INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2025, VOL. 71, NO. 1, PP. 277-284
Manuscript received December 23, 2023; revised March 2025. doi: 10.24425/ijet.2025.153571

Development of a mapping system on an autonomous vehicle using a Fully Convolutional Neural Network and Fast SLAM Algorithm

Bhakti Yudho Suprapto, Abeng Yogta, and Suci Dwijayanti

***Abstract*—There are many challenges when it comes to autonomous vehicle movement, one of which is developing an accurate and precise internal mapping system. Autonomous vehicles use internal maps to move from a starting point to destination point. Many methods are used in creating these maps, but because they still display weaknesses, further development is required. This research combines the FastSLAM 2.0 algorithm with a fully convolutional neural network (FCNN) model using the road features recognized by the FCNN algorithm as the object of observation of the FastSLAM 2.0 algorithm. This method was tested to form a map of the environment around the Faculty of Engineering, Sriwijaya University, Inderalaya Campus. In the training, the Adam optimizer and Adam combined with batch normalization (BN) model showed good accuracy: 82.07% and 78.08%, respectively. The application of this method succeeded in forming a map similar to Google Maps using the FCNN observation model. The map that was successfully formed had an IoU of 0.159 against the Google Maps map obtained with the Adam + BN model.**

***Keywords*—autonomous vehicle; FastSLAM; FCNN; mapping**

I. Introduction

ONE of the fastest-growing technologies in transportation is autonomous vehicles, that is, vehicles that can move without drivers. These vehicles must have technology that can assist in their autonomous movement, including path planning, decision-making, localization, map building, and movement control[1][2]. Gaining knowledge is essential before achieving localization and perception. If a vehicle possesses all of these abilities, including information acquisition, it can be classified as an autonomous vehicle[3].

Path planning functions can be used to determine the optimal route that can be taken from the starting point to a predetermined destination point, such that the vehicle can save time and energy by traveling efficiently [4][5]. The optimal route to take will be determined by considering various constraints and conditions. This may involve factors such as finding the shortest distance between two points or minimizing the travel time by avoiding collisions. In some cases, constraints and goals may overlap, such as attempting to minimize energy usage without exceeding a specified threshold for travel time[6]. Maps and positions are required to show the position, direction, and route to the autonomous vehicle. The position of the autonomous vehicle is an important factor in determining the vehicle's location on the road or path that it traverses on the map. Many types of maps are used, one of which is the Google Maps API. However, with this type of map, not all necessary information is provided, and the road information that is provided may be inaccurate[7]. Furthermore, this type of map does not change direction and cannot directly synchronize the display if there is a change in environmental conditions [8]. Several purposes and applications require accurate and accurate maps [9]. The use of the Google Maps API is highly dependent on GPS. However, GPS has several weaknesses, including diminished indoor accuracy [10], lack of coverage over certain areas[11], and less precise coordinates and position readings[12]. Therefore, it is necessary to develop good mapping for autonomous vehicle navigation because it is likely that autonomous vehicles will move through environments with incomplete information and areas that are not covered by GPS.

Apart from maps that are accurate and safe[13], navigation also requires the position of the vehicle and information about the vehicle's environment. Researchers usually use an algorithm known as simultaneous localization and mapping (SLAM) to solve this problem[14]. The research on SLAM has been conducted using various methods, such as the extended Kalman filter (EKF-SLAM)[15]. This method uses state vectors to store a limited number of landmarks with complex computations. Thus, it is necessary to reduce the complex EKF-SLAM computations if the method is used in outdoor areas where there are many objects. To overcome these weaknesses, the FastSLAM 1.0 algorithm (a factored solution to SLAM) has been developed. However, the FastSLAM 1.0 algorithm is weak because the movement model receives a lot of interference from the exteroceptive sensor. As a result, an algorithm known as FastSLAM 2.0 has been developed. This algorithm uses a Rao-Blackwellized method on the filter particles[16].

Another study has developed the FastSLAM 2.0 algorithm by using a hand-drawn image as an initial map for estimating building locations before then combining the FastSLAM algorithm and particle swarm optimization (PSO) to form a map [17]. However, in drawing a map, the accuracy of the scale of existing buildings and roads must receive attention. This problem is quite difficult to solve. To overcome this problem, in the present study, the principle of combining the FastSLAM 2.0 algorithm with an intelligent algorithm is developed. The current paper discusses the use of fully convolutional neural networks (FCNN) combined with FastSLAM 2.0 in forming a

Authors are with Department of Electrical Engineering, Universitas Sriwijaya, Indonesia (e-mail: bhakti@ft.unsri.ac.id, abeng.yogta01@gmail.com, sucidwijayanti@ft.unsri.ac.id).

map to be traversed by autonomous vehicles. Because of the advantages of FCNN, many researchers use this algorithm in several applications, such as object detection[18] visual tracking [19], and brain tumor segmentation[20]. It is known that FCNN is included in the smart algorithm based on a neural network. Neural networks have been widely used in various applications, such as a hexacopter controller[21], energy consumption forecasting[22], and automatic correction of the depth map of the human body[23]. With FCNN, which has good accuracy and an optimum level when compared with ordinary multilayer perceptron, this FCNN will be able to form maps that have good accuracy when combined with the FastSLAM 2.0 algorithm. The present study focuses on the implementation of combining FCNN and the FastSLAM 2.0 algorithm to form a better mapping for autonomous vehicles.

The current paper is structured as follows: Section 2 presents a brief overview of road recognition, which also contains the FastSLAM 2.0 and FCNN algorithms used. Section 3 discusses the research methods in detail. Section 4 presents and discusses the results. Section 5 outlines the conclusions and possible directions for future work.

II. ROAD RECOGNITION

Each road recognition is used to determine the position of the vehicle against the road that it traverses. When moving autonomously, the vehicle must be able to maintain its position so that it stays on the road and avoids accidents that would occur because of moving into the wrong lane. For a vehicle to maintain its position, it requires road recognition technology. Road recognition can be done using various methods, such as support vector machine [24], canny edge detection and Hough transformation [25][26], and FCNN [27].

A. Fast SLAM 2.0

The FastSLAM 2.0 algorithm comes from the SLAM algorithm, which was initially used in robot navigation in unknown environments. The robot's movement coincides with an estimate of the surrounding environment, and its position is based on sensory readings and existing landmarks. According to one study [28], the SLAM algorithm can be divided into two major groups: EKF-SLAM and FastSLAM. The EKF-SLAM algorithm typically uses an odometer and sensory measurements to obtain the mean and covariance of the EKF. However, creating maps, positions, and landmarks can interfere with their performance, so complex computations are required. To solve the problem of computational complexity in EKF-SLAM, the algorithm from the FastSLAM group is proposed for use with a Rao-Blackwellized particle filter. There are two types of FastSLAM algorithms: FastSLAM 1.0 and FastSLAM 2.0 [29]. Both of these methods use odometric readings based on measurements by the sensor to determine the position of a vehicle, but FastSLAM 1.0's dependence on sensor reliability is very high, which results in unstable localization. The FastSLAM 2.0 algorithm has better performance because localization is based on sensor readings. Nevertheless, the drawback is that its runtime efficiency is longer, especially if there are many landmarks in the surrounding area.

The FastSLAM 2.0 algorithm was developed by modifying the previous FastSLAM algorithm. The FastSLAM algorithm equation that estimates landmarks to be conditioned by the position estimation is as follows [28].

$$p(x^t, \theta | z^t, u^t, n^t) = p(x^t | z^t, u^t, n^t) \prod_{k=1}^{N} p(\theta_k | x^t, z^t, u^t, n^t)$$

where $x^t = \{x_1, x_2, [...]x_t\}$ is the set of estimated movements from time 1 to t, $z^t = \{z_1, z_2, [...]z_t\}$ is the set of all observations, and $u^t = \{u_1, u_2, [...]u_t\}$ is the set of all control inputs. The number of landmarks recorded from time 1 to t is denoted by $n^t = \{n_1, n_2, [...]n_t\}$ for each particle. $u = \{\theta_1, \theta_2, ...\theta_N\}$ is the set of N landmark positions. The FastSLAM algorithm estimates object poses, while landmarks are estimated using the EKF. Each particle has a pose $x^{t,[m]}$ and a state landmark represented by the mean and covariance of the Gaussian distribution, which is denoted by $u_{N,t}^{[m]}$ and $\sum_{N,t}^{[m]}$, respectively, where m is the index of the particle. Hence, each particle can be described as follows:

$$X^{t,[m]} =< x^{t,[m]}, u_{1,t}^m, \sum_{1,t}^{[m]} ...., u_{N,t}^m, \sum_{1,t}^{[m]} > \quad (2)$$

Regarding FastSLAM 1.0, the equation is conditioned to the previous state and control input or by the motion model, as shown in the following:

$$x_t^{[m]} \sim p\left(x_t | x_{t-1}^{[m]}, u_t\right) \quad (3)$$

From equation (3), it can be seen that the vehicle estimate does not consider flow measurements from the sensor. However, odometric measurements experience uncertainty, causing inaccurate localization results when the vehicle moves to the next point. Therefore, the FastSLAM algorithm must be improved to the FastSLAM 2.0 algorithm, as described previously[28]. In the following FastSLAM 2.0 algorithm, conditioning is not only in the motion model, but also the current sensory measurement, namely z_t, so that the sensory measurement will be able to correct the reading error on the odometer, as seen in [30]:

$$x_t^{[m]} \sim p\left(x_t | x^{t-1,[m]}, u^t, z^t, n^t\right) \quad (4)$$

B. Fully Convolutional Neural Network

FCNN is a neural network architecture that uses a convolutional layer like CNN, but it uses a fully connected layer so that the output from this architecture has the same dimensions as the input image (length, width, and channel). The architecture of the FCNN can be seen in Fig. 1 [31][32].

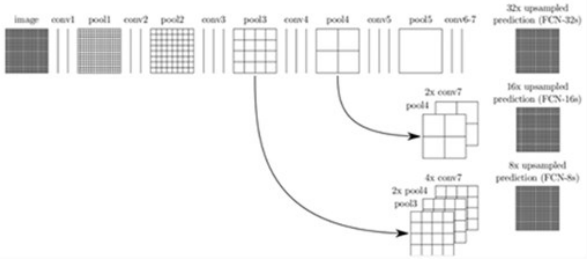


Fig.1. FCNN Architecture

In Fig. 1, there are several types of layers, namely image, convolution, pool, and up sampled. The image layer is the first layer of the FCNN architecture that contains an input image that will be processed by the FCNN to produce a targeted road image segmentation. The convolutional layer is a layer that uses

convolution operations (*) between the image layer and filter. The output of this layer is a feature map that will be processed in the next layer, namely the pooling layer. The pooling layer has a function to reduce the feature map generated by the convolutional layer. Up sampling is done to increase the output of the FCNN. The convolutional layer is formulated in (5). This equation is commonly used in convolution layer equations. Apart from these operations, there are also other types of operations at the convolution layer, such as transpose convolution, dilated convolution, and sparse convolution.

$$h(x) = f(x,y) * g(x,y) \quad (5)$$

Transpose convolution, also known as up sampling or deconvolution, is a convolutional operation that restores the initial shape of the layer after experiencing operations on the convolutional layer. The result of the convolutional layer (feature map) is smaller in size than the input layer to the convolutional layer. The convolutional feature map can be resized to the size of the initial layer preceding the convolutional layer using the transpose operation. The activation functions in the FCNN architecture are placed in all convolution layers to prevent the output value of a neuron from being too large. There are several types of activation functions, such as sigmoid, SoftMax, rectified linear unit (ReLU), and others. However, the architecture used in this FCNN only uses two types of activation functions: ReLU and SoftMax. This FCNN requires an activation function to activate or deactivate neurons, and this function must have certain characteristics, such as continuous, differentiable, and monotonically nondecreasing. The activation function used in this FCNN is ReLU, which is used in each convolutional layer output before entering the pooling layer. ReLU has several advantages, such as faster computation time and the ability to overcome the vanishing gradient problem found in the sigmoid activation function. The equation used for this activation function can be seen in (6), where $f_{ReLU}(h_{i,k})$ is the output of this activation function in neurons h in row i and column k. Another activation function of this FCNN is SoftMax.

$$f_{ReLU}(h_{i,k}) = \max(0, h_{i,k}) \quad (6)$$

The optimizer model is used to improve parameters to adjust the expected target and minimize errors in the architectural output. These optimizers are stochastic gradient descent (SGD) with Nesterov momentum and Adam optimizers. Kingma and Ban [33] introduced Adam's algorithm and tested it with the CNN algorithm with stable and fast results in eliminating errors. Each optimizer has different capabilities for minimizing errors in a model. The optimizer model requires a loss function to carry out the optimization process of calculating errors in the model. The loss function in FCNN is used to calculate the error of the FCNN model during the training process. Based on their functions, the loss function can be divided into two categories: regression loss functions and classification loss functions. Each of these has its advantages and disadvantages, as in the regression loss function based on research by Pavel Golik et al.[34].

III. Research Metodology

A. Mapping System

The method proposed in the present study is a modification of the FastSLAM 2.0 algorithm, whose observation model typically uses distance measuring sensor data such as LIDAR. In the current study, however, a monocular camera (IP Cam) was used instead. The object measured in the observation stage was the side of the road that had been detected by the FCNN algorithm–based model. The flowchart of the mapping system setup is shown in Fig. 2.

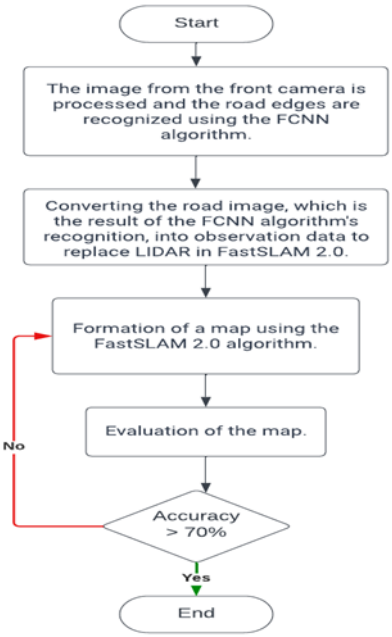


Fig.2. The Process of Map Formation Using the Proposed Method

After the front camera obtains data, the data are then processed and produce an output in the form of a recognized road image. This image is then used as an observation for the FastSLAM algorithm, as shown in Fig. 3. This observation data are obtained by comparing pixels in the image with real-world measurements taken in the field.

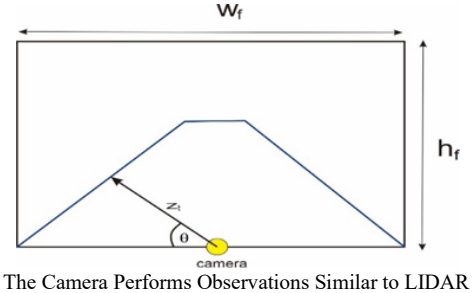
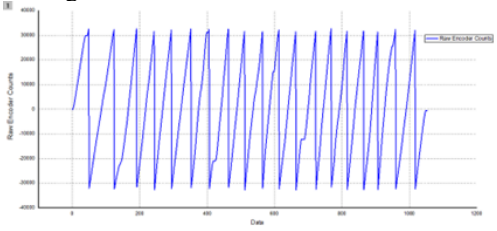


Fig.3. The Camera Performs Observations Similar to LIDAR

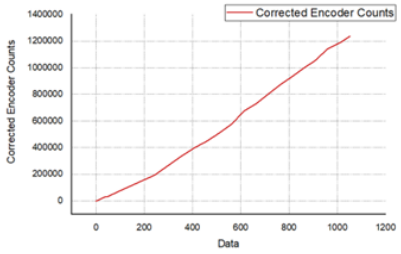
Data collection for the map formation simulation consists of distance traveled data, front camera video data, and x, y, and z-axis data from the magnetometer. These data were recorded using a data logger with a sampling time of 1500 ms/sample when vehicles were run following the route shown in Fig. 4.

Fig.4. The Route of the Electric Vehicle when Collecting Data

The yellow line illustrates the vehicle's data collection route, starting from the Department of Electrical Engineering, Sriwijaya University. After the data had been successfully retrieved, training was carried out using FCNN. In this mapping system, the mileage data were obtained from the encoder, which came from the data logger; however, these data appeared anomalous because it looked up and down. The vehicle moved forward without going backward, as shown in Fig. 5a. This anomaly was because the data were stored in integer form, which limits its display. Therefore, the data encoder needed to be compensated. The pseudocode of the compensate algorithm is shown in Fig. 6.



a. Data before compensation



b. Data after compensation

Fig.5. Data treatment

$a_t = a_{raw_{t-1}} = a^+ = a^- = \emptyset$

max_int_value = 32768

min_int_value = -32768

for $t = 1$ **to** T //loop over all data

  if $a_{raw_t} < 0$ **and** $a_{raw_{t-1}} > 0$

    $a^+ = \text{max\_int\_value} - a_{raw_{t-1}}$

    $a^- = a_{raw_t} - \text{min\_int\_value}$

  end if

  $a_t = a^+ + a^- + a_{t-1}$

end for

return a_t

Fig.6. The Encoder Data Compensation Pseudocode

After the compensation for the encoder data was carried out, the encoder data was obtained following the facts that there was a continuous increase in data. Compensated data can be seen in Fig. 5b. In addition to the encoder data, there were data obtained from the front cameras of the electric vehicles. The camera recorded the electric vehicle's journey for 25 minutes. The video was converted into an image using the Free Video to JPEG Converter software. The conversion produced 38,780 images with a 1080p resolution, resulting in a total file size of 12,607,762,453 bytes. The image was then uploaded to Google Drive so that it could be linked to Google Collaboration during the process of establishing a dataset for FCNN algorithm training. For training and performance testing, 38,780 pictures were made of ground truth, as shown in Fig. 7. Here, 2,633 data points were used as training data, while the remaining data were used to test the performance of the FCNN algorithm. The training data were considered sufficient to represent the real conditions of the road, including straight sections and both right and left turns.

Fig.7. The Process of Determining the Ground Truth

B. Road Recognition using the FCNN Algorithm

The training process was carried out to obtain an FCNN model with good training graphics and a high degree of accuracy. The road recognition process is shown in Fig. 8, in which the process consists of two paths based on the color of the arrows. The black arrows indicate the training process flow, and the green arrows represent the testing process flow. In the training, the original image or images (obtained from the road recording process on the vehicle's front camera) were resized to 320×320 following the FCNN architecture used to make the training time more efficient. After resizing, the data were formed into a dataset that was used in the training process. One by one, the images in the dataset underwent a feedforward process through the network of the FCNN architecture used. After going through the network, the FCNN architecture provided a prediction of the expected road area. This prediction was then compared with the ground truth that had previously been made using the VGG Image Annotator application. This comparison process uses cross-entropy loss to obtain the error value prediction of the model for ground truth. The error value in the cross-entropy loss is used to optimize the model so that it has a better predictive ability. This process is carried out with an optimizer model, which is improved by giving new weights to each network. The optimizers used consist of Adam, SGD, SGD Nesterov's accelerated gradient (NAG), and SGD momentum. Meanwhile, the training is done in 100 epochs. The flow of the evaluation process is marked with a green arrow, as shown in Fig. 8, in which the original image has been resized to match the input size in the FCNN architecture. The resized image was input into the FCNN model, which contained the FCNN architecture and the weights that had been trained on each network. The FCNN model provided a prediction of the recognized road area. The FCNN prediction output was then

resized back to its original size of 1920 × 1080. After resizing, the image was compared with the ground truth to evaluate the performance of FCNN by using a confusion metric. After evaluation with the confusion metric, several parameters were obtained, namely tp, tn, fp, and fn. These could then be used to calculate the accuracy value and the area under the curve (AUC).

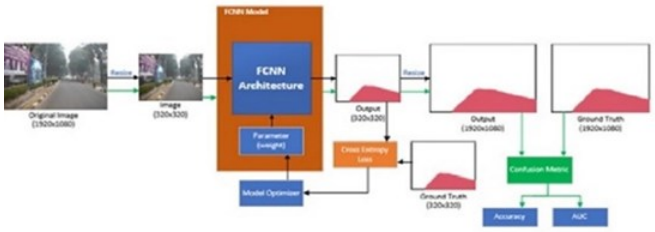


Fig. 8. Road Recognition Process Flow

The evaluation process was carried out using the confusion metric, as shown in Table I.

TABLE I
CONFUSION METRIC

	Classifier		
Ground Truth	*Label*	Positive	Negative
	Positive	tp	fn
	Negative	fp	tn

The equation for finding accuracy can be seen in (7), as follows

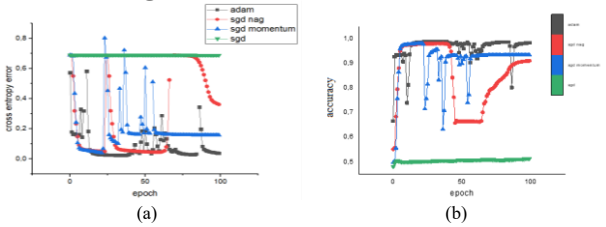
$$AUC = \frac{1}{2}\left(\frac{tp}{tp+fn} + \frac{tn}{tn+tp}\right) \quad (7)$$

The AUC is the classifier's ability to avoid misclassification. To find a measure of the similarity and diversity between the predicted image of the model and target image, an equation known as the intersection over union (IoU) / Jaccard index was used, as follows:

$$IoU = \frac{tp}{tp+fp+fn} \quad (8)$$

IV. RESULT AND DISCUSSION

At the beginning, the optimizers used consisted of Adam, SGD, SGD NAG, and SGD momentum. At this point, the training was done in 100 epochs. The errors of all optimizers can be seen in Fig. 9.



(a) (b)

Fig. 9. (a) Error Graph of Each Optimizer and (b) the Accuracy of Each Optimizer

As shown in Fig. 9, the Adam optimizer had the most stable ability in optimizing the model compared with other optimizers. The Adam algorithm was capable of quickly overcoming error fluctuations in the model and improving it rapidly. To overcome these fluctuations, BN was performed with an architecture, as shown in Fig. 10.

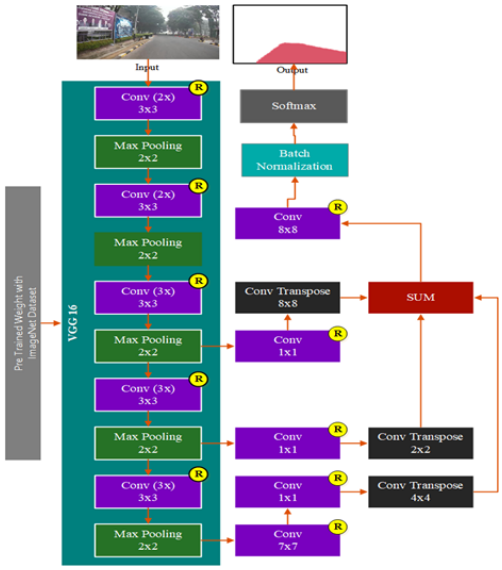
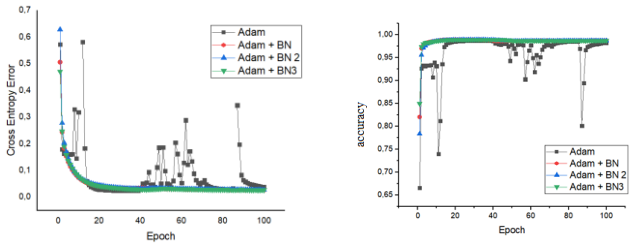


Fig. 10. FCNN Architecture with Batch Normalization.

In addition to the architecture of Fig. 10, the experiments were also done by modifying the position of the layer for batch normalization after the ReLU layer (Adam+BN2) and before the ReLU layer (Adam +BN3). The results can be seen in Fig. 11.



(a) (b)

Fig. 11. (a) Training Error with Modifying Position of Layer Batch Normalization and (b) Accuracy of Training FCNN

The errors for the Adam + BN2 and Adam + BN3 architectures were 0.0270 and 0.0268, respectively. The Adam + BN3 configuration performed better than the Adam + BN2 configuration but was not superior to architectures without batch normalization or with only one batch normalization. With one BN, the error value was 0.0264. This indicates that adding more BNs led to larger error values. Thus, the architecture of Adam with BN was later com-pared with Adam and SGD. Meanwhile, the accuracy for each condition was almost equal, which was 0.9825, 0.9873, 0.9872, and 0.9873 for Adam, Adam+BN, Adam+BN2, and Adam + BN3, respectively. To evaluate the proposed architecture, an accuracy metric was used. In the present study, three optimizer models were used for road recognition: Adam, SGD, and Adam with batch normalization (Adam + BN). The accuracy test results are shown in Fig. 12. The accuracy was calculated using the confusion matrix, as shown in (7).

TABLE II
ROAD RECOGNITION TESTING WITH ADAM

Frame	*Input*	*Output*	*Ground Truth*	Accuracy
1				0.821269
2				0.82138
3				0.821328
4				0.820930
5				0.816804
6				0.822006
7				0.822126
8				0.819515
9				0.819595
10				0.822564

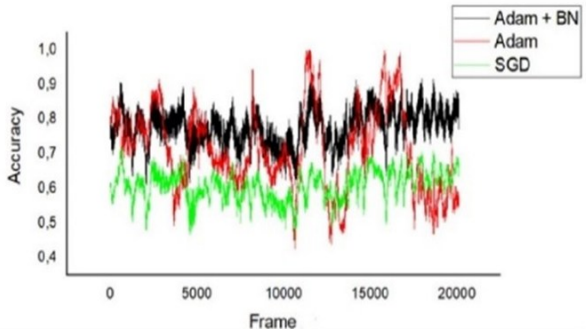


Fig. 12. Accuracy of Path Recognition per Frame on Test Data

Furthermore, road recognition testing can be carried out using Adam, SGD, and the Adam + BN optimizer. Table II shows road testing using the Adam optimizer. Table II shows that the accuracy of the Adam optimizer was quite good, with an average rate of 0.8207 or 82.07%. The introduction of the road using the SGD optimizer is shown in Table III. Table III shows that the accuracy of the SGD optimizer remained below that of the Adam optimizer, with an average rate of 0.6121 or 61.21%. Furthermore, Table IV shows an introduction to the road using the Adam + BN optimizer.

Table III shows that the accuracy of the SGD optimizer remained below that of the Adam optimizer, with an average rate of 0.6121 or 61.21%.

TABLE III
ROAD RECOGNITION TESTING WITH SGD

Frame	*Input*	*Output*	*Ground Truth*	Accuracy
1				0.613821
2				0.614082
3				0.615135
4				0.61503
5				0.608452
6				0.609921
7				0.609539
8				0.611584
9				0.609429
10				0.61392

Furthermore, Table IV shows an introduction to the road using the Adam + BN optimizer Table IV shows that the accuracy in using the Adam + BN optimizer had a rate of 0.7808 or 78.08%. When viewed based on the accuracy shown in Fig. 6, the comparison with the SGD model (with an accuracy of 0.6121) with the Adam model (with an accuracy of 0.8207) and the Adam + BN model obtained an accuracy of 0.7808, whereas the performance of the Adam model was higher than that of SGD and Adam + BN model. Likewise, with the accuracy interval, the Adam optimizer model was far from SGD and Adam + BN, as shown on the interval graph in Fig. 13. The Adam optimizer accuracy value distribution was also collected at greater accuracy points than SGD. This can also be seen in the AUC values, as shown in Table V.

Fig. 13. Graph of Adam and SGD Testing Accuracy Intervals

TABLE IV
ROAD RECOGNITION TESTING WITH THE ADAM + BN OPTIMIZER

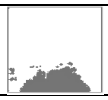
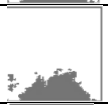
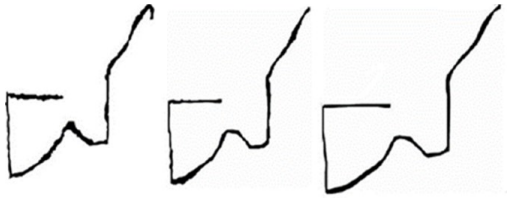
Frame	*Input*	*Output*	*Ground Truth*	Accuracy
1				0.799843
2				0.784479
3				0.792277
4				0.762074
5				0.782425
6				0.778963
7				0.753361
8				0.757267
9				0.797731
10				0.799843

TABLE V
EVALUATION OF THE FCNN ALGORITHM FOR ROAD RECOGNITION

Metric	***Adam Optimizer***	***SGD Optimizer***	***Adam + BN***
Average accuracy for each frame	71.082%	60.641%	77.5847%
AUC	0.719781	0.694611783	0.803014072

The model that successfully recognized the road features (model from the Adam optimizer) was then used as the observation object in the FastSLAM 2.0 algorithm, resulting in a map, as shown in Fig. 14. The map formed with the proposed algorithm was then compared with the map from Google Maps, which showed several additional elements, such as buildings and trees. The map formed by the proposed algorithm only recognized road objects, so Google Maps needed to be reconstructed to form a path that only showed the road on a predetermined route. This reconstruction was carried out to obtain a ground truth image showing the similarity of the map formed by the proposed algorithm to Google Maps. The reconstructed image and Google Maps are shown in Fig. 15.



(a) Adam Model (b) Adam + BN Model (c) Ground Truth

Fig. 14. Formed Map Results

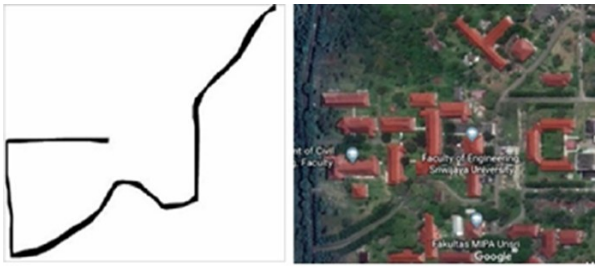


Fig. 15. Ground Truth and Google Maps

The map that was successfully formed was then compared with the ground truth map using the confusion metric to obtain the data shown in Table VI.

TABLE VI
CONFUSION MATRIX MAP FORMED (ADAM MODEL)

Ground Truth	*Label*	Map of the proposed method	
		Positive	Negative
	Positive	1260	3567
	Negative	3606	268467

Based on the confusion matrix table, the IoU/Jaccard index value could be calculated. The amount of IoU from the map formed is 0.149. For maps formed with the Adam + BN model, the IoU value is 0.159, which was better than the previous model. This value was still not sufficient; however, in terms of the route formed by the map, the proposed algorithm succeeded in forming a map according to the specified route. This weakness was because of the rel-atively low accuracy and wide interval of the model used, which created noise in the observation process of the FastSLAM 2.0 algorithm. Nevertheless, this map obtained from the proposed method can still be useful for determining the path planning for an autonomous vehicle.

CONCLUSION

The FastSLAM 2.0 method combined with the FCNN model succeeded in set-ting up a map similar to GPS-based Google Maps. The performance of the algorithm proposed in the formation of this map was still low, with an IoU value of 0.159 in the Adam + BN model. However, the best average rate of accuracy during testing of the FCNN model was 82.07% and 78.08%, found in the Ad-am and Adam + BN models, respectively. Further research will be carried out on the application of this method to an autonomous vehicle on the same route. In addition, further research should be done to consider another algorithm, such as the smooth variable structure filter (SVSF), which makes no assumptions on the noise's properties and face modeling errors and parameter uncertainties.

REFERENCES

[1] H. Cheng, *Autonomous intelligent vehicles: theory, algorithms, and implementation*. Springer Science & Business Media, 2011.

[2] H. Zhu, K.-V. Yuen, L. Mihaylova, and H. Leung, "Overview of environment perception for intelligent vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 10, pp. 2584–2601, 2017.

[3] A. Faisal, T. Yigitcanlar, M. Kamruzzaman, and G. Currie, "Understanding autonomous vehicles: A systematic literature review on capability, impact, planning and policy," *J. Transp. Land Use*, vol. 12, no. 1, pp. 45–72, 2019, doi: 10.5198/jtlu.2019.1405.

[4] Y. Yang, J. Xu, J. Zheng, and S. Lin, "Design and implementation of campus spatial information service based on google maps," in *2009 International Conference on Management and Service Science*, 2009, pp. 1–4.

[5] G. Gramajo and P. Shankar, "Path-planning for an unmanned aerial vehicle with energy constraint in a search and coverage mission," in *2016 IEEE Green Energy and Systems Conference (IGSEC)*, 2016, pp. 1–6.

[6] M. M. Costa and M. F. Silva, "A Survey on Path Planning Algorithms for Mobile Robots," *19th IEEE Int. Conf. Auton. Robot Syst. Compet. ICARSC 2019*, pp. 448–468, 2019, doi: 10.1109/ICARSC.2019.8733623.

[7] H. Li and L. Zhijian, "The study and implementation of mobile GPS navigation system based on Google Maps," in *2010 International Conference on Computer and Information Application*, 2010, pp. 87–90.

[8] S. Li, "A method for building thematic map of GIS based on Google Maps API," in *2011 19th International Conference on Geoinformatics*, 2011, pp. 1–4.

[9] Y.-C. Lee, S.-H. Park, W. Yu, and S.-H. Kim, "Topological map building for mobile robots based on GIS in urban environments," in *2011 8th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, 2011, pp. 790–791.

[10] S. Ambareesh, "Indoor navigation using QR code based on google maps for ios," in *2017 International Conference on Communication and Signal Processing (ICCSP)*, 2017, pp. 1700–1705.

[11] H. Qin *et al.*, "Autonomous Exploration and Mapping System Using Heterogeneous UAVs and UGVs in GPS-Denied Environments," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1339–1350, 2019, doi: 10.1109/TVT.2018.2890416.

[12] X.-Y. Lin, L. Liu, and H.-Z. Luo, "Research on the Method of Eliminating Gross Error of GPS Output Information," in *2011 Fourth International Conference on Information and Computing*, 2011, pp. 46–49.

[13] G. Li, H. Bao, B. Wang, and T. Wu, "Kernelised Rényi Distance for Localization and Mapping of Autonomous Vehicle," in *2017 13th International Conference on Computational Intelligence and Security (CIS)*, 2017, pp. 69–72.

[14] D. Kumiawan, A. N. Jati, and U. Sunarya, "A study of 2D indoor localization and mapping using FastSLAM 2.0," in *2016 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC)*, 2016, pp. 152–156.

[15] X. Xie, Y. Yu, X. Lin, and C. Sun, "An EKF SLAM algorithm for mobile robot with sensor bias estimation," in *2017 32nd Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, 2017, pp. 281–285.

[16] Z. Kurt-Yavuz and S. Yavuz, "A comparison of EKF, UKF, FastSLAM2. 0, and UKF-based FastSLAM algorithms," in *2012 IEEE 16th International Conference on Intelligent Engineering Systems (INES)*, 2012, pp. 37–43.

[17] K. Matsuo and J. Miura, "Outdoor visual localization with a hand-drawn line drawing map using fastslam with pso-based mapping," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 202–207.

[18] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," in *Advances in neural information processing systems*, 2016, pp. 379–387.

[19] L. Wang, W. Ouyang, X. Wang, and H. Lu, "Visual tracking with fully convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3119–3127.

[20] H. Li, A. Li, and M. Wang, "A novel end-to-end brain tumor segmentation method using improved fully convolutional networks," *Comput. Biol. Med.*, vol. 108, pp. 150–160, 2019.

[21] B. Y. Suprapto and B. Kusumoputro, "Optimized neural network-direct inverse control for attitude control of heavy-lift hexacopter," *J. Telecommun. Electron. Comput. Eng.*, vol. 9, no. 2–5, 2017.

[22] E. Yuniarti, N. Nurmaini, B. Y. Suprapto, and M. N. Rachmatullah, "Short Term Electrical Energy Consumption Forecasting using RNN-LSTM," in *2019 International Conference on Electrical Engineering and Computer Science (ICECOS)*, 2019, pp. 287–292.

[23] G. Gojić, R. Turović, D. Dragan, D. Gajić, and V. Petrović, "Automatic Corrections of Human Body Depth Maps using Deep Neural Networks," *SERBIAN J. Electr. Eng.*, vol. 17, no. 3, pp. 285–296, 2020, doi: https://doi.org/10.2298/SJEE2003285G.

[24] S. Zhou, J. Gong, G. Xiong, H. Chen, and K. Iagnemma, "Road detection using support vector machine based on online learning and evaluation," in *2010 IEEE intelligent vehicles symposium*, 2010, pp. 256–261.

[25] M. V. G. Aziz, A. S. Prihatmanto, and H. Hindersah, "Implementation of lane detection algorithm for self-driving car on toll road cipularang using Python language," in *2017 4th International Conference on Electric Vehicular Technology (ICEVT)*, 2017, pp. 144–148.

[26] H. Park, "Implementation of lane detection algorithm for self-driving vehicles using tensor flow," in *International conference on innovative mobile and internet services in ubiquitous computing*, 2018, pp. 438–447.

[27] J. Zang, W. Zhou, G. Zhang, and Z. Duan, "Traffic lane detection using fully convolutional neural network," in *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2018, pp. 305–311.

[28] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," *Aaai/iaai*, vol. 593598, 2002.

[29] C.-C. Hsu, C.-K. Yang, Y.-H. Chien, Y.-T. Wang, W.-Y. Wang, and C.-H. Chien, "Computationally efficient algorithm for vision-based simultaneous localization and mapping of mobile robots," *Eng. Comput.*, 2017.

[30] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *IJCAI*, 2003, pp. 1151–1156.

[31] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[32] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 640–651, 2016.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv Prepr. arXiv1412.6980*, 2014.

[34] P. Golik, P. Doetsch, and H. Ney, "Cross-entropy vs. squared error training: a theoretical and experimental comparison.," in *Interspeech*, 2013, vol. 13, pp. 1756–1760.