offline CNN-based method in MATLAB using RGB images. This method produced secure and accurate outcomes while reducing the required training time. The study utilized a vision-only system, eradicating the need for costly sensors such as RADAR and LiDAR. Instead, they utilized the intensity hue saturation (IHS) method to combine RGB images with depth information and validate the positioning accuracy of the CNN. To obtain precise coordinates, the researchers employed the k-means method to determine how closely an image matches street images, thereby refining the localization process. The novel CNN architecture is the main innovation of the proposed method. The network's straightforward structure allowed for effective computation while maintaining high precision. This lightweight, accurate, and dependable CNN enhanced AV localization accuracy and reduced predicted position errors by a significant margin. To evaluate the trained network's efficacy in real-time, the network was implemented in Python and connected to the car learning to act (CARLA) simulator. This enabled the online localization of a self-driving vehicle, further validating the practicability and efficacy of the proposed method. In conclusion, this study presents a robust CNN-based method for AV localization, demonstrating its accuracy, dependability, and efficacy. Using RGB images and a lightweight network, the method provides a viable alternative to expensive sensor-based systems, making it a promising solution for autonomous driving applications in the real world.

The following is an overview of this paper's structure: The second section offers a comprehensive review of the relevant literature. In section 3, the methodology of this investigation is discussed in greater detail. The fourth section provides a detailed description of the proposed method. The fifth section presents the results and analysis of the simulation. The report concludes with concluding remarks and findings in section 6.

2.Literature review

Localization is how an AV finds its location. This can be done using GPS, LiDAR, and optical odometry. Localization methods will be examined in the literature review [14].

Stenborg et al. (2018) proposed a semantic segmentation for long-term visual localization. Using semantic information enhances accuracy and robustness. Offline map development and online localization includes two phases. In the offline phase, a CNN generates a semantic map and extracts and stores features. In the online phase, the algorithm matches segmented photos with stored features to estimate the cameras pose. On a dataset, the approach beats other visual localization algorithms. However, it requires semantically segmented images, which may sometimes be available [15].

Parisotto et al. (2018) put forth an attention-based recurrent network for global posture estimation. The 6-DoF posture of an object was determined using an RGB image. Encoders extracted visual features, and decoders estimated pose. The attention method focused the network on relevant observable regions, improving position estimates. The authors trained and tested their model on several datasets and found that it outperforms several global pose estimation methods. The proposed method needed a lot of training data, which limited this research. New item categories not in the training data may not work with the proposed technique [16].

Heng et al. (2019) offered a complete AV real-time localization and 3D scene perception approach. Deep learning was used to analyse multi-camera 360-degree photos and construct a 3D scene model. Autonomous driving requires real-time, weather- and lighting-resistant systems. This paper requires real-world experimental results. The system was evaluated in a controlled environment; therefore its performance in more complex and dynamic real-world scenes is unknown. System robustness and accuracy needed real-world testing [17].

Amini et al. (2019) presented mobile robot autonomous variational end-to-end navigation and localization (VEENAL). VEENAL's variational auto encoder learned a latent representation of the robot's environment and an end-to-end navigation control policy. The method navigated and localized without maps or GPS. Navigation and localization accuracy improved in real-world experiments. VEENAL's high computing cost was negative. It was too computationally intensive to train the variational auto encoder and end-to-end control approach for low-power robots. The authors should evaluate VEENAL in simpler, dynamic scenarios, which may reduce its real-world applicability [18].

Ma et al. (2019) used sparse semantic high-definition (HD) maps to reliably localize self-driving automobiles. Semantic information from HD maps could help self-driving cars localize. They first found the HD map features visible to the automobile sensors, then used the observed properties to calculate the vehicle's pose. The authors compared their method to other state-of-the-art algorithms on a self-driving car platform dataset. They improved self-driving car localization. The paper used HD maps, which may need to be more accurate or available in real life. It may fail in dynamic environments with obsolete HD maps. The suggested method needs real-world testing to prove its reliability and resilience [19].

Yin et al. learned a similarity metric between 3 dimension (3D) LiDAR point clouds from the query scan and the map. Matching the most similar point clouds in the map may infer the query scan's global position. The proposed method performed well on a public dataset. Map data quality and completeness limited the proposed solution. Complete map data would improve the proposed method. For real-time applications, the proposed technique may be too expensive to process [20].

Wan et al. (2020) developed a multi-sensor fusion approach for urban vehicle location. LiDAR, camera, and GPS data improved vehicle localization accuracy and robustness. The authors compared their method to other cutting-edge methods using urban data. Urban multi-sensor fusion worked well in the article. Paper is limited. The proposed method required too much computing power for real-time applications. Second, the authors only tested their strategy in a few metropolitan areas, which do not reflect urban variety [21].

Chen et al. (2021) demonstrated AV image-based LiDAR localization in structured and unstructured scenarios. Neural networks extracted picture attributes and particle filters evaluated posture. Multiple datasets and comparisons to existing methods showed the method's accuracy and resilience. Pre-trained neural networks didn't generalize well. The method assumed preprocessed and calibrated LiDAR data, which may not be accurate in real life [22].

Héry et al. (2021) LiDAR, global navigation satellite system (GNSS), and HD maps enabling AV decentralized cooperative localization. The proposed solution provided reliable localization information to autonomous cars in GNSS-deficient environments. AV estimated poses with sensor data. Particle filters estimated vehicle postures in real-world experiments. Unfortunately, the system required AV to interact and share sensor measurements. Communication bandwidth may limit applications [23].

Qin et al. (2021) provided autonomous driving visual localization road map. Lanes, road boundaries, and intersections are captured via a lightweight semantic map. Only RGB images were used to create the map, which can be GNSS updated in real time to reflect environmental changes. On benchmark datasets, their strategy outperformed existing methods. In complex circumstances, the semantic map was used in the road map system. The authors also noted that the system's optical localization limits autonomy and requires other sensors [24].

Chu et al. (2021) suggested cooperative channel mapping (CCM) for vehicle localization. The system mapped the area using wireless automobile communication networks' channel state information (CSI). In challenging urban environments with non-line-of-sight (NLOS), the suggested CCM technique leveraged spatiotemporal CSI correlation to accurately estimate vehicle positions. After extensive simulations and experiments, the authors' approach accurately located autos in diverse environments. In NLOS settings, the suggested CCM methodology outperforms current vehicle localization methods. A dense network of vehicles with CSI-capable communication devices was needed for high localization accuracy. Therefore, sparse vehicle density may limit the recommended method's scalability. The method's computational complexity may limit its real-time use [25].

Ballardini et al. (2021) localized cars using 3D building models and point cloud matching. 3D maps and vehicle sensor point clouds are used to estimate the car's position. Matching 3D map and point clouds approximated vehicle position. Real-world datasets fared well. The paper needs a pre-built 3D map of the environment, which may not be available. The procedure may fail if the 3D map is unavailable or old. Point cloud-to-3D map matching took too long for real-time applications [13].

Li et al. (2021) intelligent vehicle localization using multi-sensor fusion-based semi-open navigation maps. A semi-open intelligent vehicle navigation map was created using multi-sensor fusion. The intelligent car's environment map was accurate thanks to LiDAR, IMU, and GPS. On a real-world dataset, the proposed technique performed well. LiDAR sensor accuracy, which can be expensive and unavailable in all vehicles, was used in the paper's technique. A static environment was also assumed, which can be inaccurate [26].

Liu and Guo in 2021 suggested using an extended kalman filter (EKF) and deep learning to localize cars without GPS. In challenging environments, EKF and deep learning achieve great accuracy and robustness. The EKF employed IMU data to predict the vehicle's position and velocity during GPS outages, and the deep learning model corrected it using the vehicle's previous trajectories. In short and long-term GPS outages, the proposed method worked effectively. The paper's flaw was the method's real-time performance. To test the proposed method in real-time, where the vehicle's motion is more unexpected, and the scientists used pre-recorded data. More research was needed on the method's real-time performance in diverse driving circumstances [27].

Guo et al. (2021) demonstrated semantic localization in autonomous driving utilizing HD maps and coarse-to-fine. Coarse localization with LiDAR-based mapping, semantic segmentation of point cloud data, and HD map matching were used. A structural scene dataset showed good semantic segmentation and localization using the provided strategy. The paper's flaw is that it only tested the proposed solution on one structural condition. The proposed technique could be evaluated on more diverse and complicated datasets to establish its usefulness and generalizability [28].

In 2021, Ren et al. addressed strong off-road LiDAR localization for autonomous cars. The study localizes accurately with LiDAR point cloud data. The authors' off-road studies were promising. It can handle uneven ground, grass, and impediments. Even in extreme off-road situations, the resilient and exact approach localizes. Sensor noise, occlusions, and range affect LiDAR data. The method's computing requirements aren't fully covered; therefore, a real-time implementation research is possible [29].

In 2022, Yanase et al. proposed an exact vehicle localization approach using LiDAR and radar sensors and a matching confidence framework. To improve localization system precision and endurance, the scientists suggest combining radar observations with LiDAR point cloud data. Feature extraction from sensor modalities and matching procedures estimate the vehicle's position. Real-world data was used to illustrate the correctness of the authors' localization approach. Rain and fog can destroy sensors, but their technology was more robust. Due to the recommended confidence estimate framework's capacity to measure matching outcome dependability, the system could indicate scenarios where localization may be less trustworthy. The material supplied does not directly address the approach's limitations, and identifying them would need a thorough research project [30].

The 2022 research paper by Peng et al. introduced a novel LiDAR-based dynamic localization approach. ROLL improved localization using transient mapping and LiDAR perception. Construction sites and busy cities were examined for LiDAR data. ROLL outperformed current localization methods. Temporary mapping allows the system to update and adapt to changing environments. Adaptation enhanced the system. ROLL was limited. Sensor noise and blockage affected its precision LiDAR data. Real-time application may be limited by system computing. ROLL exhibited promising LiDAR-based localization developments, providing useful information for further research and refining [31].

In 2022, Dauptain et al. employed sensor fusion, computer vision, and machine learning to create a high-level autonomous driving perception and localization system. The results proved its precision in sensing the environment and estimating vehicle position. The device has various drawbacks, including potential issues in poor conditions and high computing requirements. These limits must be addressed to optimize the system's performance for autonomous driving applications in the actual world [32].

In 2022, Lee et al. employed semantic segmentation on selected frames for real-time monocular SLAM. 3D reconstructions, camera trajectory predictions, and semantically improved maps were correct. The method enhanced scene interpretation and resource use but had size ambiguity, information loss, quick motion, occlusions, and processing overhead. Semantic segmentation improved real-time monocular SLAM systems [33].

In 2022, Kang et al. proposed GNSS-free cooperative localization for autonomous driving. AVs locate using LiDAR and V2X. The authors compared their technique to others. In GNSS-denied environments, LiDAR and V2X-based cooperative localization improved AV localization. V2X communication and Lidar data yielded high-resolution maps. Autonomous driving localization accuracy was sufficient using the proposed technique. In GNSS-denied environments, the proposed method overcomes localization failure. Autonomous cars can locate themselves using LiDAR and V2X communication. Research is scarce. Real-world results may vary. The study only investigated LiDAR and V2X technologies, although other sensor modalities and fusion approaches could improve localization accuracy. For real-time implementation in a resource-constrained system, the recommended technique's computation and communication needs were not sufficiently examined [34].

In 2023, Han et al. suggested field-based SLAM. LiDAR-visual-inertial simultaneous and mapping (LVI-SAM) is improved. They tested their technique using real-world vehicle data. Localization and mapping were more accurate and resilient with the modified LVI-SAM method. The system addressed GPS signal loss, tough terrain, and dynamic barriers. The improved LVI-SAM approach improved localization and mapping with visual input from the surroundings. The algorithm's accuracy improved with visual, odometry, and inertial sensor data. Algorithm constraints. In resource-constrained systems, the Improved LVI-SAM algorithm's computational complexity can slow real-time performance. The system was vulnerable to weak illumination and sparse visual features since it relied on visual data. To overcome these constraints and make the approach applicable to other field circumstances and vehicle platforms, more research and optimization are needed [35].

Final evaluation study is confined to different environments: Some papers analyze the suggested localization strategy in limited circumstances. A report that only assesses the strategy in controlled or structured conditions may not sufficiently portray real-world driving challenges and complexities. Sensor limits ignored: GPS, LiDAR, and IMUs localize. Studies must acknowledge sensors' constraints and uncertainties. In practice, neglecting sensor restrictions can improve localization. Not generalizable: Some authors only test their strategies on certain datasets or locations. The approach's applicability to different locations and road conditions may be questioned. Papers must demonstrate technique efficacy and robustness. Limited scalability: A localization method's scalability is its ability to execute efficiently and accurately in more complex environments or larger-scale deployments. Papers may need to test their scalability for AV deployment. The reviewed works discuss LiDAR-based AV robust localizations and offer novel methods to improve precision and robustness. LiDAR and sensor fusion help localize in difficult off-road or dynamic circumstances. These methods work in many scenarios. Disadvantages exist. Sensor noise, occlusions, and range can affect LiDAR localization precision. Some methods are computationally intensive for resource-limited systems. The evaluated works explore weather, scale ambiguity, information loss, rapid motion, occlusions, and optimization. More research is needed to overcome these constraints and optimize present methods. This includes overcoming sensor limits, considering real-time implementation, and adding sensor modalities and fusion technologies for precision and robustness.

3.Methods

Many other industries, including AV, make use of deep learning nowadays. The use of artificial neural networks (ANNs) is beneficial to the process of deep learning. This section discusses the CNN and IHS methodologies that were utilized in the theoretical component of this work. This section will discuss the suggested organizational structure.

3.1Convolutional neural networks (CNN)

Deep learning with neural networks is a subfield that falls under the general heading of the field of machine learning. Machine learning is illustrated as an example of an application of artificial intelligence [36]. CNN is well-known for its deep learning model architecture, which is used for object recognition, images categorization, and geographic localization. CNNs are built with multiple layers for the purpose of efficiently processing visual data. CNNs are built with layers that are convolutional, pooling, and fully connected (FC) [37]. Small filters detect edges and textures in a CNN layer. These filters produce feature maps from local patterns. Pooling layers shrink feature maps. Max pooling chooses the highest value in a region. This conserves computation and data. FC layers, sometimes called dense layers, link every neuron from the previous layer to the current layer. They aid data learning and classification. In order to extract spatial properties from the input image, the convolutional layers perform filtering on it. These filters locate localizations information by identifying edges, corners, and textures. Pooling reduces the number of sample features on a map, hence reducing the spatial dimensions of the map while maintaining all of the essential information. The model will become both more effective and easier to compute going forward. In order to improve real-time speed and efficiency, the CNN model ought to be improved to lessen the amount of computing complexity involved in localization. It can be computationally expensive to process massive volumes of visual information in real time for the purpose of localization. Because the processing complexity of the CNN model has been reduced, the localization process may now be completed more quickly and in a manner that is more suitable for platforms with limited resources, such as embedded devices and driverless vehicles [38]. Researchers have proposed a number of different approaches to improve the CNN's design. Reducing the size of the filters used or using depth-wise separable convolutions can help reduce the number of parameters and calculations. Both the global average pooling and the spatial pyramid pooling reduce the amount of space that is used while maintaining critical characteristics. The computational complexity of CNN models can be reduced by the use of pruning, quantization, and low-rank approximation without sacrificing speed [39]. These strategies eliminate unnecessary computations and parameters, which ultimately results in a model that is more effective. Convolution is a mathematical operation that can be used on higher-dimensional functions, such as images. There are several steps required, including element-wise multiplication, summing, and sliding a filter across the input image. This method is used to perform image transformations. Images can be thought of as functions that only exist in two dimensions. For example, A is the two-dimensional input (the picture), K is a two-dimensional filter with a size of m by n, and F is a two-dimensional feature map. Convolution of A and K results in the variable F, which can be formally stated as [40] (Equation 1):

$$F(i,j) = (A \times K)(i,j) = \sum_m \sum_n A(i+m, j+n)K(m,n) \qquad (1)$$

The spatial dimensions of the input data (both width and height) are typically reduced by a pooling layer in a neural network while the critical information is kept intact. This is one of the common functions of a pooling layer. The pooling process is applied in an independent fashion to every feature map and input channel. The max pooling operation is the most popular form of pooling, and it takes the maximum value that can be found within of a sliding window or filter. The filter iteratively moves through the input data using a step size that has been previously specified, picking the value that is highest within the window at each place to serve as the output for that region. This process is repeated for each region, which ultimately results in a map that has decreased spatial dimensions and features that are more sparsely distributed [41].

The final layer with complete connectivity outputs class scores to a classifier [42]. Softmax and support vector machines (SVMs) are the primary classifiers of ConvNets. Softmax Performance: Exponential Relationship The output layer of a neural network is used for classification. As seen in Equation 2, [40]:

$$\sigma(x_j) = \frac{e^{x_j}}{\sum_{k=1}^{n} x_k} \qquad (2)$$

Parameters and hyper parameters are required for CNN models. They regulate network conduct during training and inference [43]:

Weights: Learnable parameters interconnect various CNN layers. They keep track of training data, knowledge, and patterns. In weight training, gradient descent is used to reduce the loss function.

Biases: Each neuronal layer possesses additional properties. They allow neurons to activate even in the absence of input, thereby granting the network flexibility. Imperfections enhance model-data fit.

Hyper parameters

Learning rate: The optimization algorithm increases weights during training at a specific step size. High learning rates may overshoot ideal weights, whereas low learning rates may impede convergence. Practical training requires the correct learning rate.

Number of layers: A CNN's convolutional, pooling, and fully connected layers (FCL) are hyper parameters. Deeper networks capture more complicated information but require more computer resources and training data to avoid overfitting.

Filter size: The input data's convolutional kernels' spatial dimension is the filter size. It determines convolutional layer receptivity. Smaller filters catch minute details, while larger filters collect global patterns. Complexity and input data determine filter size.

Pooling layers reduce feature map spatial dimensions. Hyper factors that impact down sampling include pooling type (max, average) and size (pooling window dimensions). Pooling reduces computing complexity, extracts dominating features, and increases model translational invariance.

Activation function: The CNN model's activation function introduces non-linearity. The activation function depends on the model's capacity to capture complex interactions and avoid vanishing gradients. Rectified linear unit (ReLU), sigmoid, and tanh are standard.

3.2IHS for RGB and depth merging

Optical RGB sensors are the most common information source due to their adaptability and affordability. These sensors recognize, segment, monitor, and localize objects in color images using computer vision techniques. This discipline has been greatly enhanced by deep learning technology [44]. By integrating and injecting useful information from multiple input images, image fusion produces a single output image that is more beneficial and effective than all input images [45]. IHS improving fusion is widely utilized. This color analysis technique is used in image processing. Among the enhancements are spatial precision, feature perfection, and data integration. Spectral information frequently impacts the color and saturation of an image. The visual system indicates that amplitude variation is manageable and has no effect on spectral properties [46]. IHS is a third-order method due to the RGB-IHS conversion paradigm. The transform kernel has a dimension of 3×3. Numerous published studies employ unique IHS transformations with matrix value variations [47] (Equation 3):

$$\begin{bmatrix} I \\ V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{-1}{\sqrt{6}} & \frac{-1}{\sqrt{6}} & \frac{-1}{\sqrt{6}} \\ \frac{-1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 \end{bmatrix} \quad (3)$$

$$where\ H = \tan^{-1}(\frac{V_2}{V_1}), \qquad S = \sqrt{V_1^2 + V_2^2}$$

Intensity, Hue, Saturation, Red, Blue, and Green. V1 and V2 are midpoints. Through the use of special case processing, the method adjusts the levels of intensity, color, and saturation between 0 and 255. The hue, the saturation, and the intensity may all be seen with RGB cubes. Hue (H), saturation (S), and intensity (I) may all be distinguished in color photographs. Formulas in geometry can be used to convert RGB to IHS. The H that is given by [47] (Equation 4 and 5).

$$H = \begin{cases} \theta & if\ B \le G \\ 360 - \theta & if\ B > G \end{cases} \quad (4)$$

, where:

$$\theta = \cos^{-1}\left\{\frac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{(R-G)^2+(R-G)(G-B)}}\right\} \quad (5)$$

The saturation S is given by [47] (Equation 6):

$$S = 1 - (\frac{3}{(R+G+B)})[MIN(R,G,B)] \quad (6)$$

The Intensity (I) is given by [47] (Equation 7):

$$\mathrm{I} = \frac{1}{3} \times (\mathrm{R} + \mathrm{G} + \mathrm{B}) \quad (7)$$

3.3Methodological investigation

Localization is certainly one of the most essential, if not the most important component of autonomous transportation. This study's most important contribution is the achievement of outstanding localization accuracy for AVs using only a vision-based system, which paves the way for the elimination of expensive sensors such as LiDAR and RADAR. Using visual data and deep learning models, this paper investigates how to improve the precision of AV localization. Depth perception is a crucial aspect of perception for precise localization. The proposed CNN's accuracy was evaluated using an RGB image with depth and the IHS method. The camera image with unprocessed pixels is the sole input for the independent localization method. The outcome is the predicted position.

A CNN with a modified AlexNet architecture is trained offline in MATLAB using RGB images. This is accomplished by following the steps that have been outlined. After that, the trained network is implemented in the Python programming language so that the CARLA simulator may be used to perform real-time localization of an AV. A more in-depth explanation of each stage of the procedure may be found below:

Data collection and preprocessing: - Gather a set of RGB images from the CARLA simulator along with their corresponding ground truth positions of the AV.
 - The dataset needs to undergo preprocessing, including resizing the images to a consistent size and normalizing the pixel values within an appropriate range (e.g., [0, 1]).

Offline and utilizing MATLAB, CNN architecture training and design:
 -Create a customized version of the AlexNet architecture using MATLAB to satisfy the requirements of the localization project. To accommodate both the dataset and the desired output, it may be necessary to modify the number of layers, the sizes of the kernels, and the number of neurons present in FC layers. The dataset should be divided into the training set and the validation set during model training.
-Train the modified AlexNet CNN using backpropagation and stochastic gradient descent (SGD) on the training set. To monitor how well the model performs on the validation set to prevent overfitting, by modifying the model's hyper parameters and regularization procedures for optimal results. After CNN training has been completed in MATLAB, the trained model must be exported in a Python-compatible format. Tensor Flow's that are compatible with Python were used in this work.

Real-Time Python Implementation: To communicate with the CARLA simulator, a Python client-server connection was established.

- Import the exported CNN model into Python and configure it to generate predictions and accept input images.
- Establish a connection to the CARLA simulator as a client in order to acquire real-time RGB images captured by the vehicle's camera.
- To get the incoming photos ready to be uploaded to CNN, it will need to process them by resizing and normalizing them, for example.
- Using the imported CNN model, make a prediction about the position and orientation of the vehicle based on the processed images.
- Send the vehicle's anticipated location back to the CARLA simulator in order to update the simulation with the vehicle's new position.

Real-time translation in CARLA: Ensure that the Python script is properly integrated while the CARLA simulator is operating so that real-time translation can be performed.

-Using the CNN model, the Python script predicts the car's position and orientation as it travels within the CARLA simulation. These predictions enable the script to precisely update the vehicle's location. The proposed localization of AV is depicted graphically in *Figure 2*.

CNN's architecture was based on the original AlexNet structure [43]. Using an input image of 256×256 pixels, a 21-layer CNN is constructed to enhance output accuracy and reduce error with minimal training time. This network is trained using the stochastic gradient descent method (SGDM) optimization technique [48], is always more rapid and effective than gradient descent method (GDM) [49].

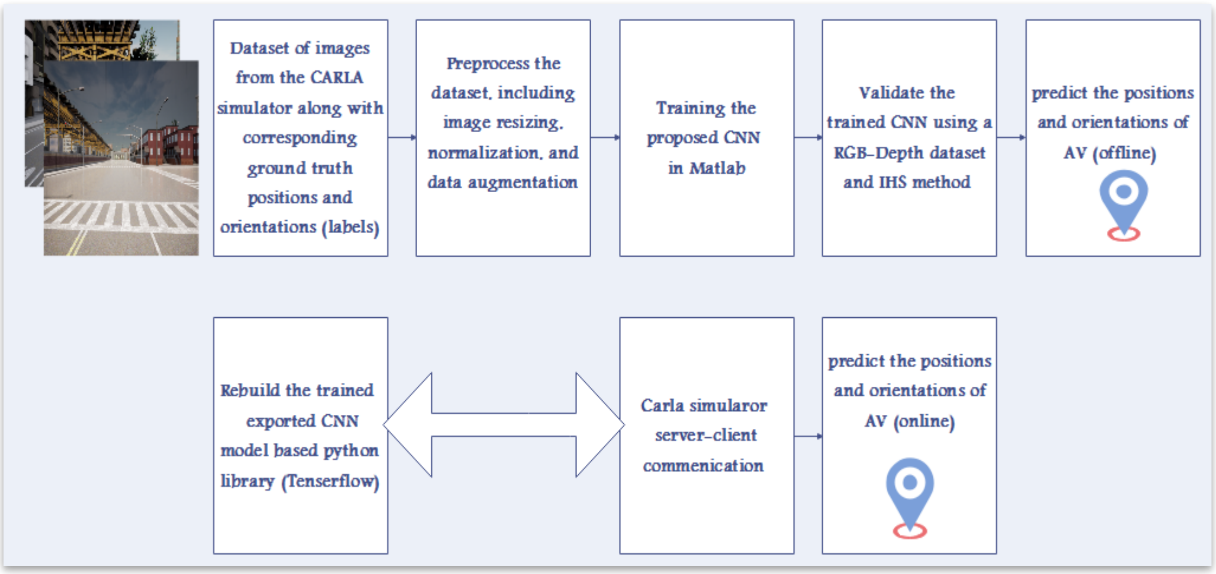

Dataset of images from the CARLA simulator along with corresponding ground truth positions and orientations (labels)
Preprocess the dataset, including image resizing, normalization, and data augmentation
Training the proposed CNN in Matlab
Validate the trained CNN using a RGB-Depth dataset and IHS method
predict the positions and orientations of AV (offline)
Rebuild the trained exported CNN model based python library (Tenserflow)
Carla simularor server-client commenication
predict the positions and orientations of AV (online)

Figure 2 Illustrates the work plan for the AV localization process

Eleven different classes are utilized in the classifier phase, and this is determined by the total number of streets. This work improved AlexNet to reduce the computational complexity and the amount of time required for training while simultaneously increasing CNN's adaptability and effectiveness. The number of convolution layers was decreased from five to three. In the newly proposed architecture, the useful properties of input images are extracted by a first convolutional layer that contains 128 filters arranged in an 11-by-11 grid. In order to further improve the features that were recovered, the second convolutional layer uses 512 filters that are 5 by 5. The next three layers are totally connected (FC), and they contain 384 filters that are 3×3. The application of mathematical operations to features through the use of FC layers helps with classification. The softmax function generates class score probability distributions in the last FC layer of the algorithm. The pooling layers come next, after the convolutional layers, and they lower the spatial dimensions while capturing higher-level characteristics. *Figure 3* illustrates how the use of ReLU activation functions can both induce non-linearity and speed up the learning process.

Depth improves generalization across image sizes but can over-specify. This study used a modified AlexNet architecture with three convolutional layers instead of five. This change optimized computational performance and layer weight management. CNNs still work well with many image categories despite this adjustment. The flowchart of the proposed network is shown in *Figure 4*.

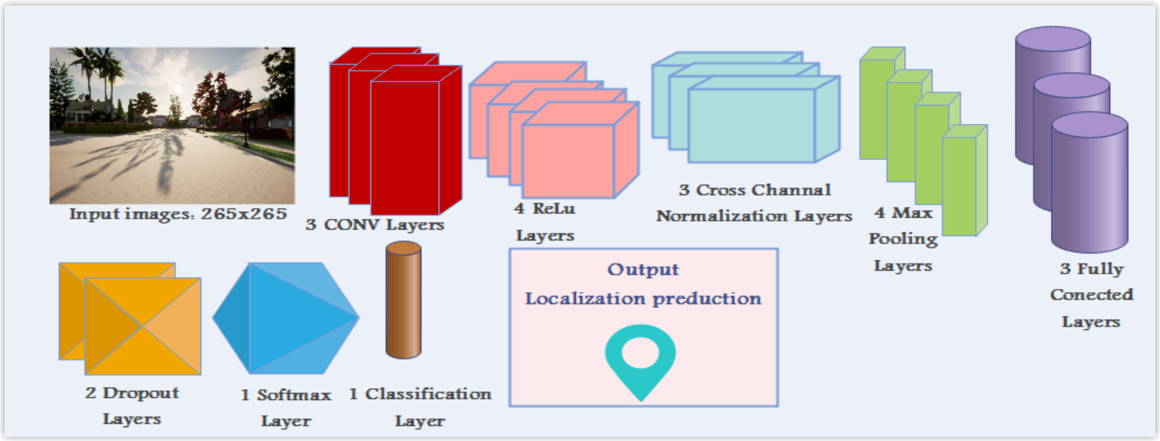


Figure 3 Demonstrates CNN's proposed design for the 21st layer

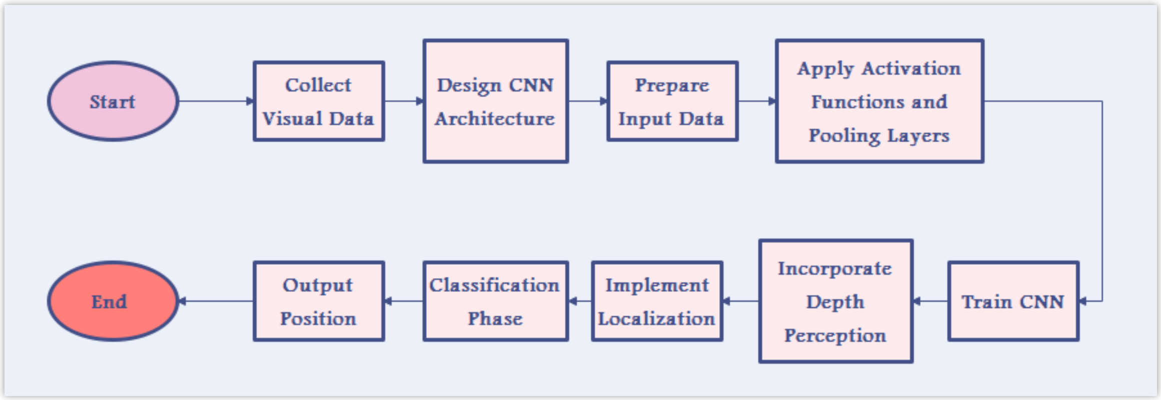


Figure 4 Flow chart of the proposed network

The algorithm of the proposed network is as follows:
Start
Input: RGB Images
Depth Images (from the IHS method)
Preprocess Images
Construct CNN Architecture (Modified AlexNet):
- Input: 256 × 256-pixel RGB Images
- Convolutional Layer 1: 128 filters (11×11)
- Convolutional Layer 2: 512 filters (5×5)
- FC Layers: 3 layers with 384 filters (3×3)
- Softmax Activation for classification
Training the CNN:
- Stochastic Gradient Descent Method (SGDM) optimization technique
Localization Process:
- Input: RGB Images
- Extract features using the CNN
- Predict position using the trained CNN

Evaluation of Localization Accuracy:
- Use RGB images with Depth information (from IHS method)
- Assess the accuracy of the CNN predictions
Improvements to AlexNet Architecture:
- Reduce the number of convolutional layers from 5 to 3
- First convolutional layer: 128 filters (11×11)
- Second convolutional layer: 512 filters (5×5)
- Three FC layers: 384 filters (3×3)
- Apply ReLU activation functions for non-linearity

Pooling Layers:
- Reduce spatial dimensions and capture higher-level features
- Applied after convolutional layers

End

4.Results

Training a supervised neural network requires a significant amount of data. The system architecture is trained and evaluated with the use of RGB images, depth images, and their corresponding coordinates. It was necessary to use a simulator in order to construct the pose label because this data was not readily available to the public. Vehicles equipped with video sensors and the CARLA system is able to record images at a predetermined frame rate. Examining how your AV algorithms react to shifts in illumination, visibility, and road conditions can be done by simulating different daytime situations (such as morning, noon, and evening) and weather conditions (such as sunny, cloudy, rainy, and foggy). The collection of images that make up this map comes from a variety of different driving conditions, see *Figure 5*. This helps uncover potential obstacles and opportunities for improvement, which ultimately results in autonomous systems that are more resilient and flexible. The generation of images with the appropriate input size for network design can be achieved by adjusting the image size and the field of vision. Images in RGB and depth can be produced by the sensor. For the purpose of producing the dataset, this study made use of the open-source CARLA simulator. With the CARLA simulator, realistic training and testing of algorithms for autonomous driving may be accomplished [50]. During the testing process, images from the 3,279 RGB training dataset are combined with the depth photos taken by the camera sensor. Data in RGB and depth can improve the accuracy of localization. The city center, nearby districts, and woodland areas are all included on the map that spans a total area of 120,000 square meters, the top view of town 1 show in *Figure 6*. The CARLA simulator's 10 city streets as well as the tunnel are depicted in *Figures 7*, this configuration corresponds closely to the town plan described in [50]. This area is accessible to AV, which can then go around and collect data. A predetermined number of photos are taken by the AV.

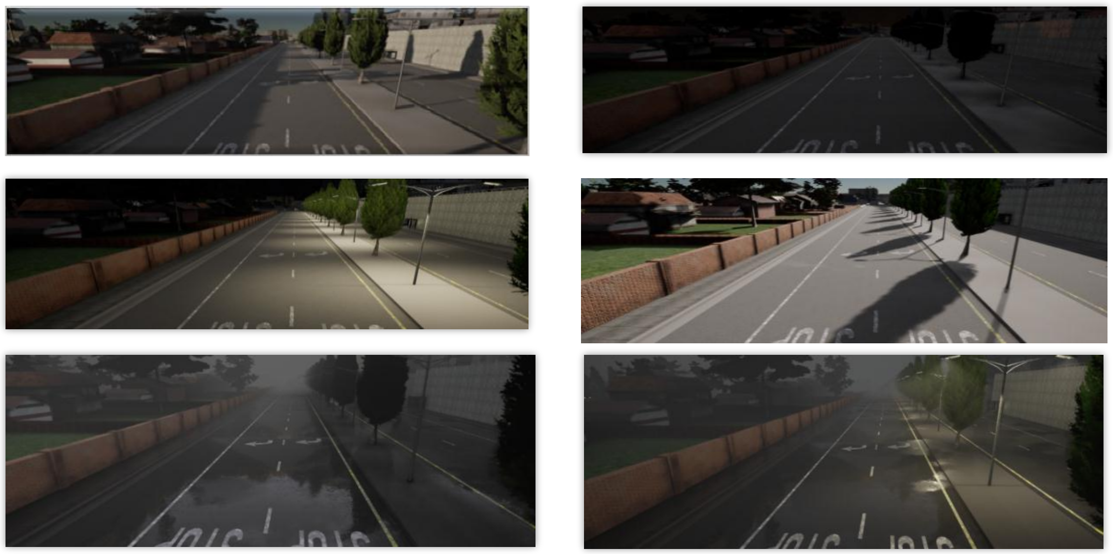


Figure 5 A Street in Town 1, different daytime (e.g., morning, noon, evening) and weather conditions (e.g., sunny, rainy, foggy)

IHS is applied to RGB and depth photos to improve dataset accuracy, comparable to the procedure employed in our published paper [51], see *Figures 8*. This method combines RGB and depth photos to depict the environment better. The automated car follows roadways in the simulated map. The car navigates and interacts with the virtual environment using RGB and depth camera sensors. This study uses CARLA to capture RGB and depth photos from several points in a virtual metropolis. These images, plus the IHS approach, enable precise localization and evaluation of the proposed localization method.

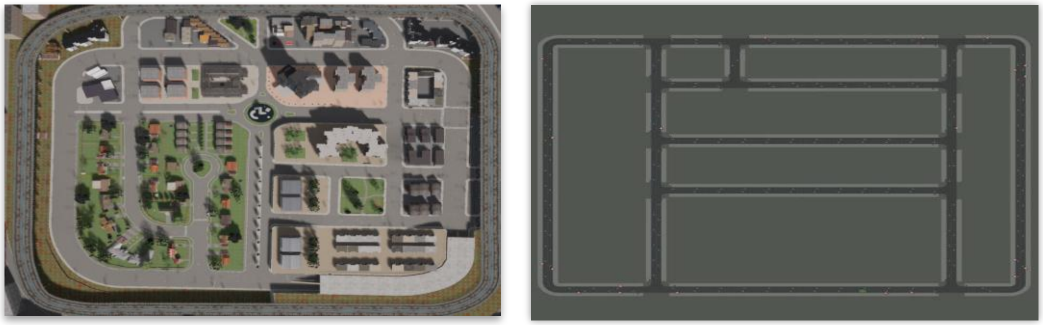


Figure 6 The top image of the proposal map in the CARLA (open-source) simulator environment.

Figure 7 The RGB image of a city roadway and a tunnel

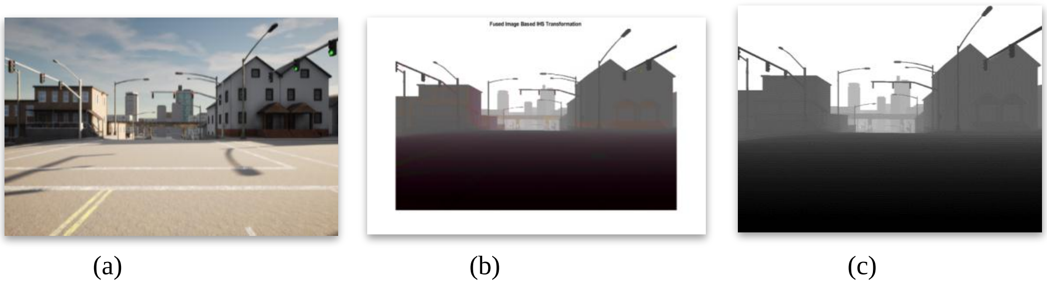


Figure 8 Images in (a) RGB format and (b) Image in RGB depth generated by the CARLA simulator, (c) the result of Combined RGB and depth pictures based on the IHS

Table 1 shows the CARLA simulator dataset used to localize AV. The dataset has 256×256 photos with three RGB channels. Three thousand two hundred seventy-nine pictures are trained and tested in the localization model. CARLA, a popular open-source autonomous driving simulator, captured these photos [50]. The simulator recreates urban and road conditions. The dataset has 11 classes. These classes may represent localization targets or landmarks the AV needs to locate accurately. The classes may feature traffic signs, lights, buildings, pedestrians, or other urban aspects. Variable image counts per class suggest an imbalance in instance distribution across classes. Some classes have more photos than others. This imbalance can influence localization model training and performance; therefore, data augmentation or class balancing may be needed during training. This dataset contains CARLA simulator photos of urban settings and situations for training and testing AV localization models. Data augmentation involves horizontally flipping photos to expand dataset size and curve representation.

Table 1 Simulator CARLA dataset

Resolution of the image	**No. of images**	**No. of classes**	**No. of images/class**
256×256×3	3279	11	Variable

The numerical repercussions, experimental setting, and training procedure parameters are outlined in *Table 2*, which can be found below. In order to identify the best possible configuration for training the CNN, its hyper parameters were optimized through a process of trial and error. During the training, which lasted for a total of 40 epochs, the complete dataset was fed into the network for a total of 40 times. The magnitude of the weight adjustment increase was defined by the initial learning rate, which was initially set to 0.001. This occurred during the training method. This variable has the potential to have a significant impact on the training dynamics and convergence pace of the network. The processing of eight samples was done throughout each iteration of the training procedure. The amount of training samples that are utilized during a single forward and reverse iteration of the network is referred to as the batch size. It has an effect on the amount of memory that is used during the training procedure, the computational efficiency, and the convergence behavior. The lesson is carried out on a computer that has an Intel Core i7-10510U processor and 8 gigabytes of RAM. The clock speed of the processor has been measured at 1.80 GHz as its normal clock speed, with a turbo clock speed of 2.30 GHz. These specs shed light on the computational resources that were put to use throughout the training process. Because MATLAB code was used in the training implementation, it may be deduced that the training technique and all of the associated actions were carried out inside the confines of a MATLAB-based programming environment. Notably, these particulars offer information regarding the experimental setup and parameters that were utilized, even though it is possible that they are not required in order to conduct an exhaustive analysis of the trained CNN model's efficiency or performance. A complete analysis and debate necessitates the inclusion of additional data and information and real time implementation.

Table 2 Parameters of training process

Experimental Setup and Parameters	**Value**
Epochs	40
Learning rate	0.001
Batch size	8
CPU	Intel Core (i7-10510U)
RAM	8 GB
CPU clock rate	1.80 GHz (base), 2.30 GHz (turbo)
Training environment	MATLAB

Equation 8 represents the mean square error (MSE) loss function that assesses the model's accuracy. The MSE loss function quantifies the average squared difference between anticipated and actual values to assess model fit training data. Minimizing MSE loss helps the model forecast accurately. The actual location is Y, and Y ̂ is the expected position [52] (Equation 8):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y - \hat{Y})^2 \qquad (8)$$

Table 3 Presents the experimental setup and parameters.

Within the experimental framework, the SGDM optimizer performs training on a backpropagation neural network. The objective of using a CNN that is based on feature selection is to localize AV. Using this strategy, useful image features can be selected to minimize the dimensionality of the data and improve localization. CNN model training elements are different depending on the task. A vehicle's location can be determined with the use of edge detection, color histograms, texture descriptors, and other visual signals. In order to evaluate the trained model, the CNN model dataset is divided into a training set consisting of 80% of the data and a test set consisting of 20%. Eighty percent of the data is used to optimize and learn patterns, while the remaining twenty percent is utilized to test the model on data that it has not before seen. The generalization of a model may be evaluated with machine learning by using this 80/20 train-test split. In 99 minutes, the CNN model will have learned from the training data and will have adjusted its parameters. Training time is determined by the complexity of the model, the quantity of the dataset, and the resources available on the computer. See *Figure 9* for further information on how 2,680 iterations carried out throughout 40 epochs led to an accuracy of 95.49 percent.

Based on the network trained using MATLAB, have proceeded with the following steps in this wok:

AV localization outcome in MATLAB:

- In the our published paper [51], AV localization results were obtained using an offline CNN-based MATLAB implementation. According to the project's specifications, the CNN architecture was modified and trained on a dataset of RGB images from the CARLA simulator. In the published paper, the localization precision obtained by the MATLAB-based implementation was discussed and presented. This included metrics such as MSE and other evaluation measures, demonstrating the accuracy of the trained CNN model in predicting the position of the AV.

Python reconstruction of CNN:

- The current work seeks to extend and complete the published paper by implementing the trained CNN model in Python.

-The previously developed and trained CNN model in MATLAB is exported in a Python-compatible format (Tensor Flow).

In Python, the model is reloaded and the configurations necessary for real-time predictions are set.
Real-Time Simulation Using the Python:
-The CARLA simulator acts as the server, while Python is used as the client to establish a connection using the Python API.
-During simulation navigation, the Python application captures real-time RGB images from the CARLA simulator's camera.
-These images are processed (e.g., resized and normalized) to conform to the CNN model's input specifications.
-Based on the processed images, the imported CNN model predicts the AV's position and orientation and provides real-time localization data.
CARLA simulator as server:
-During the simulation, the CARLA simulator functions as a server, providing the environment and receiving commands from the Python script (the client).
-The Python script continuously transmits images to the CARLA simulator for localization and receives updates on the AV's position and orientation as it moves through the simulated environment.

By recreating the CNN model in Python and connecting it to the CARLA simulator via the Python API, this study seeks to demonstrate the real-time performance of the proposed CNN-based AV localization technique in a dynamic simulation environment. The combination of MATLAB for offline training and Python for real-time deployment enables a thorough and efficient evaluation of the AV localization system, yielding valuable insights for future advancements in autonomous driving research.
Comparison and modification between our published paper and this work:

Number of Layers: This work employs a CNN with 21 layers, which is a little shallower than the published work's CNN, which has 22 layers. The change in the number of layers may have been done to make the computation more effective without significantly losing accuracy.

Size of initial Filter: The initial layer of this work features 128 filters that are each 11×11 in size, whereas the published work uses 256 filters that are each the same size. The decision to use fewer filters may have been taken to simplify the model and use fewer computational resources.

Number of images for instruction: The dataset used in this work, which has 3279 photos, is smaller than the dataset used in the published paper, which has 5283 images. The dataset has shrunk (*Table 4).*

Accuracy: The accuracy of this work is higher than that of the published work, which was 94.76%. The enhancements in localization accuracy were probably made possible by changes made to the CNN architecture and training procedure.

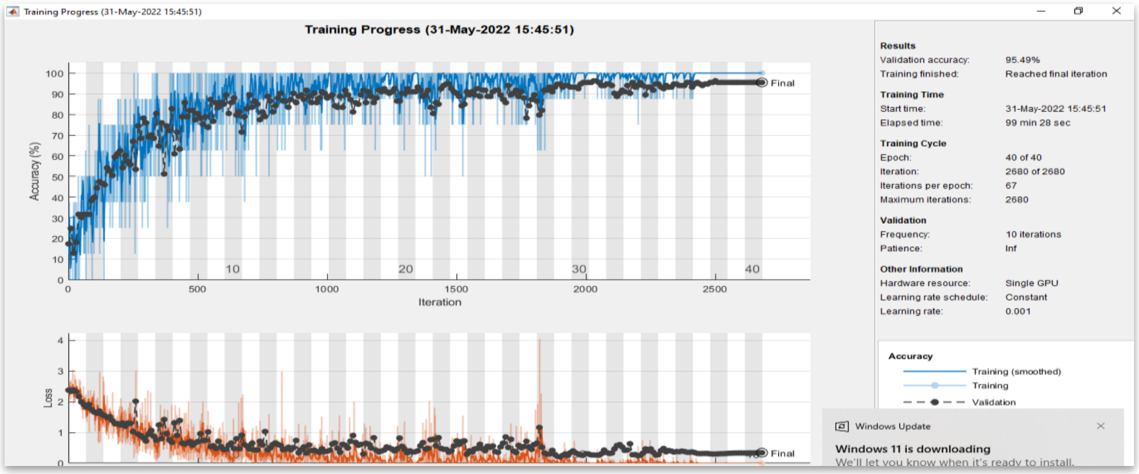
Precision: From earlier work to this study, precision increased from 0.9 to 0.96. This improvement shows that the current methodology can better discover and classify meaningful AV localization instances. An accuracy rating of 0.96 indicates a more refined and effective system with fewer false positives (FP) and better localization results. This study's precision improvement supports improving AV location accuracy, see *Table 5.*

Modification:
Localization Capability: The addition of yaw estimation in addition to (x, y) localization is the main change between this work and the published work. This work extends the localization capacity and makes it acceptable for actual autonomous driving applications by estimating the vehicle's orientation (yaw), which is crucial for precise and accurate navigation in real-world circumstances.

Implementation: This work uses Tensor Flow to implement the trained CNN model in Python and connects it in real-time to the CARLA simulator. The published study, in contrast, uses MATLAB to conduct offline testing. The dynamic and real-world testing made possible by the online implementation provides for a more accurate assessment of how well CNN performs in a virtual setting. The results of using a CNN for the localization of AV in the CARLA simulator are shown in the *Figure 10*. A comparison of the desired and actual vehicle positions as determined by the suggested CNN-based AV localization system is shown in *Figure 10*. The findings displayed in this figure show outstanding performance in precisely determining the vehicle's position. The localization method was highly precise and reliable, as evidenced by how closely the actual positions match the anticipated positions. The CNN model's ability to precisely locate the car using RGB photos is demonstrated by the small difference between the desired and real placements. In a realistic virtual world, the experiment attempted to assess the precision of the CNN-based localization strategy. The outcomes show that the AV-based CNN localized the vehicle's position and orientation with a high degree of accuracy. The exact predictions of the AV's location were made possible by the CNN's capacity to process and evaluate RGB images from the CARLA simulator. For autonomous driving systems to be safe and dependable, the vehicle needs to be able to navigate and operate successfully in challenging real-world situations. The *Figure 10* showcases the results of high-accuracy AV localization using a CNN in the CARLA simulator's Town 1 environment. The purpose of the experiment is to compare the desired AV position, represented by a blue point, with the actual AV position obtained through the CNN-based localization, depicted as an orange point. The presence of a blue point on the figure indicates the target or desired position that the AV should ideally be located at this position is likely determined by ground truth data. On the other hand, the orange point represents the actual AV position obtained through the CNN-based localization method. The localization CNN has been trained to predict the AV's position and orientation accurately using input data, which may include RGB images, depth data, or both. A successful CNN-based localization will be characterized by the orange point closely aligning with the blue point, demonstrating the network's ability to accurately estimate the AV's position in the simulated environment. A small distance between the blue and orange points indicates a high level of accuracy in the CNN-based localization, signifying that the AV's position prediction closely matches the desired position.

Table 3 Experimental setup and parameters

Method/Technique	**Description**
Optimizer	SGDM
Neural Network	Backpropagation-based
Feature Selection	CNN-based approach selects informative image features for localization
Input Dimensionality	Reduced by selecting relevant features from the image
Features	Includes edge detection, color histograms, texture descriptors, and other visual signals
Dataset Split	Training set (80%) and test set (20%)
Training Duration	99 minutes (depends on model complexity, dataset size, and computational resources)
Train-Test Split	Commonly used 80-20 split for balancing training and evaluation data
Model Performance	Achieved 95.49% accuracy after 2680
iterations	over 40 epochs
Simulation Results	Provided for CNN-based street identification and coordinate work
Output Prediction	*Figure 11* shows the successful prediction of AV position and orientation
Dataset Evaluation	Datasets categorized for visual location recognition
Coordinate Retrieval	K-Means clustering used to locate image coordinates after CNN network determines the street class
Average Distance	Between two photographs: 3.01 meters vertically and 0.5 meters horizontally when the car travels at 60 km/h
Frame Rate	Camera captures 50 frames per second



Results
Validation accuracy: 95.49%
Training finished: Reached final iteration

Training Time
Start time: 31-May-2022 15:45:51
Elapsed time: 99 min 28 sec

Training Cycle
Epoch: 40 of 40
Iteration: 2680 of 2680
Iterations per epoch: 67
Maximum iterations: 2680

Validation
Frequency: 10 iterations
Patience: Inf

Other Information
Hardware resource: Single GPU
Learning rate schedule: Constant
Learning rate: 0.001

Windows Update
Windows 11 is downloading
We'll let you know when it's ready to install.

Figure 9 The results of the accuracy and loss functions

Table 4 Comparison based on layers and filters

Metric	Previous study [51]	Proposed work
No. of Layers	22	21
Size of First Filter	256 filters (11×11)	128 filters (11×11)
No. of Images for Training	5283 images	3279 images
Localization Capability	x and y	x, y, yaw
Implementation	MATLAB Offline	MATLAB Offline +Python (Tensor Flow) Online

Table 5 Accuracy and precision comparison of the previous and current study

S. No.	Metrics	Previous study [51]	Proposed work
1	Accuracy	94.76%	95.49%
2	Precision	0.9	0.96

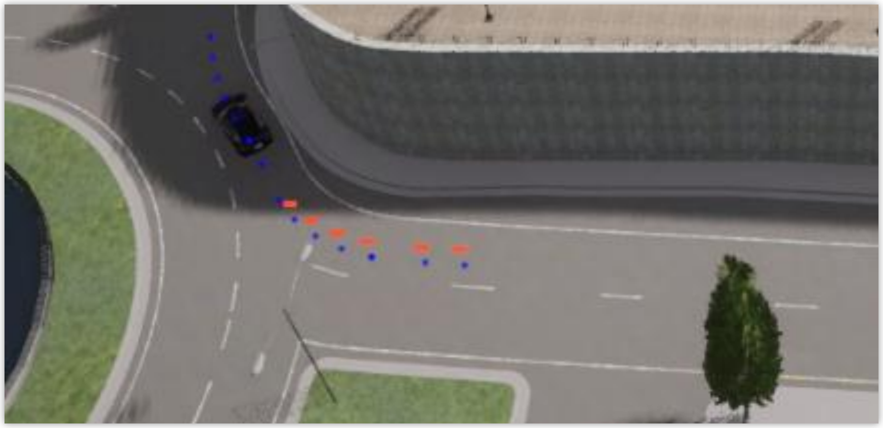


Figure 10 Desired and actual positions

Figure 11 demonstrates the error comparison between the magnitudes of position of the vehicle with the estimated localization. We add the magnitudes of the position from x, and y axis from both actual position and estimated position and plot them against time to compare the error. *Figure 12* shows the orientation magnitude curve for the ground truth and the estimated data.

The strong design and effective training of the CNN model are responsible for the good results in *Figure 11*. With the help of the modified AlexNet architecture, the network in this work is able to extract meaningful representations from the RGB images, collecting crucial elements for accurate localization. Additionally, the MATLAB training procedure makes sure that the CNN is well-optimized and generalizes to real-world circumstances successfully. The desired and actual vehicle rotations (yaw) as determined by the same CNN-based AV localization system is contrasted in *Figure 12*. The outcomes displayed in this *Figure 12* likewise demonstrate very strong performance in precisely determining the vehicle's direction. The model's ability to precisely forecast the vehicle's rotation based on the visual input is demonstrated by the fact that the real vehicle's yaw closely resembles the planned yaw. For autonomous driving to be successful, the vehicle's rotation must be correctly estimated. When traveling through complicated and dynamic situations, the AV can make well-informed decisions thanks to accurate yaw estimate. An important benefit of the suggested approach is that the CNN model can predict yaw with accuracy using RGB images without the need for pricey LiDAR or RADAR sensors. Overall, the reliability and efficiency of the proposed CNN-based AV localization system are well supported by *Figures 11* and *12*. Excellent position and yaw estimation are displayed in the results, showcasing the approach's potential for use in practical autonomous driving scenarios. A viable option for AV, removing the need for expensive sensors and improving the overall safety and effectiveness of autonomous driving systems, is the accurate localization and reliable yaw estimation obtained by a light and effective CNN model.

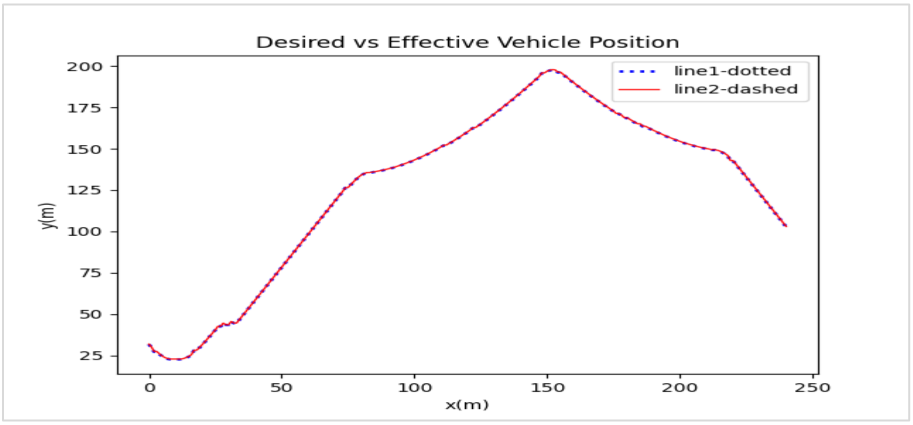


Figure 11 Desired vs effective vehicle position

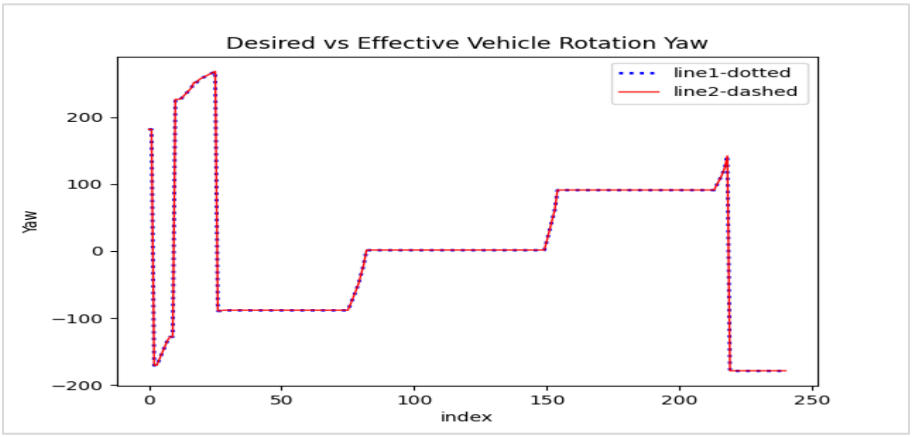


Figure 12 Desired vs effective vehicle rotation Yaw

5.Discussion

To assess the results quantitatively, the mean squared error (MSE) was computed for pose, encompassing error analysis for both position and orientation. *Table 6* presents the MSE values, illustrating the disparity between desired and actual positions within a CNN-based AV localization system. The objective of this study is to evaluate the CNN model's accuracy and precision in localizing the AV across real-world scenarios. MSE stands as a widely employed metric, gauging the variance between predicted and actual positions, thereby offering valuable insights into the performance of localization algorithms.

Within this context, the x-coordinate (horizontal position) manifests an MSE of 0.039, while the y-coordinate (vertical position) demonstrates an MSE of 0.0099, and the vehicle rotation (yaw) showcases an MSE of 0.0047. The minimal MSE values underscore the high accuracy and precision achieved by the CNN-based AV localization system. These marginal deviations between desired and actual positions signify the model's adeptness in predicting the AV's coordinates in both horizontal and vertical dimensions.

The promising results presented affirm the robustness and reliability of the CNN-based localization system. This, in turn, establishes a solid basis for real-world applications in the realm of autonomous driving. The system's commendable accuracy holds paramount significance in ensuring the secure and efficient navigation of AVs, particularly in intricate and dynamic surroundings. It's worth noting that the reported MSE values of 0.039, 0.0099, and 0.0047 denote exceptional performance. Nonetheless, further analysis and validation using diverse real-world datasets and challenging scenarios are imperative to validate the model's generalization capabilities and real-world viability. Furthermore, juxtaposing the CNN-based system with alternative localization methods and considering computational efficiency will elucidate the system's competitiveness in practical applications.

Collectively, the outcomes depicted in the figure validate the efficacy of the CNN-based AV localization system. This highlights its remarkable accuracy and potential to serve as a pivotal component in achieving dependable and precise autonomous driving capabilities.

Table 6 Quantitative result on localization accuracy MSE

Position	MSE
x-coordinate (horizontal position)	0.0399
y-coordinate (vertical position)	0.0099
Yaw (vehicle rotation)	0.0047

5.1Limitation and recommendations

Real-world driving scenarios are utilized in this research. However, additional data derived from actual driving experiences must be incorporated to fully validate the suggested method and ensure the CNN's performance in real-world scenarios.

Limited dataset size: While the utilization of a lightweight network has its advantages, the model's ability to generalize across diverse road conditions and environments may be constrained due to the relatively small dataset of 3279 photos. Enhancing the performance and robustness of the CNN could be achieved by augmenting the dataset size.

Single camera sensor: The core component of the proposed technique is an RGB camera sensor. While this approach eliminates the need for expensive LiDAR and RADAR sensors, relying solely on a single camera might be limiting in scenarios involving occlusions or complex road layouts. Exploring the fusion of multiple sensors could potentially enhance localization performance overall.

Recommendations:

Data augmentation: To address the challenge of the small dataset size, employing data augmentation methods can expand the training dataset. Techniques like rotation, flipping, and adding noise can diversify the training data, thus improving the CNN's ability to generalize across scenarios.

Sensor fusion: Although the focus of this work is on RGB camera data, future research could delve into sensor fusion techniques. Integrating information from various sensors, such as LiDAR and RADAR, could enhance localization accuracy and robustness under varying driving conditions.

Real-world testing: To validate the efficacy of the proposed strategy in real-world environments, conducting real-world testing using an actual AV is essential. This approach will help identify potential issues and opportunities for improvement.

Error analysis: A comprehensive error analysis should be undertaken to comprehend the limitations of the proposed strategy. Identifying sources of errors and uncertainty in localization can guide future enhancements and optimizations.

Performance in dynamic environments: Evaluating the CNN's performance in scenarios involving moving obstacles, pedestrians, and other vehicles is critical for real-world applications. Assessing the method's capabilities under complex and dynamic situations provides a more realistic evaluation.

The proposed CNN-based method has the potential to address these limitations and implement the suggested strategies in the future, thereby enhancing AV localization accuracy. This, in turn, positions the method as a more practical and reliable option for applications in the field of autonomous driving.

A complete list of abbreviations is shown in *Appendix I.*

6.Conclusion and future work

This study delves into the potential of utilizing a CNN for localizing AVs in urban environments, taking into account diverse meteorological conditions. The CNN-based technique holds promise for achieving moderately accurate AV localization using vision-only data, without the need for costly RADAR and LiDAR sensors. In this study, a practical computational modification of the well-known ALEXNET CNN architecture with fewer layers proves successful. The CNN model is trained offline in MATLAB using raw camera images and associated location-coordinate data for each frame. The utilization of the IHS technique, blending depth images with RGB images, facilitates accurate AV localization. Simulation results demonstrate that the proposed technique attains commendable localization accuracy with enhanced performance within a significantly shorter training time of 99 minutes and 28 seconds. The CNN-based solution outperforms conventional methods, achieving an impressive accuracy rate of 95.49%.

A notable feature of the proposed method is its capacity for seamless integration of offline MATLAB training with real-time implementation in Python through the CARLA simulator. Offline MATLAB training enables researchers to safely explore diverse architectures, hyperparameters, and training methodologies, expediting the exploration of the CNN's design space and yielding a well-optimized model.

Moreover, integrating the CNN model into the CARLA simulator enables dynamic and realistic testing within real-world scenarios. By recreating the CNN in Python and seamlessly interfacing it with CARLA, the model's performance is assessed across varied driving scenarios, weather conditions, and urban environments. Real-time implementation provides rapid insights into the model's behavior and potential refinements. The CNN-based technique demonstrates its ability to accurately and reliably localize AVs across diverse urban areas and weather conditions. The synergy between offline MATLAB training, real-time Python-based implementation, and CARLA simulator integration showcases the model's practical efficacy and sets the stage for future research involving more intricate training and validation scenarios.

Although the MSE values of 0.039, 0.0099, and 0.0047 reflect excellent performance, further investigation and validation using diverse real-world datasets and complex scenarios remain crucial. Rigorous testing of the model's robustness, generalizability, and real-world feasibility will solidify its potential for advancing AV localization technology.

Acknowledgment
None.

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

Author’s contribution statement
Shahad S. Ghintab: Conceptualization, investigation, data curation, writing – original draft, writing – review and editing, data collection, analysis and interpretation of results. **Mohammed Y. Hassan:** Supervision, investigation, writing – review and editing, investigation on challenges.

References
[1] Karur K, Sharma N, Dharmatti C, Siegel JE. A survey of path planning algorithms for mobile robots. Vehicles. 2021; 3(3):448-68.
[2] Chen S, Liu B, Feng C, Vallespi-gonzalez C, Wellington C. 3d point cloud processing and learning for autonomous driving: impacting map creation, localization, and perception. IEEE Signal Processing Magazine. 2020; 38(1):68-86.
[3] Fayyad J, Jaradat MA, Gruyer D, Najjaran H. Deep learning sensor fusion for autonomous vehicle perception and localization: a review. Sensors. 2020; 20(15):1-35.
[4] Reid TG, Houts SE, Cammarata R, Mills G, Agarwal S, Vora A, et al. Localization requirements for autonomous vehicles. SAE International Journal of Connected and Automated Vehicles. 2019:1-16.
[5] Rublee E, Rabaud V, Konolige K, Bradski G. ORB: an efficient alternative to SIFT or SURF. In international conference on computer vision 2011 (pp. 2564-71). IEEE.
[6] Lowe DG. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision. 2004; 60:91-110.
[7] Bay H, Tuytelaars T, Van GL. Surf: speeded up robust features. In computer vision–ECCV: 9th European conference on computer vision, Graz, Austria, 2006. Proceedings, Part I, 2006 (pp. 404-17). Springer Berlin Heidelberg.
[8] Fischler MA, Bolles RC. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM. 1981; 24(6):381-95.
[9] Mur-artal R, Tardós JD. Orb-slam2: an open-source slam system for monocular, stereo, and RGB-d cameras. IEEE Transactions on Robotics. 2017; 33(5):1255-62.
[10] Engel J, Koltun V, Cremers D. Direct sparse odometry. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2017; 40(3):611-25.
[11] Shotton J, Glocker B, Zach C, Izadi S, Criminisi A, Fitzgibbon A. Scene coordinate regression forests for camera relocalization in RGB-D images. In proceedings of the conference on computer vision and pattern recognition 2013 (pp. 2930-7).
[12] Kendall A, Gal Y, Cipolla R. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In proceedings of the IEEE conference on computer vision and pattern recognition 2018 (pp. 7482-91).
[13] Ballardini AL, Fontana S, Cattaneo D, Matteucci M, Sorrenti DG. Vehicle localization using 3D building models and point cloud matching. Sensors. 2021; 21(16):1-19.
[14] Zghair NA, Al-araji AS. A one decade survey of autonomous mobile robot systems. International Journal of Electrical and Computer Engineering. 2021; 11(6):4891-906.
[15] Stenborg E, Toft C, Hammarstrand L. Long-term visual localization using semantically segmented images. In international conference on robotics and automation 2018 (pp. 6484-90). IEEE.
[16] Parisotto E, Chaplot D, Zhang J, Salakhutdinov R. Global pose estimation with an attention-based recurrent network. In proceedings of the conference on computer vision and pattern recognition workshops 2018 (pp. 237-46). IEEE.
[17] Heng L, Choi B, Cui Z, Geppert M, Hu S, Kuan B, et al. Project autovision: localization and 3d scene perception for an autonomous vehicle with a multi-camera system. In international conference on robotics and automation 2019 (pp. 4695-702). IEEE.
[18] Amini A, Rosman G, Karaman S, Rus D. Variational end-to-end navigation and localization. In

international conference on robotics and automation 2019 (pp. 8958-64). IEEE.
[19] Ma WC, Tartavull I, Bârsan IA, Wang S, Bai M, Mattyus G, et al. Exploiting sparse semantic HD maps for self-driving vehicle localization. In IEEE/RSJ international conference on intelligent robots and systems 2019(pp. 5304-11). IEEE.
[20] Yin H, Wang Y, Ding X, Tang L, Huang S, Xiong R. 3D LiDAR-based global localization using siamese neural network. IEEE Transactions on Intelligent Transportation Systems. 2019; 21(4):1380-92.
[21] Wan G, Yang X, Cai R, Li H, Zhou Y, Wang H, et al. Robust and precise vehicle localization based on multi-sensor fusion in diverse city scenes. In IEEE international conference on robotics and automation 2018 (pp. 4670-77). IEEE.
[22] Chen X, Vizzo I, Läbe T, Behley J, Stachniss C. Range image-based LiDAR localization for autonomous vehicles. In international conference on robotics and automation 2021 (pp. 5802-8). IEEE.
[23] Héry E, Xu P, Bonnifait P. Consistent decentralized cooperative localization for autonomous vehicles using LiDAR, GNSS, and HD maps. Journal of Field Robotics. 2021; 38(4):552-71.
[24] Qin T, Zheng Y, Chen T, Chen Y, Su Q. A light-weight semantic map for visual localization towards autonomous driving. In international conference on robotics and automation 2021 (pp. 11248-54). IEEE.
[25] Chu X, Lu Z, Gesbert D, Wang L, Wen X. Vehicle localization via cooperative channel mapping. IEEE Transactions on Vehicular Technology. 2021; 70(6):5719-33.
[26] Li Y, Cai Y, Malekian R, Wang H, Sotelo MA, Li Z. Creating navigation map in semi-open scenarios for intelligent vehicle localization using multi-sensor fusion. Expert Systems with Applications. 2021; 184:115543.
[27] Liu J, Guo G. Vehicle localization during GPS outages with extended Kalman filter and deep learning. IEEE Transactions on Instrumentation and Measurement. 2021; 70:1-10.
[28] Guo C, Lin M, Guo H, Liang P, Cheng E. Coarse-to-fine semantic localization with HD map for autonomous driving in structural scenes. In IEEE/RSJ international conference on intelligent robots and systems 2021 (pp. 1146-53). IEEE.
[29] Ren R, Fu H, Xue H, Li X, Hu X, Wu M. LiDAR-based robust localization for field autonomous vehicles in off-road environments. Journal of Field Robotics. 2021; 38(8):1059-77.
[30] Yanase R, Hirano D, Aldibaja M, Yoneda K, Suganuma N. LiDAR-and radar-based robust vehicle localization with confidence estimation of matching results. Sensors. 2022; 22(9):1-20.
[31] Peng B, Xie H, Chen W. ROLL: long-term robust LiDAR-based localization with temporary mapping in changing environments. In IEEE/RSJ international conference on intelligent robots and systems 2022 (pp. 2841-7). IEEE.
[32] Dauptain X, Koné A, Grolleau D, Cerezo V, Gennesseaux M, Do MT. Conception of a high-level perception and localization system for autonomous driving. Sensors. 2022; 22(24):9661.
[33] Lee J, Back M, Hwang SS, Chun IY. Improved real-time monocular SLAM using semantic segmentation on selective frames. IEEE Transactions on Intelligent Transportation Systems. 2022; 24(3):2800-13.
[34] Kang MS, Ahn JH, Im JU, Won JH. LiDAR-and V2X-based cooperative localization technique for autonomous driving in a GNSS-denied environment. Remote Sensing. 2022; 14(22):1-16.
[35] Han L, Shi Z, Wang H. A localization and mapping algorithm based on improved LVI-SAM for vehicles in field environments. Sensors. 2023; 23(7):1-14.
[36] Raheem F, Abdulwahhab AA. Deep learning convolution neural networks analysis and comparative study for static alphabet ASL hand gesture recognition. Journal of Xidian University. 2020; 14(4):1871-81.
[37] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In proceedings of the IEEE conference on computer vision and pattern recognition 2016 (pp. 770-8). IEEE.
[38] Fujiyoshi H, Hirakawa T, Yamashita T. Deep learning-based image recognition for autonomous driving. IATSS Research. 2019; 43(4):244-52.
[39] Sandler M, Howard A, Zhu M, Zhmoginov A, Chen LC. Mobilenetv2: inverted residuals and linear bottlenecks. In proceedings of the conference on computer vision and pattern recognition 2018 (pp. 4510-20). IEEE.
[40] Wani MA, Bhat FA, Afzal S, Khan AI. Advances in deep learning. Springer; 2020.
[41] Alzubaidi L, Zhang J, Humaidi AJ, Al-dujaili A, Duan Y, Al-shamma O, et al. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. Journal of Big Data. 2021; 8:1-74.
[42] Abdulhussein AA, Raheem FA. Hand gesture recognition of static letters American sign language (ASL) using deep learning. Engineering and Technology Journal. 2020; 38(6):926-37.
[43] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems. 2012:1-9.
[44] Gómez VV. LiDAR-based scene understanding for autonomous driving using deep learning (Doctoral Dissertation, Universitat Politècnica de Catalunya (UPC). 2020.
[45] Mishra D, Palkar B. Image fusion techniques: a review. International Journal of Computer Applications. 2015; 130(9):7-13.
[46] Zhao X. Image fusion based on IHS transform and principal component analysis (PCA) transform. In international conference on computer technology, electronics and communication (ICCTEC) 2017(pp. 304-7). IEEE.
[47] Atiyah HA, Hassan MY. Outdoor localization in mobile robot with 3D LiDAR based on principal

component analysis and K-Nearest neighbors algorithm. Engineering and Technology Journal. 2021; 39(6):965-76.

[48] Hassan MY, Kothapalli G. Comparison between neural network based PI and PID controllers. In 7th international multi-conference on systems, signals and devices 2010 (pp. 1-6). IEEE.

[49] Zhou B, Liu J, Sun W, Chen R, Tomlin CJ, Yuan Y. PBSGD: powered stochastic gradient descent methods for accelerated non-convex optimization. In proceedings of the twenty-ninth international joint conference on artificial intelligence 2020 (pp. 3258-66).

[50] Dosovitskiy A, Ros G, Codevilla F, Lopez A, Koltun V. CARLA: an open urban driving simulator. In conference on robot learning 2017 (pp. 1-16). PMLR.

[51] Ghintab SS, Hassan MY. CNN-based visual localization for autonomous vehicles under different weather conditions. Engineering and Technology Journal. 2023; 41(2):375-86.

[52] Wu Y, Li Y, Ge X, Gao Y, Qian W. An efficient method for calculating the error statistics of block-based approximate adders. IEEE Transactions on Computers. 2018; 68(1):21-38.

Shahad S. Ghintab obtained her master's degree in control engineering from the University of Technology - Iraq in 2014. She previously held the position of Assistant Teacher at Al-Mansour University College within the Department of Medical Devices Engineering. At present, she is pursuing her Ph.D. and is in the research phase within the Department of Control Engineering at the University of Technology - Iraq. Her research interests primarily revolve around Intelligent Control Systems, Robotics, and Microcontrollers.
Email: cse.20.01@grad.uotechnology.edu.iq

Mohammed Y. Hassan successfully earned his Ph.D. in Control and Automation Engineering from the University of Technology-Iraq in 2003. Since 2004, he has held the position of Professor within the Control and Systems Engineering Department at the University of Technology, Iraq. He completed his B.Sc. degree in Electrical and Electronics Engineering at Al-Rasheed College of Engineering, Iraq, in 1989. Furthering his education, he pursued his M.Sc. and Ph.D. degrees in Control Engineering from the same institution in 1995 and 2003, respectively. His research pursuits are centered around Adaptive Control, Intelligent Control Systems, Robotics, and Microcontrollers. He actively engages as a member of the Iraqi Academics Syndicate and the Iraqi Engineers Union.
Email: mohammed.y.hassan@uotechnology.edu.iq

Appendix I

S. No.	Abbreviation	Description
1	ANN	Artificial Neural Network
2	AV	Autonomous Vehicle
3	B	Blue
4	CARLA	Car Learning to Act
5	CNN	Convolutional Neural Networks
6	CCM	Cooperative Channel Mapping
7	CSI	Channel State Information
8	DoF	Degrees of Freedom
9	EKF	Extended Kalman Filter
10	FC	Fully Connected
11	FCL	Fully Connected Layers
12	FP	False Positive
13	FN	False Negative
14	GNSS	Global Navigation Satellite System
15	GPS	Global Positioning Systems
16	GDM	Gradient Descent Method
17	H	Hue
18	HD	High-Definition
19	I	Intensity
20	HIS	Intensity Hue Saturation
21	IMU	Inertial Measurement Unit
22	LiDAR	Light Detection and Ranging
23	LVI-SAM	LiDAR-Visual-Inertial Simultaneous and Mapping
24	MSE	Mean Square Error
25	N	Total Number of Sample
26	NLOS	Non-Line-of-Sight
27	ORB	Oriented Fast and Rotated Briefly
28	RADAR	Radio Detecting and Ranging
29	R	Red
30	RGB	Red Green Blue
31	RGB-D	Red Green Blue-Depth
32	ReLU	Rectified Linear Unit
33	RANSAC	Random Sample Consensus
34	SLAM	Simultaneous Localization and Mapping
35	SGDM	Stochastic Gradient Descent Method
36	SVMs	Support Vector Machines
37	SIFT	Scale-Invariant Feature Transform
38	SURF	Speeded Up Robust Features
39	TP	True Positives
40	TN	True Negatives
41	V1, V2	Values in the Middle
	VEENAL	Variational End-to-End Navigation and Localization
42	X, Y, and $\hat{Y}$	X, Y Actual Location, and $\hat{Y}$ is the Expected Position
43	2D	2 Dimension
44	3D	3 Dimension