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*—In Soccer Robotics Competition, IRIS team archieved 3rd Position in RoboCup. In the game, IRIS Robots used Omnivision to sensing their environtment. The current Calibration method is using polynomial regression for one direction, so that the other direction is not calibrated and give incorrect data. This Final Project propose new method that use Machine Learning. The result shows that Omnivision calibration using Machine Learning is better than using polynomial.**

***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.

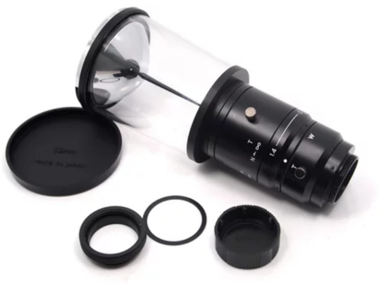


Fig. 1. Omnivision Camera.

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.

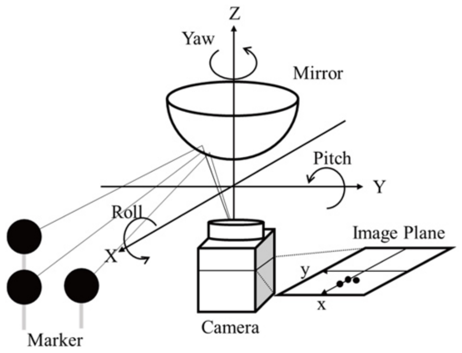


Fig. 2. Basic Concept of Omnivision Camera.

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.

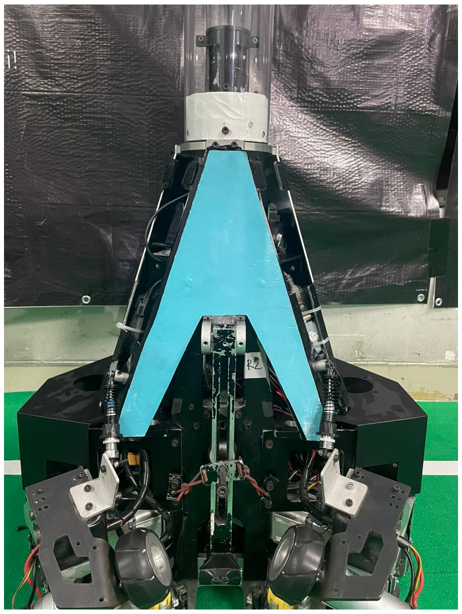


Fig. 3. IRIS Robot.

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 algorithms. 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.

9) Websocket: Websocket is a type of communication protocol based on the TCP protocol (Transmission Control Protocol). The websocket protocol makes both the sender and receiver always open their sockets so that they can communicate with each other. Compared to HTTP (Hypertext Transfer Transfer Protocol), the Websocket protocol has better latency Guan et al. [9]. Websocket applications are widely used for chat applications, online games, applications that require real-time data, and many others.

III. Design and Implementation

A. System Description

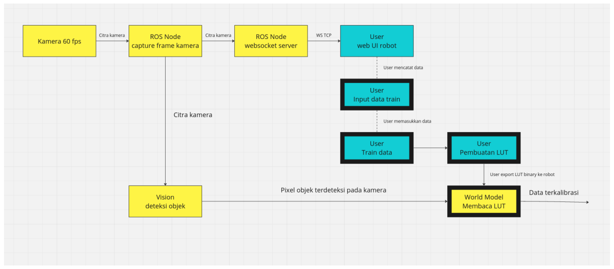


Fig. 4. Design System Block.

From the figure above, the system is divided into two colors, namely yellow and cyan. The yellow color indicates that the process is running on the robot, while the cyan color indicates that the process is running outside the robot. The calibration system focuses on the system marked with a thick border, namely Data Retrieval, Data Training, Creating a Lookup Table, and the last is reading data from the Lookup Table so that it becomes calibrated data.

B. Data Retrieval

The first thing to do before retrieving data is to prepare the robot and the marker that has been made before. The data taken is polar coordinate data both on the camera and on the field. The following formula is used to calculate polar coordinates on the camera:

$$\begin{aligned} dx &= x - x\_center\_cam \\ dy &= y\_center\_cam - y \\ r &= \sqrt{dx^2 + dy^2} \\ \theta &= \arctan(\frac{dy}{dx}) \end{aligned} \tag{2}$$

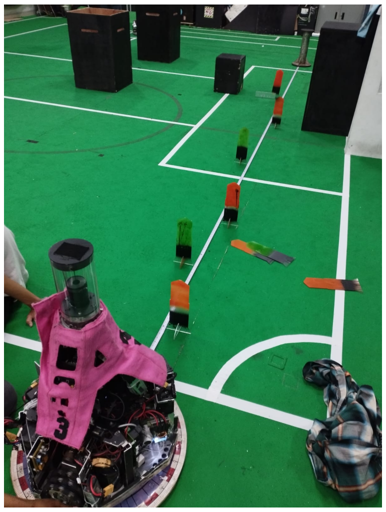


Fig. 5. Robot preparation.

Because the robot does not have a display, to access the camera needs to be done by accessing through the web program provided by the robot. The following is the view of robot camera displayed on the website.

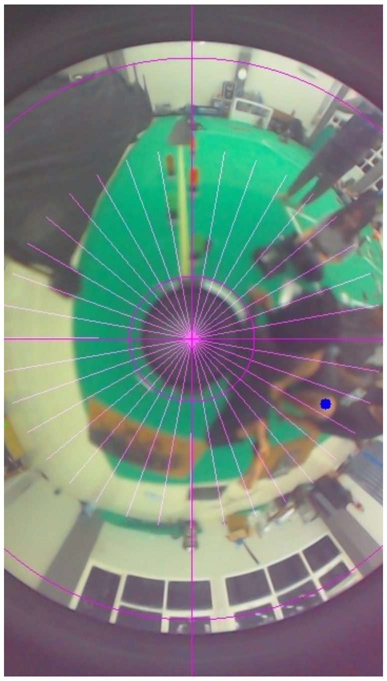


Fig. 6. Robot camera display.

From the website display, the red color on the field can be clicked and the polar coordinates on the camera will appear (according to the formula **3.1**). Then the coordinates will be saved to a file. The data taken is as follows:

C. Training data

The data that has been taken is then processed using the data training program. This program uses the Neural Network

TABLE I
TRAINING DATA.

Theta frame	Distance frame	Field distance
0 deg	123.612 px	110 cm
0 deg	148.355 px	160 cm
0 deg	162.01 px	210 cm
0 deg	171.009 px	260 cm
10 deg	149.684 px	160 cm
19 deg	162.706 px	210 cm
10 deg	171.878 px	260 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	147.681 px	160 cm
350 deg	159.797 px	210 cm
350 deg	169.577 px	260 cm

method to process the data. The following is the data training program.

The full architecture of the Neural Network used is having 2 hidden layers with 36 and 36 neurons each. The activation function used is Sigmoid with the formula as follows:

$$f(x) = \frac{1}{1+e^{-x}} \tag{3}$$

The advantage of using Sigmoid is because Sigmoid has a value range between 0 and 1 which is suitable for the data to be processed.

The loss function used is the Mean Squared Error with the formula as follows:

$$L(y,\hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2 \tag{4}$$

The optimizer used is Adam with a learning rate of 0.0001.

The epoch used is 300000 times. It can change depending on the state of the data being trained.

The full architecture of the Neural Network used is as follows:

- $\mathbf{x} \in \mathbb{R}^2$: Input data
- $\mathbf{W}^{(1)} \in \mathbb{R}^{36\times 2}$: weight layer 1
- $\mathbf{b}^{(1)} \in \mathbb{R}^{36}$: bias layer 1
- $\mathbf{W}^{(2)} \in \mathbb{R}^{36\times 36}$: weight layer 2
- $\mathbf{b}^{(2)} \in \mathbb{R}^{36}$: bias layer 2
- $\mathbf{W}^{(3)} \in \mathbb{R}^{1\times 36}$: weight layer 3
- $\mathbf{b}^{(3)} \in \mathbb{R}^{1}$: bias layer 3
- σ: sigmoid activation function

$$\textbf{Input to Hidden Layer 1}: \begin{aligned} \mathbf{z}^{(1)} &= \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)} \\ \mathbf{a}^{(1)} &= \sigma(\mathbf{z}^{(1)}) \end{aligned} \tag{5}$$

$$\textbf{Hidden Layer 1 to Hidden Layer 2}: \begin{aligned} \mathbf{z}^{(2)} &= \mathbf{W}^{(2)}\mathbf{a}^{(1)} + \mathbf{b}^{(2)} \\ \mathbf{a}^{(2)} &= \sigma(\mathbf{z}^{(2)}) \end{aligned} \tag{6}$$

$$\textbf{Hidden Layer 2 to Output Layer}: \begin{aligned} \mathbf{z}^{(3)} &= \mathbf{W}^{(3)}\mathbf{a}^{(2)} + \mathbf{b}^{(3)} \\ \mathbf{a}^{(3)} &= \mathbf{z}^{(3)} \end{aligned} \tag{7}$$

D. *Creating a Lookup Table*

After the data is trained, the data will be made into a Lookup Table. This Lookup Table contains the data that has been trained before. The following is the formula for creating a 2-dimensional Lookup Table:

$$\begin{aligned} size &= \theta\_max \times r\_max \\ index &= \theta \times r\_max + r \end{aligned} \tag{8}$$

From the formula above, the size of the Lookup Table to be created and the index of the data to be entered into the Lookup Table can be obtained. Then for the value of the Lookup Table itself comes from the formula **3.4 - 3.6**. These values will be entered into the Lookup Table according to the index calculated earlier.

E. *Reading the Lookup Table*

After the Lookup Table is created, the last thing is to read data from the Lookup Table with the formula as follows:

$$\begin{aligned} index &= \theta\_cam \times r\_max + r\_cam \\ r\_real &= \text{Lookup\_Table}[index] \\ \theta\_real &= \theta\_cam \end{aligned} \tag{9}$$

This process is done on the robot. The following is the Lookup Table reading program.

IV. Result and Discussion

A. *Calibration Result Visualization*

The visualization of the calibration results is done by comparing the data on the camera with the data on the field. This is done by entering the camera data into the *Lookup Table* that has been created before. The following is the visualization of the results obtained.

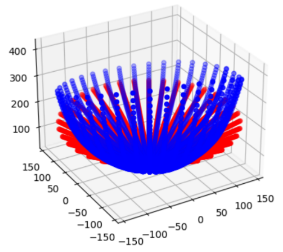
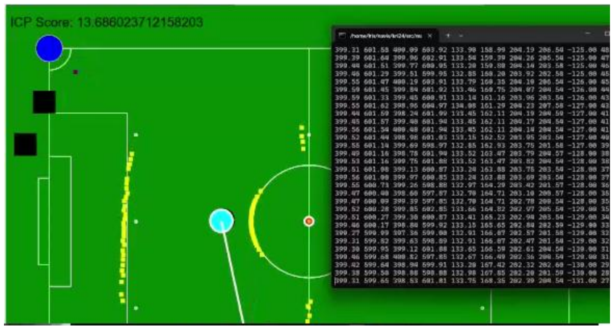


Fig. 7. Visualization of calibration results.

From the figure above, it can be seen that the data on the camera used is not entirely projected perpendicular to the field. This happens because the camera used is not installed correctly.

B. Accuracy Testing Scenario

The test is done by detecting a stationary ball on the field by rotating the robot in its position. This makes the robot able to see the ball from various angles. The test is done by taking data from the omnivision camera that has been installed on the robot and then processing the data using the *Lookup Table* that has been created before so that the ball coordinates on the field are obtained. The following is the testing scenario that was carried out:



```
399.31 601.58 400.09 603.92 133.98 158.99 204.19 206.54 -125.00 48
399.39 601.64 399.96 602.91 133.54 159.39 204.26 205.54 -125.00 47
399.44 601.51 399.77 600.95 133.20 159.80 204.14 203.58 -125.00 46
399.46 601.29 399.51 599.95 132.85 160.20 203.92 202.58 -125.00 45
399.55 601.47 400.19 603.91 133.79 160.35 204.10 206.54 -126.00 45
399.59 601.45 399.84 601.92 133.46 160.75 204.07 204.54 -126.00 44
399.59 601.33 399.45 600.91 133.14 161.16 203.96 203.54 -126.00 43
399.55 601.62 398.96 604.97 134.08 161.29 204.23 207.58 -127.00 43
399.44 601.59 398.24 601.99 133.45 162.11 204.19 204.59 -127.00 41
399.45 601.57 399.48 601.94 133.45 162.11 204.17 204.54 -127.00 41
399.56 601.54 400.48 601.94 133.45 162.11 204.14 204.54 -127.00 41
399.52 601.44 398.98 601.03 133.15 162.52 203.95 203.54 -127.00 40
399.55 601.14 399.69 598.97 132.85 162.93 203.75 201.58 -127.00 39
399.49 601.16 398.78 601.94 133.52 163.47 203.79 204.57 -128.00 38
399.53 601.16 399.76 601.88 133.52 163.47 203.82 204.54 -128.00 38
399.51 601.08 399.13 600.87 133.24 163.88 203.75 203.54 -128.00 37
399.56 601.00 399.97 600.85 133.24 163.88 203.69 203.54 -128.00 37
399.55 600.73 399.26 598.88 132.97 164.29 203.42 201.57 -128.00 36
399.47 600.40 398.60 597.87 132.70 164.71 203.10 200.57 -128.00 35
399.47 600.09 399.39 597.85 132.70 164.71 202.78 200.54 -128.00 35
399.52 600.28 399.85 602.85 133.66 164.82 202.97 205.54 -129.00 35
399.51 600.27 399.38 600.87 133.41 165.23 202.94 203.54 -129.00 34
399.46 600.17 398.84 599.92 133.15 165.65 202.84 202.59 -129.00 33
399.27 599.99 397.36 599.00 132.91 166.07 202.57 201.58 -129.00 32
399.31 599.82 399.63 598.89 132.91 166.07 202.47 201.58 -129.00 32
399.30 599.95 399.12 601.88 133.65 166.59 202.61 204.54 -130.00 31
399.46 599.68 400.82 597.85 132.67 166.49 202.36 200.54 -129.00 31
399.42 599.64 398.94 599.91 133.28 167.42 202.32 202.60 -130.00 29
399.38 599.58 398.88 598.88 132.98 167.85 202.20 201.59 -130.00 28
399.31 599.65 398.53 601.81 133.75 168.35 202.39 204.54 -131.00 27
```

Fig. 8. Testing Scenario.

The procedure is carried out at different distances between the robot and the ball on the field, namely at 120 cm, 200 cm, and 285 cm.

The formula used to calculate the ball's position on the field is as follows:

$$\begin{aligned}
dx &= x\_ball\_cam - x\_center\_cam \\
dy &= y\_center\_cam - y\_ball\_cam \\
r\_ball\_cam &= \sqrt{dx^2 + dy^2} \\
\theta\_ball\_cam &= \arctan(\frac{dy}{dx}) \\
index &= \theta\_ball\_cam \times r\_max + r\_ball\_cam \\
r\_ball\_fld &= r\_lookup[index] \\
\theta\_ball\_fld &= \theta\_ball\_cam + robot\_\theta - 90 \\
x\_ball &= robot\_x + r\_ball\_fld \times \cos(\theta\_ball\_fld) \\
y\_ball &= robot\_y + r\_ball\_fld \times \sin(\theta\_ball\_fld)
\end{aligned} \tag{10}$$

C. 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 II
BALL POSITION TEST RESULTS ON THE FIELD WITH A DISTANCE OF 120 CM.

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 III
BALL POSITION TEST RESULTS ON THE FIELD WITH A DISTANCE OF 200 CM.

Robot Angle to Ball	Ball Position X	Ball Position Y
0 deg	406.44 cm	613.13 cm
30 deg	411.09 cm	624.66 cm
60 deg	416.39 cm	627.02 cm
90 deg	411.55 cm	625.85 cm
120 deg	408.85 cm	620.01 cm
150 deg	414.2 cm	611.7 cm
180 deg	415.12 cm	601.25 cm
210 deg	412.78 cm	592.09 cm
240 deg	410.77 cm	594.57 cm
270 deg	409.66 cm	598.43 cm
300 deg	407.92 cm	599.03 cm
330 deg	406.29 cm	602.72 cm

Each table contains the ball position data on the field at different distances. Every data in that table is the ball position data on the field at a certain angle. The data is then compared with the actual ball position on the field. The comparison is done by calculating the standard deviation of the data obtained. The formula used to calculate the standard deviation is as follows:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}} \tag{11}$$

TABLE IV
BALL POSITION TEST RESULTS ON THE FIELD WITH A DISTANCE OF 285 CM.

Robot Angle to Ball	Ball Position X	Ball Position Y
0 deg	380.98 cm	593.81 cm
30 deg	382.14 cm	594.49 cm
60 deg	383.41 cm	607.18 cm
90 deg	392.49 cm	604.67 cm
120 deg	390.32 cm	606.04 cm
150 deg	391.43 cm	598.47 cm
180 deg	387.17 cm	587.89 cm
210 deg	390.16 cm	577.15 cm
240 deg	389.67 cm	585.82 cm
270 deg	389.38 cm	590.31 cm
300 deg	382.61 cm	593.25 cm
330 deg	380.73 cm	589.31 cm

Where σ is the standard deviation, x_i is the data, $\bar{x}$ is the average of the data, and n is the number of data.

From the table, the standard deviation in the ball position data on the field with a distance of 120 cm is 1.26 cm and 4.11 cm for the ball position x and y. Meanwhile, at a distance of 200 cm, the standard deviation is 3.28 cm and 12.80 cm for the ball position x and y. At a distance of 285 cm, the standard deviation is 4.40 cm and 8.92 cm for the ball position x and y.

From these results, it can be concluded that the calibration system that has been created can detect the ball position on the field well. The ball not being exactly in the middle of the field can be caused by several factors. Some of the factors causing this are the inaccuracy of the ball position in the real world, the inaccuracy of the robot position in the real world, and the inaccuracy of the robot orientation in the real world. Because basically according to the formula **4.1**, the ball position on the field is also determined by the robot's position on the field.

D. Second Accuracy Testing Scenario

The second accuracy test is by comparing the new calibration results with the old calibration that still uses the *polynomial regression* algorithm. The test is done in the same way as the previous accuracy test. However, the calculation process uses the *polynomial regression* algorithm.

E. Second Accuracy Testing Evaluation

The test is done by comparing the data obtained from the new calibration with the data obtained from the old calibration. The following is the data obtained from the test results:

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

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 old calibration is 10.01 cm and 11.32 cm for the ball position x and y. Meanwhile, in the new 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 new calibration is better than the old calibration. This happens because the camera is not installed correctly on the robot, causing the old calibration results to be inaccurate. The inaccuracy occurs because the old 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.

F. Third Accuracy Testing

The third accuracy test is placing robot somewhere on the field and then the robot will see lines around the robot. The robot will detect each pixel of line and then the robot will calculate the distance between the robot and the line. So the robot can visualize the line on the field. The test is done by taking data from the omnivision camera that has been installed on the robot and then processing the data using the *Lookup Table* that has been created before so that the line coordinates on the field are obtained. The following is the testing scenario that was carried out:

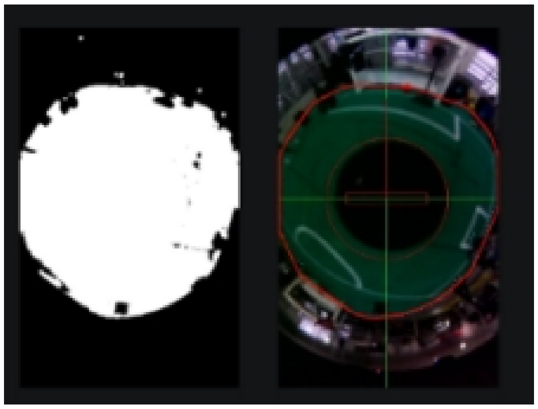


Fig. 9. Raw frame.

The left picture is the field binary image that calculated to get the line binary image.

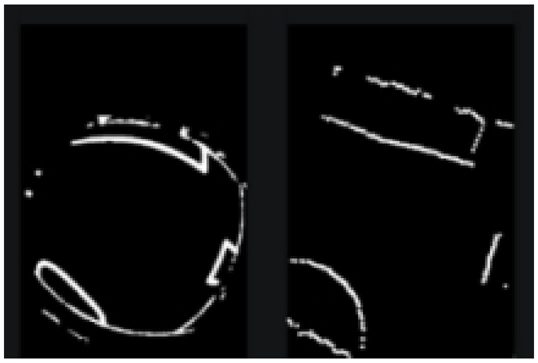


Fig. 10. Calculated frame.

The left picture is the line binary image that calculated to get the line coordinates on frame. Extracting the line coordinates on the frame is done by scanning from the center of frame to the edge of frame.

After getting the line coordinates on the frame, the robot will calculate the world coordinates of the line. The formula used to calculate the line's position on the field is as follows:

Algorithm 2 Process Lines on Frame

```
1: procedure ProcessLinesOnFrame
2:     Initialize lines_on_frame as an empty vector
3:     for angle from 0 to 360 with step size 2.5 do
4:         Initialize dist to 0
5:         for index from 0 to 320 do
6:             $x \leftarrow \text{dist} \times \cos(\text{angle}) + \text{center\_cam\_x}$
7:             $y \leftarrow \text{center\_cam\_y} - \text{dist} \times \sin(\text{angle})$
8:             if frame[y][x] == 255 then
9:                 Push (x, y) to lines_on_frame
10:            end if
11:            $dist \leftarrow \text{dist} + 1$
12:        end for
13:    end for
14: end procedure
```

$$\begin{aligned} dx &= x\_line\_cam - x\_center\_cam \\ dy &= y\_center\_cam - y\_line\_cam \\ r\_line\_cam &= \sqrt{dx^2 + dy^2} \\ \theta\_line\_cam &= \arctan(\frac{dy}{dx}) \\ index &= \theta\_line\_cam \times r\_max + r\_line\_cam \\ r\_line\_fld &= r\_lookup[index] \\ \theta\_line\_fld &= \theta\_line\_cam + robot\_\theta - 90 \\ x\_line &= robot\_x + r\_line\_fld \times \cos(\theta\_line\_fld) \\ y\_line &= robot\_y + r\_line\_fld \times \sin(\theta\_line\_fld) \end{aligned} \tag{12}$$

After the robot has finished calculating the line's position on the field, the robot will visualize the line on the field. The visualization can be seen on right picture of figure 10.

G. Computational Speed Testing Scenario

The test is done by comparing whether the new calibration system is faster than using the old 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:

$$delay\_time = time1 - time0 \tag{13}$$

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

H. 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

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

V. 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.

[9] S. Guan, W. Hu, and H. Zhou, “Real-time data transmission method based on websocket protocol for networked control system laboratory,” in *2019 Chinese Control Conference (CCC)*, 2019, pp. 5339–5344.