OMNIVISION CALIBRATION ON MOBILE ROBOT USING MACHINE LEARNING

1st Azzam Wildan Maulana
dept. of Computer Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia 60111
5024201010@student.its.ac.id

2nd Muhtadin, S.T., M.T.
dept. of Computer Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia 60111
muhtadin@te.its.ac.id

3rd Ahmad Zaini, S.T., M.T.
dept. of Computer Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia 60111
zaini@te.its.ac.id

***Abstract*—Mobile Robot is a robot that can easily move. The movement of the robot can cause the camera angle to shift. This shift can be caused by the wrong manufacturing of the camera installation or a collision on the robot. The camera angle shift will cause the camera's interpretation of the outside world to be wrong. The use of Machine Learning methods in Omnivision camera calibration can correct the wrong camera interpretation without being influenced by the Omnivision camera manufacturing and installation process. The Machine Learning used is a Multi Layer Perceptron Neural Network with an activation function in the form of a sigmoid. The results of Machine Learning will be converted into a Lookup Table which will be used in the vision computation process of the robot. This method is better than the old polynomial regression method. This can be seen from the accuracy and precision produced by the Machine Learning method which is better than the polynomial regression method. The accuracy error of the Machine Learning method is 10.84 cm while the polynomial regression method is 20.77 cm. The precision error of the Machine Learning method is 1.20 cm and 4.10 cm while the polynomial regression method is 10.01 cm and 11.32 cm. By using the Machine Learning method in Omnivision camera calibration, the robot can move better.**

***Index Terms*—Omnivision, Calibration, IRIS**

I. Introduction

There are three main parts of Mobile Robot, namely Sensor, Control, and Actuator. All the main parts are connected to each other. Sensor is used to detect the environment around the robot. Control is used to process the data from the sensor and decide the next action. Actuator is used to execute the action that has been decided by the control.

There is a sensor for Mobile Robot that can sense 360 degree around the robot. The sensor is called Omnivision Camera. The use of Omnivision Camera give more benefits because it can grab the information around robot with only one capture. The basic concept of Omnivision Camera is to use a mirror to reflect the environment around the robot. The mirror is placed in front of the camera. The camera is placed in the middle of the mirror and projected 90 degree to the ground of Robot. Not only can sense the environment around the robot, the camera can also sense within 10m distance.

The hardware installation process of Omnivision Camera is hard. There are a common mistakes that everyone can accpet it such as misplaced camera, misplaced mirror, bad mirror, misplaced the whole system, and many more. Errors like that can cause a bad representation of the environment around the robot. The bad representation of the environment can cause the robot to make a wrong decision.

II. Related Works

A. Previous Research

Polynomial Regression Method is an approximation method for data that has been provided before. The output of the Polynomial Regression method is a mathematical formula based on the data that has been provided. Polynomial regression can work efficiently even with a non-linear model Grondin et al. [1]. There is a basic formula of Polynomial Regression:

$$y = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n \tag{1}$$

where y is the dependent variable, x is the independent variable, a_0 is the intercept, a_1 is the coefficient of x, a_2 is the coefficient of x^2, and so on until a_n is the coefficient of x^n.

The Previous Research of Omnivision Camera calibration is using Polynomial Regression to calibrate Omnivision Camera. The y is the distance from the camera to the object in real world, x is distance of object from center of camera. The result of the Polynomial Regression is a mathematical formula that can be used to calculate the distance from the camera to the object.

The next step of using a Polinomial Regression is to change polar coordinates into cartesian coordinates. So the final output of the camera callibration is the cartesian coordinates of the object in the real world. And the input is cartesian coordinates of the object in the image. Here is the algorithm to calculate the cartesian coordinates of the object in the real world:

Algorithm 1 Data Calculation using Polynomial Regression

1: **procedure** CalculateCoordinatesAndAngles
2: Compute $dx \leftarrow x - x\_center\_cam$
3: Compute $dy \leftarrow y\_center\_cam - y$
4: Compute $\theta \leftarrow \arctan(\frac{dy}{dx})$
5: Compute $x\_world \leftarrow jarak\_sebenarnya \times \cos(\theta)$
6: Compute $y\_world \leftarrow jarak\_sebenarnya \times \sin(\theta)$
7: **end procedure**

With x_world representing the x coordinate in the field and y_world representing the y coordinate in the field. Then the position of the object in the field can be obtained.

B. Basic Theory/Concept

1) Callibration: Callibration is a process to rearrange the parameters in a system to match the standards that have been determined before. Callibration is widely used in measurement systems, robotic systems, automation systems, and other systems. Callibration on the camera is a process to rearrange the camera so that it can see objects at a distance that matches the actual distance. Camera callibration is usually done by software by entering camera data into the computation process. The process can change camera data according to the actual data.

Fig. 1. Omnivision Camera.

2) Omnivision Camera: Omnivision Camera is a camera that can see 360 degrees around the camera Chen and Lee [2]. The range of view of the Omnivision Camera is unlimited depending on the resolution of the camera itself and the construction of the mirror. Basically, the Omnivision Camera is a regular camera that is shot into a convex mirror so that the camera's view can be in all directions.

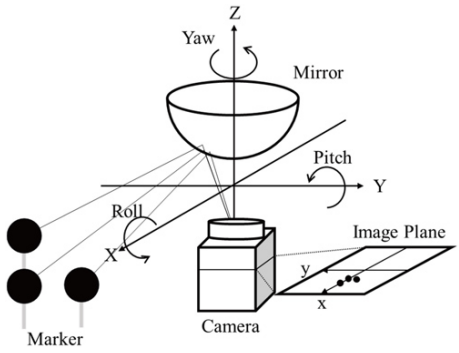

Fig. 2. Basic Concept of Omnivision Camera.

From the figure above, the Omnivision camera is a camera that faces 90 degrees to the flat plane and is shot into a convex mirror. The convex mirror will reflect the light that enters the camera and vice versa. This work makes the Omnivision camera can see in all directions.

3) Neural Network: Neural Network is a part of Machine Learning. Neural Network is created to overcome nonlinearity problems in a model Tang et al. [3]. Basically, Neural Network is just a collection of Neurons connected by a Weight and Bias. In addition to Weight and Bias, there is also a forward propagation using an activation function or commonly called a transfer function. In addition to forward propagation, there is also backward propagation using a loss function.

4) Activation Function*:* Activation function is a function used to transfer data at each layer in the Neural Network. In the Neural Network, the Activation function acts as forward propagation, which is the journey from the input layer to the output layer. Some activation functions have saturation values usually valued at 1, for example, Sigmoid which is valued in the range 0 to 1. There is also another activation function, namely tanh which is valued in the range -1 to 1 Kaloev and Krastev [4].

5) Loss Function*:* Loss function is part of the Neural Network that aims to provide feedback to the model about the good or bad phase of training. Loss function in the Neural Network works on the Backward Propagation path, which is from the output layer to the input layer. There are several types of loss functions, one of which is MSE (Mean Squared Error). MSE loss function is better used on data with low fluctuation values Dohi et al. [5].

Not only the loss function, there is another theory called Optimizer. Optimizer is an algorithm that can be used to determine the learning rate of the training system. The learning rate in the Neural Network is used to set how fast the model will converge to its data.

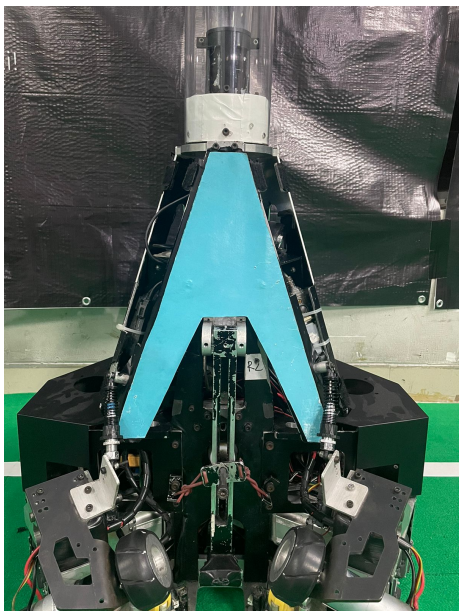

Fig. 3. IRIS Robot.

6) Mobile Robot*:* Mobile Robot is a robot designed to move or move easily. The performance of the Mobile Robot is largely focused on tracking systems, localization, and al-

gorithms. All of these are based on good sensing capabilities Lee et al. [6].

Sensor that commonly used in Mobile Robot is camera. The camera can be a regular camera or an Omnivision Camera. The use of Omnivision Camera can make the robot see in all directions, but the pre-processing of the data is more difficult than a regular camera.

7) OpenCV*:* OpenCV is an open-source library developed by Intel with the C/C++ programming language. OpenCV provides many algorithms related to Computer Vision Yildirim et al. [7]. OpenCV is widely used for object detection based on color, shape, size, and other needs of the program.

8) Robot Operating System: ROS or Robot Operating System is a platform that stands on Linux and is useful for synchronizing parts of the robot Quang et al. [8]. ROS is widely used as the core of processing data from a robot starting from processing sensor data to becoming actuator data. ROS provides modular programming concepts with a publish/subscribe method for its IPC (Inter Process Communication). In addition to facilitating data transfer between processes, ROS also provides a timer with a default scheduler that follows the Default Linux Scheduler, namely the Priority-based scheduler. This allows users to set the priority of each part of their robot.

III. Calibration Process Design

Omnivision Camera Calibration using Machine Learning methods consists of two things, namely the calibration data acquisition process and the calibration data learning process. The calibration data acquisition process is worked by humans by matching data from the camera with real world data. Meanwhile, the learning process is carried out by a computer using the Multi-Layer Perceptron method which will be made into a Lookup Table.

A. Data Acquisition

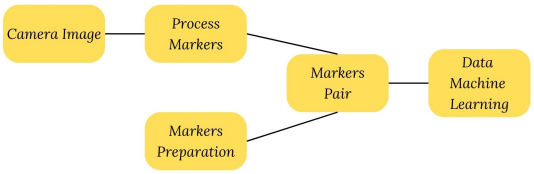

Fig. 4. Data Acquisition Block Diagram.

The first thing to do is to get an image from the Omnivision camera. This can be obtained through a program made by the author. The program will display the camera image on the website. Here is the image of the website that displays the Omnivision camera image:

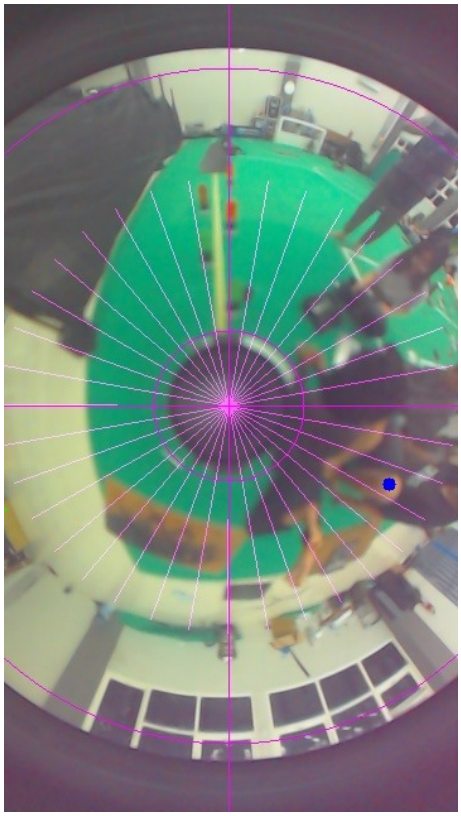

Fig. 5. Camera Image View.

Next step is to prepare markers in the field. The markers are arranged according to the desired data. The placement of the markers varies from 60 cm to 460 cm from the robot. Here is the image of the marker placement in the field:

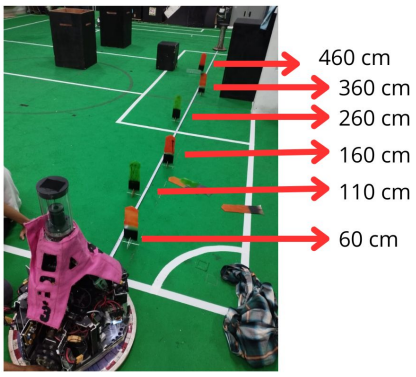

Fig. 6. Markers on Field.

Next step is to process the markers on the camera coordinates.

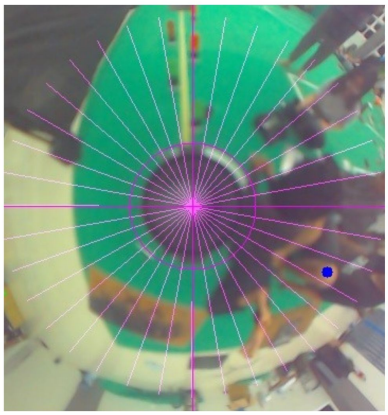

Fig. 7. Camera Image View Zoomed.

From that website, the markers on the image can be clicked to get the coordinates of the click on the camera. Then the coordinates will be processed to become polar coordinates in the form of distance and direction with the origin in the center of the camera. Here is the process of changing the marker coordinates on the camera into polar coordinates:

Algorithm 2 Cartesian Coordinate into Polar Coordinate

1: **procedure** CAMERAMERKER2POLAR
2: $dx \leftarrow x\_clicked\_point - x\_center\_cam$
3: $dy \leftarrow y\_center\_cam - y_clicked\_point$
4: $\theta \leftarrow \arctan(\frac{dy}{dx})$
5: $jarak \leftarrow \sqrt{dx^2 + dy^2}$
6: **end procedure**

Next step is to pair the distance and direction with the distance in the real world. The pairing is based on the marker number that has been arranged in the field. Here is an example of the data that has been paired:

TABLE I
DATA TRAINING.

Theta on Camera	Distance on Camera	Distance pada dunia
0 deg	81 px	60 cm
0 deg	105 px	110 cm
0 deg	143.355 px	160 cm
10 deg	81 px	60 cm
10 deg	104.738 px	110 cm
10 deg	149.684 px	160 cm
10 deg	171.878 px	260 cm
20 deg	81.2 px	60 cm
20 deg	127.343 px	110 cm
20 deg	149.378 px	160 cm
20 deg	162.029 px	210 cm
20 deg	173.398 px	260 cm
...	...	...
350 deg	81.394 px	60 cm
350 deg	105.721 px	110 cm
350 deg	147.681 px	160 cm
350 deg	159.797 px	210 cm
350 deg	169.577 px	260 cm

The data will be used as training data for the learning process.

B. Calibration Process

Fig. 8. Calibration Process Block Diagram.

After all the training data is obtained, the data will be normalized first. The purpose of normalization is to make the data used in the learning process have the same range. In addition, to be able to utilize the activation function used. Here is the normalization formula used:

$$normalized\_direction = \frac{direction\_raw\_value}{180} - 1 \quad (2)$$

$$normalized\_camera\_distance = \frac{camera\_distance\_raw\_value}{160} - 1 \quad (3)$$

$$normalized\_world\_distance = \frac{world\_distance\_raw\_value}{600} - 1 \quad (4)$$

Using the normalization formula mentioned above, the new data can be obtained as follows:

TABLE II
NORMALIZED DATA TRAINING.

A	B	C	D	E	F
0 deg	81 px	60 cm	-1	-0.49	-0.90
0 deg	105 px	110 cm	-1	-0.34	-0.82
0 deg	143.36 px	160 cm	-1	-0.10	-0.73
10 deg	81 px	60 cm	-0.94	-0.49	-0.90
10 deg	104.74 px	110 cm	-0.94	-0.35	-0.82
10 deg	149.68 px	160 cm	-0.94	-0.06	-0.73
10 deg	171.88 px	260 cm	-0.94	0.07	-0.57
20 deg	81.2 px	60 cm	-0.89	-0.49	-0.90
20 deg	127.34 px	110 cm	-0.89	-0.20	-0.82
20 deg	149.38 px	160 cm	-0.89	-0.07	-0.73
20 deg	162.03 px	210 cm	-0.89	0.01	-0.65
20 deg	173.40 px	260 cm	-0.89	0.08	-0.57
...	...	...	...	...	...
350 deg	81.39 px	60 cm	0.94	-0.49	-0.90
350 deg	105.72 px	110 cm	0.94	-0.34	-0.82
350 deg	147.68 px	160 cm	0.94	-0.08	-0.73
350 deg	159.80 px	210 cm	0.94	-0.00	-0.65
350 deg	169.58 px	260 cm	0.94	0.06	-0.57

With A stand for Direction on Camera, B stand for Distance on Camera, C stand for Distance in the World, D stand for Normalized Direction on Camera, E stand for Normalized Distance on Camera, and F stand for Normalized Distance in the World.

After the normalized data is obtained, the data will be processed using the Multi-Layer Perceptron method. The architecture of the Multi-Layer Perceptron is as follows:

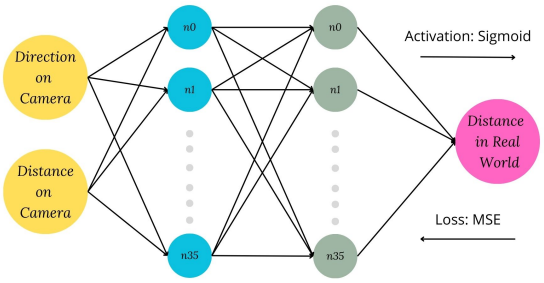

Fig. 9. NN MLP Architecture.

The architecture of the Multi-Layer Perceptron consists of 2 hidden layers, each with 36 neurons. Meanwhile, for the input layer consists of 2 neurons in the form of $jarak\_objek\_pada\_kamera$ and $arah\_objek\_pada\_kamera$. The output layer consists of 1 neuron in the form of $jarak\_objek\_pada\_dunia$. The activation function used is Sigmoid. The use of Sigmoid will make the training results more varied and more non-linear.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{5}$$

The training process is carried out using the architecture according to the image above. The training process is carried out for 300000 epochs. After the training process is complete, a model will be obtained whose input is the distance and direction on the camera with the output in the form of distance in the real world.

To get the distance and direction data according to the training data in table 1, the data must be denormalized first. Here is the denormalization formula used:

$$real\_world\_r = normalized\_real\_world\_r \times 600 + 600 \tag{6}$$

Using the formula above, the actual distance data in the real world can be obtained. The data will be made into a Lookup Table which will then be used in the next process. The making of the Lookup Table is done with two formulas, namely the formula to determine the size of the Lookup Table and the second is the formula to determine the value of each element of the Lookup Table. Here is the formula to determine the size of the Lookup Table:

$$size = \theta\_max \times r\_max \tag{7}$$

Here is the formula to determine the value of each element of the Lookup Table:

$$index = \theta \times r\_max + r \tag{8}$$

By using the index, the value of each element of the Lookup Table can be obtained. Then the Lookup Table will be saved into a binary file in the robot operating system.

IV. ROBOT IMPLEMENTATION

The implementation design on the robot to test the calibration results is by detecting the ball in the field. Here are the steps taken on the IRIS Robot:

Fig. 10. Robot Implementation Block Diagram.

The camera image on the IRIS Robot is obtained through the Omnivision camera. The image is then processed by a C++ program that has been made by the author.

The next step after getting the image on the camera is to do color segmentation on the image. Color segmentation is done by comparing the YUV value in the image with the predetermined YUV value. The predetermined YUV value is the YUV value of the ball color. So the output of color segmentation is a binary image. Here is an example of a binary image generated from color segmentation:

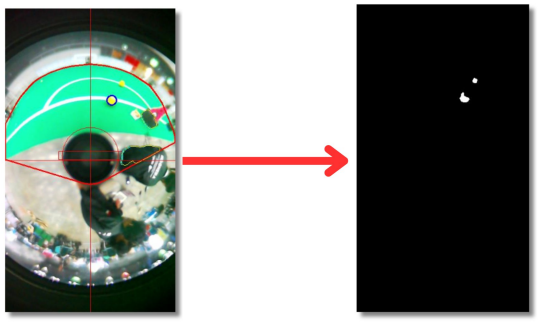

Fig. 11. Binary and RGB Image.

After getting the binary image, the next step is to find the object coordinates in the image. To do this, the following procedure is used:

Algorithm 3 Find Ball on Camera

procedure FINDBALL
  for angle from 0 to 360 with step size 2.5 **do**
    Initialize dist to 0
    for index from 0 to 320 **do**
      $x \leftarrow \text{dist} \times \cos(\text{angle}) + \text{center\_cam\_x}$
      $y \leftarrow \text{center\_cam\_y} - \text{dist} \times \sin(\text{angle})$
      if ball_frame_binary[y][x] == 255 **then**
        $ball\_x\_on\_kamera \leftarrow x$
        $ball\_y\_on\_kamera \leftarrow y$
        Break
      end if
      $dist \leftarrow \text{dist} + 1$
    end for
  end for
end procedure

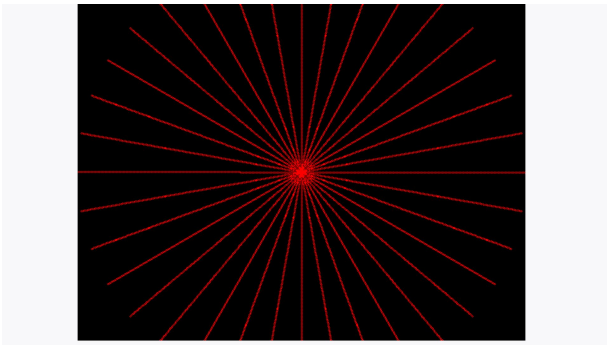

Fig. 12. Linear Scanning Visualization.

The meaning of the procedure is to do linear scanning from the center of the camera outwards with an angle of 2.5 degrees. At each angle, a linear scanning will be done from the center of the camera outwards with a distance of 1 pixel. If the pixel at that coordinate has a value of 255, then the coordinate is the ball coordinate on the camera.

Next, after getting the Cartesian ball coordinates on the camera, the next step is to change the Cartesian coordinates into polar coordinates. Here is the procedure for changing Cartesian coordinates into polar coordinates:

Algorithm 4 Ball Cartesian Coordinate to Polar Coordinate

procedure BALLCARTESIAN2POLAR
  $dx \leftarrow ball\_x\_on\_cam - x\_center\_cam$
  $dy \leftarrow y\_center\_cam - ball\_y\_on\_cam$
  $ball\_\theta\_cam \leftarrow \arctan(\frac{dy}{dx})$
  $ball\_distance\_cam \leftarrow \sqrt{dx^2 + dy^2}$
end procedure

Next, after getting the distance and direction data of the ball on the camera, the distance of the ball in the real world can be obtained by reading the Lookup Table. Here is the procedure for reading the Lookup Table:

Algorithm 5 Reading Lookup Table

procedure READLOOKUPTABLE
  $index \leftarrow ball\_\theta\_cam \times r\_max + ball\_distance\_cam$
  $ball\_distance\_world\_local \leftarrow lookup\_table[index]$
  $ball\_\theta\_world\_local \leftarrow ball\_\theta\_cam$
end procedure

After getting the distance and direction of the ball in the real world, the position of the ball in the real world can be obtained. However, the ball position is still a local ball position, which means that the ball position is still relative to the robot. To get the global ball position, the ball position must be added to the robot's position and orientation. Here is the procedure to get the global ball position:

Algorithm 6 Local to Global World Model for Ball

procedure BALLLOCAL2GLOBAL
  $ball\_\theta\_global \leftarrow angle\_px + robot\_pose\_\theta - 90$
  $ball\_x\_global \leftarrow robot\_pose\_x + ball\_distance\_world\_local \times \cos(\text{DEG2RAD} \times ball\_\theta\_global)$
  $ball\_y\_global \leftarrow robot\_pose\_y + ball\_distance\_world\_local \times \sin(\text{DEG2RAD} \times ball\_\theta\_global)$
end procedure

By using the procedure above, the position of the ball in the real world can be obtained in the form of $ball\_x\_global$ and $ball\_y\_global$.

V. RESULT AND DISCUSSION

A. Accuracy Testing Scenario

The accuracy test is done by placing the ball in the middle of the field with the actual coordinates of 400, 600. Then place the robot at a distance of 120 cm, 200 cm, and 285 cm from the ball and rotate the robot in place. Every 30 degrees, the robot will record the ball position on the field resulting from the

calibration implementation. Then the results will be calculated the *Euclidean Distance* between the actual ball position and the ball position resulting from the calibration. The following is the testing scenario that was carried out:

Fig. 13. Testing Scenario.

B. Accuracy Testing Evaluation

While the test is running, the robot will rotate in its position to detect the ball on the field. The robot will rotate 30 degrees each time. The robot will rotate 12 times so that the robot can see the ball from all directions. After the robot has finished rotating, the robot will stop and the data will be recorded. The data obtained is the ball position on the field. The data is then compared with the actual ball position on the field. There are three tests that have been carried out, namely at a distance of 120 cm, 200 cm, and 285 cm. The following is the evaluation of the test results. The following data is obtained from the test results:

TABLE III
BALL POSITION TEST RESULTS ON THE FIELD WITH A DISTANCE OF 120 CM USING MACHINE LEARNING.

Angle to Ball	Ball Pos X	Ball Pos Y	Distance Error
0 deg	407.74 cm	596.67 cm	8.42 cm
30 deg	409.41 cm	600.3 cm	9.41 cm
60 deg	409.65 cm	600.54 cm	9.66 cm
90 deg	407.52 cm	598.2 cm	7.73 cm
120 deg	407.18 cm	600.07 cm	7.18 cm
150 deg	409.98 cm	598.35 cm	10.11 cm
180 deg	410.66 cm	593.41 cm	12.53 cm
210 deg	410.84 cm	590.28 cm	14.55 cm
240 deg	410.01 cm	590.8 cm	13.59 cm
270 deg	409.9 cm	594.28 cm	11.43 cm
300 deg	408.91 cm	590.14 cm	13.28 cm
330 deg	408.8 cm	591.57 cm	12.18 cm

From that data, we can calculate the average error distance. The following is the calculation of the average error distance:

$$\text{Avg error} = \frac{\sum \text{Distance Error}}{12} = \frac{123.64}{12} = 10.3 \text{ cm} \quad (9)$$

So that we can conclude the Accuracy Error is 10.3 cm.

C. Precision Testing Testing Scenario

The Precision Testing test is comparing the Machine Learning Calibration results with the Polynomial Regression Calibration that still uses the *polynomial regression* algorithm. The test is done in the same way as the previous accuracy test. However, the Camera Calibraion process uses the *polynomial regression* algorithm.

D. Precision Testing Testing Evaluation

The test is done by comparing the data obtained from the Machine Learning Calibration with the data obtained from the Polynomial Regression Calibration. The following is the data obtained from the test results:

TABLE IV
BALL POSITION TEST RESULTS ON THE FIELD WITH A DISTANCE OF 120 CM USING MACHINE LEARNING.

Robot Angle to Ball	Ball Position X	Ball Position Y
0 deg	407.74 cm	596.67 cm
30 deg	409.41 cm	600.3 cm
60 deg	409.65 cm	600.54 cm
90 deg	407.52 cm	598.2 cm
120 deg	407.18 cm	600.07 cm
150 deg	409.98 cm	598.35 cm
180 deg	410.66 cm	593.41 cm
210 deg	410.84 cm	590.28 cm
240 deg	410.01 cm	590.8 cm
270 deg	409.9 cm	594.28 cm
300 deg	408.91 cm	590.14 cm
330 deg	408.8 cm	591.57 cm

TABLE V
BALL POSITION TEST RESULTS ON THE FIELD WITH A DISTANCE OF 120 CM USING THE POLYNOMIAL REGRESSION.

Robot Angle to Ball	Ball Position X	Ball Position Y
0 deg	370.18 cm	576.31 cm
30 deg	376.24 cm	585.42 cm
60 deg	383.91 cm	597.98 cm
90 deg	392.99 cm	605.67 cm
120 deg	390.23 cm	610.94 cm
150 deg	400.49 cm	598.89 cm
180 deg	385.97 cm	582.89 cm
210 deg	398.61 cm	577.15 cm
240 deg	390.76 cm	585.02 cm
270 deg	386.32 cm	588.91 cm
300 deg	378.69 cm	585.55 cm
330 deg	370.32 cm	579.11 cm

From the data, it can be seen that the standard deviation in the ball position data on the field with a distance of 120 cm using the Polynomial Regression Calibration is 10.01 cm and 11.32 cm for the ball position x and y. Meanwhile, in the Machine Learning Calibration, the standard deviation is 1.26 cm and 4.11 cm for the ball position x and y. It can be seen that the Machine Learning Calibration is better than the Polynomial Regression Calibration. This happens because the camera is not installed correctly on the robot, causing the Polynomial Regression Calibration results to be inaccurate. The inaccuracy occurs because the Polynomial Regression Calibration only uses one direction as a reference for the polynomial regression model. Whereas, in reality, the model formula for each camera direction will always be different.

E. Computational Speed Testing Scenario

The test is done by comparing whether the Machine Learning Calibration system is faster than using the Polynomial Regression Calibration system. The test is done by recording the time after calibration and subtracting it from the time before calibration. So that an estimate of the time it takes for the system to perform the calculation process for calibration can be obtained. The following formula is used to calculate the delay time:

$$delay\_time = time1 - time0 \tag{10}$$

Where $time1$ is the time after calibration and $time0$ is the time before calibration.

F. Computational Speed Testing Evaluation

From the tests that have been carried out, there are 10 iterations that have been done. The following is the data obtained from the test results:

TABLE VI
DIFFERENCE IN COMPUTATIONAL SPEED TEST RESULTS

Iteration to-	New Calibration Time	Old Calibration Time
0	119 ns	176 ns
1	76 ns	86 ns
2	25 ns	107 ns
3	31 ns	90 ns
4	32 ns	88 ns
5	36 ns	87 ns
6	37 ns	88 ns
7	34 ns	93 ns
8	37 ns	89 ns
9	35 ns	90 ns

With New Calibration Time stand for the Machine Learning Calibration system and Old Calibration Time stand for the Polynomial Regression Calibration system.

From the graph, it can be seen that the Machine Learning Calibration system is 53.2 ns faster than the Polynomial Regression Calibration system. With an average of 46.2 ns for the Machine Learning Calibration and 99.4 ns for the Polynomial Regression Calibration. This is because the Machine Learning Calibration system only uses a *Lookup Table* containing camera calibration data. Whereas the Polynomial Regression Calibration system uses the polynomial regression algorithm which takes longer.

VI. CONCLUSION

From the research that has been done, it can be concluded that the omnivision camera calibration method using Machine Learning is better than the omnivision camera calibration method using polynomial regression. This can be seen from the ability of the omnivision camera calibration method using Machine Learning to calibrate the omnivision camera in all directions. While the omnivision camera calibration method using polynomial regression can only calibrate the omnivision camera in one direction. In addition, the omnivision camera calibration method using Machine Learning also requires a shorter execution time than the omnivision camera calibration method using polynomial regression.

REFERENCES

[1] F. Grondin, H. Tang, and J. Glass, "Audio-visual calibration with polynomial regression for 2-d projection using svd-phat," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 4856–4860.

[2] C.-H. L. Chen and M.-F. R. Lee, "Global path planning in mobile robot using omnidirectional camera," in *2011 International Conference on Consumer Electronics, Communications and Networks (CECNet)*, 2011, pp. 4986–4989.

[3] H. Tang, H. Li, and Z. Yi, "A discrete-time neural network for optimization problems with hybrid constraints," *IEEE Transactions on Neural Networks*, vol. 21, no. 7, pp. 1184–1189, 2010.

[4] M. Kaloev and G. Krastev, "Comparative analysis of activation functions used in the hidden layers of deep neural networks," in *2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, 2021, pp. 1–5.

[5] N. Dohi, N. Rathnayake, and Y. Hoshino, "A comparative study for covid-19 cases forecasting with loss function as aic and mse in rnn family and arima," in *2022 Joint 12th International Conference on Soft Computing and Intelligent Systems and 23rd International Symposium on Advanced Intelligent Systems (SCIS&ISIS)*, 2022, pp. 1–5.

[6] G. D. S. Lee, K. S. Lee, H. G. Park, and M. H. Lee, "Optimal path planning with holonomic mobile robot using localization vision sensors," in *ICCAS 2010*, 2010, pp. 1883–1886.

[7] M. Yildirim, O. Karaduman, and H. Kurum, "Real-time image and video processing applications using raspberry pi," in *2022 IEEE 1st Industrial Electronics Society Annual On-Line Conference (ONCON)*, 2022, pp. 1–6.

[8] H. D. Quang, T. N. Manh, C. N. Manh, D. P. Tien, M. T. Van, D. H. T. Kim, V. N. T. Thanh, and D. H. Duan, "Mapping and navigation with four-wheeled omnidirectional mobile robot based on robot operating system," in *2019 International Conference on Mechatronics, Robotics and Systems Engineering (MoRSE)*, 2019, pp. 54–59.