JIRAE, Vol. 4, No. 2, October 2019, 57-62
e-ISSN 2407-7259
DOI: 10.9744/jirae.4.2.57-62

Omni-Directional Mobile Robot Control using Raspberry Pi and Jetson Nano

Evert Oneil[1,a], Indar Sugiarto[2,b]
[1,2] Electrical Engineering, Petra Christian University, Surabaya, Indonesia
[a]evertoneil1@gmail.com, [b]indar@ieee.org

Abstract. This paper describes a process of designing a low-cost robot that moves omni-directionally that capable of detecting simple objects around it while moving. The robot uses three DC motors with encoder as feedback. The overall movement control is done by a Raspberry Pi that is embedded into the robot. It also uses ROS (Robotic Operating System) as the framework. To detect certain objects, the robot is equipped with an infrared sensor for measuring the object distance, and a camera for capturing the image of the object that will be processed by a Nvidia Jetson Nano. By using inverse kinematics and odometry calculations, the robot can move smoothly with error at about 9.5% on the x-axis and 8.1% on the y-axis, measured at the robot's final position. The robot can detect objects using infrared sensors reliably with error at about 0.87%, however, it cannot measure the object size precisely due to various factors.

Keywords: Robot, omnidirectional, omniwheel, object detection, kinematics.

1. Introduction

Robot design and programming is a core subject in engineering that has rich aspects to be explored, especially for educational purposes. By self-developing robust but low-cost robots, students and researchers can sharpen their knowledges while exploring various novelty aspects in robotics. This is in line with one of the opinions of the Team ABI Research: "Flexibility and efficiency have become the main differentiators in a system, in order to cope with volatile product demand, seasonal peaks, and rising consumer shipping expectations" [1].

Seeing that flexibility and efficiency are important components in mobile robotics, a robot needs to be designed with capability of moving easily to specific locations with wider scope of space. For this reason, the robot needs to have good mobility and maneuverability, which depends on its wheel configuration.

One disadvantage of conventional wheels is that they are limited in motion; thus, the robot cannot move sideways without preliminary maneuvers. This problem can be overcome with special omni-directional type wheels that allow the robot to be driven in all directions.

In this paper, the process of building such a mobile robot is described. In the future, this robot can also be equipped with a robotic arm to create a mobile manipulator. For this purpose, the robot needs to know the distance of the object from the robot, along with the height and width information of the object so that the manipulator arm can grip object correctly.

This paper is organized as follows. In section 2, the design of the omni-directional mobile robot is described. In section 3, we evaluate the performance of the robot. Finally, in section 4, we conclude our work.

2. Research Methods

The following figure shows the logical connection between core components of our mobile robot.

As shown in Figure 1, the Raspberry Pi acts as a controller that will receive input from the user to determine the destination position and direction of motion of the robot.

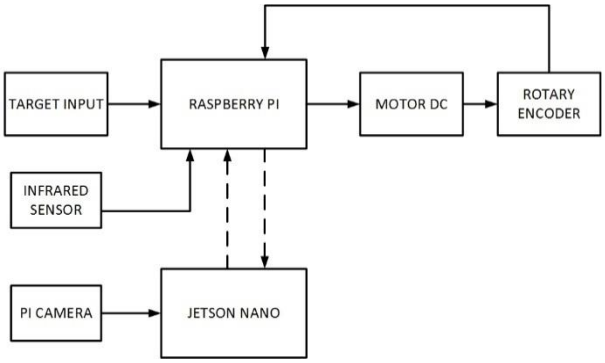

Figure 1. Block diagram of an omni-directional robot system

The Raspberry Pi will send data to drive a DC motor, and the encoder will read the wheel's speed and make it as feedback so that the Raspberry Pi can determine the robot's mileage.

Infrared sensor input is used to detect the distance from the robot and the destination objects. The output from the camera will be processed by Jetson Nano. Camera image processing is carried out at Jetson Nano to read the height and width of the detected object, then the reading data will be sent to the Raspberry Pi. Both microcontrollers (Raspberry Pi and Jetson Nano) are connected on the same network and use ROS for communication between controllers.

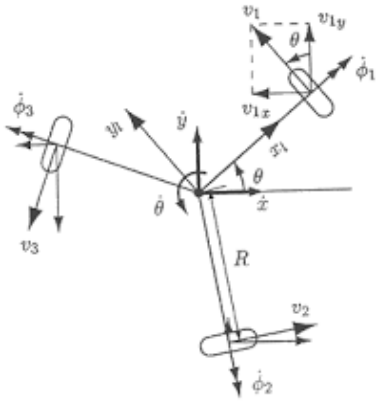

Figure 2. Three wheeled robot's kinematics

Figure 2 shows the basic inverse kinematics calculation of an omni-directional 3-wheel robot system [2]. And the final equation can be written as follows:

$$\begin{aligned}\dot{\phi}_1 &= (-\sin(\theta+\alpha_1)\cos(\theta)\dot{x}_L + \cos(\theta+\alpha_1)\cos(\theta)\dot{y}_L + R\dot{\theta})/r \\ \dot{\phi}_2 &= (-\sin(\theta+\alpha_2)\cos(\theta)\dot{x}_L + \cos(\theta+\alpha_2)\cos(\theta)\dot{y}_L + R\dot{\theta})/r \\ \dot{\phi}_3 &= (-\sin(\theta+\alpha_3)\cos(\theta)\dot{x}_L + \cos(\theta+\alpha_3)\cos(\theta)\dot{y}_L + R\dot{\theta})/r\end{aligned} \quad (1)$$

With the description of several variables according to this study:

α_1 is the angle between the x-axis and wheel 1, 270^O
α_2 is the angle between the x axis and wheel 2, 30^O
α_3 is the angle between the x axis and wheel 3 ,150^O
R is the distance of the wheel to the center point, which is 0.12 m.
r is the radius of the wheel, which is 0.0246 m
θ is the initial angle of the robot, which is 0

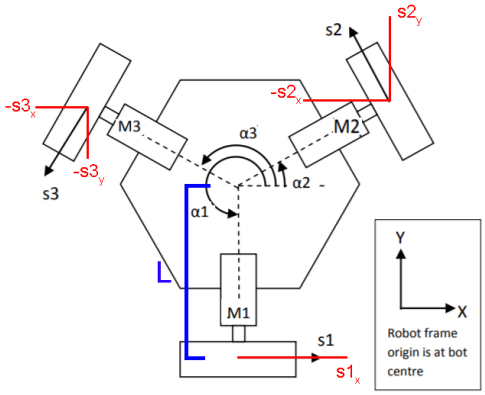

Figure 3. Odometry calculation

$$\begin{bmatrix}\phi 1\\ \phi 2\\ \phi 3\end{bmatrix} = \begin{bmatrix}2/3 & 0 & 1/3\\ 0 & 1/\sqrt{3} & 1/3\\ -1/3 & -1/\sqrt{3} & 1/3\end{bmatrix}\begin{bmatrix}x/r\\ y/r\\ \theta.3L/r\end{bmatrix} \quad (2)$$

This equation is implemented in the program to determine each rotation angle needed to input the intended x, y, θ position [3]. The units x and y are in meters, while θ is in radians. Units ϕ1, ϕ2, ϕ3 in radians.

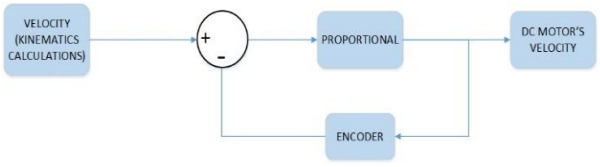

Figure 4. Motor's speed control diagram

Figure 4 is a motor speed control design where the motor speed's value as results of inverse kinematics calculation will be controlled with proportional control, using an encoder as feedback that reads current speed.

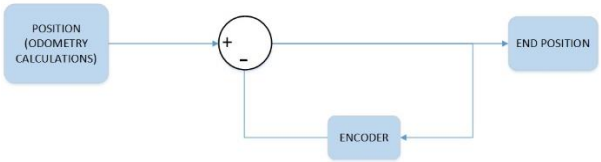

Figure 5. Robot's movement diagam

To regulate the movement of the robot, the encoder acts as feedback to estimate the movement of the robot. So once the robot has reached the target position, then the motor will immediately stop.

To detect the distance between the robot and the object , it is used an infrared sensor. The **SHARP GP2Y0A21** sensor used can read a range of 10 - 80 cm. Calibration is performed to determine the conversion equation from the sensor, the calibration process is done using a black rectangular image measuring 12 cm x 12 cm. The regression equation is:

$$y = 12320\ x\text{-}1{,}073 \quad (3)$$

Where 'x' is the reading number from the sensor and 'y' is the reading value of the distance of the sensor to the object in centimeters.

To detect the size of the object in front of the camera, an object reading program is required by the camera. The camera used is the **Raspberry v2 camera** module which will be processed using openCV.

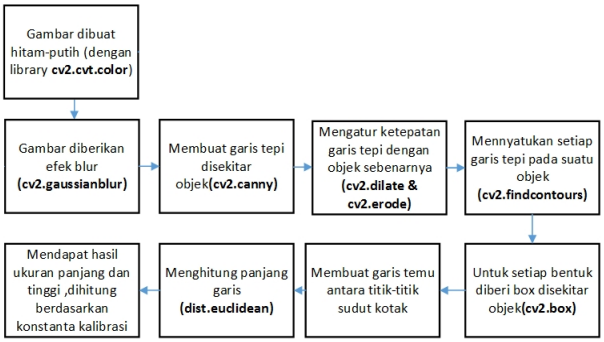

Figure 6. The process of object sizes measurements

To calculate the size of an object that is read, it needs to determine the ratio between the pixels read in the image and the distance of the camera to the object [4]. First it should be 'calibrated' with a reference object first. The reference object must be known for its original size.The calibration result is:

$$y = 661.84x\text{-}1{,}002 \quad (4)$$

Where 'x' is the distance that is read and 'y' is a dividing constant to calculate the size of objects. Next for the implementation of the program are: Actual size = pixel distance / constant.

3. Result and Discussion

3.1 Motor's characteristic test

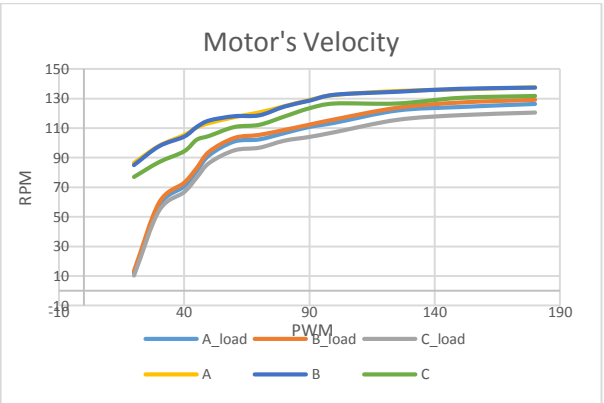

Figure 7. Motor's velocity readings with encoder.

The graph shown in **Figure 7** shows the difference in motor speed at no load and under load conditions. After obtaining characteristic data from each motor, a regression equation can be made to convert the desired speed numbers to the PWM numbers needed for each motor. Through the table data above, the regression equation is obtained using Microsoft Excel:

$$PWM\_a = -4E\text{-}10y6 + 2E\text{-}07y5 - 4E\text{-}05y4 + 0.0043y3 - 0.2179y2 + 6.0869y - 42.92 \quad (5)$$
$$y = 7E\text{-}06v4 - 0.0016v3 + 0.1341v2 - 4.0937v + 52.613$$

$$PWM\_b = 9E\text{-}08y5 - 3E\text{-}05y4 + 0.0032y3 - 0.1748y2 + 4.2613y - 12.657 \quad (6)$$
$$y = -3E\text{-}10v6 + 2E\text{-}07v5 - 4E\text{-}05v4 + 0.0049v3 - 0.2711v2 + 7.9921v - 65.117$$

$$PWM\_c = 1E\text{-}07y5 - 3E\text{-}05y4 + 0.0036y3 - 0.1794y2 + 3.9339y - 5.0599 \quad (7)$$
$$y = -3E\text{-}10v6 + 2E\text{-}07v5 - 5E\text{-}05v4 + 0.0052v3 - 0.2882v2 + 8.468v - 69.855$$

Equations (5), (6), (7) are regression equations where v is the value of velocity required for each motor and PWM_a, PWM_b, PWM_c is the PWM value given to the motor.

To be able to reach the speed of each motor more accurately, a proportional controller is used, so the determination of the motor rotational speed is closer to the target. In this system, using proportional controller of 0.7, obtained from testing with trial and error methods.

3.2 Kinematics and odometry calculation test

Table 1. Final Position test

Input[cm]			End [cm]		Error[cm]	
X	Y	Time[s]	X	Y	x	y
0	40	4	8	40	8	0
0	80	4	14	78	14	2
0	120	4	7	125	7	5
40	0	4	38	16	2	16
80	0	4	86	3	6	3
120	0	4	135	5	15	5
0	-40	4	-7	-35	7	5
0	-80	4	-20	-77	20	3
0	-120	4	-15	-116	15	4
-40	0	4	-38	-15	2	15
-80	0	4	-82	3	2	3
-120	0	4	-131	-20	11	20
Average Error					9.08	6.75

For diagonal robot movements, there is an average error of 9.51% on the x-axis and 8.12% on the y coordinate. All robots are set to reach the target position within 4 seconds, except the target position x = 120, y = 120 and x = -120, y = -120 where the robot's movement time is set at 5 seconds because the motor 3's speed has exceeded the maximum (for 4 seconds). Looking at the data shown in **Table 2**, if the final position is small (in this test x = ± 40, y = ± 40), it causes an error that tends to be greater, because the required motor speed is below 20 rpm so that proportional control has difficulty reaching that point.

Table 2. Final Position test (diagonal)

Input[cm]			End [cm]		Error [%]		
X	Y	T [s]	X	Y	x	y	Avrg
40	40	4	35	38	12.5	5	8.7
80	80	4	73	65	8.7	18.7	13.7
120	120	5	125	102	4.1	15	9.5
					0	0	0
40	-40	4	38	-37	5	7.5	6.2
80	-80	4	72	-75	10	6.2	8.1
120	-120	4	115	-110	4.1	8.3	6.2
					0	0	0
-40	40	4	-33	37	17.5	7.5	12.5
-80	80	4	-70	74	12.5	7.5	10
-120	120	4	-110	111	8.3	7.5	7.91
					0	0	0
-40	-40	4	-35	-36	12.5	10	11.2
-80	-80	4	-73	-78	8.75	2.5	5.6
-120	-120	6	-108	-118	10	1.67	5.83
Average error					9.51	8.12	8.81

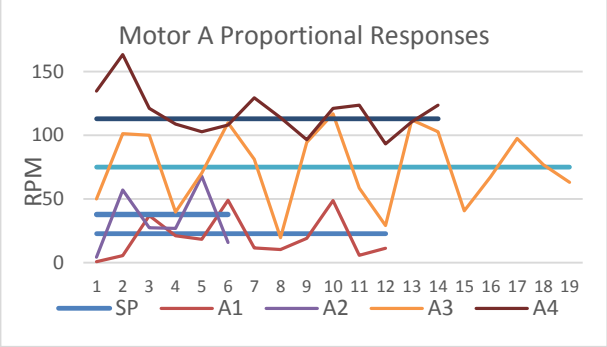

Figure 8. Motor A Poropotional responses

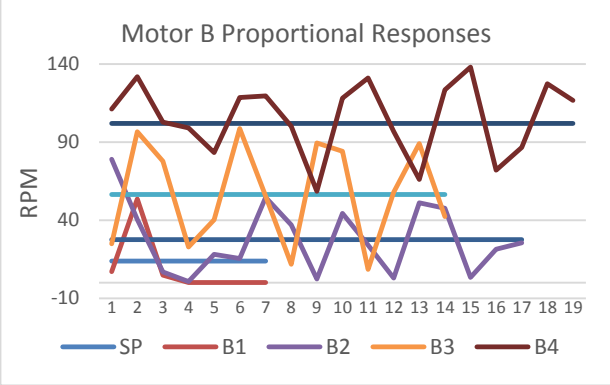

Figure 9. Motor B Poropotional responses

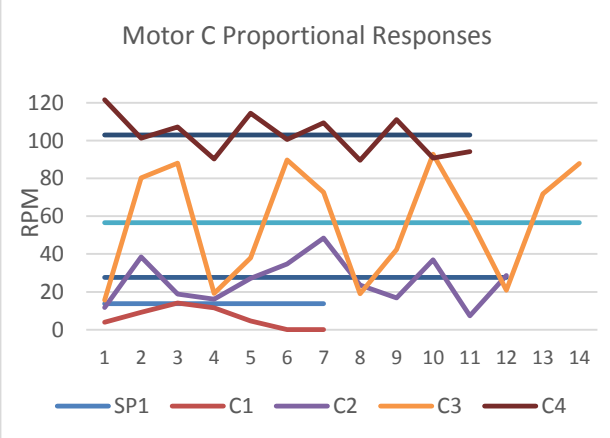

Figure 10. Motor C Poropotional responses

Looking at the responses of the three motors used, several points can be observed as follows:

1. All three motors find it difficult to achieve low speed numbers, especially for speeds below 25 rpm, indicated by the error number from the proportional control response that is of high value.
2. The current control parameter setting, which is 0.7, is more suitable and tends to be more stable at high speeds, especially at speeds above 90 rpm. This is indicated by the error number of the proportional control response which is smaller.

3.3 Distance measurement with Infrared sensor

This test is carried out to determine the use of infrared sensors used in reading the distance of an object with a robot. The object used for testing is a rectangular radio with a height of 8.6 cm and a width of 13.4 cm.

Table 3. Infrared sensor to measure distance

Test	Real Distance [cm]	Sensor reading [cm]	Error [%]
1	10	9.99	0.1
2	15	15.15	1
3	20	20.45	2.25
4	25	25.01	0.04
5	30	30.51	1.7
6	35	35.23	0.65
7	40	40.65	1.62
8	45	45.15	0.33
9	50	50.5	1
10	55	55.12	0.21
11	60	60.52	0.86
12	65	63.8	1.84
13	70	70.08	0.11
14	75	76.8	2.4
15	80	82.6	3.25
	Average Error		**0.87**

In previous experiments, objects with a flat surface were used. Then an experiment was carried out using a tubular object whose surface was not flat, dan change the position of the object. The test method remained the same as before. This is to determine the effect of the object's surface dan position on distance reading with an infrared sensor.

Table 4. Distance measurement on different object's surfaces

Distance [cm]	Flat Surfaces		Curved surface	
	Distance [cm]	Error [%]	Distance [cm]	Error [%]
20	20.45	2.25	18.12	9.40
30	30.51	1.70	28.16	6.13
40	38.45	3.87	36.4	9.00
50	49.55	0.90	52.5	5.00
60	61.35	2.25	62.8	4.67
70	63.8	8.86	78.1	11.57

Table 5. Distance measurement on different object's position

Distance [cm]	Parallel		Tilted 15°	
	Distance [cm]	Error [%]	Distance [cm]	Error [%]
30	30.58	1.9	31.73	5.77
40	41.48	3.7	30.42	23.95
50	50.85	1.70	39.1	21.80
60	61.12	1.87	42.8	28.67
70	70.15	0.21	34.5	50.71

As the result, the curved surface and the tilted object do affect the reading by infrared sensor, the curved or tilted object potentially has a greater error number.

3.4 Object size measurement using pi camera and Jetson Nano

Following the test of distance measurement, object dimension calculation is also performed. For this, the Nvidia Jetson Nano is used for the image processing. Due to the physical constraints, the system cannot detect more than one object and the object being measured must have height above 7.5 cm (height of the sensor to the ground surface).

For testing, we used various mundane objects such as a radio, a food box, a flashlight, a DVD box, a bottle, and a lump of sewing threads (see Figure 11).

Figure 11. Various common objects for the object size measurement experiment.

Here are the measurement results.

Table 6. Radio's size measurement

Distance [cm]	Real size [cm]		Camera reading [cm]		Error[%]	
	width	height	width	length	width	Height
20			13.63	8.57	1.72	0.35
25			13.6	8.66	1.49	0.70
30			13.77	8.65	2.76	0.58
35			13.51	8.4	0.82	2.33
40			13.98	8.61	4.33	0.12
45	13.4	8.6	14.33	8.57	6.94	0.35
50			13.88	8.42	3.58	2.09
55			14.08	8.37	5.07	2.67
60			13.2	8.3	1.49	3.49
65			13.38	8.47	0.15	1.51
70			13.7	8.4	2.24	2.33
Rata-Rata			13.73	8.49	2.78	1.50

Table 7. Food box 's size measurement

Distance [cm]	Real size [cm]		Camera reading [cm]		Error[%]	
	width	height	width	height	width	height
20			13.2	17.22	20.00	7.62
30			12.55	16.66	14.09	4.13
40	11	16	12.32	15.55	12.00	2.81
50			11.7	15.48	6.36	3.25
60			11.64	16.65	5.82	4.06
70			13.1	15.88	19.09	0.75
Rata-Rata			12.42	16.24	12.89	3.77

As a note, the food box measurements have an average error of 12.89% because, at distance of 20cm, the object covers all the camera's reading screens so the size reading is invalid. The average error for the height is 3.77%.

Table 8. Flashlight's size measurement

Distance [cm]	Real size [cm]		Camera reading [cm]		Error [%]	
	width	height	width	height	width	height
20	12.55	24.88	14.4	12.1	14.74	51.37
30			6.08	4.97	51.55	80.02
40			12.77	24.44	1.75	1.77
50			12.31	23.52	1.91	5.47
60			10.83	19.67	13.71	20.94
70			11.67	21.44	7.01	13.83
Rata-Rata			11.34	17.69	15.11	28.90

For reading the size of a round flashlight, the average number of errors is 15.11% and 28.9%, because at distance of 20 cm and 30 cm, the size of the object is too large for the camera reading.

Table 9. DVD box's size measurement

Distance [cm]	Real size [cm]		Camera reading [cm]		Error [%]	
	width	height	width	height	width	height
20	38.85	24.85	4.75	2.2	87.77	91.15
30			4.56	2.15	88.26	91.35
40			5.88	3.55	84.86	85.71
50			42.12	25.12	8.42	1.09
60			39.85	24.88	2.57	0.12
70			39.25	23.76	1.03	4.39
Rata-Rata			22.74	13.61	45.49	45.63

Measurement using a DVD box is only effective at distances greater than 50 cm, because at closer distances, the camera screen is not enough to capture the entire object.

Table 10. The balm bottle's size measurement

Distance [cm]	Real size [cm]		Camera reading [cm]		Error[%]	
	width	height	width	length	width	Height
20	5.1	3.6	11.8	12.3	131.37	241.6
30			8.44	10.01	65.49	178.
40			7.22	8.02	41.57	122.7
50			5.41	6.78	6.08	88.33
60			6.35	9.5	24.51	163.8
70			21.7	10.18	325.4	182.7
Rata-Rata			10.15	9.47	99.08	162.9

Measurement of the size of the balsam bottle object cannot be carried out, because the height of the object is only 3.6 cm, while the placement of the sensor is 7 cm from the ground surface. So it cannot be detected by an infrared sensor, causing the absence of distance readings and calculations for measurements with the camera cannot be processed correctly.

Table 11. A lump of sewing thread 's size measurement

Distance [cm]	Real size [cm]		Camera reading [cm]		Error [%]	
	width	height	width	height	width	height
20	7.5	11.5	6.40	10.66	14.67	7.30
30			7.53	11.2	0.40	2.61
40			7.45	10.88	0.67	5.39
50			7.21	10.34	3.87	10.09
60			6.85	9.44	8.67	17.91
70			7.47	10.87	0.40	5.48
Rata-Rata			7.15	10.57	4.78	8.13

From the above experiments, we observe the following:

- The system cannot detect objects with the same background color as the object.
- The shape of objects that can be measured is square, rectangular, round, tube. Beyond this form, the reading becomes less accurate.
- To be able to read its size, objects must all enter the camera reading frame. If it is too close to the camera, it might produce invalid readings.

Figure 12 shows the sketch and the physical appearance of our robot.

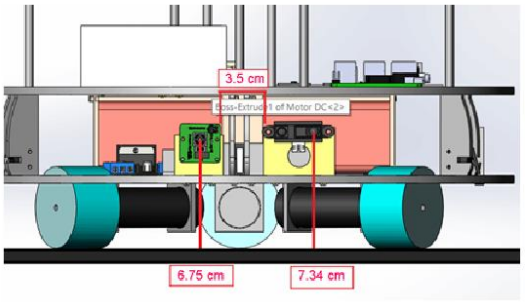

Figure 12. The sketch and the real robot.

4. Conclusion

This paper describes the process of designing a 3-wheel omni-directional mobile robot that is controlled using a Raspberry Pi for its direction and can detect simple objects in front of it. Inverse kinematics was used to determine the direction and speed of each motor, and the odometry was used to estimate the movement of the robot's position. From these two calculations, the robot can move and stop at the position inputted by the user with an error rate of 9.51% on the x-axis and 8.12% on the y-axis for the robot's final position. This is influenced by the speed of the robot to reach that point, the proportional parameters used, and the distance between the final position and the initial position. Reading the distance of an object using an infrared sensor, we got an average error of 0.87%. The factors that influence this reading accuracy are the angle of placement of the object to the sensor and the surface of the object being measured. For the object detection, the OpenCV library was used that utilizes the color contrast of

objects against the background to detect an object. From the results of testing the size of objects using 6 objects of different sizes, this method has the ability to read object sizes in front of the robot with an average error of 30.02% for length readings and 41.8% for object height readings. This detection accuracy was influenced by object size, object distance from the camera, and object detection methods used. For future work, we plan to combine this omni-directional robot platform with a manipulator to create an adaptive mobile manipulator.

Acknowledgements

This work is supported partially by Petra Christian University through the Special Grant (No.009/SP2H/LT-MULTI/LPPM-UKP/III/2020 and No.016/SP2H/LT-MULTI/LPPM-UKP/III/2020).

References

1. ABI Research, "50,000 Warehouses to Use Robots by 2025 as Barriers to Entry Fall and AI Innovation Accelerates", 2019, March 26. [Online] Available: abiresearch. Com.
2. X. Li and A. Zell, "Motion control of an omnidirectional mobile robot," *Lect. Notes Electr. Eng.*, vol. 24 LNEE, pp. 181–193, 20
3. K. Drive, "A Simple Introduction to Omni Roller Robots," no. April, 2015.
4. A. Rosebrock, "Measuring size of objects in an image with Open CV," 28 March 2016. [Online]. Available: https://www. pyimagesearch.com/2016/03/28/measuring-size-of-objects-in-an-image-with-opencv/.
5. P. Malheiros, J. Gonçalves, and P. Costa, "Towards a more Accurate Infrared Distance Sensor Model," *Manuf. Syst. Eng. Unit*, no. Sept, 2010.
6. M. Taufiqqurohman and N. F. Sari, "Odometry Method and Rotary Encoder for Wheeled Soccer Robot," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 407, no. 1, 2018.
7. J. J. Parmar and C. V Savant, "Selection of Wheels in Robotics," *Int. J. Sci. Eng. Res.*, vol. 5, no. 10, pp. 339–343, 2014.
8. F. Fahmizal, D. U. Rijalussalam, M. Budiyanto, and A. Mayub, "Trajectory Tracking pada Robot Omni dengan Metode Odometry," *J. Nas. Tek. Elektro dan Teknol. Inf.*, vol. 8, no. 1, p. 35, 20.