JURNAL INFOTEL
Informatics - Telecommunication - Electronics
Website Jurnal : http://ejournal.st3telkom.ac.id/index.php/infotel
ISSN : 2085-3688; e-ISSN : 2460-0997

Object Position Estimation based on Dual Sight Perspective Configuration

Dion Setiawan[1], Maulana Ifdhil Hanafi [2], Indra Riyanto[3], Akhmad Musafa[4*]
[1,2,3,4] Fakultas Tenik, Universitas Budi Luhur
[1,2,3,4]Ciledug Raya Street, Jakarta Selatan 12260, DKI Jakarta, Indonesia
*Corresponding email: akhmad.musafa@budiluhur.ac.id

Received 29 September 2020, Revised 20 December 2020, Accepted 08 May 2021

Abstract — Development of the coordination system requires the dataset because the dataset could provide information around the system that the coordination system can use to make decisions. Therefore, the capability to process and display data-related positions of objects around the robots is necessary. This paper provides a method to predict an object's position. This method is based on the Indoor Positioning System (IPS) idea and object position estimation with the multi-camera system (i.e., stereo vision). This method needs two input data to estimate the ball position: the input image and the robot's relative position. The approach adopts simple and easy calculation technics: trigonometry, angle rotations, and linear function. This method was tested on a ROS and Gazebo simulation platform. The experimental result shows that this configuration could estimate the object's position with Mean Squared Error was 0.383 meters. Besides, R squared distance calibration value is 0.9932, which implies that this system worked very well at estimating an object's position.

Keywords – Object Detection, Position Coordinate Estimation, 3D Simulation, ROS

I. Introduction

The development of the coordination system requires a dataset [1] because the data could provide information around the system. The command or coordination system uses the dataset to produce a command set further. Therefore, processing and displaying data related to positions (in coordinate format) is necessary. The dataset represents the information and stuffs layout around the robot. Then, a command set containing each robot's next behavior is created by the command system as a result.

Various approaches and models are submitted to sample the conditions around a system or robot based on their sensor types. One of those approaches is, using a sensor configuration based on an omnidirectional camera configuration [2]. The system could detect objects around the system in a circular area within a certain radius using this sensor configuration. On the other hand, this configuration could calculate a distance between the robot and the object even with a simple regression function [3].

The localization approaches (methods) are under development so that the robot can retrieve data from the surrounding environment; one of them is Monte Carlo Localization [4]. The generated result by using this method is a map. Then this method would generate a better result in an indoor area [5], [6]. Meanwhile, this method requires a high capability computation hardware and the use of appropriate sensors such as LiDAR [7], [8]. Otherwise, systems with lower computation capability (standard microcontrollers) would degenerate the system's performance.

In this research, a simple method that can estimate an object's coordinate is proposed and named the Dual-Sight Perspective configuration. The method is tested in the realm of the simulation system. The created technique consists of trigonometry and angle rotations functions. This configuration is expected to ease the computation and could be implemented on a low computation capability hardware.

II. RESEARCH METHODS

A. Camera Configuration

Fundamentally, detecting objects must require sensors. In this case, the desired output sensor data is in the form of an image. Therefore, the right sensor to use is a vision sensor such as a camera. The vision sensor has various configurations available to use, such as mono or perspective vision [9], stereo vision [10], and omnidirectional vision [11], shown in Fig.1, to detect objects. Further, in the case of detecting objects using omnidirectional cameras, a few available methods are Blob Tracking, Hough Circle Transform [12], [13], and even by using a neural network method [14].

The configuration of stereo vision is an example or even a part of a multi-camera system, for other examples of multi-camera systems available. The usage of multi-camera systems is distance measurement systems [15], 3d image reconstruction [16], object's or robot's control position system [17], [18], and indoor positioning systems [19].

In this study, omnidirectional camera configuration is the best option because this camera has almost or even more than 180 degrees FOV. Furthermore, the single-point omnidirectional camera is desirable. Because the captured image is, but it would need a geometry correction or calibration [20].

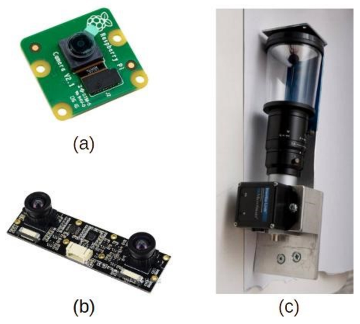

Fig.1. (a) Perspective Camera; (b) Stereo Camera; (c) Omnidirectional Camera.

B. Indoor Positioning System

In researches discussing indoor positioning systems (IPS), there are few approaches classified based on their signal categories, such as radio frequency (RF), light, sound, and magnetic field [21]. For example, one of these methods based on radio frequency signals is Wi-Fi connection [22], [23], Bluetooth connections [24], and ZigBee protocol [25]. On the IPS using Wi-Fi signals, there are two available methods, *received signal strength (RSS) measurement* and *time and space attributes of the received signal (TSARS)* [26]. For example, usually, there are more than two hotspots (access points) as the anchor of RSS measurement (shown in Fig.2). This technique would require *trilateration* or a *multilateration* technique.

C. System Modelling

At last, the designed configuration is adopting and modifying the other researches before. The stereo camera configuration with different camera positions showing an interesting point. The Dual-Sight Perspective idea came out by reviewing the camera's position while every camera is mounted on a robot and intentionally moves around as the omnidirectional camera mounted on soccer robots.

Then, the indoor position system gives an interesting point too. The RSS measurement result is obtained on some specific distance value. Finding the equal function is calculated using the regression method, so the signal strength measured next would be calibrated using the generated function [27]. The alpha distance value is obtained in an image form for the Dual-Sight Perspective design, so the alpha value is in pixel value. Then the pixel value could be calibrated into real distance value in meter.

At last, dual sight perspective configuration is a kind of combined method between stereo vision and indoor positioning system (Fig.3). First, the camera model used in stereo vision is changed into omnidirectional camera configuration and the camera position would be various since it is mounted on each robot. Then, the acquisition data system adopts the IPS method while the collected data is the object and robot distance in pixel unit inside the image captured.

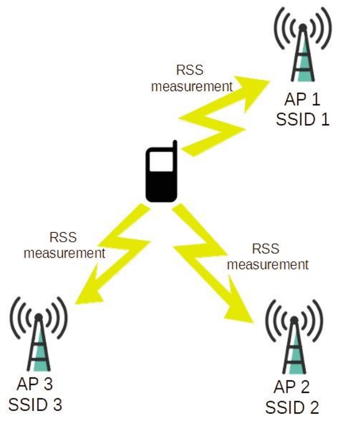

Fig.2. RSS Measurement-based Indoor Positioning System

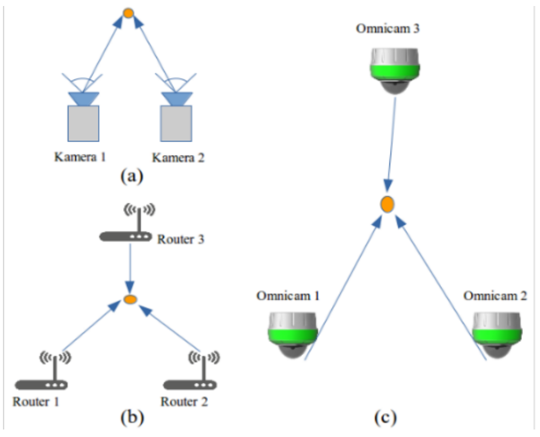

Fig.3. (a) Stereo Vision; (b) IPS; (c) Dual-Sight Perspective

D. Data Acquisition System

At a glance, the DSP configuration will look like the IPS system when it is online [22]. This is because all of the robots are doing a raw data collecting procedure. Every robot captures images with cameras (e.g., soccer ball) and calculates its pose and orientation on the field (see Fig.4). The collected data are shared into the ROS system that runs the main algorithm.

The ROS system consists of four support systems. There are Plugin, Data Pre-processor, Estimation Position algorithm, and Coordination system (not included in this research). The plugin works as a data buffer between ROS and Gazebo as this study uses both platforms. The plugin sends data from ROS to Gazebo and otherwise.

The data pre-processor block contains two algorithms, image processing, and the robot's pose parser algorithm. Image processing is a method to detect and recognize the estimated object. The image processing is divided into 3 sequences, HSV filtering, Morphology filtering, and Blob tracking algorithm. The output of image processing is detected coordinate object in the image captured known as centroid data *(pixel x, pixel y)*.

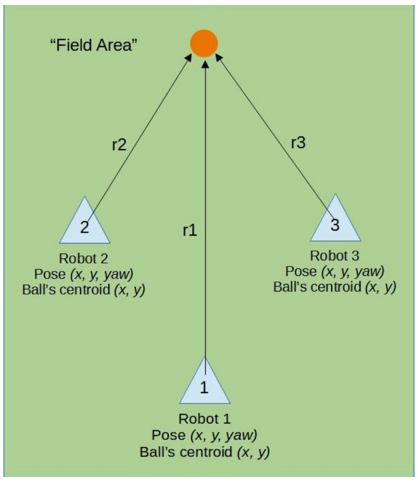

Fig.4. Data Acquisition Illustration

E. Simulation System

As the initial step, this system runs in a three-dimensional simulation system. Gazebo simulator is used to provide the world simulation and ROS to run the programs needed. It is required to build a 3D design of each element used for the simulation initially before proceeding to a simulation. [28].

A Gazebo and ROS simulation package designed for the Robocup MSL named *simatch* [29] is available as open-source, which the package is licensed under Apache 2.0 license. The package contains the robot's design, MSL field texture, goals, and the ball based on the Robocup MSL game (Fig.5). Unfortunately, the omnidirectional camera and robot's odometry system are not modeled inside the package [30]. Additional plugin (omnidirectional camera, and odometry system) is added inside the robot's model (Fig.6).

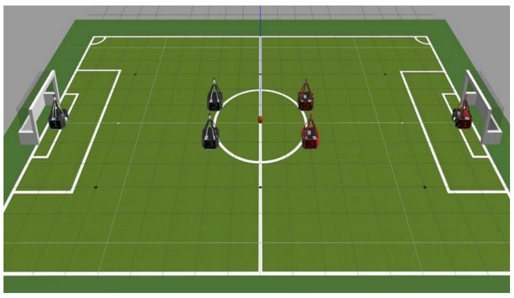

Fig.5. Simatch World Design

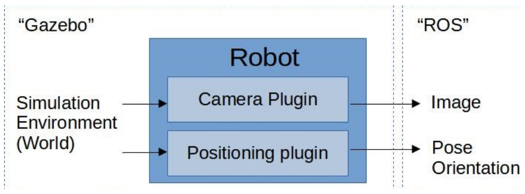

Fig.6. Sensor Block Diagram

Fig.6 shows the sensor diagram modeling. There are no physical sensors used. Instead, two plugins are used to take data from the simulation system. Camera plugin is a plugin for taking the image data inside the simulator. The output image is set at 1080x1080 pixels resolutions and up to 180 degrees of FOV. The positioning plugin takes the information about all object's positions and orientation data in Gazebo. According to represents specific data for each robot, a parser program is used, which is explained before.

Overall, based on the system modeling section in this part, every model is built in the ROS system. The ROS graph structure is shown in Fig.7. While the simulation system is running, the built ROS system also runs simultaneously. The programs containing every algorithm created are executed following ROS standard communication system (publish/subscribe topics) [31].

The acquired data is an image captured by the simulated camera on the robot. The detected range is until almost half of the field on one side. However, this plugin creates the ideal calibrated image. Fig.8 shows a sample image captured by a simulated camera. There are elements inside the image captured. The zero image point locates at the top left corner, where the image represents a matrix form [32]. The Center image point is the point that occurs at the center of the image, as the resolution is 1080x1080 pixels, so the center point is located at the coordinate (539, 539). The yellow dashed line represents the distance and direction between the ball and the center of the robot.

F. Computer Vision

The main objective in this point is to identify the specified object (ball). The input image is filtered by two methods, HSV filtering and Morphology Filtering. First, the image coloring format is converted from RGB into HSV format as follows [32]. Then, implementing a threshold value in the HSV formatted image, the HSV format image is filtered, and it focuses on the ball's color in binary format like in [33]. The morphology filter intends to increase (dilate/open operation) or reduce (erosion/close operations) the selected structure area as follows in [32]. The open operation is chosen for the morphology filter method in this study. If the robot can identify the ball, then the ball's position to the robot could be estimated.

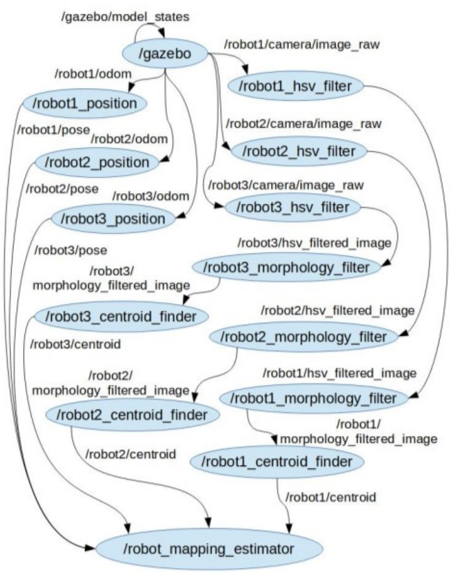

Fig.7. ROS Graph Structure

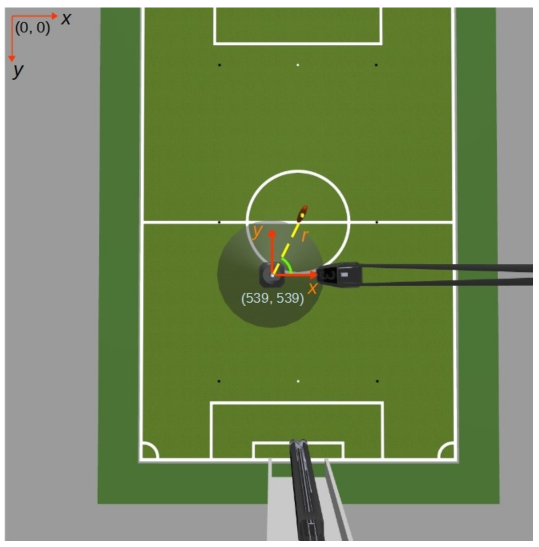

Fig.8. Example Image Captured by Robot 2

The result of the filtering process is a morphology filtered binary image. Then the blob tracking is done using this image as input. Blob tracking is a method to detect a selected area depending on information surroundings. Then, the blob result is taken to the center point of the selected area (called "centroid") to be processed using the moments' equation. Equations (1) and (2) can be used to determine the centroid position as follows [32]. The output value is in Cartesian coordinate format (C_x, C_y) and pixel unit (see Fig.9).

G. Pixel to Meter Calibrator Function

This process uses the blob tracking algorithm. The algorithm is only to find the position of the centroid position. So, the centroid data is sampled by moving the ball in front of the robot per 0.1 meters. The ball's initial position is 0.4 meters from the robot's middle point. This sampling task is done while the distance between the robot and the ball is 9 meters (Fig.11). To acquire the ball's centroid and robot's middle distance see dashed line in Fig.8, the modified Pythagoras method is used (Euclidean distance) as follows [34]. The formula is shown as (3). In comparison, (4) is the generated function of distributed data in Fig.10 with the linear regression analysis method.

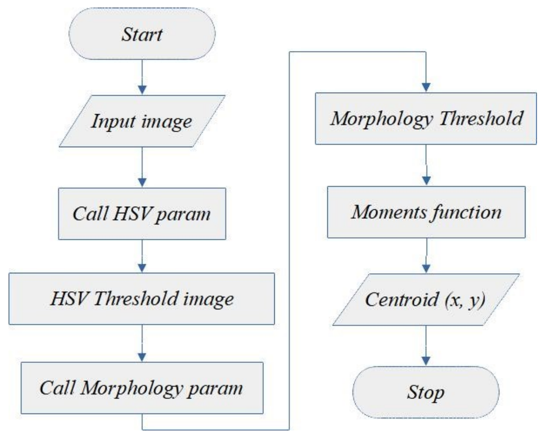

Fig.9. Blob Tracking Algorithm

$$C_x = \frac{M_{10}}{M_{00}} \tag{1}$$

$$C_y = \frac{M_{01}}{M_{00}} \tag{2}$$

$$r = \sqrt{(C_x - 539)^2 + (Cy - 539)^2}$$
$$r = \sqrt{\Delta Cx^2 + \Delta Cy^2} \tag{3}$$

$$d = 0.0163 * r + 0.2786 \tag{4}$$

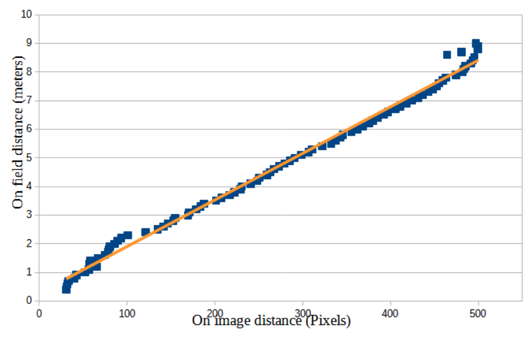

Fig.10. The Sampled Distance Data

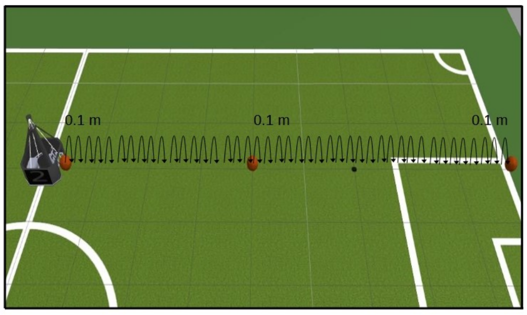

Fig.11. Distance Sampling to Produce Equation (4)

H. Quaternion to Euler Converter Algorithm

This part is intended as a pose data collecting algorithm and data converting algorithm because the robot's yaw angle data is required to estimate the ball's position. The Gazebo system provides the positioning data in Quaternion format (*x, y, w, z*), while the data is required in Euler format (*roll, pitch, yaw*). There is no specific function Quaternion to Euler conversion function used. Instead of using the "getRPY()" function, convert a Quaternion format to an Euler format. The pseudo-code below shows the Quaternion-to-Euler conversion algorithm.

Pseudo-code 1: Quaternion-Euler Converter

```
Function Main
  Input gazeboposedata
  Declare Real Array pose2d[3]
  Assign pose2d[0] = gazeboposedata[0]
  Assign pose2d[1] = gazeboposedata[1]
  Declare Real Array q[4]
  Declare Real Array m(q)[3][3]
  Declare Real roll, pitch, yaw
  Assign m = getRPY(roll, pitch, yaw)
  Assign pose2d[3] = yaw
  Output pose2d
End
```

I. Estimation Function

Figure 8 is an example captured image when the omnidirectional camera is in an active state. The image's initial point (0, 0) is located at the top-left corner because an image is equivalent to be a matrix, as mentioned in [32]. The *referring point* is located at the center of the image located at the coordinate of (539, 539). The point is used as the reference point of the ball to a robot's distance to estimate the point's calculation.

Referring to Fig.8, the dashed line's length is an important value. First, the length is calculated using (3). Then, the ball's angle value towards the x refers point axis is calculated. Finally, the angle is determined using the trigonometry invers method [35], such as (5). The angle value is written in radian value, so the degree value input must be converted to a radian value.

$$\alpha = \operatorname{acos}\left(\frac{C_x}{r}\right) \tag{5}$$

There are two conditions in this process. First, when the C_y value is below the y value center of the image variable ($C_y \leq 539$). Second, when C_y value is above y value center of image variable ($C_y > 539$). Both of the condition gives two different results using if are calculated using (5). Equation (6) is provided to encounter this problem. The equation adds a rotation effect. When the condition occurs like the second condition, the result deviates more with a doubled real distance between ball and robot. The dtr Symbol is a variable containing the value of 0.017444 (or 3.14/180). This value will convert the angle in degree value to be a radian value. The condition mentioned above is described in Fig.12.

$$\alpha = \left((360 \cdot dtr) - \operatorname{acos}\left(\frac{C_x}{r}\right)\right) \tag{6}$$

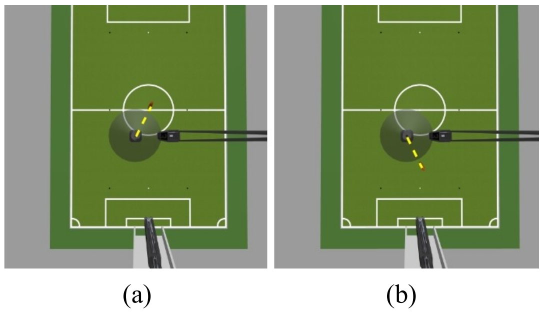

Fig.12. Illustration of Both Cy Conditions; (a) $C_y \leq 539$, (b) $C_y > 539$.

The next step is to determine the amount of influence caused by the robot's direction. The process is described in (7) and (8). The result value is a degree value between the robot's headings and the ball's direction to the robot. As the change of the ball's direction, the image's centroid point also changes. Equations (9) and (10) to calculate the new centroid point can be used for each x and y coordinate value.

$$\alpha_0 = 90 * dtr - yaw \tag{7}$$

$$\alpha_1 = \alpha - \alpha_0 \tag{8}$$

$$C_{x1} = r.\cos\alpha_1 \tag{9}$$

$$C_{y1} = r.\sin\alpha_1 \tag{10}$$

After that, calibrating the new centroid point by using (4). The result of calibration is moved to the variable (d_x, d_y) while the centroid point (C_{x1}, C_{y1}) is substituting the x value in (4). The transformed equation is shown as (11) and (12).

$$d_x = 0.0163 \cdot C_{x1} + 0.2786 \tag{11}$$

$$d_y = 0.0163 \cdot C_{y1} + 0.2786 \tag{12}$$

The result of (11) and (12) is a delta distance value in the meter. So the distance in pixel before (C_{x1}, C_{x2}) has been calibrated into the real distance in meter. The last, counting the ball's position using (13) for x element and (14) for y components.

$$P_{b_x} = d_x + P_{r_x} \tag{13}$$

$$P_{b_y} = d_y + P_{r_y} \tag{14}$$

On (13) and (14), P_r means robot position on the field. The Pr_x and Pr_y elements represent each coordinate elements, Pr_{th} is the element of the robot's direction written in radian value. At last, the result of (13) and (14) is the ball's estimated position. The value is written in meter. By combining all of the equation, the final result will be:

a) Condition 1, $C_y \leq 539$

$$Pb_x = \left(0.0163r * \cos\left(\text{acos}\frac{\Delta Cx}{r} - (90 * dtr - Pr_{th})\right) + 0.2786\right) + \mathrm{Pr_x} \tag{15}$$

$$Pb_x = \left(0.0163r * \cos\left(\text{acos}\frac{\Delta Cx}{r} - (90 * dtr - Pr_{th})\right) + 0.2786\right) + \mathrm{Pr_y} \tag{16}$$

b) Condition 2, $C_y > 539$

$$Pb_x = \left(0.0163r * \cos\left((360 * \mathrm{dtr}) - (\text{acos}\frac{\Delta Cx}{r} - (90 * dtr - Pr_{th}))\right) + 0.2786\right) + \mathrm{Pr_x} \tag{17}$$

$$Pb_y = \left(0.0163r * \cos\left((360 * \mathrm{dtr}) - (\text{acos}\frac{\Delta Cx}{r} - (90 * dtr - Pr_{th}))\right) + 0.2786\right) + \mathrm{Pr_y} \tag{18}$$

J. Testing Scenarios

At last, the system has been tested under four different scenarios. The first scenario is illustrated in Fig.13, the second scenario is illustrated in Fig.14, the third scenario is illustrated in Fig.15, the last scenario is illustrated in Fig.16. The black hairline inside those figures is illustrating a 1-meter distance on the x or y-axis.

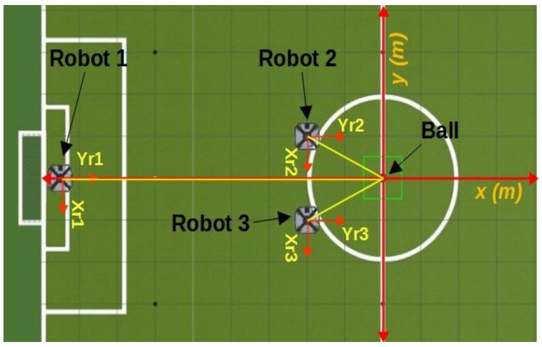

Fig.13.Illustration of Scenario 1.

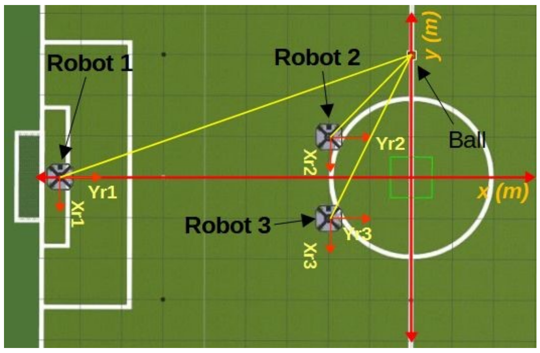

Fig.14. Illustration of Scenario 2.

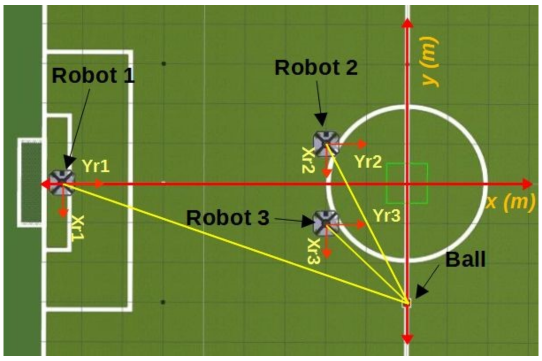

Fig.15. Illustration of Scenario 3.

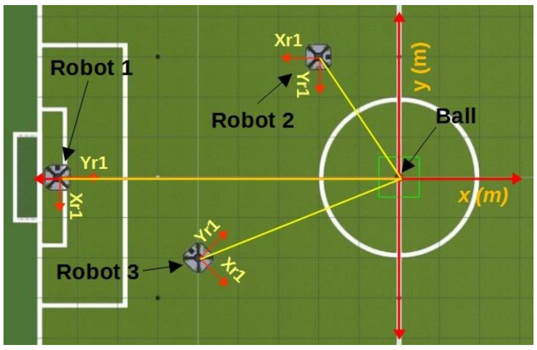

Fig.16. Illustration of Scenario 4.

In the first scenario, robot 1's orientation is located (-8.5, 0, 0). Robot 2's orientation is (-2, 1, 0) and the robot 3's orientation is (-2, -1, 0). Then, the ball is located on the center of the field representing the zero coordinate or (0, 0). All of the robots face the same direction (to the right side of the picture shown by "Yr" annotation in Fig.13). The pose value above represents the *x* and *y* axes plus the robot's heading. The *x* and *y* elements represent the coordinate of the robots locating on the map (field). The *yaw* element shows the value of which degree the robots are facing. For the second and the third scenario, all of the robots are locating in the same initial poses. The only difference is that the ball is moved around the y axis with the same coordinate and headings. Therefore, the ball is posed at (0, 3) for the second scenario. In contrast, the ball is posed at a coordinate of (0, -3) for the third scenario. Finally, the robots are posed in different coordinates and heading courses except for robot 1 because robot 1 plays the role of the goalkeeper for the last one. In summary, robot 2 is located on (-2, 3) with the heading angle is -90 degrees, and robot 3 is located on (-5, -2) with a heading angle is 45 degrees.

K. Estimation Coordinates Errors

The estimation coordinates are compared with the actual ball's coordinate to get the estimation error (deviation) values. The Root Mean Square Error (RMSE) to find the deviation errors is shown in (21). Suppose that (P_{bx}, P_{by}) is the estimated values and (P_{rx}, P_{ry}) is the actual coordinate. Then, M is the number of estimation points taken as the number of scenarios that follow [27].

$$E = \sqrt{\frac{1}{M}\sum_{M=1}^{M}(P_{rx}-P_{bx})^2 + (P_{ry}-P_{by})^2} \quad (19)$$

III. RESULTS

A. Acquired Data

Figure 17 until Fig.20 show the illustration of the estimated ball's coordinate, respectively. The graph illustrates the viewpoint of the field used in the simulation. The maximum x and y have been shrunk to a smaller resolution according to make the deviation is could be recognized well. Every dots inside the image are the coordinate point due to the estimation coordinate system on the field. The points displayed in the graph are the average value of each robot's sample that has been calculated before. As an example, the obvious blue dot in Fig.17 is the average value from robot 1's sampled object's position data.

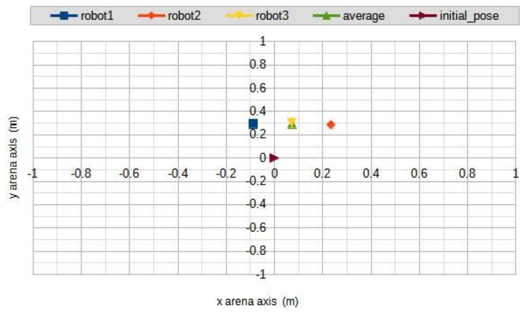

Fig.17. Ball Estimation Map for Scenario 1.

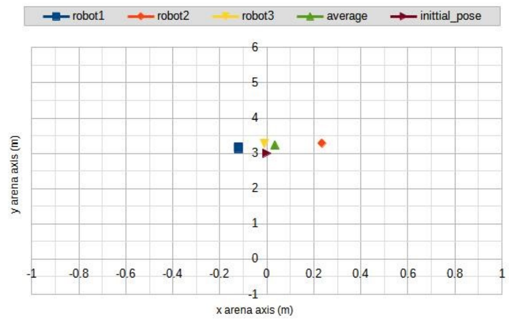

Fig.18. Ball Estimation Map for Scenario 2.

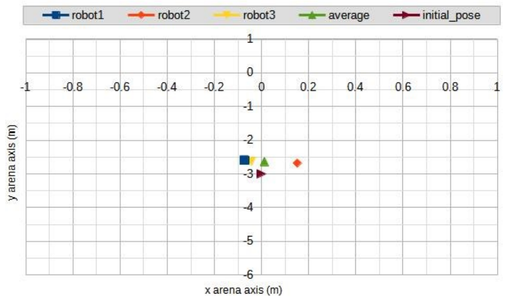

Fig.19. Ball Estimation Map for Scenario 3.

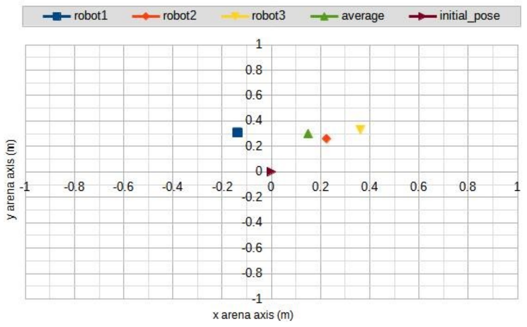

Fig.20. Ball Estimation Map for Scenario 4.

Because the robots detect and calculate the estimation separately, the estimation results will produce three different points. Of course, this causes differences in the estimation results. Therefore, to produce a uniform point, it is necessary to calculate the average of each element from the ball's coordinate point. The final result is determined by calculating the average values from all of the estimated results using (20) and (21). The average results are already shown in Fig.17 until Fig.20 and Table 1.

$$B_x = \frac{\Sigma P_{bx}}{\Sigma i} \quad (20)$$

$$B_y = \frac{\Sigma P_{by}}{\Sigma i} \quad (21)$$

Table 1. Average Estimation Results

Test	Ball's Coordinate				Square Error	RMSE
	X_0	Y_0	X_1	Y_1		
1	0	0	0.07279	0.2948	0.294	0.383
2	0	3	0.04137	3.2447	0.061	
3	0	-3	0.01252	-2.652	0.121	
4	0	0	0.149655	0.2995	0.112	

IV. DISCUSSION

The omnidirectional camera on the soccer robots usually uses the reflective convex mirror, as shown in Fig.1c. Therefore, the captured image will have a distortion effect and need calibration. However, implementing the distortion and calibration on camera simulation burdens the computer that is running the simulation. Therefore, according to a fine simulation, the distortion and image calibration is skipped in this study. As a result, the captured image looks like in Fig.8, then considered as a calibrated image and ready to be processed.

Figure 10 shows the data comparison between the distance values in pixel into the real distance in meters between robots to the ball. The graph shows the linear function written as (4). By implementing a linear analysis, the R squared value is 0.9932. This result only applies to this study because this study uses a simulation system that every condition can be considered a perfect condition.

Consider that a robot is located at the coordinate of (-8.5, 0), a heading angle of about 360 degrees. In contrast, the ball is located at (0, 0) and does not have the heading angle. Last, the ball is located on the image's coordinate at (539, 40). In a word, these data are shown in Table 2.

Table 2. Example of the Captured Data

Robot	Robot's Pose			Ball's pos. (m)		Image (pixel)	
	X	Y	Th	X	Y	X	Y
Robot 1	-8.5	0	360			539	40
Robot 2	-2	1	360	0	0	600	420
Robot 3	-2	-1	360			475	453

According to the data in Table 2 above, the best equation to be used is (15-16). First, the ball is detected on the image at coordinate (539, 40) pixels for Robot 1. Then, it is known that the robot's center point is located at (539, 539) of the image as follows these calculations.

$$r = \sqrt{(539-539)^2 + (40-539)^2}$$
$$= \sqrt{0^2 + (-499)^2}$$
$$= 499$$

$$Pb_x = (0.0163 \cdot 499 \cdot \cos\left(acos\frac{0}{499} - ((90 \cdot 0.0174) - 3.14)\right)$$
$$+0.2786) + (-8.5)$$

$$= \left(8.134 \cdot \cos\left(1.57 - (1.57 - 3.14)\right) + 0.2786\right) +$$
$$(-8.5)$$
$$= (8.134\ \cdot \cos(0.001) + 0.2786) + (-8.5)$$
$$= (8.134\ \cdot (0.999) + 0.2786) + (-8.5)$$
$$= (8.126 + 0.2786) + (-8.5)$$
$$= (8.4046) + (-8.5)$$
$$= -0.095$$

$$Pb_y = (0.0163 \cdot 499 \cdot \sin\left(acos\frac{0}{499} - ((90 \cdot 0.174) - 3.14)\right)$$
$$+0.2786) + 0$$

$$= \left(8.134 \cdot \sin\left(1.57 - (1.57 - 3.14)\right) + 0.2786\right)$$
$$+0$$
$$= (8.134\ \cdot \sin(0) + 0.2786) + 0$$
$$= (8.134\ \cdot 0 + 0.2786) + 0$$
$$= (0 + 0.2786) + 0$$
$$= 0.2786 + 0$$
$$= 0.278$$

Robot 1 predicted that the ball is located at (-0.0954, 0.2786) coordinate based on the calculation above. Besides, the prediction results by Robots 2 and 3 are shown in Table 3. The average result of the predicted ball's coordinate is (0, 0.294). The average value is compared to the actual ball's position, i.e. (0, 0). The squared error will be calculated as below.

$$Squared\ Error = (0-0)^2 + (0.294-0)^2$$
$$= 0.294\ m$$

Table 3. Example of Result Data

Robot	Ball's Coordinate		Average	
	X	Y	X	Y
Robot 1	-0.095	0.278		
Robot 2	0.234	0.284	0	0.294
Robot 3	-0.139	0.321		

The estimated ball's positions number 2 to 4 in Table 1 are calculated in the same way. The different SE values in Table 1 are caused by the

$$RMSE = \sqrt{\frac{1}{M}\sum_{m=1}^{m} SE_m} = \sqrt{\frac{1}{4}0.588} = 0.383$$

Then, all of SE in Table 1 was used to calculate the root mean square error (19). The root means square error from 4 testing scenarios in this study is 0.383 meters. Therefore, it can be concluded that the prediction of the object's coordinate position is fairly accurate. So, this configuration could be considered to be used in a lower computational capability system to estimate an object's position.

V. CONCLUSION

The output of this system is the object's coordinate written as (*x, y*). The dual sight perspective configuration can predict the object's position with the estimated error of 0.383 meters. While the R squared value of distance in the image to real simulated distance value is noted as 0.9932. Furthermore, then the corresponding result is obtained in the realm of simulation. However, the dual sight perspective configuration still needs to be developed in other ways and be tested on the real-world applications.

REFERENCES

[1] P. Long, T. Fanl, X. Liao, W. Liu, H. Zhang, dan J. Pan, "Towards Optimally Decentralized Multi-Robot Collision Avoidance via Deep

Reinforcement Learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Mei 2018, pp. 6252–6259, doi: 10.1109/ICRA.2018.8461113.

[2] H. Lu, H. Zhang, S. Yang, dan Z. Zheng, "Vision-based ball recognition for soccer robots without color classification," in *2009 IEEE International Conference on Information and Automation, ICIA 2009*, 2009, no. December 2013, pp. 916–921, doi: 10.1109/ICINFA.2009.5205049.

[3] Y. Prakoso, "Desain dan Implementasi Pengukuran Posisi Bola Menggunakan Kamera 360 Derajat Pada Robot Sepak Bola," 2017.

[4] D. Fox, W. Burgard, F. Dellaert, dan S. Thrun, "Monte Carlo Localization: efficient position estimation for mobile robots," in *Proc. Natl. Conf. Artif. Intell.*, no. Handschin 1970, hal. 343–349, 1999.

[5] A. C. Almeida, S. R. J. Neto, dan R. R. A. Bianchi, "Comparing vision-based monte-carlo localization methods," in *Proceedings - 15th Latin American Robotics Symposium, 6th Brazilian Robotics Symposium and 9th Workshop on Robotics in Education, LARS/SBR/WRE 2018*, 2018, pp. 408–412, doi: 10.1109/LARS/SBR/WRE.2018.00084.

[6] W. Xiaoyu, L. Caihong, S. Li, Z. Ning, dan F. U. Hao, "On adaptive monte carlo localization algorithm for the mobile robot based on ROS," in *Chinese Control Conference, CCC*, 2018, vol. 2018-July, pp. 5207–5212, doi: 10.23919/ChiCC.2018.8482698.

[7] D. Talwar dan S. Jung, "Particle Filter-based Localization of a Mobile Robot by Using a Single Lidar Sensor under SLAM in ROS Environment," *Int. Conf. Control. Autom. Syst.*, vol. 2019-Octob, no. Iccas, pp. 1112–1115, 2019, doi: 10.23919/ICCAS47443.2019.8971555.

[8] X. Chen *et al.*, "OverlapNet: Loop Closing for LiDAR-based SLAM," no. i, 2020, doi: 10.15607/rss.2020.xvi.009.

[9] X. Wang, B. Zhou, J. Ji, dan B. Pu, "Recognition and distance estimation of an irregular object in package sorting line based on monocular vision," *Int. J. Adv. Robot. Syst.*, vol. 16, no. 1, pp. 1–12, 2019, doi: 10.1177/1729881419827215.

[10] R. A. Setyawan, R. Sunoko, M. A. Choiron, dan P. M. Rahardjo, "Implementation of stereo vision semi-global block matching methods for distance measurement," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 12, no. 2, pp. 585–591, 2018, doi: 10.11591/ijeecs.v12.i2.pp585-591.

[11] N. Setyawan, N. Mardiyah, K. Hidayat, Nurhadi, dan Z. Has, "Object detection of omnidirectional vision using PSO-neural network for soccer robot," *Int. Conf. Electr. Eng. Comput. Sci. Informatics*, vol. 2018-Octob, pp. 117–121, 2018, doi: 10.1109/EECSI.2018.8752833.

[12] E. M. Pamungkas, B. A. A. Sumbodo, dan I. Candradewi, "Sistem Pendeteksi dan Pelacakan Bola dengan Metode Hough Circle Transform, Blob Detection, dan Camshift Menggunakan AR.Drone," *IJEIS (Indonesian J. Electron. Instrum. Syst.*, vol. 7, no. 1, pp. 1, 2017, doi: 10.22146/ijeis.15405.

[13] T. Manda, A. Triyono, H. Fitriyah, M. Hannats, dan H. Ichsan, "Deteksi Jarak Bola Pada Robot Kiper Sepak Bola Menggunakan Hough Circle Transformation Berbasis Raspberry Pi," *J. Pengemb. Teknol. Inf. dan Ilmu Komput. Univ. Brawijaya*, vol. 3, no. 2, pp. 8937–8943, 2019.

[14] H. K. Kim, J. H. Park, dan H. Y. Jung, "An Efficient Color Space for Deep-Learning Based Traffic Light Recognition," *J. Adv. Transp.*, vol. 2018, pp. 1-12, 2018, doi: 10.1155/2018/2365414.

[15] M. E. Ashoori dan M. Mahlouji, "Measuring the Distance between the Two Vehicles Using Stereo Vision with Optical Axes Cross," *Mod. Appl. Sci.*, vol. 12, no. 1, pp. 165, 2017, doi: 10.5539/mas.v12n1p165.

[16] P. Li, T. Qin, dan S. Shen, "Stereo Vision-Based Semantic 3D Object and Ego-Motion Tracking for Autonomous Driving," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11206 LNCS, pp. 664–679, 2018, doi: 10.1007/978-3-030-01216-8_40.

[17] V. Tuominen, "The measurement-aided welding cell—giving sight to the blind," *Int. J. Adv. Manuf. Technol.*, vol. 86, no. 1–4, pp. 371–386, 2016, doi: 10.1007/s00170-015-8193-9.

[18] H. Deng, Q. Fu, Q. Quan, K. Yang, dan K. Y. Cai, "Indoor Multi-Camera-Based Testbed for 3-D Tracking and Control of UAVs," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 6, pp. 3139–3156, 2020, doi: 10.1109/TIM.2019.2928615.

[19] Y. N. Naggar, A. H. Kassem, dan M. S. Bayoumi, "A Low Cost Indoor Positioning System Using Computer Vision," *Int. J. Image, Graph. Signal Process.*, vol. 11, no. 4, pp. 8–25, 2019, doi: 10.5815/ijigsp.2019.04.02.

[20] D. Scaramuzza, "Omnidirectional Camera," in *Computer Vision*, 2014, hal. 552–560.

[21] R. F. Brena, J. P. García-Vázquez, C. E. Galván-Tejada, D. Muñoz-Rodriguez, C. Vargas-Rosales, dan J. Fangmeyer, "Evolution of Indoor Positioning Technologies: A Survey," *J. Sensors*,

vol. 2017, pp.1-21, 2017, doi: 10.1155/2017/2630413.

[22] I. Riyanto dan Y. M. Akbar, "Local Area Positioning System (LAPS) for indoor navigation and tracking system and building electricity energy saving," *Int. J. Simul. Syst. Sci. Technol.*, vol. 17, no. 32, pp. 1–7, 2016, doi: 10.5013/IJSSST.a.17.32.10.

[23] J. Duque Domingo, C. Cerrada, E. Valero, dan J. A. Cerrada, "Indoor positioning system using depth maps and wireless networks," *J. Sensors*, vol. 2016, pp.1-8, 2016, doi: 10.1155/2016/2107872.

[24] V. Cantón Paterna, A. Calveras Augé, J. Paradells Aspas, dan M. A. Pérez Bullones, "A Bluetooth Low Energy Indoor Positioning System with Channel Diversity, Weighted Trilateration and Kalman Filtering," *Sensors (Basel).*, vol. 17, no. 12, 2017, doi: 10.3390/s17122927.

[25] M. Uradzinski, H. Guo, X. Liu, dan M. Yu, "Advanced Indoor Positioning Using Zigbee Wireless Technology," *Wirel. Pers. Commun.*, vol. 97, no. 4, pp.6509–6518, 2017, doi: 10.1007/s11277-017-4852-5.

[26] S. Xia, Y. Liu, G. Yuan, M. Zhu, dan Z. Wang, "Indoor fingerprint positioning based on Wi-Fi: An overview," *ISPRS Int. J. Geo-Information*, vol. 6, no. 5, 2017, doi: 10.3390/ijgi6050135.

[27] H. Pujiharsono, D. Utami, dan R. D. Ainul, "Trilateration Method For Estimating Location in RSSI-Based Indoor Positioning System Using Zigbee Protocol," *J. Infotel*, vol. 12, no. 1, pp. 8–13, 2020, doi: 10.20895/infotel.v12i1.380.

[28] A. Jalil, "Robot Operating System (ROS) dan Gazebo Sebagai Media Pembelajaran Robot Interaktif," *Ilk. J. Ilm.*, vol. 10, no. 3, pp.284–289, 2018, doi: 10.33096/ilkom.v10i3.365.284-289.

[29] Z. Zhou, W. Yao, J. Ma, H. Lu, J. Xiao, dan Z. Zheng, "Simatch: A Simulation System for Highly Dynamic Confrontations between Multi-Robot Systems," in *Proceedings 2018 Chinese Automation Congress, CAC 2018*, 2019, pp.3934–3939, doi: 10.1109/CAC.2018.8623698.

[30] J. Xiao, D. Xiong, W. Yao, Q. Yu, H. Lu, dan Z. Zheng, "Building software system and simulation environment for RoboCup MSL Soccer robots based on ROS and Gazebo," in *Studies in Computational Intelligence*, vol. 707, Springer, 2017, pp. 597–631.

[31] J. M. O'Kane, *A gentle introduction to ROS*, no. 2.1.3. 2016.

[32] A. Kadir, *Langkah Mudah Pemrograman OpenCV & Python*. Jakarta: PT Elex Media Komputindo, 2019.

[33] R. Dikarinata, I. K. Wibowo, M. M. Bachtiar, dan M. A. Haq, "Searching Ball around ROI to Increase Computational Processing of Detection," *IES 2020 - Int. Electron. Symp. Role Auton. Intell. Syst. Hum. Life Comf.*, pp. 207–212, 2020, doi: 10.1109/IES50839.2020.9231903.

[34] K. Czyzewska, "GENERALIZATION OF THE PYTHAGOREAN THEOREM," *Demonstr. Math.*, vol. 24, no. 1–2, 2018, doi: 10.1515/dema-1991-1-228.

[35] M. Occhiogrosso, *Graphs of trigonometric functions*. Milliken Publishing Company, 2007.