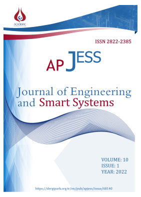

Research Paper
Academic Platform Journal of Engineering and Smart Systems (APJESS) 10(1), 48-56, 2022
ISSN: 2822-2385
Received: 15-May-2021 Accepted: 22-Nov-2021
https://doi.org/10.21541/apjess.1060763

Realization of the Autonomous Driving System on the Experimental Vehicle

*[1]Namig Aliyev, [2]Mehmet Turan Guzel, [3]Oguzhan Sezer

[1]Department of Computer Engineering, Sakarya University, Sakarya, Turkey,
namig.aliyev@ogr.sakarya.edu.tr
[2] Department of Computer Engineering, Sakarya University, Sakarya, Turkey,
mehmet.guzel@ogr.sakarya.edu.tr
[3] Department of Computer Engineering, Sakarya University, Sakarya, Turkey,
oguzhan.sezer1@ogr.sakarya.edu.tr

Abstract

Running control software on limited computing resources is considered one of the toughest problems. In this study, an autonomous driving software has been developed that can safely complete the map by tracking the lanes and avoiding obstacles on a robot vehicle with limited hardware components. The data was simplified with the image processing technique and the neural network was trained. Overfitting was prevented by hyperparameter tuning and synthetic data augmentation. In order to avoid obstacles, optical flow was calculated by detecting corners every 4 seconds and was used to find the focus of expansion of the vehicle. Time-to-collision was found with the FOE and the distance between the previous position and the current position of the detected point. Optimization was made by averaging the values of close points. The balance mechanism was created according to the TTC difference calculated on the right and left parts of the vehicle.

Keywords: Convolutional Neural Network, Overfitting, Hyperparameter Tuning, Data augmentation, Lane tracking, Optical Flow, Focus of Expansion, Time to Collision

1. INTRODUCTION

The autonomous driving system is one of the most popular smart autonomous systems recently. Nowadays, it is aimed to minimize driver-related errors with autonomous driving systems. Today, we can say that autonomous driving systems have speed control with radar and distance sensors, lane tracking, and lane change after cameras on the vehicle. Autonomous vehicles are one of the most effective use cases where hardware and software work together. The hardware enables the vehicle to move and communicate with a range of cameras, sensors, while the software processes information and provides control.

Today, many automobile companies are attempting to produce cars with autonomous driving systems. We can say Tesla company as the leading company. The cars they produce have a full automation driving system. The data set is collected in real-time from approximately 1 million vehicles. 70,000 GPU's are trained per hour. It is capable of semantic segmentation, object recognition, depth estimation. There are 1000 different estimates per step each time. Some companies use the LIDAR device to model depth prediction and 3D perception. Depth prediction is a fundamental task in perceiving the 3D environment around us [1].

In this study, lane tracking, which is one of the two most important abilities in autonomous vehicles, and the ability to avoid obstacles for the robot vehicle to drive freely without hitting any obstacle are discussed. The main purpose of this research was to develop lane tracking and obstacle avoidance capabilities with different methods and solutions for an autonomous driving system on an experimental vehicle.

The robot vehicle, remote control module, and experimental map that constitute the hardware part of the project were prepared. Raspberry Pi module on the vehicle forms the brain of the vehicle. Raspberry PI communicates with the remote-control module via wireless network (RF24) and computer via embedded software. The Raspberry PI module, which plays the role of the brain of the system in the later stages of the project, was renewed with the Coral Dev Board [2] device developed by Google for artificial intelligence model's due to its inadequate performance.

Supervised Learning [3, 4] approach, which is one of the Machine Learning [5, 6, 30] techniques, was used for lane tracking. With the help of a remote-control device, the robot vehicle was moved along the track and dataset collection was carried out through the camera on the vehicle. This dataset created consists of images and action information taken at

[*] Corresponding Author

the time of that image. Then, the images taken were simplified with image processing techniques, and the strip lines were brought to the fore. At this stage, Convolutional Neural Network [7] was used while creating an artificial intelligence model. CNN is a type of artificial neural network developed to solve problems such as image classification, object detection, and style transfer. Since the images are our main data source, it was decided to use CNN.

The target problem for avoiding obstacles is the calculation of contact time or time to collision. The most important feature focused here was the calculation of the time until the collision, ie the contact time. In this direction, corner detection, optical flow focus of expansion, and collision time were calculated instantaneously on the image taken from the camera [8 - 10]. The balance calculation has been made for the right and left body of the robot vehicle and a decision mechanism has been created to avoid obstacles. The robot vehicle has been provided to move without hitting any obstacle.

There are many studies on road lane tracking in the literature. Bounini and Farid [11] obtained a result by detecting corners on the data taken from the camera image. J. Han, D. KIM [12] using 2D Lidar sensors, they were able to gather information about the environment and keep the vehicle within the lane by performing road boundary extraction. There are a few studies investigating vehicle obstacle avoidance using only information extracted from the camera image. Kachluche Souhila and Achour Karim [13] measured the distance to objects using optical flow and corner detection.

2. PROPOSED METHOD

2.1. Lane Tracking

In this section, simplification of lane information with computer vision techniques, and data set collection are given initially to enable the robot vehicle to move autonomously by following the lane information on the experimental map. Next, a detailed description of the designed network architecture and training is provided. To achieve the successful model, hyper parameter tuning and data augmentation, and finally, the testing process is explained.

The dataset collection process will be performed by moving the robot vehicle over the experimental environment with the help of a remote control. The images taken from the camera correctly positioned on the vehicle will first be recorded in the filing system by simplifying the lane information with image processing techniques. At the same time, the action information of the car at the time the image is taken is recorded in the filing system simultaneously with the images.

2.1.1. Simplifying Lane Information with Computer Vision Techniques

Preparing the images in the data set that we will give to the neural network in accordance with the purpose is the most influential factor in the result of the developed neural network model. If the image is messy, difficult to understand, and the neural network is not able to distinguish the features in the image, the error values of the model will be high and the operation is nothing but a waste of time. For this purpose, the images taken from the camera on the robot vehicle were first simplified with image processing techniques. First, the color space change was perform It is planned to increase frames per second in the future and add new capabilities using 2D Lidar and Google Coral Dev Board. It is planned to increase frames per second in the future and add new capabilities using 2D Lidar and Google Coral Dev Board. It is planned to increase frames per second in the future and add new capabilities using 2D Lidar and Google Coral Dev Board. med on the image. Many color spaces are supported in the OpenCV library and you can convert between them. In the first step, the image was converted from RGB color space to grayscale color space [14].

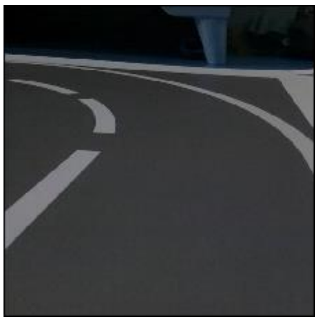

Figure 1. Change from R.G.B. color space to grayscale color space

Figure 1. shows the image obtained by converting the camera image taken on the robot vehicle from RGB colour space to grayscale colour space. In the image in the grayscale colour space obtained, the stripe lines are desired to be prominent. With the help of the Canny [15] edge detection algorithm, the strip lines required on the image were made more prominent.

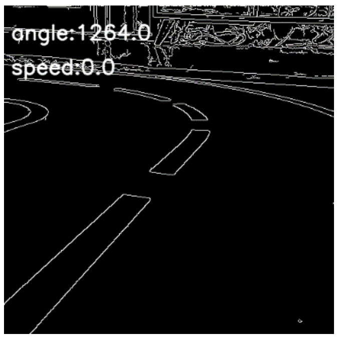

Figure 2. Transformation of grayscale image with Canny edge detection algorithm

The image taken from the camera has been successfully simplified and made ready for the use of the neural network. If you pay attention to the upper left corner of the figure, you can see the angle and speed, which are the action information of the vehicle at the time the image is taken from the camera.

2.1.2. Data Set Collection and Editing

The collection of the data set will be carried out by moving the robot vehicle over the experimental environment with the remote control. In the Supervised Learning machine learning approach we will use, the data to be trained should be given to the learner as x and y outputs. While the vehicle is controlled remotely on the experimental environment we have prepared before, the image from the camera, the current servo angle and engine speed are recorded in the filing system simultaneously.

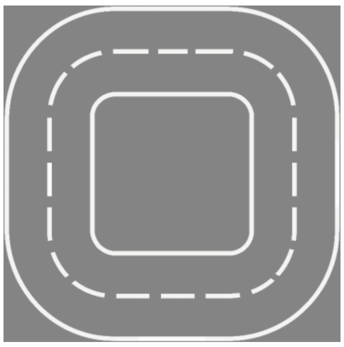

Figure 3. Experimental map on which the robot vehicle will be moved

In the Supervised Learning approach, the data should be given to the trainer as x and y outputs. While the robot vehicle was being moved over the experimental environment, the servo angle information, which is the current action information, was recorded in the filing system along with the camera image. According to the general structure of the experimental map, the servo x angle values in the data collected vary between 1270 and 600.

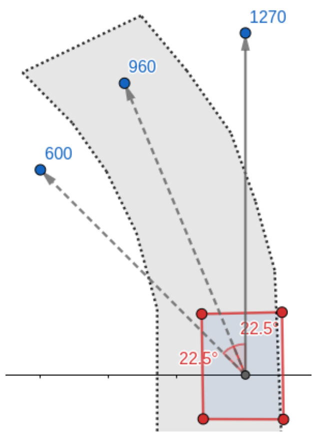

Figure 4. The angles the robot vehicle takes at the moment of movement.

Angle data collected at this stage has a complex structure and needs to be simplified. For this purpose, while the robot vehicle is moving on the experimental map, the angle information as *FLAT*, *MIDDLE*, and *SHARP* is updated in the filing system by simultaneously looking at its location on the map and the instant angle information from the computer. Alternatively, the angle information, which is a parameter of the data set, can be compressed between 0 and 1 for linear regression [16, 17], allowing linear estimation.

2.1.3. CNN Neural Network Model

While the robot vehicle is in motion, it should analyze the environmental conditions and make control predictions. Environmental conditions consist of data collected in the previous topic. The robot vehicle needs a system that can use this data and make predictions.

Artificial neural networks are Artificial Intelligence structures that are trained with the given data and can make predictions according to the information they learn.

In this subsection, the design of the architecture and the training of the model presented initially. Next, the hyper parameter tuning [18-19], and data augmentation [20], and finally the testing and result are explained in detail.

2.1.4. Designing and Training the Model

Before creating a neural network model, the neural network structure to be used is decided by considering the data set, project conditions and properties. The main source of data consists of images. The neural network is required to be predicted according to the images and action information taken from the camera. The neural network will distinguish the features in the images taken from the camera and perform the learning and prediction processes. Therefore, it was decided to use convolutional neural networks at this stage of the project.

Data set consists of binary color pictures simplified with Canny edge detection algorithm and angle information, which is the action information at the time the picture is taken. The stored images were resized to 128 x 128 pixels before being transferred to the model. The action information, *FLAT*, *MIDDLE* and *SHARP*, are updated to correspond to 0, 1, and 2, respectively. Due to the general structure of neural networks, the complexity of the structure is directly proportional to the estimation time. Therefore, it is important that the model to be designed has a simple structure. On the other hand, the education period of the models with a simple structure is short and time saving is obtained. Another issue in neural networks is that there are no rules for establishing the best model. For this reason, until we find the model with which we have achieved high performance, the models have been designed by taking the available data into consideration.

The steps to be taken during the training of our artificial neural network model are as follows. The first step is to read and store the data set and mix it randomly. The second step is to separate 70% of the data set as training data and 30% as test data. After separating the training and test data set, an image from the training data set is given to our model and the weights are updated according to the error value. Figure 5. shows the flow chart representing the training process of the model.

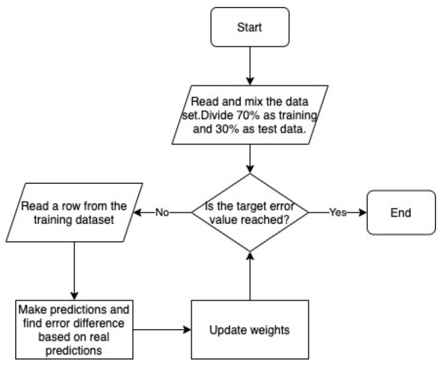
Start
Read and mix the data set.Divide 70% as training and 30% as test data.
Is the target error value reached?
No
Yes
End
Read a row from the training dataset
Make predictions and find error difference based on real predictions
Update weights

Figure 5. The algorithm flow chart representing the training process of neural network model.

The first layer of the first model prepared is a convolution layer with 32 x 32 depth and 3 x 3 filter dimensions. Input data is 128 x 128 x 2 size simplified image with Canny edge detection algorithm. Relu, the next layer activation function, has been applied. Two Max-Pooling layers were then applied. Filter dimensions of the Max-Pooling layers are determined as 2 x 2. By applying Max-Pooling layers in succession, which yields a feature map of the 32 × 32 × 32 size. Next, the Flatten and Dense layer added.[21].

```
             Name          Type              Shape
0       conv2d_48        Conv2D  (None, 128, 128, 32)
1   activation_24    Activation  (None, 128, 128, 32)
2 max_pooling2d_30 MaxPooling2D    (None, 64, 64, 32)
3 max_pooling2d_31 MaxPooling2D    (None, 32, 32, 32)
4      flatten_26       Flatten         (None, 32768)
5        dense_25         Dense             (None, 3)
-------------------------------------------------------
Total params: 98915
```

Figure 6. First CNN model summary.

Considering the Raspberry PI module performance, the first model was kept simple and the total number of calculated parameters was obtained as 98915.

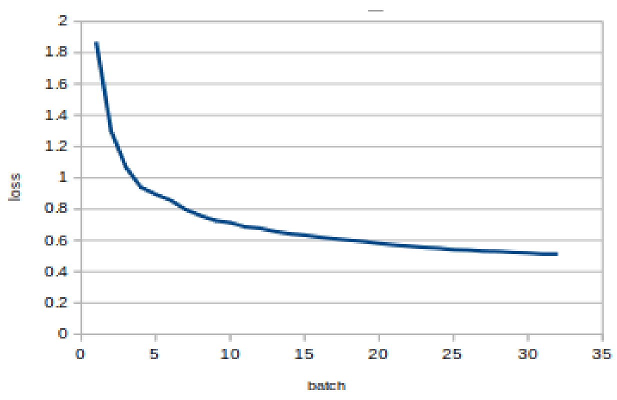

Figure 7. Loss graph of the first CNN model

When the loss graph in Figure 7. is examined, it is seen that during the batch of 32 pictures each, it goes to overfitting quickly [22, 23]. It has been observed that the loss of the neural network model rapidly approaches zero at the end of one epoch.

The result we will get here is that the dropout layer used to reduce overfitting is insufficient. In order to eliminate this problem caused by the fact that the dataset consists of few and similar images, data diversity will be increased by data augmentation. Thus, a more general model that can respond to real-life problems will be obtained by considering parameters such as lighting conditions and noise in the image.

2.1.5. Hyperparameter Tuning and Data Augmentation

While designing a model in artificial neural networks, there is no rule to reach a successful model. There is no rule to be followed in line with the information obtained from the studies conducted on this subject in the world so far. The improvement of the model is done by techniques such as trial-and-error method, hyperparameter tuning [18] and data augmentation [24].

One of the hyperparameter adjustment is to prevent overfitting. The dropout layer which have been added with 0.25 value to the model is a regularization approach [21] that helps reduce dependent learning between neurons. Another hyperparameter regulation is increase the computable parameter counts. The Max-Polling layer has been removed and a new convolution layer of 16 x 16 depth and 3 x 3 filter dimensions has been added.

```
             Name          Type              Shape
0       conv2d_49        Conv2D  (None, 128, 128, 32)
1       conv2d_50        Conv2D  (None, 128, 128, 16)
2 max_pooling2d_32 MaxPooling2D    (None, 64, 64, 16)
3      flatten_27       Flatten         (None, 65536)
4      dropout_25       Dropout         (None, 65536)
5        dense_26         Dense             (None, 3)
-------------------------------------------------------
Total params: 201843
```

Figure 8. Third neural network with hyperparameters tuned.

The total number of parameters calculated after tuning was obtained as 201843. The data given to the model were augmented by producing synthetic data with the data augmentation technique [24].

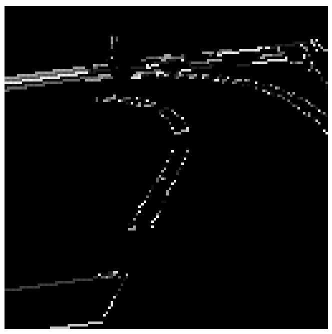

Figure 9. Synthetic image created through data augmentation.

Figure 9. shows the synthetic image obtained after applying flip, shift, and zoom to the real image. These variations in the data set enable the trained model to achieve a similar performance in different image conditions. Thus, a more general solution will be achieved. The loss graph of the model trained after data augmentation is shown in Figure 10.

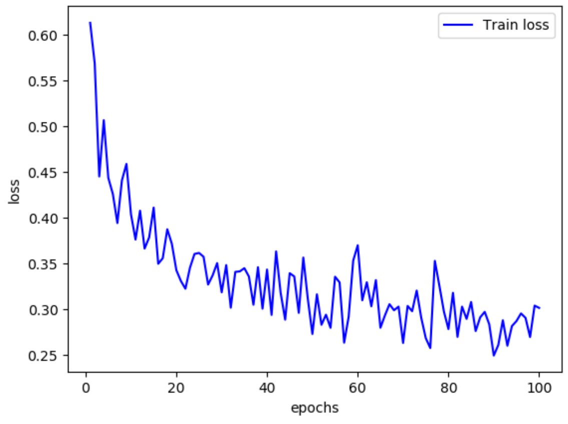

Figure 10. Loss graph of the model trained after data augmentation

There is an obvious improvement in overfitting [23] rate compared to previous training. It is seen that the loss ratio that converges rapidly to zero at the batch level before is now decreasing at the epoch level. At the end of 100 epoch, the lowest loss value was reached as 0.25 in the 90th epoch.

New synthetic data were generated by making changes in the augmentation parameters to reduce the loss value even smaller. Zoom ratio decreased from 0.4 to 0.2, flip angle from 40 degrees to 10 degrees. The changes applied here will make less distortion of the lane information in the image and help achieve the goal of obtaining a more general model. The highest performing neural network model has been retrained with the new dataset, and the expected reduction in loss data occurred.

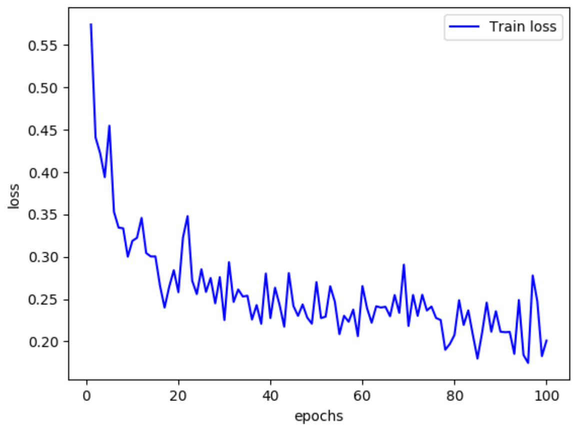

Figure 11. Loss graph of the most successful CNN model.

As seen in the graph, the 95th epoch has reached 0.17 loss value. Thus, it was observed that the change made in the data augmentation parameters had an effect on the decrease of the loss value. The final model was trained three times over 100 epochs with synthetic data. The number of frames per second, which represents the reaction speed of the vehicle, reached the maximum 14 frames per second which is the best result of all time.

2.2. Obstacle Avoidance with Optical Flow

While an autonomous robot vehicle is moving in a constant velocity, the time until the collision can be found without any knowledge of the distance to be traveled or the velocity the robot is moving [8]. Calculating the time to collision is one of the practical optical flow uses. The optical flow knowledge is extracted from the image sequence taken from the Google camera placed in the robotic vehicle, and then the time until the robot reaches a particular area is determined. Calculated collision times are considered separately as collision times on the left and right of the image. Depending on whether the difference between the collision times of the left and right side is higher or lower than a certain threshold value, the vehicle is ordered to ignore the obstacle in front of it or to take action.

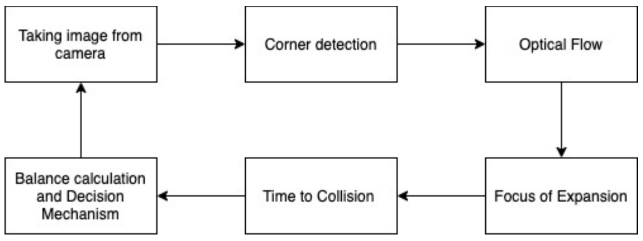

Figure 12. The flow char representing obstacle avoidance procedure [26].

In this section, corner detection, and calculating optical flow are introduced in the first place. After, the focus of expansion and time to collision calculation procedures are explained. Next, the balance strategy and decision mechanism are explained in detail and the movement of the vehicle is presented according to the decision produced by the mechanism.

2.2.1. Corner Detection with FAST (Features from Accelerated Segment Test)

It is necessary to extract the optical flow information from the image sequence taken from the camera. To find the optical flow between consecutive frames, the motion of a pixel feature set should be tracked. Features in the image are points of interest that provide rich picture content information, and these points are not affected by intensity changes in the image [27].

Using the FAST [26] algorithm, which is known for its high performance in real-time images, corner detection performed in the real-time image sequence.

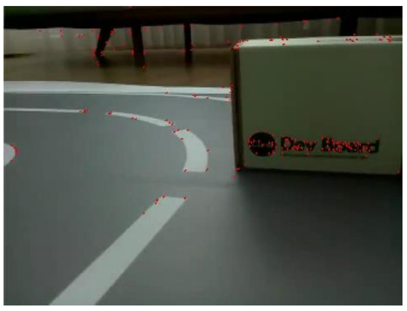

Figure 13. Corner detection on the image with FAST algorithm.

Figure 13. shows the image formed after applying the corner detection algorithm on the image. The corner detection process is run every 50th iteration of the runtime, which means that the corners are refreshed at approximately 3-4 second intervals. The detected corners are stored on a vector for later use in calculating the optical flow.

2.2.2. Calculating Optical Flow

For the optical flow to be computable, a selected point on the first image must change its location on the next image. While the selected point is moving, the shape of the light reflected on that point is constantly changing and optical flow occurs. In other words, the vehicle must be moving in order to obtain optical flow with the robot vehicle. The most widely used Lucas-Kanade [27] method was used to calculate the optical flow between consecutive frames. The vector containing the vertices detected by the FAST corner detection algorithm is given to the function and it returns two vectors containing the (x, y) coordinates of the previous and next points. Now that the changing coordinates of a corner point in the previous and ongoing frame are known, an arrow can be drawn from the previous position to the next position. In other words, an arrow is drawn in the direction of the point's movement in consecutive frames if the tracked corner point exists (detected) in the next frame. Suppose (x_2, y_1) and (x_2, y_2) are the coordinates of the point in the previous and next squares.

$$angle = arctan\left(\frac{y2-y1}{x2-x1}\right) \qquad (1)$$

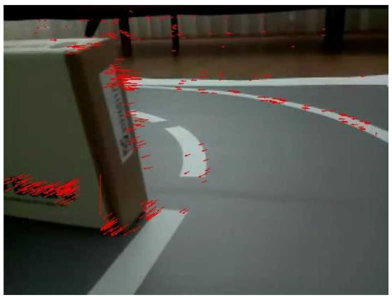
Figure 14. Arrows were drawn in the direction of movement of the points.

$$arrow_x = x2 + len * cos\left(angle + \frac{3.14}{180}\right)$$
$$arrow_y = y2 + len * sin\left(angle + \frac{3.14}{180}\right) \qquad (2)$$

2.2.3. Calculating Focus of Expansion

The motions of objects moving around are projected to the eyes of the observer as two fundamental motions. An optical flow field is formed as a result of the projection of the translation and rotation fundamental motions into an image plane [8]. Rotational motion can be imagine as flow vectors produced as a result of the surrounding objects shifting left or right as the robot vehicle turns left or right.

Translation motion occurs when the camera is moving forward or backward. If the camera moves backward, it creates an area called a focus of contraction (FOC) where the flow vectors converging around a point. On the contrary, if it moves forward, it creates an area called the focus of expansion (FOE) where the flow vectors diverge around from a central point.

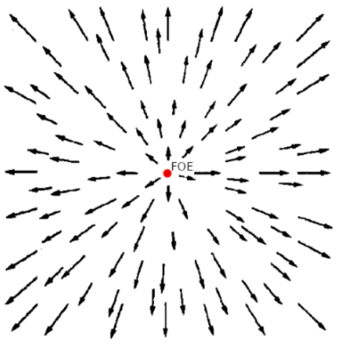

Figure 15. Diverging flow vectors and focus of expansion during forward translation motion.

Any two vectors are needed to calculate the focus of expansion. If the place where these two vectors meet can be determined, the focus of expansion is found. The least-squares [28] solution of all available flow vectors was used to find focus of expansion. Each optical flow vector has a previous point and delta. Let pt = (x, y) be the x and y coordinates of the previous position of an optical flow vector. Let v = (u, v) be the x and y coordinate differences between the previous and current position of the optical flow vector.

$$A = \begin{bmatrix} a_{00}\, a_{01} \\ \cdots \;\; \cdot\cdot \\ a_{n0}\, a_{n1} \end{bmatrix} \qquad b = \begin{bmatrix} b_0 \\ \cdot\cdot \\ b_n \end{bmatrix} \qquad (3)$$

In matrix A it should be known as a_{i0} = -v and a_{i1} = u, and in matrix B, each value is obtained by b_i = xv – yu. The focus of expansion is calculated using the least-squares method and inversion of the matrices.

$$FOE = (A^T A)^{-1} A^T b$$
$$= \begin{bmatrix} \sum a_{i0}\, b_i \sum a_{j1}^2 - \sum a_{j1}\, b_i \sum a_{j0}\, a_{j1} \\ -\sum a_{i0}\, b_i \sum a_{j0}\, a_{j1} + \sum a_{i1}\, b_i \sum a_{j0}^2 \end{bmatrix} - \frac{1}{\sum a_{j0}^2 a_{j1}^2 - (\sum a_{i0} a_{i0})^2} \qquad (4)$$

The OpenCV library has the necessary functionality for the matrix inversion method. The function is given matrices A and b and an empty matrix of 2x1 dimensions. Additionally, the DECOMP_QR flag was added for QR decomposition [29].

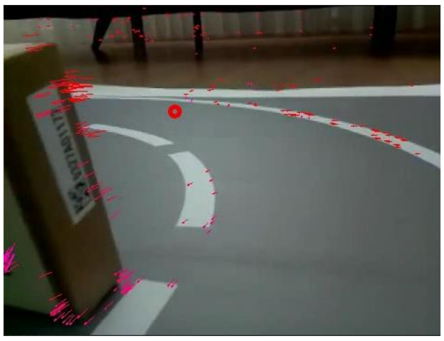
Figure 16. Calculation of the focus of expansion on the image taken from the camera. The FOE is shown by the red circle in the image.

2.2.4. Calculating Time to Collision

The most valuable information we can obtain for the robot vehicle to avoid obstacles is the determination of the contact time or the time until the collision. This information can be found without requiring any information about the distance to travel or the velocity the robot is moving [8].

The studies carried out up to this stage were to obtain information to be used in the TTC because when calculating the TTC, optical flow vectors and FOE are required. Let p = (x, y) be the x and y positions of an optical flow vector and FOE = (x, y) be the x and y positions of a focus of expansion. Let v = (u, v) be the x and y coordinate differences between the previous and current positions of the optical flow vector.

$$TTC = \sqrt{\frac{(p_x - foe_x)^2 + (p_y - foe_y)^2}{u^2 + v^2}} \tag{5}$$

The locations of the detected points on the image vary according to the position and movement of the objects captured by the robot's camera. There are situations where the detected points on the image are not equably positioned in the whole image plane. This means there are more corners in some parts of the image and fewer corners in others. This imbalance caused by the objects and movements in the frame can be avoided by using 16 x 16 dimensional matrices.

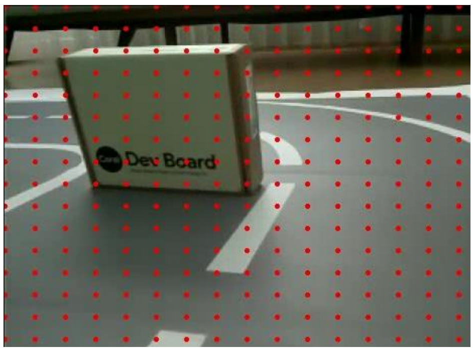
Figure 17. Drawing the matrix on the image.

After calculating the TTC of each flow vector, it is collected at one of the closest A matrix points in the image. In matrix B, the number of TTCs collected at each matrix point is stored.

$$A = \begin{bmatrix} \sum ttc_i & \cdots & \sum ttc_i \\ \vdots & \vdots & \vdots \\ \sum ttc_i & \ddots & \sum ttc_i \end{bmatrix}_{16x16} \qquad b = \begin{bmatrix} a_{00} & \cdots & a_{0n} \\ \vdots & \vdots & \vdots \\ a_{n0} & \ddots & a_{nn} \end{bmatrix}_{16x16} \tag{6}$$

ttc$_i$ is the time until the collision of a flow vector, and a in matrix B is the number of flow vectors. Using matrices A and b, the average TTC for each matrix point can be calculated.

$$TTC[i] = \frac{A[i]}{b[i]} \tag{7}$$

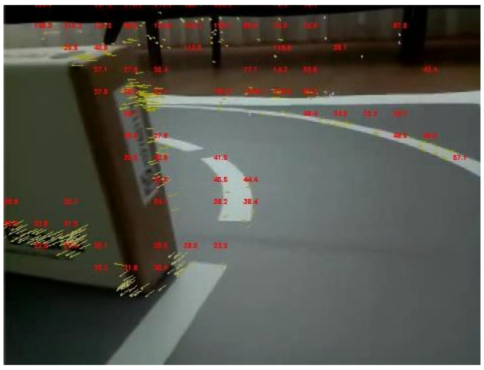
Figure 18. Shows the average collision times. It seen that vectors with the large optical flow on the image have low TTC values.

2.2.5. Balance Calculation and Decision Mechanism

The basic idea is that when the robot is in motion, close objects move faster than farther objects on the retina. Also, closer objects cover the field of view more, causing greater optical flows. In the region where the optical flow is greater, the collision time is low and the robot vehicle must go to the other side and avoid from the obstacle. Kachluche Souhila and Achour Karim in their article [13], they present a different perspective in which the robot car moves away from the side where there is greater optical flow.

The image taken from the camera is divided into two parts to give the vehicle balance. Balance can be achieved by minimizing the difference between collision times on the left and right side of the vehicle.

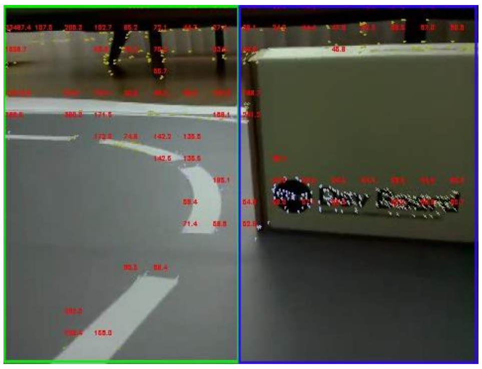
Figure 19. Dividing the image into two parts.

The following control formula is used to calculate the difference between collision times on the left and right side of the vehicle.

$$\Delta(F_L - F_R) = \frac{\sum||TTC_L|| - \sum||TTC_R||}{\sum||TTC_L|| + \sum||TTC_R||} \tag{8}$$

Here (F_L-F_R) is the difference between the forces on both sides of the robot body and TTC is the average of the collision time in the visual half-field on one side. The difference between the forces calculated in equation 8 is a linear number and varies between 0 and 1.The balance mechanism was applied to the robot vehicle. Due to the environmental conditions of the experimental environment, the threshold value of the difference between the left and right TTC was determined as 0.5. In cases where the threshold value is exceeded, the robot vehicle will be given the necessary rotation order. As shown in Figure 21, the left collision time is calculated as (3895.2), the right collision time (101820.7) and the difference in forces (0.9) were found. It is decided to turn right, because the left collision time is less than the right, and the difference in forces is greater than 0.5.

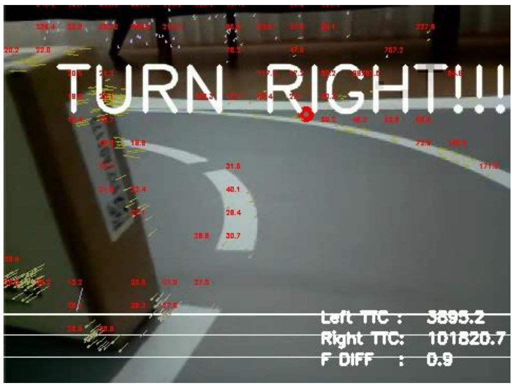

Figure 20. Avoiding the box to the left of the robot vehicle.

In Figure 21, it can be seen that the robot vehicle is given a forward motion command. Left collision time (993.2), right collision time (339.8) and difference in forces (0.1) were found.

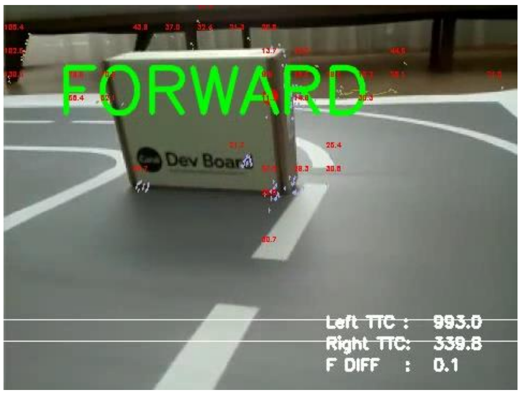

Figure 21. Robot vehicle is commanded to go forward.

3. CONCLUSION

This article presented develop lane tracking and obstacle avoidance capabilities with different methods and solutions for an autonomous driving system on an experimental vehicle.

Considering the Raspberry PI module performance, the first model was kept simple and fast overfitting was observed due to the small dataset. In order to develop prediction of the model and prevent overfitting, the Dropout layer was added, the dataset was enlarged by generating synthetic data, and the number of computable parameters was increased. The final model has been trained three times over 100 epochs with synthetic data.

Training of the neural network model was done with simplified images. During the test stage, the images taken from the camera should be similar to the images used in the training stage of the neural network. The similarity emphasized here is that the images are in the same colour space and simplified. For this reason, the images taken from the camera during the test stage are instantly simplified and then transmitted to the neural network. The prepared neural network model produces 0, 1, and 2 as output. These correspond to the values for FLAT, MEDIUM, and SHARP, respectively. Since these labels are obtained by simplifying the rotation angle of the servo, the rotational motion is provided by converting the predictions back to the servo angle with the help of an algorithm.

In order to avoid obstacles, optical flow was calculated by detecting corners every 4 seconds by FAST algorithm and was used to find the focus of expansion of the vehicle. Time-to-collision was found with the FOE and the distance between the previous position and the current position of the detected point. There are situations where the detected points on the image are not equably positioned in the whole image plane. For this reason, the values of the close points are averaged and placed in the 16x16 matrix. The balance mechanism was created according to the TTC difference calculated on the right and left parts of the vehicle.

Frames per second, representing the vehicle's response speed, reached 14 frames per second when following the lane, 20 frames when avoiding obstacles, and 12 frames when both modules were working together. And the vehicle completed the map safely without hitting any obstacle.

It is planned to increase frames per second in the future and add new capabilities using 2D Lidar and Google Coral Dev Board.

Acknowledgement: This work was supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK), Grant No: 1919B011903963

REFERENCES

[1] Casser, Vincent, et al. "Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 33. No. 01. 2019.
[2] Google LLC, “Get started with the Dev Board.” 2020.
[3] Caruana, Rich, and Alexandru Niculescu-Mizil. "An empirical comparison of supervised learning algorithms." Proceedings of the 23rd international conference on Machine learning. 2006.
[4] Zhu, Xiaojin, and Andrew B. Goldberg. "Introduction to semi-supervised learning."Synthesis lectures on artificial intelligence and machine learning 3.1 (2009): 1-130.
[5] Alpaydin, Ethem. “Introduction to machine learning.” MIT press, 2020.
[6] Schalkoff, Robert J. "Pattern recognition." Wiley Encyclopedia of Computer Science and Engineering (2007).
[7] Albawi, Saad, Tareq Abed Mohammed, and Saad Al-Zawi. "Understanding of a convolutional neural network."2017 International Conference on Engineering and Technology (ICET). Ieee, 2017.
[8] O’Donovan, Peter. "Optical flow: Techniques and applications."International Journal of Computer Vision (2005): 1-26.

[9] Beauchemin, Steven S., and John L. Barron. "The computation of optical flow."ACM computing surveys (CSUR) 27.3 (1995): 433-466.

[10] Barron, John L., and Neil A. Thacker. "Tutorial: Computing 2D and 3D optical flow."Imaging science and biomedical engineering division, medical school, university of manchester 1 (2005).

[11] Bounini, Farid, et al. "Autonomous vehicle and real time road lanes detection and tracking." 2015 IEE Vehicle Power and Propulsion Conference (VPPC). IEEE, 2015

[12] Han, J., et al. "Road boundary detection and tracking for structured and unstructured roads using a 2D lidar sensor." International Journal of Automotive Technology 15.4 (2014): 611-623

[13] Souhila, Kahlouche, and Achour Karim. "Optical flow based robot obstacle avoidance."International Journal of Advanced Robotic Systems 4.1 (2007): 2.

[14] Saravanan, C. "Color image to grayscale image conversion."2010 Second International Conference on Computer Engineering and Applications. Vol. 2. IEEE, 2010.

[15] Xu, Zhao, Xu Baojie, and Wu Guoxin. "Canny edge detection based on Open CV. "2017 13th IEEE international conference on electronic measurement & instruments (ICEMI). IEEE, 2017.

[16] Edwards, Allen L. An introduction to linear regression and correlation. No. 04; QA278. 2, E3 1984.. 1984

[17] Montgomery, Douglas C., Elizabeth A. Peck, and G. Geoffrey Vining. Introduction to linear regression analysis. John Wiley & Sons, 2021.

[18] Bergstra, James, et al. "Algorithms for hyper-parameter optimization."25th annual conference on neural information processing systems (NIPS 2011). Vol. 24. Neural Information Processing Systems Foundation, 2011.

[19] Feurer, Matthias, and Frank Hutter. "Hyperparameter optimization." Automated Machine Learning. Springer, Cham, 2019. 3-33.

[20] Mikołajczyk, Agnieszka, and Michał Grochowski. "Data augmentation for improving deep learning in image classification problem."2018 international interdisciplinary PhD workshop (IIPhDW). IEEE, 2018.

[21] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." The journal of machine learning research 15.1 (2014): 1929-1958.

[22] Ciliberto, Carlo, Lorenzo Rosasco, and Alessandro Rudi. "A consistent regularization approach for structured prediction." Advances in neural information processing systems 29 (2016): 4412-4420.

[23] Hawkins, Douglas M. "The problem of overfitting. "Journal of chemical information and computer sciences 44.1 (2004): 1-12.

[24] Takahashi, Ryo, Takashi Matsubara, and Kuniaki Uehara. "Data augmentation using random image cropping and patching for deep cnns." IEEE Transactions on Circuits and Systems for Video Technology 30.9 (2019): 2917-2931.

[25] Fleet, David, and Yair Weiss. "Optical flow estimation." Handbook of mathematical models in computer vision. Springer, Boston, MA, 2006. 237-257.

[26] Viswanathan, Deepak Geetha. "Features from accelerated segment test (fast)." Proceedings of the 10th workshop on Image Analysis for Multimedia Interactive Services, London, UK. 2009.

[27] Lucas, B. and Kanade, T. 1981. An iterative image registration technique with an application to stereo vision.In Proc. Seventh International Joint Conference on Artificial Intelligence, Vancouver, Canada, pp. 674–679.

[28] Levenberg, Kenneth. "A method for the solution of certain non-linear problems in least squares." Quarterly of applied mathematics 2.2 (1944): 164-168.

[29] Gander, Walter. "Algorithms for the QR decomposition." Res. Rep 80.02 (1980): 1251-1268.

[30] Satti, Satish Kumar, et al. "A machine learning approach for detecting and tracking road boundary lanes." ICT Express 7.1 (2021): 99-103