Bitlis Eren Üniversitesi Fen Bilimleri Dergisi
BİTLİS EREN UNIVERSITY JOURNAL OF SCIENCE
ISSN: 2147-3129/e-ISSN: 2147-3188
VOLUME: 13 NO: 2 PAGE: 384-392 YEAR: 2024
DOI:10.17798/bitlisfen.1341929

Layer Selection for Subtraction and Concatenation: A Method for Visual Velocity Estimation of a Mobile Robot

Mustafa Can BINGOL[1*]

[1] *Department of Electrical-Electronics Engineering, Burdur Mehmet Akif Ersoy University, Burdur 15100, Türkiye*

(ORCID: 0000-0001-5448-8281)

Keywords: Deep learning, Mobile robot, Velocity estimation.

Abstract
Kinematic information such as position, velocity, and acceleration is critical to determine the three-dimensional state of the robot in space. In this study, it is aimed at estimating, visually, the linear and angular velocity of a mobile robot. Additionally, another aim of this study is to determine the suitability of the concatenation or subtraction layer in the Convolutional Neural Network (CNN) that will make this estimate. For these purposes, first, a simulation environment was created. 9000 pairs of images and the necessary velocity information were collected from this simulation environment for training. Similarly, 1000 pairs of images and velocity information were gathered for validation. Four different CNN models were designed, and these models were trained and tested using these datasets. As a result of the test, the lowest average error for linear velocity estimation was calculated as 0.93e-3m/s, and the angular velocity estimation was measured as 4.37e-3rad/s. It was observed that the results were sufficient for linear and angular velocity prediction, according to the statistical analysis of errors. In addition, it was observed that the subtraction layer can be used instead of the concatenation layer in the CNN architectures for hardware-limited systems. As a result, visual velocity estimation of mobile robots has been achieved with this study, and the framework of CNN models has been drawn for this problem.

1. Introduction

Robotic systems must be able to measure their kinematic information, such as position, velocity, and acceleration. Also, these systems must be able to control this information according to the desired behavior in order to move autonomously in an environment. For example, Aydogmus and Boztas controlled the linear and angular velocities of a mobile robot by using the pure pursuit algorithm [1]. In another study, the design and analysis of a whole-body controller were realized for a velocity-controlled robot mobile manipulator [2]. The velocity of an omnidirectional wheeled mobile robot was controlled using computed voltage control with visual feedback [3]. In another study, a novel path-planning algorithm for Ackermann mobile robots was been developed. This algorithm is based on B-spline curves and the ant colony algorithm [4]. An omnidirectional mobile robot was real-time navigated in a dynamic environment by using a velocity obstacle and a hybrid A* algorithms [5].

Artificial intelligence is frequently used in many fields, from health to robotics. To illustrate, a liver tumor diagnosis was realized using deep learning techniques and CT - MR imaging [6]. In another study, an industrial robot was converted into a collaborative robot that did not harm people by utilizing deep learning techniques [7]. Artificial intelligence can be subdivided into two categories: deep learning and machine learning. In this study, Convolutional Neural Network (CNN) architectures,

[*]Corresponding author: *mcbingol@mehmetakif.edu.tr*

Received: 12.08.2023, Accepted: 07.06.2024

a sub-branch of deep learning, were used. CNN is a frequently used structure in robotic systems. For example, delamination was predicted using multimodal 1D CNN during the drilling process of carbon fiber-reinforced plastics [8]. Park et al. developed a novel vision-based autonomous pick and place method utilizing CNN [9]. A farm robot that formed fertilizing and cropping was developed using CNN [10].

In this study, a CNN-based estimation of the linear and angular velocity of a mobile robot is aimed at. There are few similar studies to the current study in the literature. For example, the velocity of omnidirectional mobile robots was estimated using an optical mouse [11]. In another study, state estimation of a mobile robot that was slip-velocity-aware was realized via invariant Kalman filtering and disturbance observer [12]. Arteaga–Pérez and Nuño designed a velocity observer for the consensus in delayed robot networks [13]. When the studies in the literature are examined, studies on velocity estimation of a mobile robot using artificial intelligence methods are limited.

From the past to the present, many researchers have worked on the control and positioning of mobile robots. In fact, the reason why these problems have not been solved is that the exact position of mobile robots is not known. In the current study, in addition to the methods in the literature, the velocity of a mobile robot was estimated by interpreting the images taken at different moments in a CNN-based architecture. In addition, different models have been tested for the selection of the layer to connect the input images in this CNN structure. In this way, it will contribute to the effective use of electronic hardware on a mobile robot.

The rest of the study consists of 3 sections. The first of these sections is Material and Method. In this section, the simulation environment, how the data is obtained, and the CNN models are discussed. Another section is findings. In this section, the training process of CNN models and statistical results according to the validation dataset are presented. The last section is the conclusion. This section presents the general conclusions of the study and future work.

2. Material and Method

2.1. Environment and Data Collection

One of the objectives of this study is to estimate the linear and angular velocity of a mobile robot based on visual data. For this purpose, one mobile robot and one camera that provides visual data are required. The linear velocity of a mobile robot is the linear displacement per unit time in the direction of its X-axis. The angular velocity of a mobile robot is the angular displacement per unit time in its Z axis. The linear (v) and angular (ω) velocities of a mobile robot were presented in Equations 1 and 2. In these equations, R, L, ω_R, and ω_L symbolise the wheel radius, the wheelbase of the robot, the angular velocity of the right wheel, and the angular velocity of the left wheel, respectively. In this study, the experimental environment presented in Figure 1 was formed. This experimental environment was designed using the Webots R2023 program, which is a robot simulator.

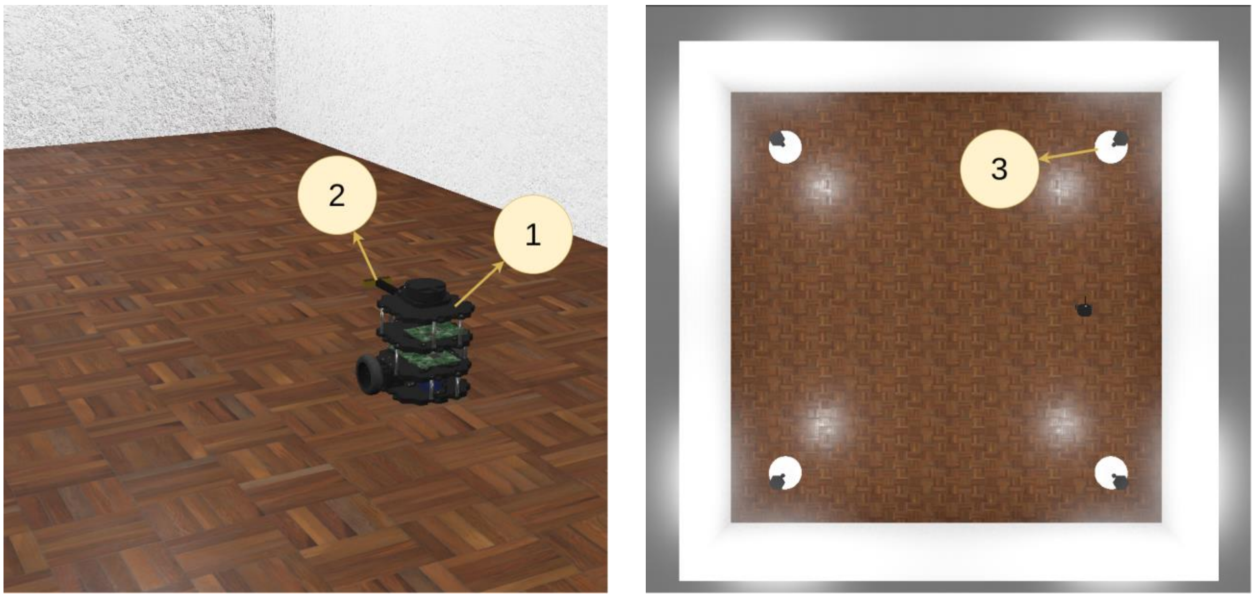
$$v = \frac{R}{2}(\omega_R + \omega_L) \tag{1}$$

$$\omega = \frac{R}{L}(\omega_R - \omega_L) \tag{2}$$

In Figure 1-a, the mobile robot and camera are symbolized by 1 and 2, respectively. In Figure 1-b, lamps are visualized with number 3. In this study, Turtlebot3 Burger was selected as the mobile robot. Turtlebot3 Burger mobile robot is 138mm x 178mm x 192mm in size and weighs 1 kg. In addition, its wheel diameter, linear velocity, and angular velocity are 66mm, 0.22m/s and 2.84 rad/s, respectively. The module in the simulator program was used as the camera. The camera’s resolution, FOV (field of view), and exposure were selected as 224px x 224px, 0.785, and 1, respectively. In this study, the parquet pattern commonly used in indoor spaces such as a daily home was preferred as the ground. The simulation environment dimensions were chosen as 5m x 5m. Other parameters of the simulation were selected as default values and presented in Table 1.

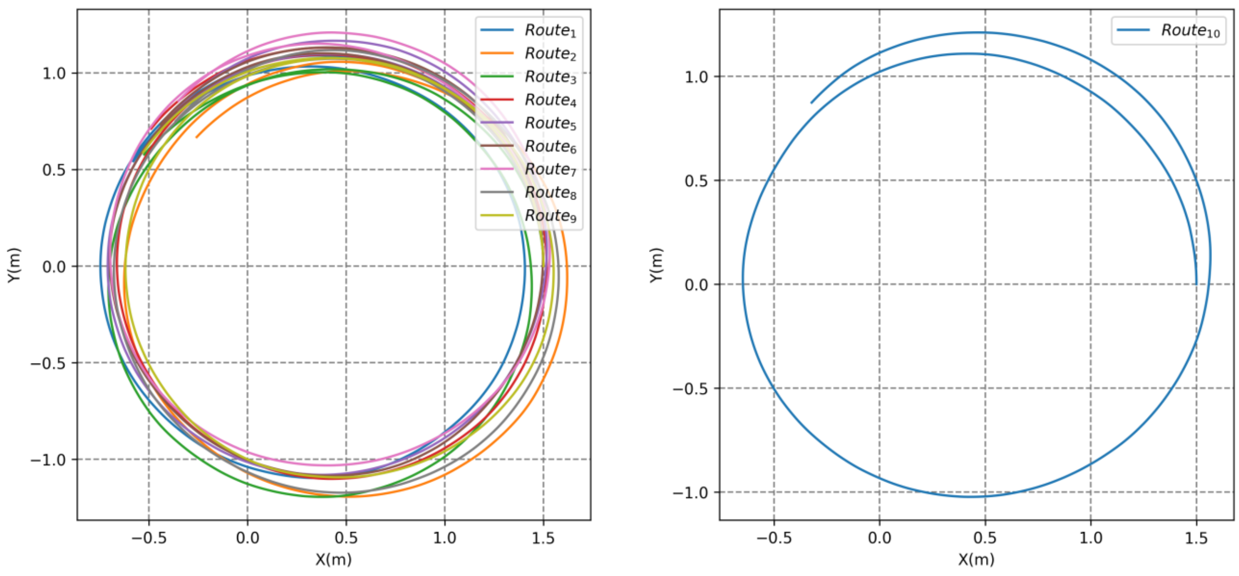
Table 1. The simulation parameters.

Parameter Description	Value
Gravity	9.81 m/s^2
Basic time step	64 ms
3D display frame per second	60
Physics disable linear velocity threshold	0.01
Physics disable angular velocity threshold	0.01
Drag force scale	30
Drag torque scale	5



(a) (b)

Figure 1. The experimental environment; a- Isometric view, b- Top view.



(a) (b)

Figure 2. Tracked routes by the robot; a- Training dataset routes, b- Validation dataset route.

After the simulation environment was designed, the 10 routes presented in Figure 2 were followed by the robot. These routes were created by randomly selecting linear and angular velocities between 0.1 and 0.2 (m/s-rad/s). Nine of these routes (Figure 2-a) were used for training, and one (Figure 2-b) was used for validating.

The robot followed each route for 1 minute and 4 seconds. During this tracking, it obtained images of the ground presented in Figure 3. There is one simulation time or 64 ms between the image on the left side of Figure 3 and the image on the right side. To express it discretely, the left image is at time t-1 while the right image is at time t.

Two datasets are created by recording the images presented in Figure 3 with linear and angular velocities when the routes in Figure 2 are followed by the robot. The training dataset, one of these two data sets, contains a total of 9000 images and their velocities. The validation dataset contains a total of 1000 images and their velocities.

Figure 3. Ground images at two different times.

After the datasets were created, the images went through a pre-processing stage. This pre-processing stage includes reducing the resolution of the images to 64px x 64px and converting their color from RGB to gray.

2.2. CNN Models

The purpose of this study is to reveal the difference between concatenation and subtraction layers in architectures that can predict the linear and angular velocities of a mobile robot visually. To this end, 4 different architectures presented in Figure 4 were trained with data obtained from simulation environments. Subtraction layers were used in Model 1 and Model 2. Concatenation layers were used in Model 3 and Model 4. Convolution and maximum pooling layers were added to the model after input layers were used to obtain the properties of the input layer in Model 1 and Model 3. In Model 2 and Model 4, input data was directly extracted or merged without a feature extraction process.

When the models presented in Figure 4 are analyzed, it is seen that the input layer sizes are 64 x 64 x 1. Also, the input of each model consists of two input layers. The inputs of these layers are images taken at different times. The convolution layer is the layer where intensive mathematical operations are performed, and the number of filters and kernel size are given in brackets, respectively. The stride size of this layer is 1 x 1, and the padding is selected as valid. In the maximum pooling layer, the pool size is given in brackets. In this layer, padding is selected as valid.

The subtraction layer applies the subtraction process to the incoming data. The concatenate layer combines the incoming data one after the other. A flatten layer is used to convert incoming matrices or tensors into vectors. A dropout layer is added to the models to solve the over-fitting problem. The hyper-parameters of the dropout layer are presented in brackets. The dense layer is the layer where the input data affects the entire output data. In these layers, the number of neurons is given in brackets. In addition, there are batch normalization and activation layers after the dense layer, except for the output layer. The batch normalization layer provides the statistical regulation necessary for the training process to proceed in a healthier way. The activation layer provides the learning process by passing the input data through a certain function. In this study, all activation functions except the output layer were chosen as rectified linear units (ReLU). In addition to these hyper-parameters, Model 1, Model 2, Model 3, and Model 4 structures consist of 7.3M, 7.3M, 7.7M and 7.3M learnable parameters, respectively.

3. Results and Discussion

Input data from datasets was pre-processed after the datasets were obtained from the simulation environment. After pre-processing, input and output data were generated to train. The input data were augmented by changing their brightness by 25% in the training phase. In this way, the system was able to produce more robust results for different brightness. In addition, the output data were normalized by subtracting the mean and dividing the result by its standard deviation. In this stage, the designed models were trained with the Adaptive Moment Estimation (Adam) algorithm for 50 steps. In the first 5 steps of this training, the learning rate is 5e-4, and the learning rate between steps 6-19 is 1e-4. In the remaining training process, the learning rate was applied as 1e-5. The training times of the models are 19.687, 16.875, 22.500 and 16.875 minutes, respectively. Mean Squared Error (MSE) was used as the loss function during this training. Loss-Epoch plots during the training period are presented in Figure 5.

Figure 5 shows that the training and validation processes of the networks did not have any high bias or high variance problems. During this training, a mini-batch size of 8 was chosen, and the input data in one mini-batch of the validation dataset is presented in Figure 6.

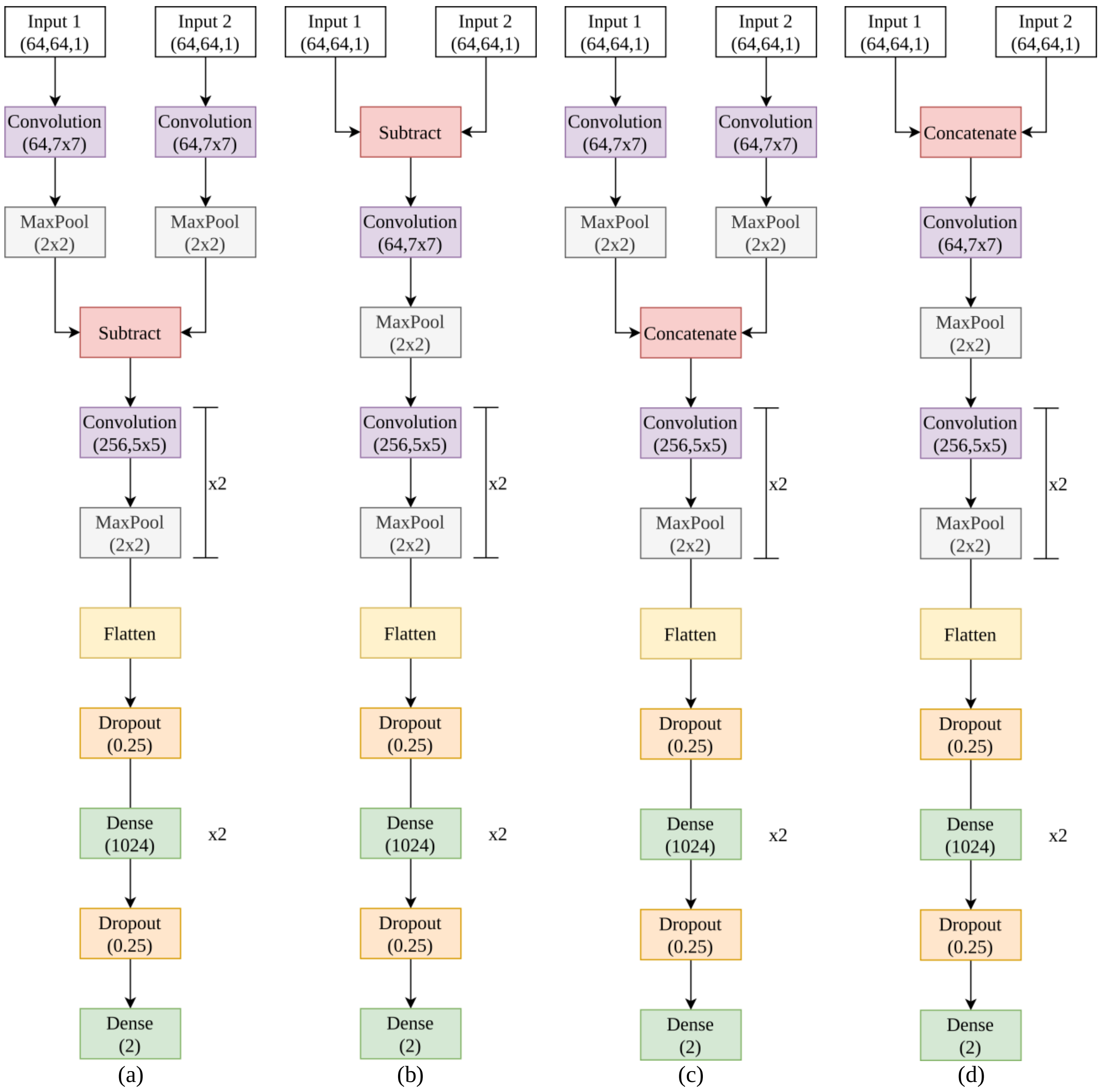


Figure 4. CNN architectures; a-Model 1, b-Model 2, c-Model 3, d-Model 4.

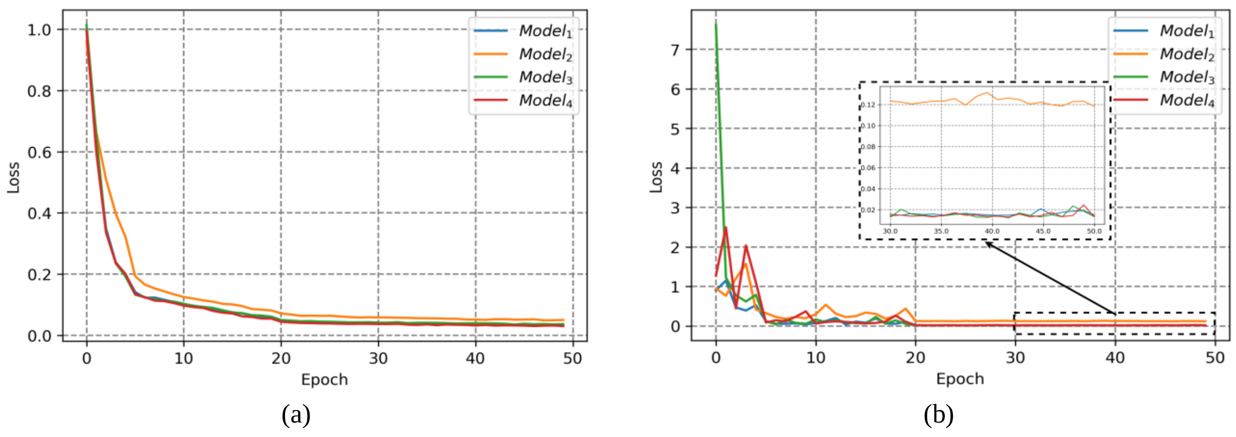


Figure 5. Loss function graphs; a-Train loss, b-Validation loss.

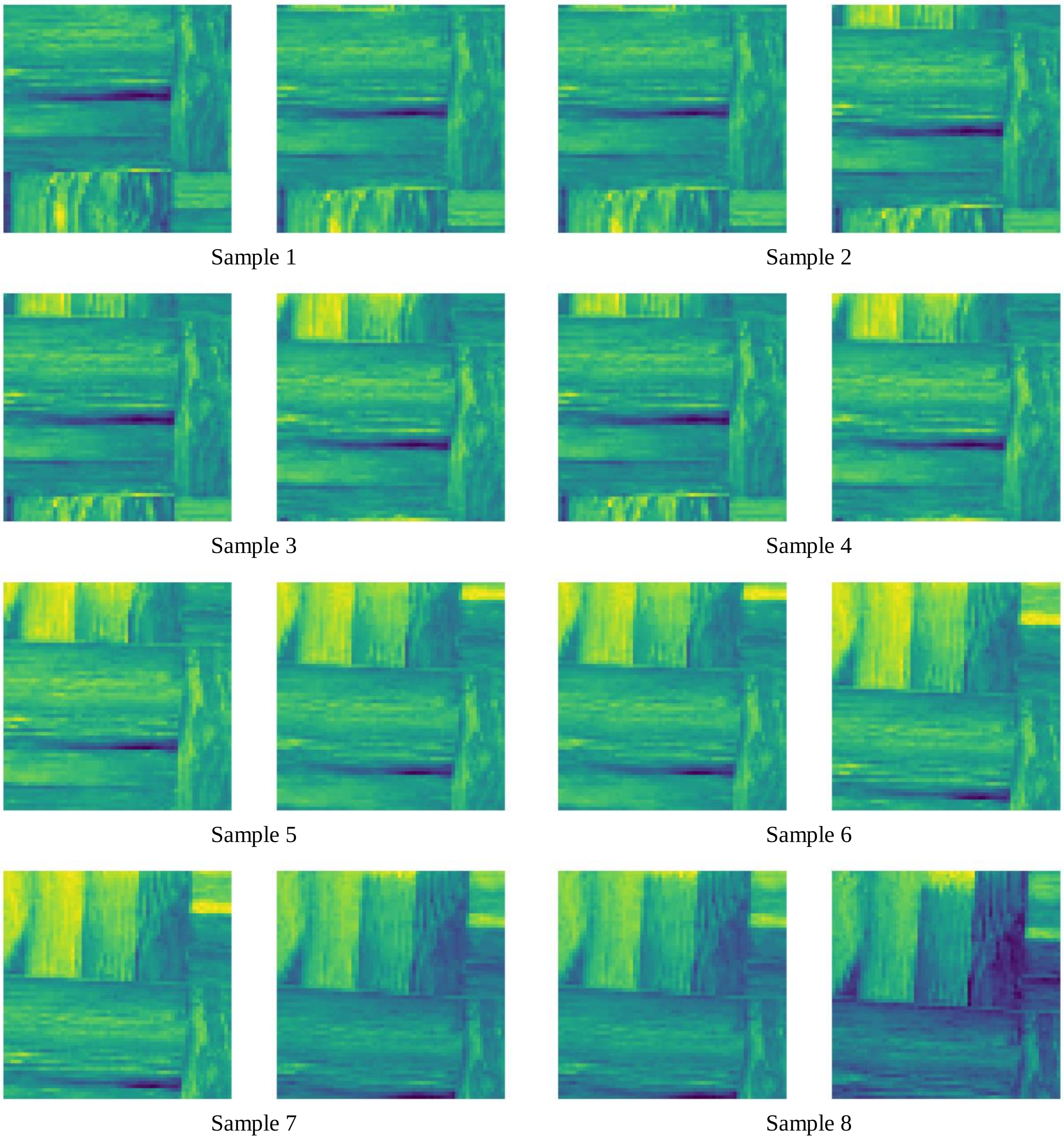
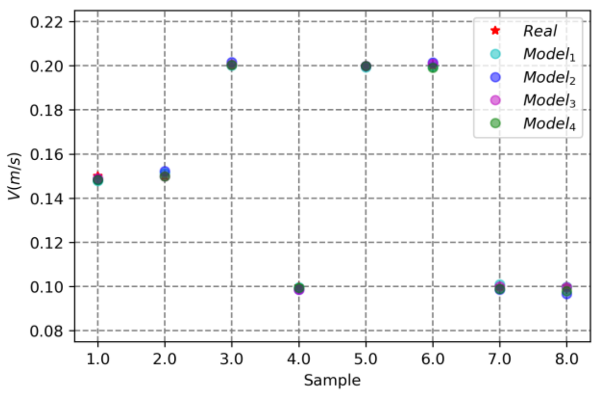


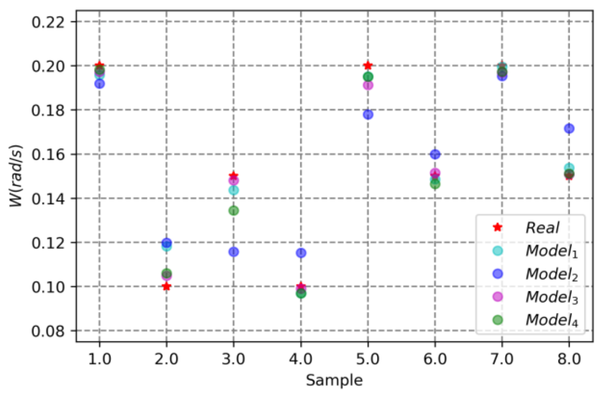
Figure 6. Mini-batch samples of the validation dataset.

The results generated against the inputs in Figure 6 are presented in Figure 7. The error amount of the data presented in Figure 7 is given in Figure 8. Figure 8 shows the measured error rates for a mini-batch. As this data is limited, it is insufficient for a comparison of model performance. For this reason, a descriptive analysis of the error between predicted and actual results for the 1000 data points in the validation dataset is presented in Table 2.

Table 2 shows the mean, standard deviation, standard error, and minimum and maximum values of the errors between the linear and angular velocity estimates and the actual value. The differences between linear (9.57e-3) and angular velocity (40.07e-3) errors are due to the units of both measurements. For example, if degrees/s were used instead of rad/s or km/h instead of m/s, the difference between linear and angular velocity would be different. It was investigated whether these calculated error values differed between the groups, and this result is presented in Tables 3 and 4.

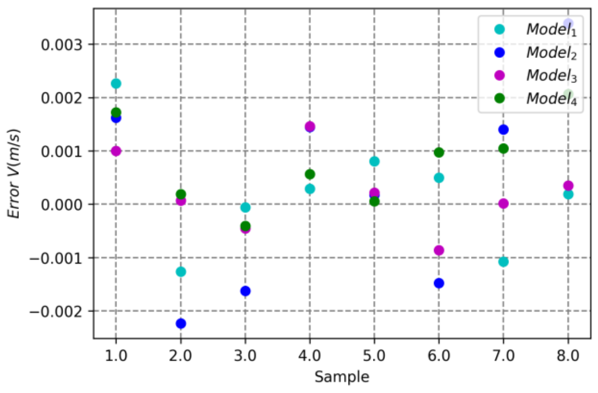


(a)

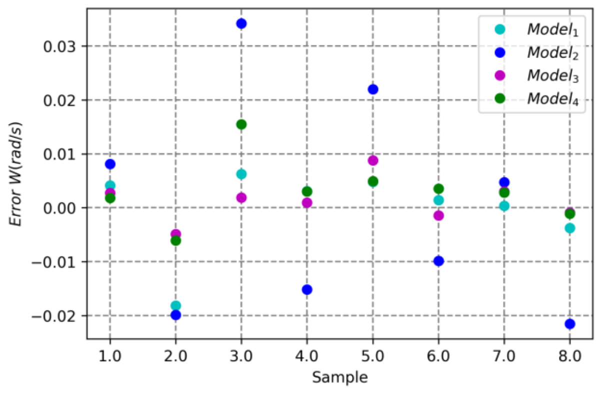


(b)

Figure 7. Velocity predictions for mini-batch samples of the validation dataset; a- Linear velocities, b- Angular velocities.



(a)



(b)

Figure 8. Errors of velocity predictions for mini-batch samples of the validation dataset; a- Linear velocities, b- Angular velocities.

Table 2. Descriptive analysis of errors.

	Model	**N**	**Mean**	**Std. Deviation**	**Std. Error**	**Minimum**	**Maximum**
|Error V| (m/s)	1	1000	1.03e-3	0.81e-3	2.56e-5	1.04e-6	9.57e-3
	2	1000	1.95e-3	1.85e-3	5.86e-5	2.51e-6	21.87e-3
	3	1000	1.01e-3	1.00e-3	3.16e-5	1.25e-6	9.43e-3
	4	1000	0.93e-3	0.86e-3	2.72e-5	0.05e-6	7.56e-3
|Error W| (rad/s)	1	1000	4.66e-3	5.22e-3	16.52e-5	20.94e-6	40.07e-3
	2	1000	14.43e-3	13.53e-3	42.79e-5	4.05e-6	88.06e-3
	3	1000	4.37e-3	4.84e-3	15.53e-5	4.62e-6	40.50e-3
	4	1000	6.92e-3	4.96e-3	15.71e-5	0.61e-6	36.08e-3

Table 3. Model comparison according to the linear velocity error.

Group 1	Group 2	p
Model 1	Model 2	**p<0.001**
	Model 3	0.99
	Model 4	0.29
Model 2	Model 1	**p<0.001**
	Model 3	**p<0.001**
	Model 4	**p<0.001**
Model 3	Model 1	0.99
	Model 2	**p<0.001**
	Model 4	0.42
Model 4	Model 1	0.29
	Model 2	**p<0.001**
	Model 3	0.42

Table 4. Model comparison according to the angular velocity error.

Group 1	Group 2	p
Model 1	Model 2	**p<0.001**
	Model 3	0.84
	Model 4	0.60
Model 2	Model 1	**p<0.001**
	Model 3	**p<0.001**
	Model 4	**p<0.001**
Model 3	Model 1	0.84
	Model 2	**p<0.001**
	Model 4	0.97
Model 4	Model 1	0.60
	Model 2	**p<0.001**
	Model 3	0.97

According to Tables 3 and 4, a significant difference was observed between Model 2 and the other groups for both linear and angular velocity (p<0.001). There is no statistically significant difference between the other groups (p>0.001).

4. Conclusion and Suggestions

In this study, it is aimed at estimating the linear and angular velocity of a mobile robot as visualized by a CNN and to compare whether the concatenation or subtraction layer of different images at different times will be more effective in this CNN structure. For these purposes, firstly, a simulation environment was designed, and training and validation datasets were obtained from this environment. Then, 4 different CNN architectures were trained on the training dataset, and these models were tested on the validation dataset. The statistical results of the tests are presented in Tables 2, 3, and 4. Based on these data, the worst result was produced by Model 2. The results of other models are better than this model. In addition, no statistical difference was observed between the other models except Model 2. Based on this finding, each of the three models can be used. However, if structures similar to Model 1 using the subtraction layer are used, a few convolutional layers should be used first for feature extraction. In addition, the extraction layer used in Model 1 will be more suitable for designs made with limited capacity processors where the input and output data sizes are the same as the concatenating layer found in Model 3 and Model 4. If there is no hardware constraint in the designed system, Model 4, which can be more easily designed compared to other models, can be used. The other aim of this study is to estimate linear and angular velocity. When Tables 2, 3, and 4 are examined, it is clear that the system designed for linear and angular velocity estimation is quite useful.

In future studies, data will be obtained from real mobile robots for complex routes, and similar studies will be repeated. In addition, it will be tried to make the proposed algorithm usable not only for mobile robots but also for all robots in general.

Conflict of Interest Statement

There is no conflict of interest between the authors.

Statement of Research and Publication Ethics

The study is complied with research and publication ethics.

References

[1] G. Boztaş and Ö. Aydoğmuş, “Implementation of Pure Pursuit Algorithm for Nonholonomic Mobile Robot using Robot Operating System,” *Balkan Journal of Electrical and Computer Engineering*, vol. 9, no. 4, pp. 337–341, Oct. 2021, doi: 10.17694/bajece.983350.

[2] M. Li et al., “Design and analysis of a whole-body controller for a velocity controlled robot mobile manipulator,” *Science China Information Sciences*, vol. 63, no. 7, Jul. 2020, doi: 10.1007/s11432-019-2741-6.

[3] A. Saenz, V. Santibañez, E. Bugarin, A. Dzul, H. Ríos, and J. Villalobos-Chin, “Velocity Control of an Omnidirectional Wheeled Mobile Robot Using Computed Voltage Control with Visual Feedback: Experimental Results,” *Int J Control Autom Syst*, vol. 19, no. 2, pp. 1089–1102, Feb. 2021, doi: 10.1007/s12555-019-1057-6.

[4] F. Huo, S. Zhu, H. Dong, and W. Ren, “A new approach to smooth path planning of Ackerman mobile robot based on improved ACO algorithm and B-spline curve,” *Rob Auton Syst*, vol. 175, May 2024, doi: 10.1016/j.robot.2024.104655.

[5] M. U. Shafiq et al., “Real-time navigation of mecanum wheel-based mobile robot in a dynamic environment,” *Heliyon*, vol. 10, no. 5, Mar. 2024, doi: 10.1016/j.heliyon.2024.e26829.

[6] B. Lakshmipriya, B. Pottakkat, and G. Ramkumar, “Deep learning techniques in liver tumour diagnosis using CT and MR imaging - A systematic review,” *Artificial Intelligence in Medicine*, vol. 141. Elsevier B.V., Jul. 01, 2023. doi: 10.1016/j.artmed.2023.102557.

[7] M. C. Bingol and O. Aydogmus, “Practical application of a safe human-robot interaction software,” *Industrial Robot*, vol. 47, no. 3, pp. 359–368, May 2020, doi: 10.1108/IR-09-2019-0180.

[8] J. G. Choi, D. C. Kim, M. Chung, S. Lim, and H. W. Park, “Multimodal 1D CNN for delamination prediction in CFRP drilling process with industrial robots,” *Comput Ind Eng*, vol. 190, Apr. 2024, doi: 10.1016/j.cie.2024.110074.

[9] J. Park, M. B. G. Jun, and H. Yun, “Development of robotic bin picking platform with cluttered objects using human guidance and convolutional neural network (CNN),” *J Manuf Syst*, vol. 63, pp. 539–549, Apr. 2022, doi: 10.1016/j.jmsy.2022.05.011.

[10] C. Cruz Ulloa, A. Krus, A. Barrientos, J. del Cerro, and C. Valero, “Robotic Fertilization in Strip Cropping using a CNN Vegetables Detection-Characterization Method,” *Comput Electron Agric*, vol. 193, Feb. 2022, doi: 10.1016/j.compag.2022.106684.

[11] S. Kim and S. Lee, “Robustness analysis of mobile robot velocity estimation using a regular polygonal array of optical M,” *in IFAC Proceedings Volumes (IFAC-PapersOnline)*, 2008. doi: 10.3182/20080706-5-KR-1001.0769.

[12] X. Yu et al., “Fully Proprioceptive Slip-Velocity-Aware State Estimation for Mobile Robots via Invariant Kalman Filtering and Disturbance Observer,” Sep. 2022, [Online]. Available: http://arxiv.org/abs/2209.15140

[13] M. A. Arteaga-Pérez and E. Nuño, “Velocity observer design for the consensus in delayed robot networks,” *J Franklin Inst*, vol. 355, no. 14, pp. 6810–6829, Sep. 2018, doi: 10.1016/j.jfranklin.2018.07.001.