Navigation and Guidance for Autonomous Quadcopter Drones Using Deep Learning on Indoor Corridors

Ahmad Wilda Yulianto[1], Dandhi Yudhit Yuniar[2], Yoyok Heru Prasetyo[3]
[1,2,3]Digital Telecommunication Network Study Program,
Department of Electrical Engineering, State Polytechnic of Malang, 65141, Indonesia
[1]ahmadwildan@polinema.ac.id, [2]dhandi.yudhit@gmail.com, [3]yoyok.heru@polinema.ac.id

***Abstract*—Autonomous drones require accurate navigation and localization algorithms to carry out their duties. Outdoors drones can utilize GPS for navigation and localization systems. However, GPS is often unreliable or not available at all indoors. Therefore, in this research, an autonomous indoor drone navigation model was created using a deep learning algorithm, to assist drone navigation automatically, especially in indoor corridor areas. In this research, only the Caddx Ratel 2 FPV camera mounted on the drone was used as an input for the deep learning model to navigate the drone forward without a collision with the wall in the corridor. This research produces two deep learning models, namely, a rotational model to overcome a drone's orientation deviations with a loss of 0.0010 and a mean squared error of 0.0009, and a translation model to overcome a drone's translation deviation with a loss of 0.0140 and a mean squared error of 0.011. The implementation of the two models on autonomous drones reaches an NCR value of 0.2. The conclusion from the results obtained in this research is that the difference in resolution and FOV value in the actual image captured by the FPV camera on the drone with the image used for training the deep learning model results in a discrepancy in the output value during the implementation of the deep learning model on autonomous drones and produces low NCR implementation values.**

***Keywords*—Quadcopter, ResNet50V2, CNN, deep learning, python, TensorFlow.**

I. INTRODUCTION

Unmanned Aerial Vehicles (UAV) or drones are flying vehicles that can be operated remotely by the pilot or can control themselves (autonomous). Over the last few decades, drone technology has been widely used by research institutes as research objects because of the flexibility and performance of drones that can be utilized for various human purposes, such as in agriculture, mining, construction, geology, archaeology, surveys, inspections, firefighting, photography, and other sectors. Drones can be classified based on several factors such as size, average takeoff weight, control configuration, and degree of autonomy. One of the popular drones is the quadcopter. A quadcopter is a type of drone that uses a four-rotor drive. Quadcopters are also included in multi-rotor drones which generally fall under the category of vertical-takeoff-and-landing (VTOL) vehicles with the ability to hover in place [1].

In carrying out their duties as utilization for humans, drones require accurate navigation and localization methods and algorithms [2], both drones with remote manual control and autonomous drones. In most outdoor environments, autonomous drone navigation has been successfully carried out by utilizing a global positioning system (GPS). The Global Positioning System (GPS) helps obtain the drone's position and orientation, aiding in navigation [3]. On the other hand, the disadvantage of GPS is that it is often unreliable or unavailable at all in most indoor environments, such as inside buildings, urban environments, and underwater. This makes the drone task difficult and complex to navigate.

Several solutions have been proposed for indoor autonomous navigation. One of them is Simultaneous Localization and Mapping (SLAM). Using a laser range finder, RGB-D sensor, or single camera, a 3-D map of an unknown indoor environment and its position on the map can be inferred for autonomous flight [4]. Another solution is based on stereo vision. By calculating the disparity between stereo images, depth can be estimated [5]. The SLAM method is not practical for drones because it requires heavy computing to build 3D models. In addition, 3D structures that are built often do not perform well in environments without traceable features (e.g., walls). The depth estimated by stereo vision shows poor performance in areas without texture and may suffer from specular reflection. The fact that most publicly available quadcopters only have one internal camera makes the solution impractical [6].

Therefore, in this research, an autonomous drone navigation model will be designed that allows the quadcopter to automatically navigate indoors, especially in the corridor area. This model does not require a proximity sensor but uses a single camera for navigation determination [7]. The approach is to train the regression model by utilizing one of the Deep learning algorithms, Convolutional Neural Network (ConvNet), from the corridor image data set [8]. The model's output in this study is a value that is a reference measure for the direction of the quadcopter when flying. System implementation begins when the ground station consistently receives visual input images from the camera mounted on the quadcopter and then returns flight commands (yawing, pitching, and rolling) from the results of the trained model. The quadcopter is expected to be able to fly from end to end of various types of corridors without colliding with the corridor walls.

II. METHODS

A. System Workflow

In the system flowchart in Fig.1. shows the system workflow design. First, after the drone and GCS are connected, the program

on the GCS will make the drone fly vertically as high as one meter. At the same time, the drone will also send images captured by the FPV camera to the GCS. The image received by the GCs will be processed by the deep learning model, and the output of the model will issue a command for the next action to be carried out by the drone. The command will then be sent to the drone. commands can be drone move forward, roll right, roll left, yaw right, yaw left, and land. The process of sending images by drones, processing images, and sending commands by GCS will continue to be carried out until a landing command is issued by GCS.

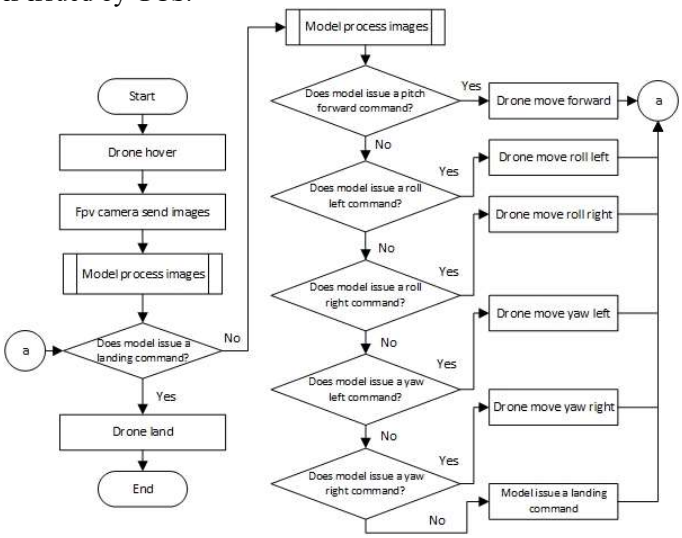


Figure 1. System Workflow

B. *System Block Diagram*

The block diagram of the system shows the overall system design, where there are two places for information processing, namely the vehicle and the ground control station. The vehicle is equipped with a camera module connected to a computer using telemetry, Pixhawk as flight control, electronic speed controller as a speed controller for the four brushless motors. At the Ground Control Station, there is a ROTG, the liaison between the camera and the computer, and different telemetry to send navigation commands from the deep learning model output to the quadcopter. The block diagram of the system is shown in Fig. 2.

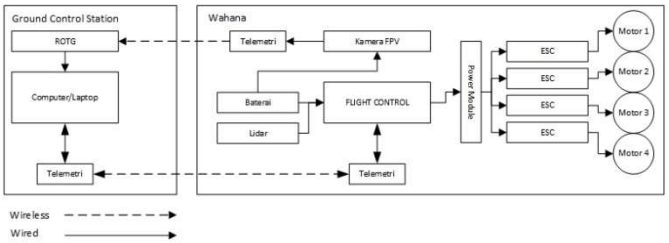


Figure 2. System Block Diagram

C. *Quadcopter Manufacturing Planning*

In this study, an assembled quadcopter is used to apply the deep learning model that has been created. The use of an assembled quadcopter can provide advantages in future development. The assembled quadcopter design can be seen in Fig. 3. and Fig. 4. Explanation of the parts on the assembled quadcopter as follows:

1. Propeller as a quadcopter propulsion in order to fly.
2. Frame as the body frame on the quadcopter to put all the components.
3. Electronic Speed Controller (ESC) as a regulator of motor rotation speed.
4. Flight Controller (FC) Pixhawk as the main controller of the quadcopter.
5. FPV camera to get a picture that is in front of the quadcopter
6. Motors.
7. Lidar sensor as a sensor to determine the height of the quadcopter.
8. Optical Flow sensor to help stabilize the quadcopter.

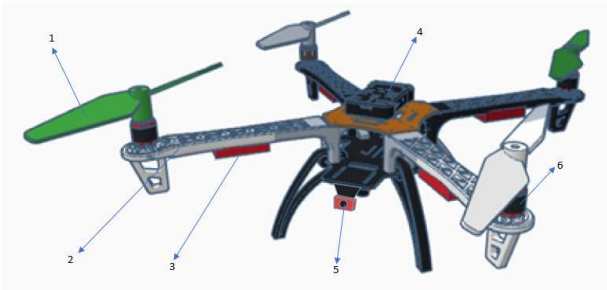


Figure 3. Top view quadcopter design

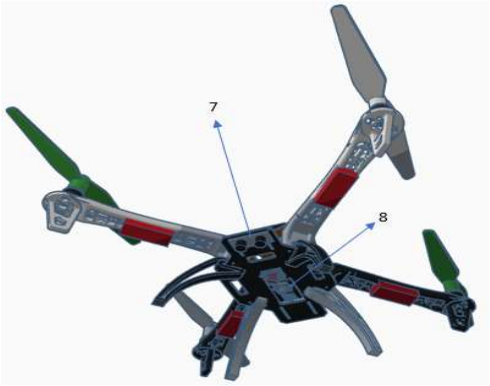


Figure 4. Bottom view quadcopter design

D. *Dataset*

The dataset used in this study is named NITRCorrV1 which has been created and used in this study [2], the data used for this study amounted to 21000 training images and 600 testing images for the translation model and used 21000 training images and 300 testing images for the rotation model. This dataset provides ground truth in terms of deviations with respect to the Central bisector line (CBL) corridor. The CBL line is used to measure the translation deviation and rotation of the quadcopter at the center of the corridor. Fig. 5. is an example of images on the dataset used.

Figure 5. Images on dataset

E. Proposed Model Deep Learning Architecture

1) Architecture

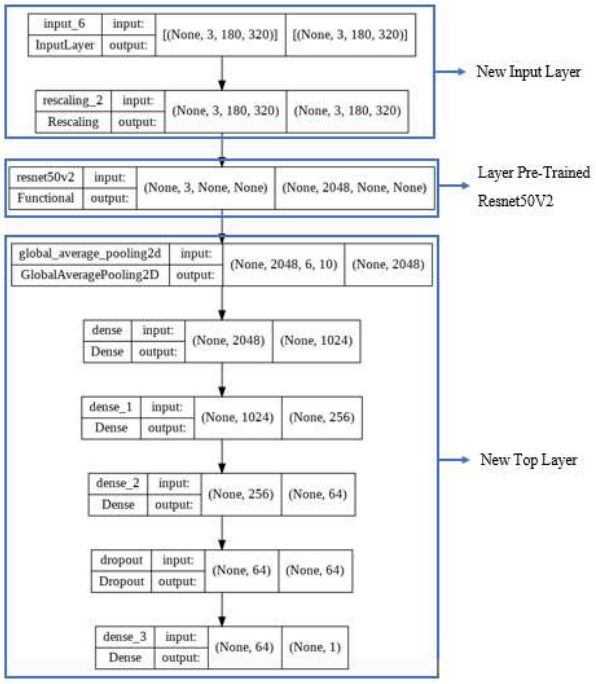


Figure 6. Architecture Model

The model architecture in this study uses the Pre-Trained Model Resnet50V2 which has been provided in the TensorFlow framework. The architecture of the model can be seen in Fig. 6. When using Transfer Learning Techniques, it takes changes to the top layer or layer classification in the model used and changes to the input layer according to research needs. The input layer is changed according to the existing dataset and in accordance with the research objectives. The shape of the input layer will be changed to (3, 180, 320) with the format (channel image, height, width) and rescaling is carried out to change the input scale to [-1,1]. The choice of ResNet50V2 was because in previous studies [2] ResNet50 produced MSE, MAE and MRE values as the second smallest evaluation metrics after DenseNet-161. ResNet50V2 is a development of ResNet50 with a smaller internal size of 103 compared to ResNet 50 of 107, has a better Top-1 Accuracy and Top-5 Accuracy [9].

2) Optimizer

Optimizer in this study using Adam. Previously, training experiments were carried out only on the rotational model and not the entire model using Adam, RMSprop and SGD with the same model architecture and model parameters as in Table I. The results of loss, loss validation, evaluation metrics and validation evaluation metrics on Adam are the smallest as shown in Fig. 7, Fig. 8., and Fig. 9.

3) Loss Function

The loss function in this study will use two loss functions, namely Mean Absolute Error (MAE) and Mean Squared Error (MSE). Both loss functions will be trained as a whole to choose the smallest loss before being implemented on the quadcopter.

4) Metrics

The Metrics Model in this study uses the Mean Squared Error (MSE) which is recommended from this study [10]. MSE was chosen because it was in accordance with the considerations, namely because this study compared several different predictions, using the same value and scale and there was a zero value in the prediction.

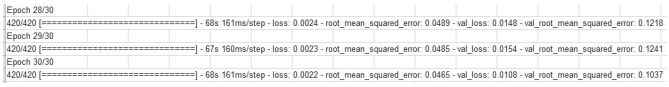


Figure 7. Model results using Adam Optimizer

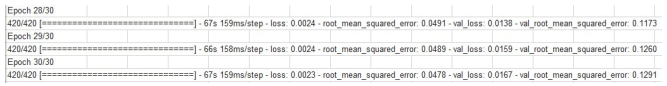


Figure 8. Model results using the RMSprop Optimizer

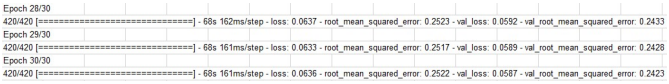


Figure 9. Model results using the SGD Optimizer

5) Parameters

TABLE I
PARAMETERS MODEL

Parameter Model	***Value***
Number of Epoch	50
Learning Rate	0.0001
Number of Epoch Fine-Tuning	10
Learning Rate Fine-Tuning	0.00001
Optimizer	Adam
Evaluation Metrics	MSE
Loss Function	MAE and MSE
Size of Batch	50
Re-scaling	1/127.5

The model parameter is a value that forms a model, all the parameters in Table I can be changed according to the research objectives. In this study using Adam as the optimizer, MSE as evaluation metrics, learning rates of 0.0001, batch size of 50, number of epochs of 50 and rescaling to make a range of values from -1 to 1. Only one loss function was chosen to be applied to the quadcopter of the two. The loss function used is MAE and MSE.

III. RESULTS AND DISCUSSION

A. Quadcopter Planning Results

The results of the quadcopter planning include assembling all the components used to make the quadcopter such as flight controller, motor, ESC, battery, FPV camera. The placement of each component is shown in Fig. 10 and Fig. 11.

Figure 10. Quadcopter front view

Explanation of the parts on the assembled quadcopter as follows:

1. Flight Controller (FC) Pixhawk 2.4.8.
2. Holybro 100 MW 915 MHz Telemetri Radio V3.
3. Propeller type 1045.
4. Motor Brushless 920Kv.
5. FPV Camera Caddx Ratel 2.
6. Battery Lippo 4S.
7. Frame 450 Multicopter.
8. Race Ranger VTX FPV Transmitter

9. PX4Flow.
10. Lidar sensor.
11. Electronic Speed Controller (ESC) 40A Spider.

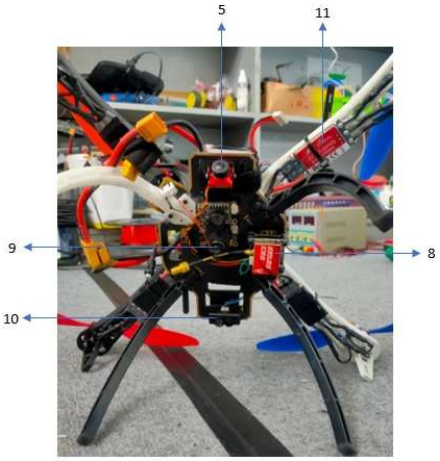


Figure 11. Quadcopter bottom view

B. *Model Results with MSE loss function*

The choice of loss function in the model lies in the model compiler section. In the compiler model section, there is the use of the Adam optimizer which functions as a model learning tool to minimize losses caused by the MSE loss function. The evaluation metric used is the same as the loss function, MSE.

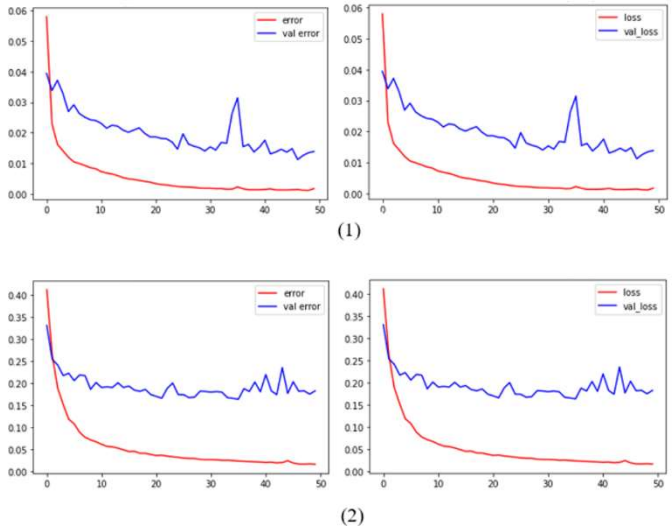


Figure 12. Model performance results with loss function MSE before fine-tuning. Rotational model (1), Translation model (2)

The top and bottom left images are the results of the metrics used, while the right images are the results of the loss function. The right and left images look the same because the loss function and evaluation metrics used are the same, using MSE. Fig. 12. (1) is the result of the rotation model with a loss value of 0.0017, a validation loss of 0.0139, a mean squared error of 0.0017, and a validation of the mean squared error of 0.0139.

Fig. 12. (2) is the result of the translation model with a loss value of 0.0168, a validation loss of 0.1827, a mean squared error of 0.0168, and a validation of a mean squared error of 0.1827. The difference in the validation values generated by the two models can identify overfitting or can be caused by a lack of epochs in training.

Fig. 13 is the result of the performance of the rotation and translation model using the loss function MSE after fine-tuning. The rotation model got a loss value of 0.0010, a validation loss of 0.0009, a mean squared error of 0.0010, and a validation of the mean squared error of 0.0009, there was a much better improvement after fine-tuning each validation value. These results indicate an overfitting problem before the fine-tuning can be completed.

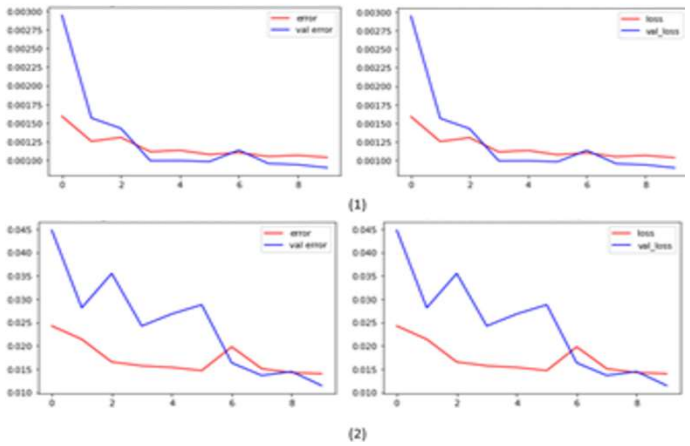
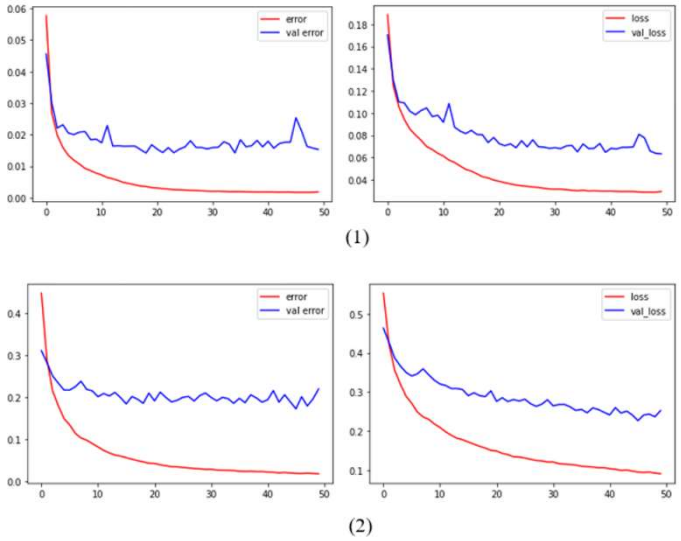


Figure 13. Model performance results with loss function MSE after fine-tuning. Rotational model (1), Translation model (2)

In the translation model from Fig. 13, the model gets a loss value of 0.0140, a validation loss of 0.0114, a mean squared error of 0.0140, and a validation of the mean squared error of 0.0114. These results make the model much better than before fine-tuning, just like the rotation model, the overfitting problem can be solved.

C. *Model Results with MAE loss function*

In this section, the model uses MAE loss function and evaluation metrics uses MSE and the same parameter settings as in the model with MSE loss function.



v

Figure 14. Model performance results with loss function MAE before fine-tuning. Rotational model (1), Translation model (2)

Results of the model in this section can be seen in Fig. 14. The image on the left is the result of the metrics used, while the image on the right is the result of the loss function. Fig. 14. (1) is the result of the rotation model with a loss value of 0.0293, a loss validation of 0.0633, a mean squared error of 0.0018 and a validation of the mean squared error of 0.0153. Fig. 14. (2) is the result of the translation model with a loss value of 0.0908, a loss validation of 0.2528, a mean squared error of 0.0177 and a

validation of the mean squared error of 0.2199. The difference in the validation values generated by the two models can identify overfitting or can be caused by a lack of epochs in training.

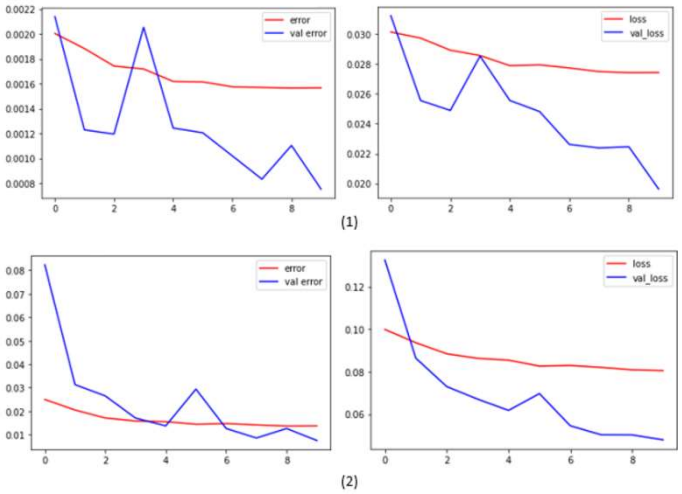


Figure 15. Model performance results with loss function MAE after fine-tuning. Rotational model (1), Translation model (2)

Fig. 15. is the result of the performance of the rotation and translation model using the loss function mae after fine-tuning. The rotation model gets a loss value of 0.0277, a validation loss of 0.0181, a mean absolute error of 0.0015 and a validation of the mean absolute error of 0.0007. These results are much better than the model before fine-tuning, the value of each validation is close to the loss value and the error indicates overfitting indications can be overcome.

In the translation model from Fig. 15., the loss value is 0.0803, the validation loss is 0.0486, the mean absolute error is 0.0148 and the mean absolute error validation is 0.0079. These results make the model much better than before fine-tuning, just like the rotation model, the overfitting problem can be solved.

TABLE II
OVERALL PERFORMANCE RESULTS OF MODEL

Model	MAE		MSE	
	Loss	Error	Loss	Error
	Val loss	Val error	Val loss	Val error
Rotation	0.0293	0.0018	0.0017	0.0017
	0.0633	0.0153	0.0139	0.0139
Rotation after Fine Tuning	0.0277	0.0015	0.0010	0.0010
	0.0181	0.0007	0.0009	0.0009
Translation	0.0908	0.0177	0.0168	0.0168
	0.2528	0.2199	0.1827	0.1827
Translation after Fine Tuning	0.0803	0.0148	0.0140	0.0140
	0.0486	0.0079	0.0114	0.0114

Table II shows the overall performance results of all models using both MAE and MSE loss functions and the model results before and after fine-tuning. Loss, error and validation values are obtained from the two smallest values in the model using the MSE loss function. Therefore, a model with an MSE loss function was chosen to be implemented on a quadcopter as a tool to generate navigation commands.

D. *Results of Model Implementation on Quadcopter*

The model and quadcopter that have been made will be tested in the corridor of the 3rd floor of the Electrical Building of the State Polytechnic of Malang.

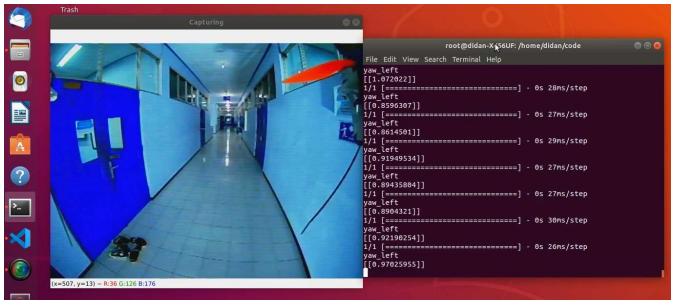


Figure 16. Display on GCS during model implementation on quadcopter

Quadcopter will be controlled autonomously starting from take-off to landing at the end of the corridor. The success parameter of this test is using NCR (No-Collision-Ratio). NCR is the number of quadcopter tests that managed to pass through the corridor without a collision divided by the number of tests performed.

TABLE III
TEST RESULTS OF DEEP LEARNING MODEL AND QUADCOPTER IN THE CORRIDOR

Test	Place	Take-off	Through the corridor	Flight distance
1st	Corridor 3rd floor	Succeed	Not successful	1 meter
2nd	Corridor 3rd floor	Succeed	Not successful	0 meter
3rd	Corridor 3rd floor	Succeed	Not successful	0 meter
4th	Corridor 3rd floor	Succeed	Succeed	5 meters
5th	Corridor 3rd floor	Succeed	Succeed	5 meters
6th	Corridor 3rd floor	Succeed	Not successful	0 meter
7th	Corridor 3rd floor	Succeed	Not successful	0 meter
8th	Corridor 3rd floor	Succeed	Not successful	0 meter
9th	Corridor 3rd floor	Succeed	Not successful	0 meter
10th	Corridor 3rd floor	Succeed	Not successful	0 meter

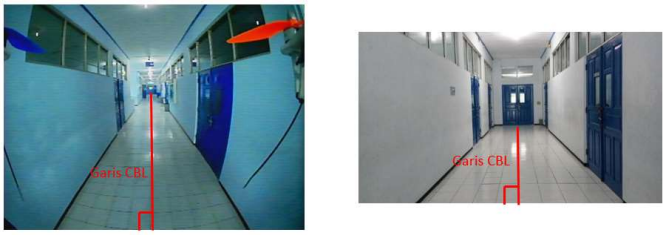
Based on Table III, the test was carried out ten times and the quadcopter successfully passed the corridor twice, then the NCR value was 0.2.

E. *Analysis of Model Implementation Results on Quadcopter*

The analysis is carried out on each model output with a different image input, image input from the fpv camera and image input from the cellphone camera. The images from the FPV camera have a field of view (FOV) value of 165 degrees, while the cellphone camera has no FOV value.

The image from the cellphone camera resembles the image used to train each model. The resolution on the fpv camera is 1200 TVL or the equivalent of 1.2 megapixels, while the images from the cellphone camera are 13 megapixels and the image resolution used to train the model is 0.05 megapixels. Each test image will have the same ground truth as the reference using the

CBL line. Images with the same ground truth mean the location and direction of facing the same quadcopter as in Fig. 17.



(a) (b)

Figure 17. (a) FPV camera results, (b) Mobile camera results

This test is carried out in order to find out which parts are not in line with expectations and cause failure in the whole system. Fig. 17. is an image from an FPV camera and an image from a cellphone camera with identical CBL lines. The CBL lines in both images form a 90 degrees angle and divide the image into two nearly equal planes. The calculation for the rotation model is counting the number of pixels from the CBL line to the left side of the image and normalization is carried out so that the values are in the range 0 to 1, both images are expected to get the output value of the rotation model close to 0.5 or from 0.4 to 0.6. The calculation for the translation model is to calculate the angle produced by the CBL line with the bottom side of the image and normalize it so that the value is in the range 0 to 3.14, both images are expected to get the output value of the translation model from 1.47 to 1.67.

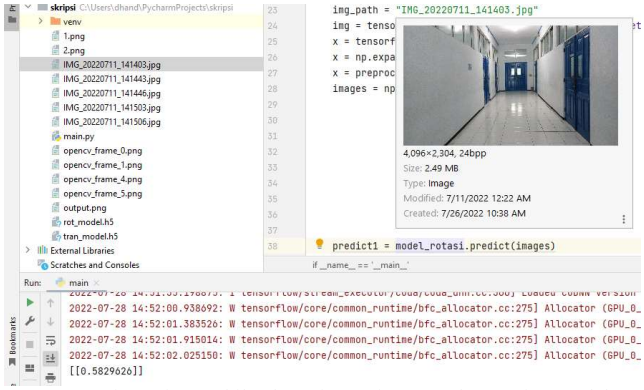


Figure 18. Testing using mobile phone image input on the rotation model

Fig. 18. shows the results of the model for rotation and produces an output of 0.5. The calculation for the rotation model and the suitability of the image with the label shows the correct results. The input image is shown in Fig. 17. which shows the quadcopter right in the middle of the corridor. The calculation for the rotation model is to calculate the number of pixels from the CBL line to the left side of the image with a value range between 0 to 1 after normalization. In this result, the CBL imaginary line will stretch to divide the two image planes equally and produce almost the same number of pixels. The pixel values will be normalized and produce a value of 0.5 which means the model produces the correct output.

Fig. 19. shows the results of the model for translation and produces an output of 1.54. The calculation for the translation model and the suitability of the image with the label shows the correct results. The input image is shown in Fig. 17. which shows the quadcopter right in the middle of the corridor. The calculation for the translation model is to calculate the angle produced by the CBL line with the left side of the image with a range of values from 0 to 3.14 after normalization. In this result, the CBL imaginary line will stretch to divide the two image planes equally and produce an angle value of 90 degrees. The angle value will be normalized and produce a value of 1.54 which means the model produces the correct output.

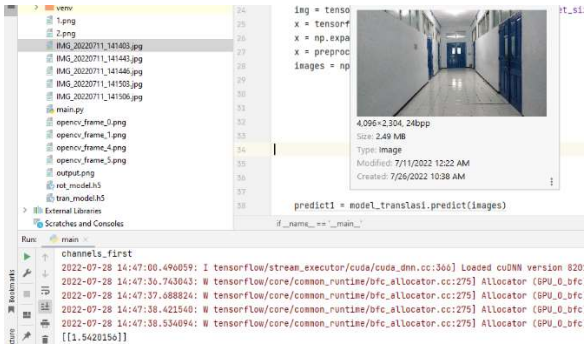


Figure 19. Testing using mobile phone image input on the translation model

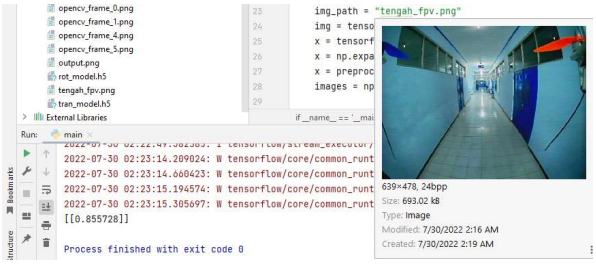


Figure 20. Testing using FPV camera image input on the rotation model

Fig. 20. shows the results of the model for rotation and produces an output of 0.85. The calculation for the rotation model and the fit of the image with the label shows wrong results. The input image is shown in Fig. 17. which shows the quadcopter right in the middle of the corridor. In this result, the CBL imaginary line will stretch to divide the two image planes equally and produce the same number of pixels. The pixel value will be normalized and produce a value of 0.85 which means the model produces an incorrect output.

Figure 21. Testing using FPV camera image input on the translation model

Fig. 21. shows the results of the model for translation and produces an output of 0.88. The calculation for the translation model and the suitability of the image with the label shows the wrong result. The input image is as shown in Fig. 17. which shows the quadcopter right in the middle of the corridor. In this result, the CBL imaginary line will spread the two image planes equally and produce an angle value of 90 degrees. The angle value will normalize and produce a value of 0.88, which means the model produces an incorrect output.

IV. CONCLUSION

Based on the background, problems, planning, implementation, testing, and discussion, it is concluded that the results of the implementation of the two models on the

quadcopter get a low NCR value of 0.2. These results are influenced by differences in the image processed by the model from the camera image on the quadcopter with the image used for model training. The difference between the two images is the FOV value and resolution of the image. The image produced by the Caddx Ratel 2 FPV camera has an FOV of 165 degrees and has a resolution of 1.2 megapixels, while the images used for deep learning model training have no FOV value and have a resolution of 0.05 megapixels.

REFERENCES

[1] T. Elmokadem, "Advanced Algorithms of Collision Free Navigation and Flocking for Autonomous UAVs," no. October, 2021, [Online]. Available: http://arxiv.org/abs/2111.00166.

[2] R. P. Padhy, S. Ahmad, S. Verma, S. Bakshi, and P. K. Sa, "Localization of unmanned aerial vehicles in corridor environments using deep learning," *Proc. - Int. Conf. Pattern Recognit.*, pp. 9423–9428, 2020, doi: 10.1109/ICPR48806.2021.9412096.

[3] K. Gryte, J. M. Hansen, T. A. Johansen, and T. I. Fossen, "Robust navigation of UAV using inertial sensors aided by UWB and RTK GPS," *AIAA Guid. Navig. Control Conf. 2017*, 2017, doi: 10.2514/6.2017-1035.

[4] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid, "RSLAM: A system for large-scale mapping in constant-time using stereo," *Int. J. Comput. Vis.*, vol. 94, no. 2, pp. 198–214, 2011, doi: 10.1007/s11263-010-0361-7.

[5] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy, "Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environments," *Unmanned Syst. Technol. XI*, vol. 7332, p. 733219, 2009, doi: 10.1117/12.819082.

[6] D. K. Kim and T. Chen, "Deep Neural Network for Real-Time Autonomous Indoor Navigation."

[7] B. Bender, M. E. Atasoy and F. Semiz, "Deep Learning-Based Human and Vehicle Detection in Drone Videos," 2021 6th International Conference on Computer Science and Engineering (UBMK), 2021, pp. 446-450, doi: 10.1109/UBMK52708.2021.9558888.

[8] D. K. Behera and A. Bazil Raj, "Drone Detection and Classification using Deep Learning," 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS), 2020, pp. 1012-1016, doi: 10.1109/ICICCS48265.2020.9121150.

[9] "Keras Applications." https://keras.io/api/applications/ (accessed Jul. 21, 2022).

[10] A. Jierula, S. Wang, T. M. Oh, and P. Wang, "Study on accuracy metrics for evaluating the predictions of damage locations in deep piles using artificial neural networks with acoustic emission data," *Appl. Sci.*, vol. 11, no. 5, pp. 1–21, 2021, doi: 10.3390/app11052314.