Engineering and Technology Journal
Journal homepage: https://etj.uotechnology.edu.iq

CNN-based Visual Localization for Autonomous Vehicles under Different Weather Conditions

Shahad S. Ghintab, Mohammed Y. Hassan

Control Engineering Dept., University of Technology-Iraq, Alsina'a street, 10066 Baghdad, Iraq.
*Corresponding author Email: cse.20.01@grad.uotechnology.edu.iq

HIGHLIGHTS

- A Convolutional Neural Network (CNN) was developed for autonomous localization layer-by-layer in urban driving situations.
- To check the positional accuracy of the CNN, RGB images are combined with depth images using the IHS method.
- With an accuracy rate of 94.74%, the simulation results demonstrated the effectiveness of the suggested strategy.
- The Simulation findings demonstrate the superiority of the suggested technique for different weather conditions.

ARTICLE INFO

Handling editor: Rana F. Ghani

Keywords:
Autonomous Vehicle; Localization; Deep Learning; Different weather conditions; HIS, K-mean Algorithm.

ABSTRACT

Autonomous vehicles (AV) are expected to improve, transform, and revolutionize ground transportation. Previous techniques are dependent on localization employing pricey inaccuracy Global Positioning sensor. Furthermore, the performance loss is caused by drifting errors of Simultaneous Localization and Mapping. Regarding categorizing and analyzing texture, cameras are much more accessible and practical. This work contributes to obtaining high accuracy for AV localization and reducing errors in predicted positions. Based on the light, accurate, and robust proposed Convolutional Neural Network (CNN), it will scale down the computational complexity and shorten the training time. Considering various weather and time of day conditions such as bright sunny, hard rain noon, and wet cloudy noon with a vision-only system Red, Green, and Blue (RGB) low-cost camera sensors. To check the positional accuracy of the CNN, RGB images are combined with depth images using the IHS method. The k-Mean technique evaluates the similarity between a specific image and all street images to obtain precise coordinates. The Simulation findings demonstrate the superiority of the suggested technique for different weather conditions, which has an accuracy of up to 94.74% and a Mean Squared Error MSE in a distance of 0 meters, as opposed to [19], where the MSE in the projected position is 4.8 meters. Another indication of the proposed method's effectiveness is that it yielded reliable results when its validity was tested on images from a dataset that had not been trained.

1. Introduction

Autonomous Vehicle (AV) is a target for many academics and engineers across generations because it is one of the most intriguing engineering projects of the modern era. The globe anticipates using AV to prevent accidents caused by driver mistakes and to conserve parking spaces, especially in urban areas [1]. The autonomous navigation problem is typically broken down into four key elements: perception, localization, path planning, and control. Although various AV systems may differ, they all must give a solution [2]. For operating on small and residential streets, AV requires near centimeter level.

This section discusses current methods that have handled this problem, explains why these methods were invalid for AV localization, and explain why the proposed method is superior to those methods. There are many methods for localization, such as the Global Positioning System (GPS); however, they are sometimes faulty and unavailable due to the lack of network coverage. [3]. In urban canyons, where structures may induce Non-line-of-sight (NLOS) or multi-path errors, the accuracy of a localization estimate derived using Global Navigation Satellite Systems (GNSS) is typically limited [4]. A tiny offset can result in a significant error when readings are accumulated over time, as in the case of the Inertia Measurement Unit (IMU) and Simultaneous Localization and Mapping (SLAM), which are some sources of performance loss caused by drifting errors [5], [6], and [7]. The requirement for an accurate system model is one of the main difficulties in estimate techniques. It can be challenging to create an accurate model in some circumstances, particularly for complex systems with high dynamics. Additionally, most sensors have intrinsic uncertainties that are typically incompatible with the system model, leading to an erroneous model [8]. Since end-to-end learning is made possible by deep learning in this circumstance, it is not necessary to

http://doi.org/10.30684/etj.2022.135917.1289
Received 14 September 2022; Accepted 24 October 2022; Available online 09 December 2022
2412-0758/University of Technology-Iraq, Baghdad, Iraq

individually model the mathematical properties of each sensor. A valuable tool for time-series data processing is the Convolutional Neural Network (CNN). The study should be enhanced by testing it in actual challenging driving situations, similar to Kim et al.'s study [9], which merged GNSS and IMU data using long short-term memory (LSTM), the multilayer recurrent neural networks (RNN) used to improve the accuracy and the stability of the GNSS complete solutions for AV. Parisotto et al. [10] offered end-to-end deep-learning-based SLAM algorithms taught model with pose selection, graph optimization, and local pose estimation models for AV. Chen et al. developed an encode-regress network that creates six Degrees of Freedom (DoF) postures for AV without depth maps [11]. Eigen et al. [12] introduced a multi-scale network to assess depth in outdoor scenarios for AV utilizing high-quality data obtained from ResNet. Byun et al. [13] proposed the relative positioning of the vehicle to the lane and stop-line markers in the lateral and longitudinal axes as an innovative technique to increase the vehicle's positional accuracy. Essential conclusions from earlier experiments employing dated, incorrect approaches for self-driving cars suffer a lack of generalization.

Using CNNs for vision-based localization has several benefits, but its main benefit is its ability to preprocess images through its layers. In recent years, CNNs have shown to be more effective at matching images and detecting objects. They may render conventional feature detection methods obsolete in the future. The co-learning of features with the classifier is one of the critical benefits of the CNN architecture, giving it an edge over hand-crafted features paired with traditional classifiers [14]. The best technique for localizing AV-based camera image data is CNNs. Constant runtime and natural navigation, or the lack of infrastructure improvement, are advantages of deep learning for localization. Unprocessed image features have also been extracted using deep learning and machine learning [15]. A CNN methodology for AV localization was developed because of this precise and trustworthy procedure. However, when using modern methods like deep learning, there are several limitations in previous studies, such as:

- Gathering data on the complexity of driving surroundings using pricey sensors kinds.
- Ignorance of different weather and time-of-day conditions.
- Using networks with multiple layers, which slow training.
- Manage a limited amount of data.

The CNN employed in this paper for autonomous localization uses an RGB camera sensor, which accepts camera images as input and produces position predictions as an output to localize the vehicle accurately. A minimum structure, reliable, and robust proposed network helps achieve great accuracy for AV localization and reduces predicted location error with minimal training time.

This work suggests a CNN by the layer-by-layer approach to develop network architecture from scratch until it finds a workable solution-based visual localization for the AV in urban driving situations. This is under varying weather and time of day conditions, such as bright sunny, hard rain noon, and wet cloudy noon. Regarding categorizing and analyzing texture, cameras are much more accessible and practical. Light Detection and Ranging (LIDAR) operates quite well in explicit situations compared with other sensors used for detection, but its performance accuracy decreases noticeably in wet or snowy settings. The IHS method is used to merge RGB photos with depth images and verify the positional correctness of the CNN. To acquire precise coordinates, the K-Mean technique is used to evaluate how closely one image resembles all street images.

This work's contributions include the following:

- The main innovation of the proposed work is a novel CNN that is computationally effective due to its straightforward architecture.
- Due to the light, precise, and robust suggested network, the proposed work achieves high accuracy for AV localization and reduced error in anticipated positions with minimal training time.
- Used several weather scenarios based on a vision-only system to provide the groundwork for potentially eliminating pricey sensors like RADAR and LIDAR.
- The fact that the proposed method produced accurate findings when its validity was checked using images from a dataset that had not been trained on it is evidence of its strength.

The structure of this paper is as follows: in Section 2, the methodology is explained. Then, the recommended strategy is described in Section 3. Next, section 4 presents the simulation results and analysis of the results. Finally, the conclusions are given in Section 5.

2. Methodology

Deep learning fundamental idea is based on artificial neural networks (ANN). Over the past few years, deep learning has been utilized more and more in various applications, including autonomous vehicles. This section discusses the theoretical aspects of this work's CNN and IHS algorithms.

2.1 Convolutional Neural Networks (CNN)

Deep learning is an effective method for picture recognition in autonomous vehicles and can automate the feature extraction process (such as object detection and semantic segmentation) [2]. CNN uses the convolution operation, a subset of deep learning systems, to process visual pixels. The layers are arrayed in width, height, and depth, distinguishing its architecture from a Multilayer Perceptron (MLP). Additionally, there are gaps in the connections between the layers and CNN's neurons. According to Figure 1, a Typical CNN comprises the following layers [16]:

- Input layer: This includes information about the input image.
- Convolution layers: This layer's convolution technique extracts significant features from the image.
- Pooling layers: These layers, which are between two convolution layers, reduce some of the spatial information in the convoluted image while keeping the spatial information that matters most.
- Fully connected layer: This connects all weights and neurons and acts as a classifier.
- The output layer is where the classification probability, the final result, is stored. References thoroughly explain the theoretical component and mathematical formulae for CNN [16], [17], and [18].

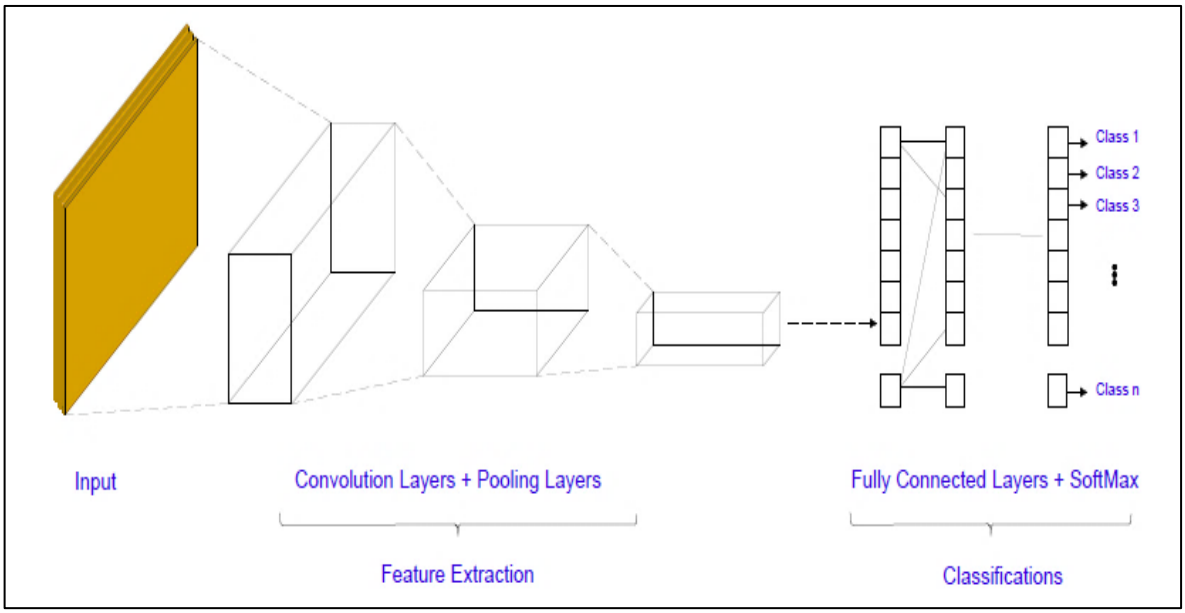

Figure 1: CNN layers for object detection and classification [16]

2.2 Intensity Hue Saturation (IHS) for RGB and Depth Merging

Because of their versatility and low cost, optical RGB sensors are the most commonly used information source. Computer vision algorithms are used to assess the color images produced by these sensors for issues such as object detection, segmentation, tracking, optical flow, and so on. With the advent of deep learning technologies, this field of study has improved substantially [19]. Image fusion combines and injects essential information from multiple input photos to create a single output image that is more useful and effective than the sum of the input images [20]. One of the most prominent fusion approaches is the IHS approach to sharpening. This approach of color analysis has become widely used in image processing. The goal is to improve spatial precision, feature perfection, and the convergence of numerous data sets. A photograph's color and saturation frequently reflect spectral information. The visual system can determine that the amplitude variation is manageable and has no effect on spectral features [21]. The RGB IHS conversion paradigm establishes IHS as a third-order technique, which is widely accepted. The transform kernel is a 3 x 3 matrix. According to various published studies, the following definition employs distinct IHS transformations, resulting in substantial differences in matrix values [22]:

$$\begin{bmatrix} I \\ V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{-1}{\sqrt{6}} & \frac{-1}{\sqrt{6}} & \frac{-1}{\sqrt{6}} \\ \frac{-1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 \end{bmatrix}, H = \tan^{-1}\left(\frac{V_2}{V_1}\right),\ S = \sqrt{V_1^2 + V_2^2} \quad (1)$$

, where I stands for Intensity, H for Hue, S for Saturation, and R stands for Red, and (V1 and V2) are values in the center. The algorithm scales intensity, hue, and saturation between 0 and 255 and employs special case processing. The RGB color cubes depict hue, saturation, and intensity. In color photographs, intensity is separated from color data (Hue and Saturation). Geometric formulas convert RGB to IHS color. Gives Hue H by [22]:

$$H = \begin{cases} \theta & if\ B \le G \\ 360 - \theta & if\ B > G \end{cases} \quad (2)$$

Where:

$$\theta = \cos^{-1}\left\{\frac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{(R-G)^2+(R-G)(G-B)}}\right\} \quad (3)$$

The saturation S is given by [22]:

$$S = 1 - \frac{3}{(R+G+B)}[MIN(R,G,B)] \quad (4)$$

The Intensity I is given by [22]:

$$I = \frac{1}{3}(R+G+B) \quad (5)$$

3. Proposed Approach

One of the most crucial components of autonomous driving, if not the most crucial one, is localization. End-to-end learning describes the training of neural networks from start to finish without human assistance. End-to-end learning's goal is to automatically train the system to learn internal models of the required processing stages, including identifying beneficial road characteristics, using only the input signal, as shown in Figure 2.

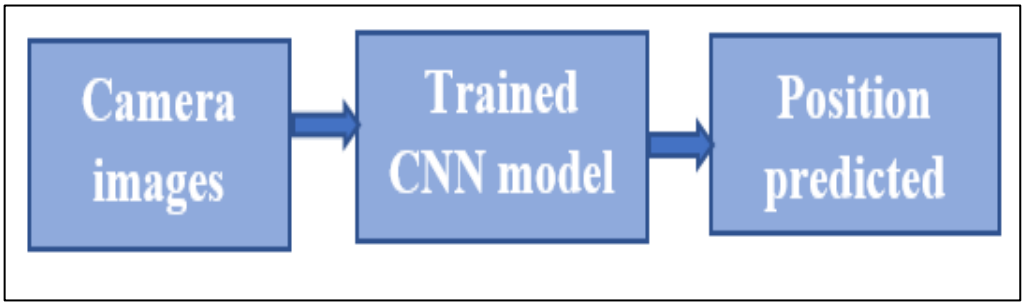

Figure 2: Diagram of the proposed autonomous localization system

The main contribution of this study is the achievement of good localization accuracy for AV with just a vision-based system, opening the door to the idea of eschewing expensive sensors like the LIDAR or the RADAR in a variety of weather/time circumstances, such as clear sunny, hard rain midday, and wet cloudy noon in the urban area. To improve localization accuracy in AV, this work aims to employ deep learning models on visual data. Instead of decreasing the parameters of some of the well-known neural networks that can be used for autonomous driving, start from scratch and create network architecture layer by layer until we arrive at a suitable solution. Then, increase the number of layers or add new levels progressively until the returns become less sensitive to validation loss by starting with very few layers and parameters. Depth, one of the crucial factors in precise localization, is the most excellent component of perception. This work utilizes an RGB image with Depth created using the IHS approach to evaluate the accuracy of the recommended CNN. The only input for the independent localization system is the camera image in its raw pixel form. The result is the projected position. Figure 3 depicts the suggested localization for autonomous vehicles.

In order to deliver the network's performance in terms of locating the AV with high accuracy, the network has been defined with 22 layers using 256 x 256 input images, which is suitable to increase the output's accuracy and decrease error with minimal training time. Additionally, as the number of layers rises, the computations become more complex, which slows down network training. This network is trained using the Stochastic Gradient Descent with Momentum (SGDM) optimization strategy [23], which is consistently speedier and more effective than the (GDM) approach [24]. In the classifier step, 11 classes are used, depending on the number of streets.

The proposed architecture is used in this study to shorten training times, increase the applicability of the proposed CNN for batch image classes, and reduce computing complexity. The first convolutional layer of this algorithm has 256 filters with 11 by 11 dimensions. The second convolutional layer contains 512 filters with 5 * 5 size. Three FC layers and 384 filters with a 3 * 3-pixel size follow the first two convolutional layers. The likelihood of scoring prediction scores, or the Softmax function, which creates a distribution across the number of classes, is connected to the final FC layer. Grouping layers follow some convolutional layers. Due to the reduced spatial dimension of the representation, pooling layers allows for the abstraction of abstract hierarchies of filters with better receptivity while also decreasing computation after each layer. As a result, CNN frequently trains quicker while using ReLU as its activation function. Figure 4 displays the CNN design information.

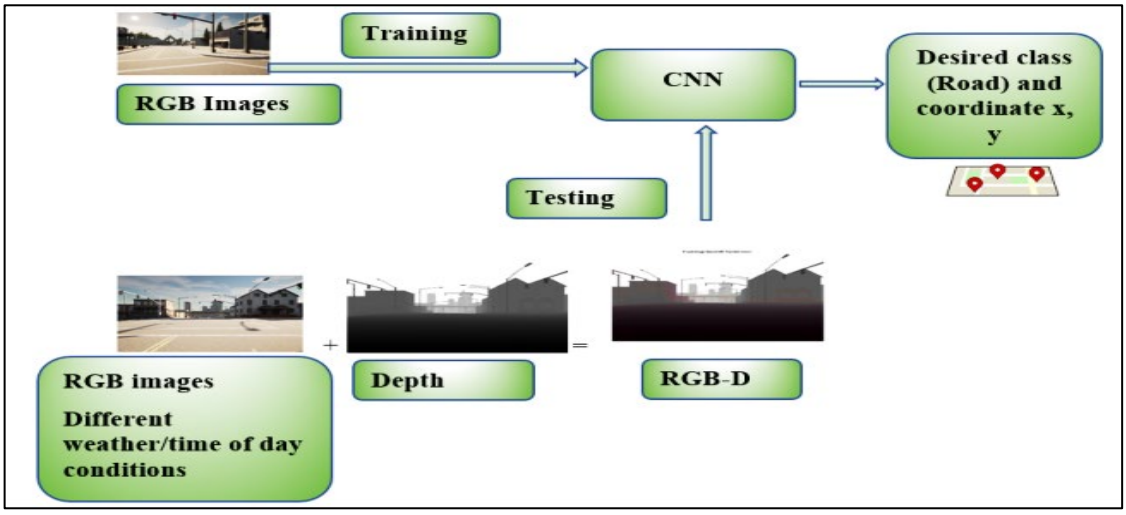

Figure 3: The proposed localization for Autonomous vehicle

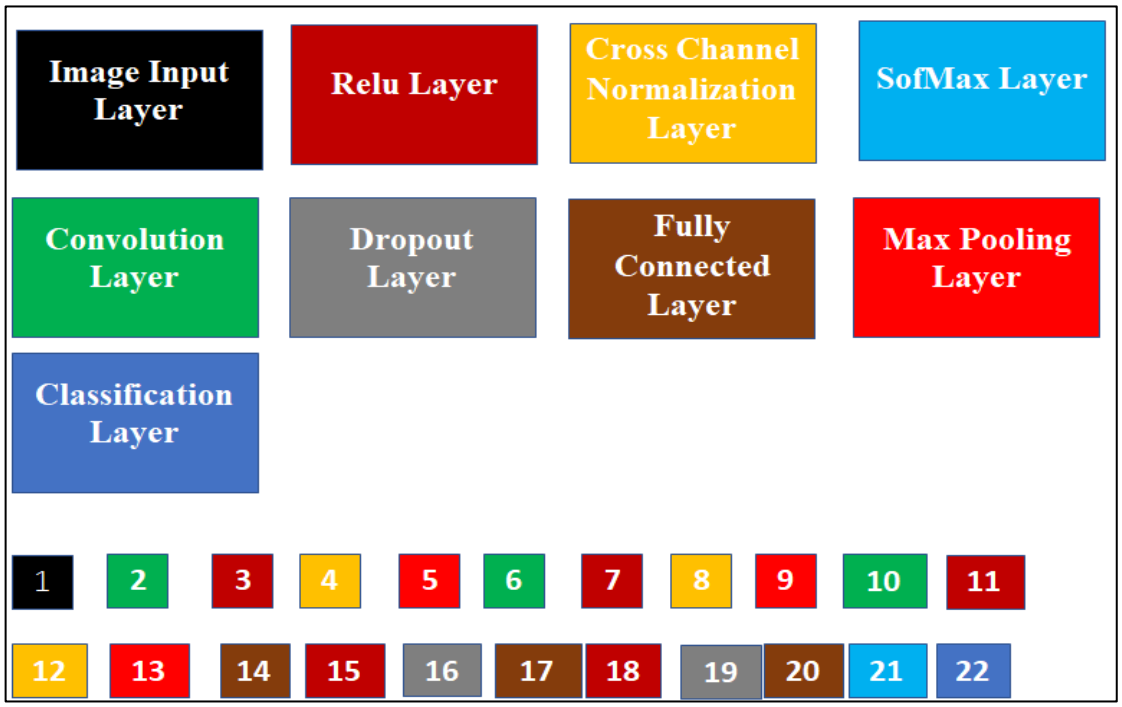

Figure 4: 22-layer proposed end-to-end CNN architecture

While concurrently getting the highest performance feasible, in this case, the autonomous and the light model (computationally least expensive) is used. The model with the fewest parameters that affect its memory footprint and computations is typically the least expensive in terms of computation. Therefore, various feature maps, kernel sizes, and layer types and sizes all affect computational performance. The model's ultimate architecture resulted from testing various layer counts, convolutional layer kernel sizes, feature map counts, location of pooling layers, and finally, the size and number of fully connected layers. Use more convolutional layers since they are what make feature extraction possible. For example, a line that the autonomous vehicle must follow is one of the properties needed for autonomous driving. The convolutional layer was able to extract in the first experiment. However, only using one convolutional layer was not enough to achieve our objectives; more feature maps are needed.

4. Simulation Results and Discussion

The data sets for this work were collected using the CARLA (Car Learning to Act) simulator, an open-source simulator, by having an autonomous vehicle navigate a map while being outfitted with a depth camera and RGB sensor [24]. This software makes driving preparations for various climatic conditions and times of the day. 5283 RGB images comprise the training datasets, while the testing phase embeds an equivalent depth image in each frame. The map utilized to collect the data, which includes the city center, neighborhoods, and wooded regions, is depicted in Figure 5. It has a surface area of approximately 120,000 square meters. Figures 6 and 7 show a street and a tunnel in different weather conditions, respectively.

Figure 5: Top view of the planned map within the context of the CARLA simulator (open-source simulator) [24]

Figure 6: RGB image of a street in bright sunny, hard rain noon, and wet cloudy noon, respectively [25]

Figure 7: RGB image of the Tunnel in bright sunny, hard rain noon, and wet cloudy noon, respectively [24]

In CARLA, the camera sensor is mounted to the vehicle and may take pictures at a predetermined frame rate. Figures 8, 9, and 10 illustrate how the camera sensor generates images in RGB and depth under bright, precise settings, respectively. The IHS method is then used to merge the images. Finally, an autopilot with RGB sensors and a depth camera was mounted on the AV, and it was instructed to travel the city streets as indicated on the map. The data sets used in the paper were created using this process.

Figure 8: RGB images from CARLA simulator [24]

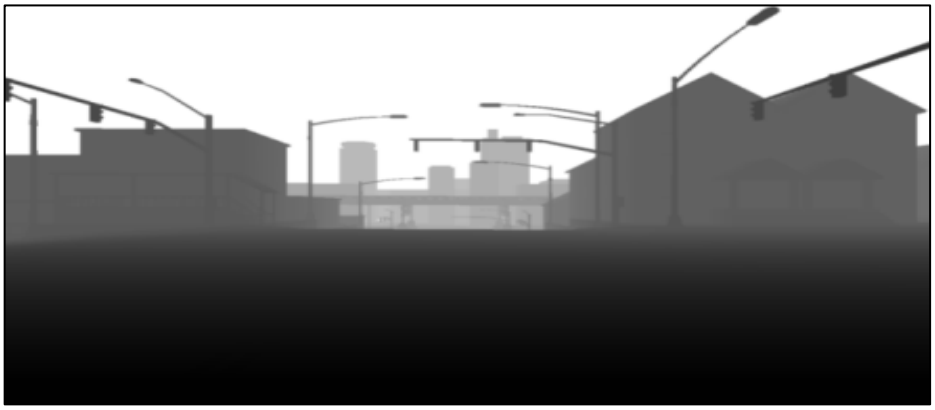

Figure 9: RGB Depth image from CARLA simulator [24]

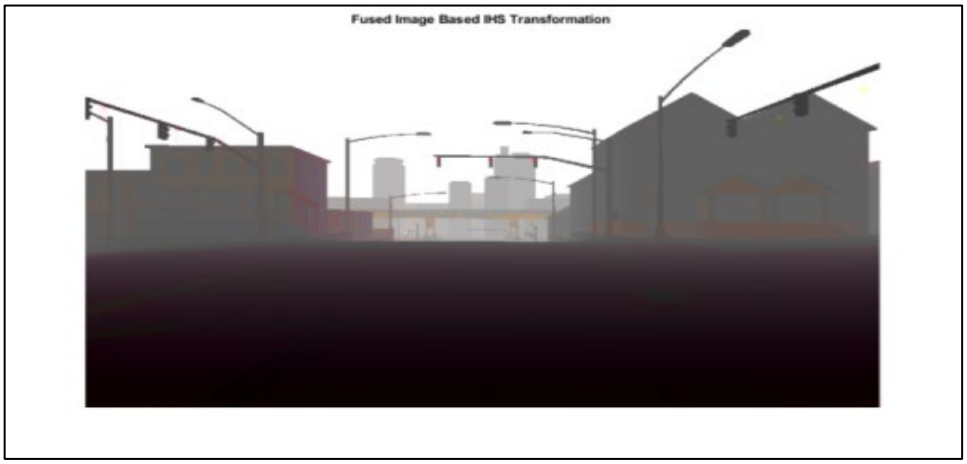

Figure 10: RGB, Depth images merged based on IHS

When the weather changes and at different times of day, the capacity of the majority of sensors changes significantly. For instance, LIDAR works well while the sky is clear, but its accuracy suffers greatly when it rains or snows. This study uses a camera that can adapt to changing lighting conditions and weather [26]. Data augmentation is a straightforward approach to double the dataset size and highlight curves by flipping the images in vertical access. The model is trained using data already collected from the CARLA simulator [24], as shown in Table 1. The simulated findings are produced using MATLAB. The CNN network results show no deviation from the correct placement. Other methods, however, have a clear departure, calling for exact self-driving maps that take a lot of time, money, and resources to produce and are resource-intensive.

Table 1: Dataset from CARLA simulator

Image resolution	$\mathbf{256 \times 256 \times 3}$
Number of images	5283
Number of images/class	Variable
Number of classes	11

The best CNN hyperparameter settings for training the datasets were discovered through trial and error: the max number of epochs is 25, the maximum iteration is 2875 iterations; 115 iterations per epoch, starting learning rate is 0.001, and the batch size is 8A PC with a 1.80GHz and 2.30GHz Intel (R) Core(TM) i7-10510U CPU and RAM (8 GB) is used the code is written under MATLAB environment.

The Mean Square Error (MSE) loss function is the performance metric used to assess modeling accuracy, as shown in Equation (6). If n predictions are made from a sample of n observations, Y is the variable encoding the actual location and is the predicted position [27]:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y - \hat{Y})^2 \quad (6)$$

The backpropagation neural network is trained via the SGDM optimizer. 80% of the data for training and 20% for testing; the training process consumed 166 minutes. The camera records 50 frames per second. Since the car is moving at 60 km/h, the average distance between the two succeeding photos is 3.01 meters in vertical and 0.5 meters in horizontal directions. The accuracy of 25 epochs and 2875 iterations is 94.74%; see Figure 11. The proposed work for utilizing CNN to identify a street and determine its coordinates is provided with the simulation results.

Based on the output class number and the coordinate values, three photos of various streets were evaluated, and the locations were correctly predicted. As a result, they are improved compared to simulation results from reference 19. Initially, multiple classes were developed, each comprising a unique pair of coordinates and pictures shot from various perspectives. Then, these datasets were categorized to test if each location could be distinguished from the others using only a visual cue. Finally, to locate a match between the suggested image and the images in the predicted class, the coordinates of this image are obtained using K-Mean clustering. After finding the match, the value of the coordinates for x and y is recovered, and the position's measuring unit is in meters, as shown in Figure 12.

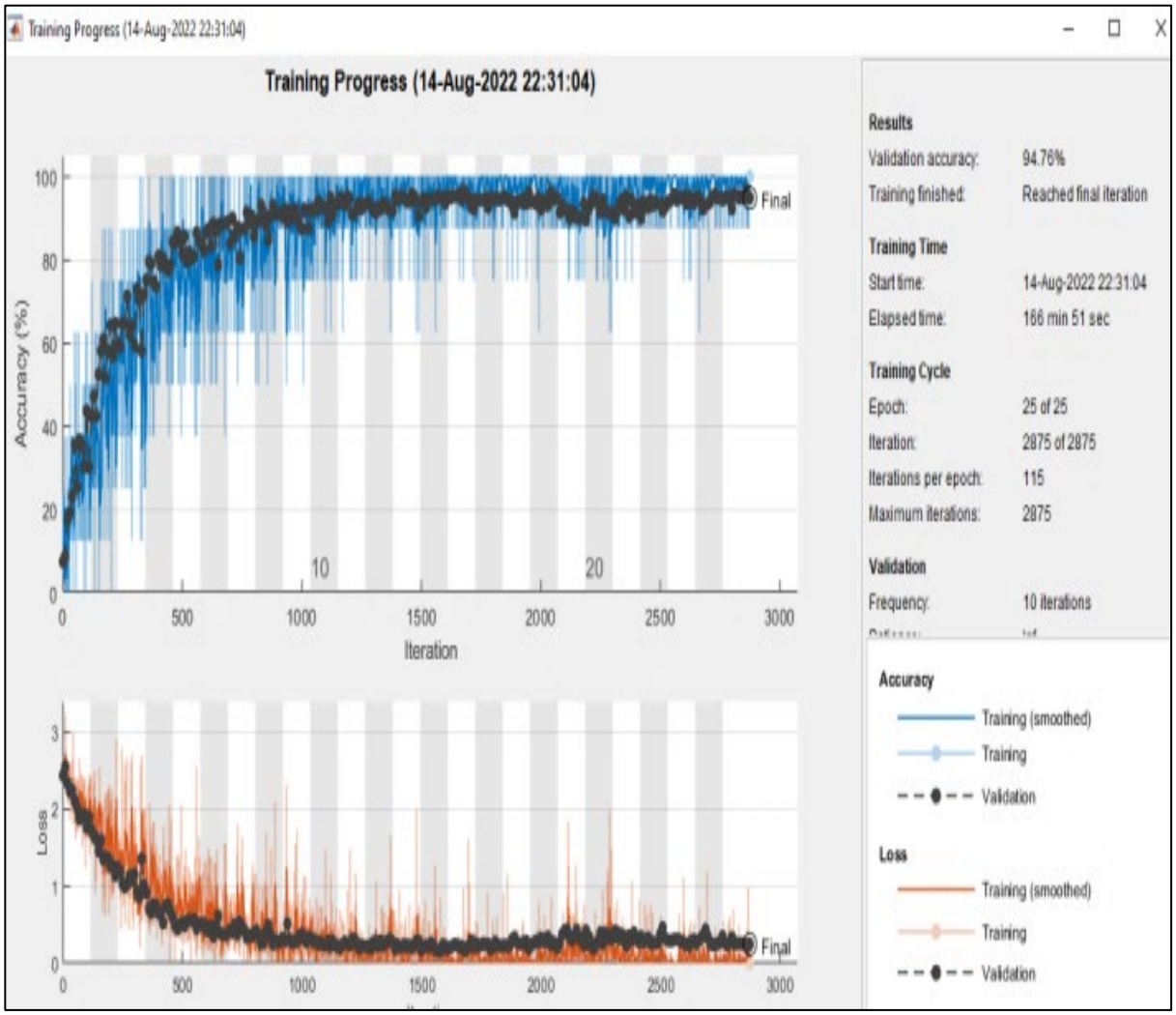

Figure 11: Spending time on training and training accuracy

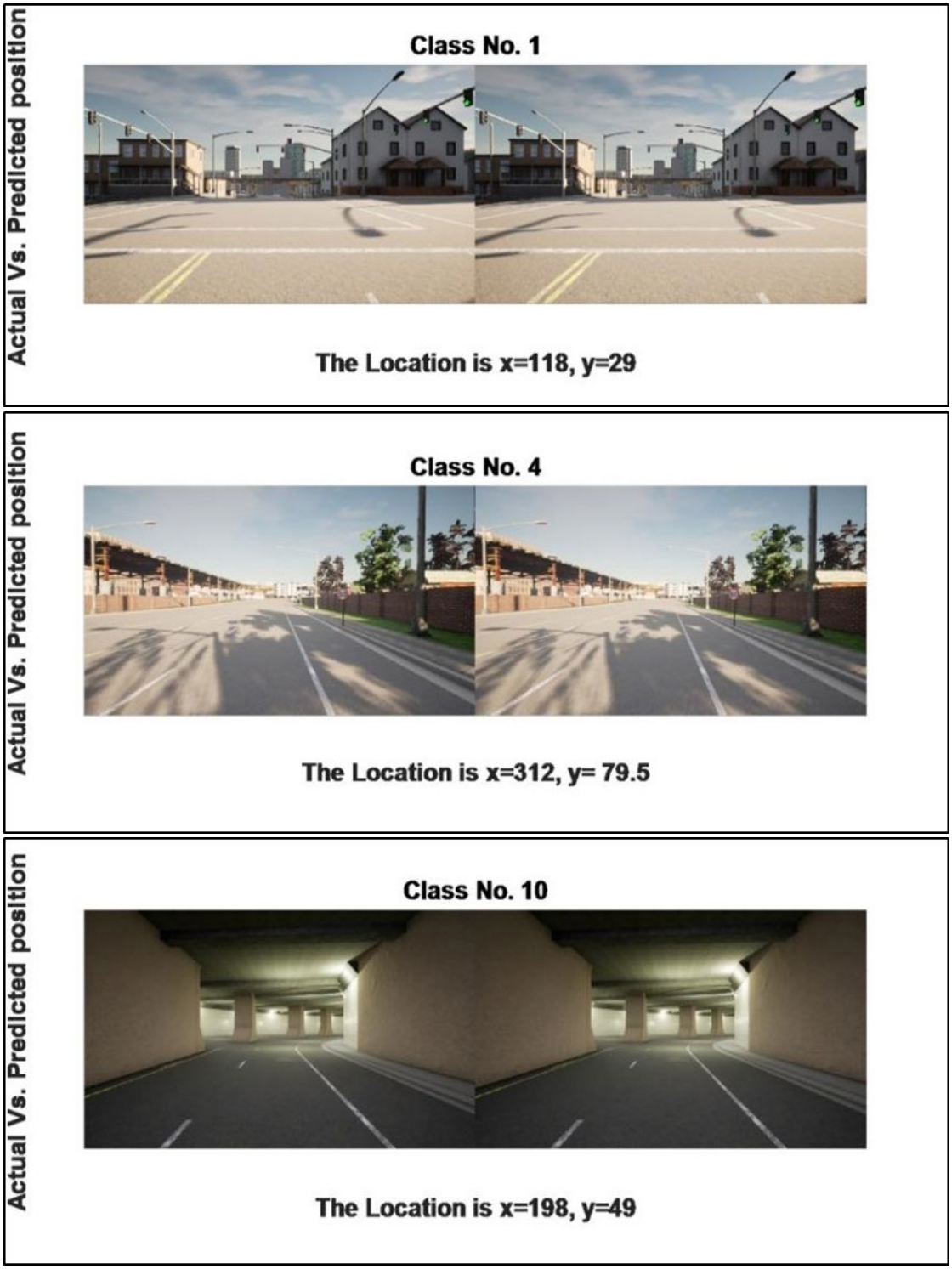

Figure 12: Actual vs. Predicted class (Road) and the coordinate

In a self-driving car simulator, the effectiveness of autonomous driving was evaluated. Results show that the suggested approach has dramatically enhanced the performance and robustness of the localization network when compared with reference No. 19. This is up to 94.74% accurate and has a Mean Squared Error (MSE) of 0 meters, in contrast to reference No. 19, which has an MSE at the projected position of 4.8 meters. The best performance is attainable while employing the light, computationally the least expensive model. The model with few parameters affecting its memory requirements and computations typically has the lowest computational cost. As a result, the number of feature mappings, kernel sizes, layer type, and size all affect computational performance. The proposed network was built by starting with a minimal number of layers and gradually adding more to achieve a tiny network and the least computational cost. It was designed to efficiently localize self-driving automobiles without using commercially available networks. Several specifications for the proposed work and the work in reference No. 19 are listed in Table 2.

Table 2: Comparison between the proposed network with Reference No. 19

Model	Mean error	No of layers	Execution time	No of neurons in FCL
CNN-based RGB-D in the training stage [18]	4.8 m	29 layers	11 ms	1024 neuron
CNN-based RGB in the training stage	0 m	22 layers	5 ms	1000 neuron

One may validate the network's robustness by choosing an image in a different weather situation and not training the network on it (mid-rain noon). This type of test is interesting as the images will differ from what exists in the training data set, indicating to what degree the CNN network has learned to generalize. The network predicts the class and precise coordinate position. It means obtaining the superior generalization capabilities of our model, as shown in Figure 13.

Figure 13: Actual vs. Predicted class (Road) and the coordinate (validation subset)

Evaluating evaluation criteria for deep learning tasks is essential for developing an efficient classifier. Training and testing are used during two key stages of a typical data categorization process. During the training phase, it is used to improve the classification algorithm. This indicates that the evaluation measure is used to discriminate and choose the optimum solution, such as a discriminator that can produce an exceptionally accurate forecast of impending evaluations about a particular classifier. Currently, the constructed classifier's effectiveness is assessed using the evaluation metric, for example, as an evaluator utilizing confidential data during the model testing step. The number of negative and positive occurrences that are effectively classified, as shown in Eq. 7, is defined as TN and TP. The quantity of incorrectly identified positive and negative instances is also defined as FN and FP, respectively. The following list includes some of the most popular evaluation metrics [20]:

1) Accuracy: Determines the proportion of correctly predicted classes to all tested samples (Eq. 7).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{7}$$

2) Sensitivity or Recall: Used to determine the percentage of positive patterns that are appropriately classified (Eq. 8).

$$Sensitivity = \frac{TP}{TP+FN} \tag{8}$$

3) Specificity: Used to determine the percentage of negative patterns that are accurately classified (Eq. 9).

$$Specificity = \frac{TN}{FP+TN} \tag{9}$$

4) Accuracy: Used to correctly determine the positive patterns predicted by each pattern in a positive class (Eq. 10).

$$Precision = \frac{TP}{TP+FP} \tag{10}$$

Evaluation matrices were calculated for all data classes, as shown in Table 3.

Table 3: Evaluation Metrics

Actual Classes	**----------1--------2--------3--------4--------5--------6--------7--------8--------9--------10--------11-----**										
TP	9.00	7.00	11.00	8.00	23.00	7.00	11.00	16.00	12.00	13.00	12.00
FP	1.00	0.00	2.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00
FN	0.00	0.00	1.00	0.00	0.00	1.00	1.00	0.00	1.00	0.00	0.00
TN	123.00	126.00	119.00	125.00	110.00	125.00	121.00	117.00	119.00	120.00	121.00
Preci.	0.90	1.00	0.85	1.00	1.00	1.00	1.00	1.00	0.92	1.00	1.00
Sensi.	1.00	1.00	0.92	1.00	1.00	0.88	0.92	1.00	0.92	1.00	1.00
Speci.	0.99	1.00	0.98	1.00	1.00	1.00	1.00	1.00	0.99	1.00	

5. Conclusion

Due to the light, accurate, and durable suggested network in various weather situations based on a vision-only system, this study contributed high accuracy for AV localization and minimized error in projected positions with short training times. In this study, CNN was used as a feasible solution for the autonomous localization of self-driving cars in urban environments under various temporal and weather conditions. The CNN obtained decent localization accuracy for AV composites with a vision-only system. Thus, it opens the door to doing away with the requirement for expensive sensors like RADAR or LIDAR. Due to the relatively simple architecture, it was a computationally efficient method to build the network architecture layer by layer until it met the requirements of accurate localization. The suggested network was trained using both the raw camera images and each frame's location-coordinate information. The IHS algorithm used depth images with RGB images to evaluate the accuracy of the learned data. By offering high localization accuracy and enhancing performance with less training time, where the training time was decreased to 166 minutes and 51 seconds, the simulation results were improved compared to the results achieved in reference 19 with respect to MSE. Results demonstrate that the suggested approach has significantly improved the performance and robustness of the localization network. With an accuracy of 94.74 percent, the simulation results demonstrated the proposed network's superiority. Our model's higher generalization abilities were obtained when the weather differed from the training settings in the validation scenario.

Author contribution

All authors contributed equally to this work.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Data availability statement

Not applicable.

Conflicts of interest

The authors of the current work do not have a conflict of interest.

References

[1] S. Chen, B. Liu, C. Feng, C. Vallespi-Gonzalez, and C. Wellington, 3D Point Cloud Processing and Learning for Autonomous Driving, arXiv:2003.00601v1 [cs.CV], 2020. https://doi.org/10.48550/arXiv.2003.00601

[2] J. Fayyad, M. A. Jaradat, D. Gruyer, H. Najjaran,Deep Learning Sensor Fusion for Autonomous Vehicles Perception and Localization: A Review Deep Learning Sensor Fusion for Autonomous Vehicle Perception and Localization: A Review, Sensors, 20 (2020) 4220. https://doi.org/10.3390/s20154220

[3] T. G. R. Reid et al., Localization Requirements for Autonomous Vehicles, SAE Int. J. Connect. Autom. Veh., 2 (2019) 173-190. https://doi.org/10.4271/12-02-03-0012

[4] A. L. Ballardini, S. Fontana, D. Cattaneo, M. Matteucci, and D. G. Sorrenti, Vehicle localization using 3D building models and point cloud matching, Sensors, 21 (2021) 5356. https://doi.org/10.3390/s21165356

[5] R. Pirník, M. Hruboš, D. Nemec, T. Mravec, and P. Božek, Integration of inertial sensor data into control of the mobile platform, Adv. Intell. Syst. Comput., 511 (2017) 271–282. https://doi.org/10.1007/978-3-319-46535-7_21

[6] P. Božek, Y. Nikitin, P. Bezák, G. Fedorko, and M. Fabian, Increasing the Production System Productivity Using Inertial Navigation, Manuf. Technol., 15 (2015) 274-278. https://doi.org/10.21062/ujep/x.2015/a/1213-2489/MT/15/3/274

[7] T. T. O. Takleh, N. A. Bakar, S. A. Rahman, R. Hamzah, and Z. A. Aziz, A brief survey on SLAM methods in autonomous vehicle, Int. J. Eng. Technol., 7 (2018) 38–43. https://doi.org/10.14419/ijet.v7i4.27.22477

[8] A. Noureldin, A. El-Shafie, and M. Bayoumi, GPS/INS integration utilizing dynamic neural networks for vehicular navigation, Inf. Fusion, 12 (2011) 48–57. https://doi.org/10.1016/j.inffus.2010.01.003

[9] H. U. Kim , T. S. Bae, Deep learning-based GNSS network-based real-time kinematic improvement for autonomous ground vehicle navigation, J. Sensors, 2019 (2019)8. https://doi.org/10.1155/2019/3737265

[10] E. Parisotto, D. S. Chaplot, J. Zhang, and R. Salakhutdinov, Global Pose Estimation with an Attention-based Recurrent Network, 2018. https://doi.org/10.48550/arXiv.1802.06857

[11] D. Chen, Y. Yu, X. Gao, Semi-Supervised Deep Learning Framework for Monocular Visual Odometry,(2019)

[12] D. Eigen, C. Puhrsch, R. Fergus, Depth map prediction from a single image using a multi-scale deep network, Adv. Neural Inf. Process. Syst., 3 (2014) 2366–2374.

[13] J. Byun, M. Roh, K.-I. Na, J. C. Sohn, and S. Kim, LNAI 7508 - Navigation and Localization for Autonomous Vehicle at Road Intersections with Low-Cost Sensors, 2012.

[14] Bag, S. Deep Learning Localization for Self-driving Cars, 2017.

[15] S. Alzu'bi and Y. Jararweh, Data Fusion in Autonomous Vehicles Research, Literature Tracing from Imaginary Idea to Smart Surrounding Community, 020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC), Paris, France, 2020, 306-311. https://doi.org/10.1109/FMEC49853.2020.9144916

[16] . Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, Deeper depth prediction with fully convolutional residual networks, Proc. - 2016 4th Int. Conf. 3D Vision, 3DV, 2016, 239–248. https://doi.org/10.1109/3DV.2016.32

[17] G. Plastiras, C. Kyrkou, and T. Theocharides, Efficient convnet-based object detection for unmanned aerial vehicles by selective tile processing, ACM Int. Conf. Proceeding Ser., 2018, 1–6. https://doi.org/10.1145/3243394.3243692

[18] L. Alzubaidi et al., Review of deep learning: concepts, CNN architectures, challenges, applications, future directions, J. Big Data, 8 (2021). https://doi.org/10.1186/s40537-021-00444-8

[19] V. Vaquero Gomez, Lidar-Based Scene Understanding for Autonomous Driving Using Deep Learning, Tesi doctoral, 2020.

[20] D. Mishra and B. Palkar, Image Fusion Techniques: A Review, Int. J. Comput. Appl., 130 (2015) 7–13. https://doi.org/10.5120/ijca2015907084

[21] F. A. Al-Wassai, N. V. Kalyankar, and A. A. Al-Zuky, The IHS Transformations Based Image Fusion, 2 (2011).

[22] H. Atiyah and M. Y., Outdoor Localization in Mobile Robot with 3D LiDAR Based on Principal Component Analysis and K-Nearest Neighbors Algorithm, Eng. Technol. J., 39 (2021) 965–976. https://doi.org/10.30684/etj.v39i6.2032

[23] M. Y. Hassan and G. Kothapalli, Comparison between neural network based PI and PID controllers, 2010 7th Int. Multi-Conference Syst. Signals Devices, SSD-10, no. May 2014, 2010, 1-6. https://doi.org/10.1109/SSD.2010.5585598

[24] B. Zhou, J. Liu, W. Sun, R. Chen, C. Tomlin, and Y. Yuan, pbSGD: Powered stochastic gradient descent methods for accelerated nonconvex optimization, Int. Jt. Conf. Artif. Intell., 2021-Janua,2020,3258–3266. https://doi.org/10.24963/ijcai.2020/451

[25] Q.-Y. Zhou, J. Park, V. Koltun, Open3D: A Modern Library for 3D Data Processing, arXiv:1801.09847v1 [cs.CV], 2018. https://doi.org/10.48550/arXiv.1801.09847

[26] Y. Wu, Y. Li, X. Ge, Y. Gao, W. Qian, An Efficient Method for Calculating the Error Statistics of Block-Based Approximate Adders, IEEE Trans. Comput., 68 (2019) 21–38. https://doi.org/10.1109/TC.2018.2859960