KALIBRASI KAMERA *OMNIVISION* PADA *MOBILE ROBOT* MENGGUNAKAN *MACHINE LEARNING*

Azzam Wildan Maulana
Departemen Teknik Komputer
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia 60111
5024201010@student.its.ac.id

Muhtadin, S.T., M.T.
Departemen Teknik Komputer
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia 60111
muhtadin@te.its.ac.id

Ahmad Zaini, S.T., M.T.
Departemen Teknik Komputer
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia 60111
zaini@te.its.ac.id

***Abstrak*—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.**

***Kata kunci*—Omnivision, Calibration, IRIS**

I. PENDAHULUAN

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.

II. TINJAUAN PUSTAKA

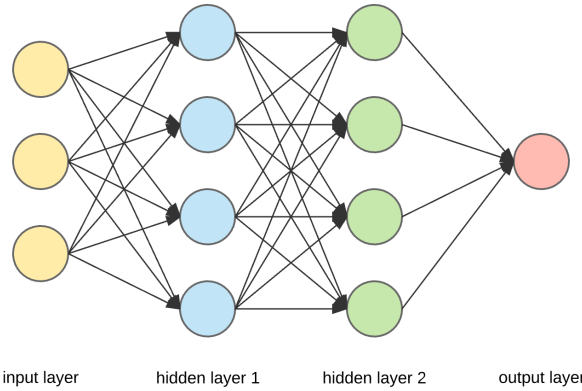
A. Hasil penelitian/perancangan terdahulu

Metode Regresi Polinomial adalah sebuah metode pendekatan terhadap data-data yang telah disediakan sebelumnya. Hasil keluaran dari metode Regresi Polinomial ini berupa rumusan matematika berdasarkan data-data yang telah disediakan. Regresi polinomial dapat bekerja secara efisien meskipun dengan model yang non-linear Grondin et al. [1].

B. Teori/Konsep Dasar

1) Kamera Omnivision*:* Kamera *omnivision* adalah sebuah kamera yang bisa melihat 360 derajat sekitar kamera tersebut Chen and Lee [2]. Jarak pandang kamera *omnivision* tidak terbatas tergantung dari resolusi kamera itu sendiri dan konstruksi cerminnya. Pada dasarnya kamera *omnivision* adalah kamera biasa yang ditembakkan ke sebuah cermin cembung sehingga pandangan kamera tersebut bisa ke segala arah.

Gambar 1: Kamera Omnivsion.



Gambar 2: Neural Network.

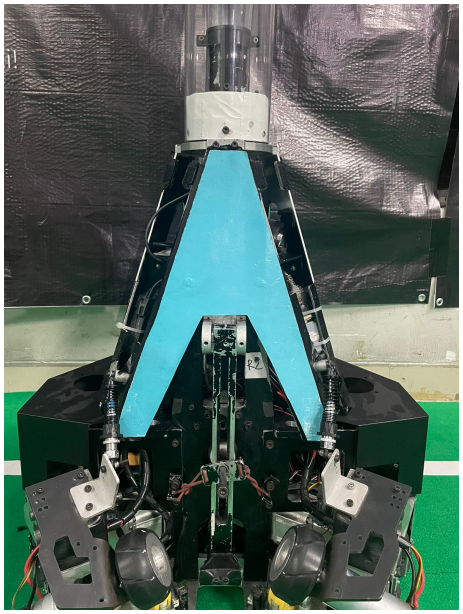
2) Neural Network: Neural network merupakan bagian dari Pembelajaran Mesin. *Neural network* diciptakan untuk mengatasi masalah ketidaklinearan pada sebuah model Tang et al. [3].

Pada dasarnya, *Neural network* hanyalah sekumpulan *Neuron* yang terhubung oleh sebuah *Weight* dan *Bias*. Selain *Weight* dan *Bias*, ada juga namanya *forward propagation* menggunakan *activation function* atau biasa disebut transfer function. Selain *forward propagation*, terdapat *backward propagation* menggunakan *loss function*.

3) Activation Function*: Activation function* adalah sebuah fungsi yang digunakan untuk men-transfer data pada masing-masing layer pada *Neural Network*. Dalam *Neural Network*, *Activation function* berperan sebagai *forward propagation* yaitu perjalanan dari layer input menuju layer output. Beberapa *activation Function* memiliki nilai saturasi biasanya bernilai 1 contohnya Sigmoid yang bernilai pada interval 0 sampai 1. Ada juga *activation Function* lain yaitu tanh yang bernilai pada interval -1 sampai 1 Kaloev and Krastev [4].

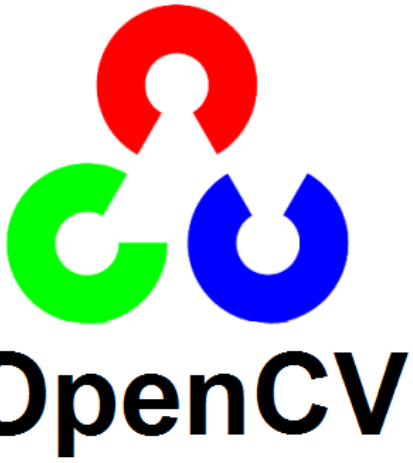
4) Loss Function*: loss function* adalah bagian dari *Neural Network* yang bertujuan untuk memberikan umpan balik pada model tentang baik atau buruknya fase *training*. *loss function* pada *Neural Network* bekerja pada jalur *Backward Propagation* yaitu dari layer output menuju layer input. Ada beberapa macam *loss function* salah satunya adalah MSE (*Mean Squared Error*). MSE *loss function* lebih baik digunakan pada data dengan nilai fluktuasi yang rendah Dohi et al. [5].

Selain *loss function*, ada sebuah teori lagi yaitu Optimizer. Optimizer adalah sebuah algoritma yang bisa digunakan untuk menentukan *learning rate* sistem training. *Learning rate* pada *Neural Network* digunakan untuk mengatur seberapa cepat model akan konvergen terhadap data-datanya.



Gambar 3: Robot IRIS tampak depan.

5) Mobile Robot*: Mobile Robot* adalah sebuah robot yang didesain agar bisa bergerak atau berpindah tempat dengan mudah. Performa *Mobile Robot* banyak dikhususkan pada sistem tracking, lokalisasi, dan algoritma. Ketiga hal tersebut berdasar pada kemampuan sensing yang baik Lee et al. [6]. Pada umumnya, sensor yang digunakan adalah kamera baik itu kamera biasa maupun kamera *omnivision*. Penggunaan kamera *omnivision* dapat membuat robot melihat ke segala arah, Namun pre-processing datanya yang lebih sulit dibandingkan dengan kamera biasa.



Gambar 4: Logo OpenCV.

6) OpenCV*: OpenCV* adalah library open-source yang dikembangkan oleh Intel dengan bahasa pemrograman C/C++. *OpenCV* menyediakan banyak algoritma yang berhubungan dengan Visi Komputer Yildirim et al. [7]. OpenCV banyak digunakan untuk deteksi objek baik itu berdasarkan warna, bentuk, ukuran, dan lain lain sesuai kebutuhan program.

Gambar 5: Logo ROS.

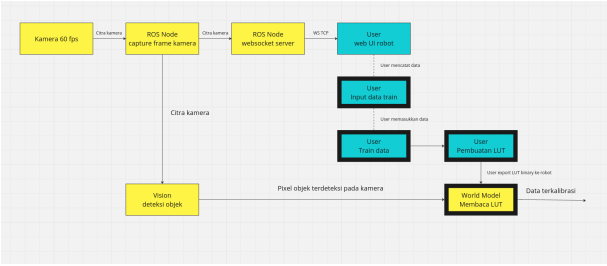
7) Robot Operating System: ROS atau *Robot Operating System* adalah sebuah platform yang berdiri diatas Linux dan berguna untuk sinkronisasi bagian bagian dari robot Quang et al. [8]. ROS banyak digunakan sebagai inti pemrosesan data dari sebuah robot mulai dari pemrosesan data sensor hingga menjadi data aktuator. ROS menyediakan konsep modular programming dengan metode publish/subscribe untuk IPC (*Inter Process Communication*) nya. Selain memudahkan

untuk transfer data antar proses, ROS juga menyediakan timer dengan scheduler default nya mengikuti Default Linux Scheduler yaitu Priority-based scheduler. Hal itu memungkinkan pengguna untuk mengatur prioritas masing-masing bagian dari robotnya.

8) Websocket: Websocket adalah jenis protokol komunikasi berbasis protokol TCP (*Transmission Control Protocol*). Protokol websocket membuat kedua pengirim dan menerima untuk selalu membuka socket nya agar bisa saling komunikasi. Dibandingkan dengan HTTP (*Hypertext Transfer Transfer Protocol*), protokol Websocket memiliki latensi yang lebih baik Guan et al. [9]. Aplikasi websocket banyak digunakan untuk aplikasi obrolan (chat), game online, aplikasi yang membutuhkan data *realtime*, dan masih banyak lainnya.

III. Desain dan Implementasi

A. Deskripsi Sistem



Gambar 6: Tampilan website robot.

Dari gambar diatas dapat dilihat bahwa sistem terbagi menjadi dua warna yaitu warna kuning dan warna cyan. Warna kuning menandakan bahwa proses tersebut berjalan pada robot sedangkan warna cyan menandakan bahwa proses tersebut berjalan di luar robot. Sistem kalibrasi berfokus pada sistem yang ditandai dengan ukuran border yang tebal yaitu Pengambilan data, *Training* data, Pembuatan *Lookup Table*, dan yang terakhir adalah membaca data dari *Lookup Table* tersebut sehingga menjadi data yang telah terkalibrasi.

B. Pengambilan Data

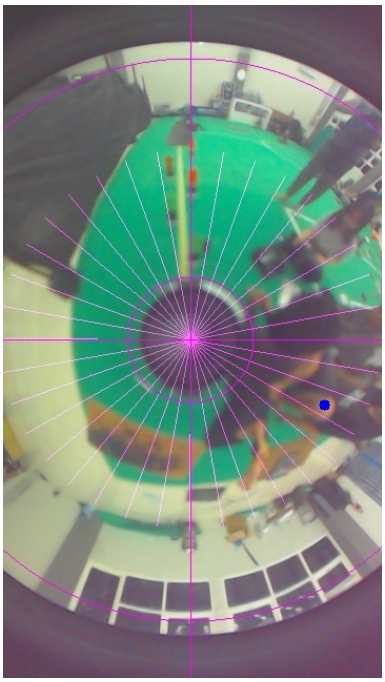
Hal yang dilakukan sebelum pengambilan data adalah menyiapkan robot dan marker yang telah dibuat sebelumnya. Adapun data yang diambil merupakan data koordinat polar baik itu pada kamera maupun pada lapangan. Berikut rumus yang digunakan untuk menghitung koordinat polar pada kamera:

$$\begin{aligned} dx &= x - x\_center\_cam \\ dy &= y\_center\_cam - y \\ r &= \sqrt{dx^2 + dy^2} \\ \theta &= \arctan(\frac{dy}{dx}) \end{aligned} \tag{1}$$

Dikarenakan robot tidak memiliki display, maka untuk mengakses kamera perlu dilakukan dengan mengakses melalui program web yang telah disediakan oleh robot. Berikut adalah

Gambar 7: Persiapan robot.

program robot untuk mengakses kamera dan mengirim ke web mengunakan *websocket*.



Gambar 8: Tampilan website robot.

Dari tampilan website tersebut, warna merah pada lapangan bisa diklik dan akan muncul koordinat polar pada kamera (sesuai pada rumus **3.1**). Lalu koordinat-koordinat tersebut akan disimpan ke dalam file. Adapun data yang diambil adalah sebagai berikut:

C. Training data

Data yang telah diambil kemudian diolah dengan menggunakan program *training* data. Program ini menggunakan metode *Neural Network* untuk mengolah data tersebut. Berikut adalah program *training* data.

TABEL I
DATA *training*.

Theta frame	Jarak frame	Jarak lapangan
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

Adapun arsitektur *Neural Network* yang digunakan adalah memiliki 2 *hidden layer* dengan masing-masing 36 dan 36 *neuron*. *Activation function* yang digunakan adalah *Sigmoid* dengan rumusan sebagai berikut:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

Keuntungan menggunakan *Sigmoid* adalah karena *Sigmoid* memiliki rentang nilai antara 0 dan 1 yang sesuai dengan data yang akan diolah.

Loss function yang digunakan adalah *Mean Squared Error* dengan rumusan sebagai berikut:

$$L(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{3}$$

Optimizer yang digunakan adalah *Adam* dengan *learning rate* sebesar 0.0001.

Adapun epoch yang digunakan adalah sebanyak 300000 kali. Hal itu bisa berubah tergantung keadaan data yang di-*train*.

Adapun Arsitektur penuh dari *Neural Network* yang digunakan adalah sebagai berikut:

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

Input ke Hidden Layer Pertama :

$$\begin{aligned} \mathbf{z}^{(1)} &= \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)} \\ \mathbf{a}^{(1)} &= \sigma(\mathbf{z}^{(1)}) \end{aligned} \tag{4}$$

Hidden Layer Pertama ke Hidden Layer Kedua :

$$\begin{aligned} \mathbf{z}^{(2)} &= \mathbf{W}^{(2)}\mathbf{a}^{(1)} + \mathbf{b}^{(2)} \\ \mathbf{a}^{(2)} &= \sigma(\mathbf{z}^{(2)}) \end{aligned} \tag{5}$$

Hidden Layer Kedua ke Output Layer :

$$\begin{aligned} \mathbf{z}^{(3)} &= \mathbf{W}^{(3)}\mathbf{a}^{(2)} + \mathbf{b}^{(3)} \\ \mathbf{a}^{(3)} &= \mathbf{z}^{(3)} \end{aligned} \tag{6}$$

D. Pembuatan Lookup Table

Setelah data di-*train* maka data tersebut akan dijadikan *Lookup Table*. *Lookup Table* ini berisi data-data yang telah di-*train* sebelumnya. Berikut adalah rumusan untuk membuat *Lookup Table* 2 dimensi:

$$\begin{aligned} size &= \theta\_max \times r\_max \\ index &= \theta \times r\_max + r \end{aligned} \tag{7}$$

Dari rumusan tersebut bisa didapat ukuran dari *Lookup Table* yang akan dibuat dan index dari data yang akan dimasukkan ke dalam *Lookup Table*. Kemudian untuk nilai dari *Lookup Table* itu sendiri berasal dari rumusan **3.4 - 3.6**. Nilai-nilai tersebut akan dimasukkan ke dalam *Lookup Table* sesuai dengan index yang telah dihitung sebelumnya.

E. Pembacaan Lookup Table

Setelah *Lookup Table* dibuat maka yang terakhir adalah membaca data dari *Lookup Table* dengan rumusan sebagai berikut.

$$\begin{aligned} index &= \theta\_cam \times r\_max + r\_cam \\ r\_real &= \textit{Lookup\_Table}[index] \\ \theta\_real &= \theta\_cam \end{aligned} \tag{8}$$

Proses tersebut dilakukan pada robot. Berikut adalah program pembacaan *Lookup Table*.

IV. HASIL DAN PEMBAHASAN

A. Visualisasi Hasil Kalibrasi

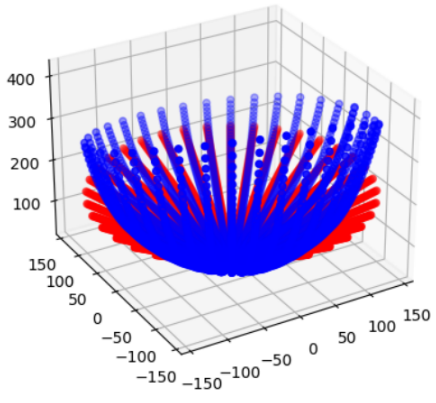
Visualisasi hasil kalibrasi dilakukan dengan cara data pada kamera dengan data pada lapangan. Hal itu dilakukan dengan cara memasukkan data pada kamera ke dalam *Lookup Table* yang telah dibuat sebelumnya. Berikut adalah hasil visualisasi yang didapat.

Dari gambar tersebut, dapat dilihat bahwa data pada kamera yang digunakan tidak sepenuhnya diproyeksikan tegak lurus 90 derajat dengan lapangan. Hal itu terjadi karena kamera yang digunakan tidak terpasang dengan benar.

B. Skenario Pengujian Akurasi

Pengujian dilakukan dengan cara mendeteksi bola yang diam di lapangan dengan memutar robot pada posisinya sendiri. Hal itu membuat robot dapat melihat bola dari berbagai sudut. Pengujian dilakukan dengan mengambil data dari kamera omnivision yang telah terpasang pada robot lalu memroses data tersebut menggunakan *Lookup Table* yang telah dibuat sebelumnya sehingga didapat koordinat bola pada lapangan. Berikut adalah skenario pengujian yang dilakukan:

Prosedur tersebut dilakukan pada jarak robot dengan bola pada lapangan yang berbeda-beda yaitu pada 120 cm, 200 cm, dan 285 cm.



Gambar 9: Visualisasi hasil kalibrasi.

Gambar 10: Skenario Pengujian.

Adapun rumusan yang digunakan untuk menghitung posisi bola pada lapangan adalah sebagai berikut:

$$
\begin{aligned}
dx &= x\_bola\_cam - x\_center\_cam \\
dy &= y\_center\_cam - y\_bola\_cam \\
r\_bola\_cam &= \sqrt{dx^2 + dy^2} \\
\theta\_bola\_cam &= \arctan(\frac{dy}{dx}) \\
index &= \theta\_bola\_cam \times r\_max + r\_bola\_cam \\
r\_bola\_lap &= r\_lookup[index] \\
\theta\_bola\_lap &= \theta\_bola\_cam + robot\_pose\_\theta - 90 \\
x\_bola &= robot\_pose\_x + r\_bola\_lap \times \cos(\theta\_bola\_lap) \\
y\_bola &= robot\_pose\_y + r\_bola\_lap \times \sin(\theta\_bola\_lap)
\end{aligned} \tag{9}
$$

C. Evaluasi Pengujian Akurasi

Dari pengujian yang telah dilakukan, didapat data sebagai berikut:

Dari tabel tersebut didapat standar deviasi pada data posisi bola pada lapangan dengan jarak 120 cm adalah 1.26 cm dan 4.11 cm untuk posisi bola x dan y. Sedangkan pada jarak 200 cm didapat standar deviasi 3.28 cm dan 12.80 cm untuk posisi bola x dan y. Pada jarak 285 cm didapat standar deviasi 4.40 cm dan 8.92 cm untuk posisi bola x dan y.

Dari hasil tersebut dapat disimpulkan bahwa sistem kalibrasi yang telah dibuat dapat mendeteksi posisi bola pada lapangan dengan baik. Bola tidak tepat berada di tengah lapangan bisa disebabkan oleh beberapa faktor. Adapun beberapa faktor penyebabnya adalah ketidaktepatan posisi bola pada dunia aslinya, ketidaktepatan posisi robot pada dunia aslinya, dan ketidaktepatan orientasi robot pada dunia aslinya. Karena pada

TABEL II
HASIL PENGUJIAN POSISI BOLA PADA LAPANGAN DENGAN JARAK 120 CM.

Sudut Robot ke Bola	Posisi Bola X	Posisi Bola 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

TABEL III
HASIL PENGUJIAN POSISI BOLA PADA LAPANGAN DENGAN JARAK 200 CM.

Sudut Robot ke Bola	Posisi Bola X	Posisi Bola 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

TABEL IV
HASIL PENGUJIAN POSISI BOLA PADA LAPANGAN DENGAN JARAK 285 CM.

Sudut Robot ke Bola	Posisi Bola X	Posisi Bola 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

dasarnya sesuai dengan rumus **4.1**, posisi bola pada lapangan juga ditentukan oleh posisi robot pada lapangan.

D. Skenario Pengujian Akurasi Kedua

Pengujian akurasi kedua adalah dengan cara membandingkan hasil kalibrasi baru dengan kalibrasi lama yang masih menggunakan algoritma *regresi polynomial*. Pengujian dilakukan dengan cara yang sama seperti pengujian akurasi sebelumnya. Namun, proses perhitungannya menggunakan algoritma *regresi polynomial*.

E. Evaluasi Pengujian Akurasi Kedua

TABEL V
HASIL PENGUJIAN POSISI BOLA PADA LAPANGAN DENGAN JARAK 120 CM MENGGUNAKAN KALIBRASI LAMA.

Sudut Robot ke Bola	Posisi Bola X	Posisi Bola 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

Dari data tersebut dapat dilihat bahwa standar deviasi pada data posisi bola pada lapangan dengan jarak 120 cm menggunakan kalibrasi lama adalah 10.01 cm dan 11.32 cm untuk posisi bola x dan y. Sedangkan pada kalibrasi baru standar deviasi 1.26 cm dan 4.11 cm untuk posisi bola x dan y. Dapat dilihat bahwa kalibrasi baru lebih baik dibandingkan dengan kalibrasi lama. Hal itu terjadi karena pemasangan kamera pada robot yang tidak tepat sehingga menyebabkan hasil kalibrasi lama menjadi tidak akurat. Ketidakakuratan tersebut terjadi karena kalibrasi lama hanya menggunakan satu arah saja sebagai referensi untuk model regresi polynomial. Sedangkan, pada kenyataannya rumus model untuk masing-masing arah kamera akan selalu berbeda.

F. Skenario Pengujian Kecepatan Komputasi

Pengujian dilakukan dengan cara membandingkan apakah sistem kalibrasi baru lebih cepat dibandingkan menggunakan sistem kalibrasi lama. Pengujian dilakukan dengan cara mencatat waktu setelah kalibrasi lalu menguranginya dengan waktu sebelum kalibrasi. Sehingga bisa didapat perkiraan waktu lamanya sistem melakukan proses perhitungan untuk melakukan kalibrasi. Berikut rumus yang digunakan untuk menghitung waktu delay:

$$delay\_time = waktu1 - waktu0 \tag{10}$$

Dimana $waktu1$ adalah waktu setelah kalibrasi dan $waktu0$ adalah waktu sebelum kalibrasi.

lclclcl

TABEL VI
HASIL PENGUJIAN PERBEDAAN KECEPATAN KOMPUTASI

Iterasi ke-	Waktu Kalibrasi Baru	Waktu Kalibrasi lama
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

G. Evaluasi Pengujian Kecepatan Komputasi

Dari pengujian yang telah dilakukan, didapat data sebagai berikut:

Dari grafik tersebut dapat dilihat bahwa sistem kalibrasi baru lebih cepat 53.2 ns dibandingkan dengan sistem kalibrasi lama. Dengan rata-rata Kalibrasi baru membutuhkan waktu 46.2 ns sedangkan kalibrasi lama membutuhkan waktu 99.4 ns. Hal tersebut disebabkan karena sistem kalibrasi baru hanya menggunakan *Lookup Table* yang berisi data kalibrasi kamera. Sedangkan sistem kalibrasi lama menggunakan algoritma *regresi polynomial* yang membutuhkan waktu lebih lama.

V. KESIMPULAN

Dari hasil penelitian yang telah dilakukan, dapat disimpulkan bahwa metode kalibrasi kamera omnivision menggunakan *Machine Learning* lebih baik daripada metode kalibrasi kamera omnivision menggunakan regresi polinomial. Hal ini dapat dilihat dari kemampuan metode kalibrasi kamera omnivision menggunakan *Machine Learning* yang dapat meng-kalibrasi kamera omnivision pada semua arah. Sedangkan metode kalibrasi kamera omnivision menggunakan regresi polinomial hanya dapat meng-kalibrasi kamera omnivision pada satu arah saja. Selain itu, metode kalibrasi kamera omnivision menggunakan *Machine Learning* juga membutuhkan waktu eksekusi yang lebih singkat daripada metode kalibrasi kamera omnivision menggunakan regresi polinomial.

PUSTAKA

[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.