

Comparative Study of Computer Vision Based Line Followers using Raspberry Pi and Jetson Nano

Gunawan Dewantoro, Jamil Mansuri, and Fransiscus Dalu Setiaji
Faculty of Electronics and Computer Engineering, Satya Wacana Christian University
Jl. Diponegoro 52 – 60, Salatiga 50711
e-mail: gunawan.dewantoro@uksw.edu

Abstract—The line follower robot is a mobile robot which can navigate and traverse to another place by following a trajectory in the form of black or white lines. This robot can also assist human in carrying out transportation and industrial automation. However, this robot also has several challenges with regard to the calibration issue, incompatibility on wavy surfaces, and also the light sensor placement due to the line width variation. Robot vision utilizes image processing and computer vision technology for recognizing objects and controlling the robot motion. This study discusses the implementation of vision based line follower robot using a camera as the only sensor used to capture objects. A comparison of robot performance employing different CPU controllers, namely Raspberry Pi and Jetson Nano, is made. The image processing uses an edge detection method which detects the border to discriminate two image areas and mark different parts. This method aims to enable the robot to control its motion based on the object captured by the webcam. The results show that the accuracies of the robot employing the Raspberry Pi and Jetson Nano are 96% and 98%, respectively.

Keywords: *line follower, image processing, Raspberry Pi, Jetson Nano*

Abstrak—Robot line follower adalah jenis robot mobile yang dapat bergerak memindahkan dirinya ke tempat lain dengan mengikuti garis lintasan yang berwarna hitam atau putih. Robot ini juga dapat membantu tugas manusia dalam melakukan otomatisasi transportasi dan juga robot industri. Namun robot ini juga memiliki beberapa kelemahan di antaranya sulit dikalibrasi, tidak cocok digunakan pada permukaan yang bergelombang, dan peletakan sensor cahaya sangat dipengaruhi lebar tipisnya garis yang diikuti. Robot vision menggunakan teknologi pemrosesan citra dan computer vision dalam tugasnya, baik untuk pengenalan obyek maupun kendali gerak robot. Penelitian ini membahas tentang perbandingan unjuk kerja robot menggunakan Raspberry Pi dan Jetson Nano yang diintegrasikan dengan webcam sebagai sensor yang menangkap obyek dan selanjutnya dilakukan pengolahan citra. Pengolahan citra menggunakan metode deteksi tepi yang berfungsi mendeteksi garis tepi yang membatasi dua wilayah. Penggunaan metode ini bertujuan agar robot dapat melakukan kendali gerak berdasarkan tangkapan obyek yang diperoleh oleh webcam. Hasil percobaan menunjukkan ketepatan robot dengan Raspberry Pi melewati rintangan sebesar 96% dan Jetson Nano 98%.

Kata kunci: *line follower, pemrosesan citra, Raspberry Pi, Jetson Nano*

I. INTRODUCTION

Robot is one of the devices in the field of mechatronics. During carrying out the task, the robot follows the trajectory which has been previously determined by the user. The robot control system is a system used by humans to determine the robot's motion pattern [1]. One of the easy-to-control and widely used robots is the line follower robot. The line follower robot can automatically traverse along the line, which is generally black or white. This robot aims to assist human tasks in transportation and industrial automation by means of a predefined path [2]–[5]. The line follower robot utilizing light sensors has been tested, and the results show that the line follower robot can walk following the black line properly. However, this line

follower robot still has shortcomings due to the line sensor sensitivity dependence with respect to the robot speed [6]. On top of that, this robot also has several challenges with regard to the calibration issue, incompatibility on wavy surfaces, and also the light sensor placement due to the line width variation. Therefore, a vision robot is proposed to overcome these shortcomings. The robot vision uses image processing and computer vision technology for recognizing object and controlling robot motion.

Some previous works were carried out to realize a vision based line follower robot. In [7], a camera was utilized to capture the image of track and then it is converted into bitmap image. The Least Square Method was utilized to follow the predefined path. By calculating the slope of line, the turning angle of the robot was determined. Meanwhile,

other research applied a PID control algorithm to adjust the robot on the line. In [8] – [9], the line follower robot was equipped with a webcam mounted on the vehicle and connected to Matlab platform.

The robot vision requires a processing unit which is responsible for processing the digital images and controlling the robot motion. Therefore, a compact and portable processor is essential to support the robot mobility and performance [10]. Some platforms and development boards have been chosen so as to apply the color detection scheme on the line follower robot. The Arduino platform and Pixy camera were used to distinguish the path and the background using various approaches: the Otsu's Method, thresholding based on color mixtures, and color tracking through hue and saturation [11]. The motion control was fully automated based on the path line center. Employing a four-legged robot platform and a camera as the only sensor, the robot is capable of successfully following a line. In [12], the hardware setup of an autonomous robotic vehicle was developed based on color detection to navigate the vehicle in different directions. Raspberry Pi camera and OpenCV were used to access images which are the indications of the directions of the movements such as stop, left turn, right turn.

Another work designed and constructed a line follower robot with a TDRB-D5M camera connected to a Cyclone IV FPGA in a development board DE0-NANO [13]. The computer vision algorithms were programmed into the FPGA to guide the robot through a line of one of three possible colors by masking pixels in the camera image. Based on the color detected, the robot controlled the motors to keep up the sequence and advance at different speeds in the line.

The advantages of camera compared to the light sensor for line tracking are explained in [14]. The additional computational time is compensated by numerous benefits. This includes the fact that the travel speed decreases only in turns, while the robot can go on higher speed in the straight lines without any trouble. The speed also varies depending on the next element of the road captured by the camera. In contrast to the line sensor, the vulnerability to the vibration of the robot is also much lower.

In this study, a line follower was built using Raspberry Pi and Jetson Nano separately to compare their performance using the same image processing algorithm. The Raspberry Pi and Jetson Nano are chosen because, they both are categorized as mini CPUs which include processor, memory, peripherals, and Python compiler. The clock speed of these mini CPUs are approximately ten times faster than that of Arduino boards, making them suitable for dealing with real-time image processing applications. A webcam was used as a sensor which captures objects and the edge detection method was employed to mark the lines [15]. This method is intended so that the robot can traverse based on the information extracted from the image captured by the webcam. This paper is organized as follows: Section II explains the methods used in this study. Section III presents and discusses the results. Finally,

section IV concludes the findings.

II. METHODS

The computer vision based line follower consists of several main components such as a processing unit, two motor drivers, four DC motors, and a webcam. Fig. 1 shows a block diagram of the hardware used.

The webcam serves as the line reader which captures image and send it to the mini CPU, i.e. Raspberry Pi or Jetson Nano, as the brain of the line follower robot. These mini CPUs are in charge of receiving input data, and inferring the motion command to the driver motor. Then, the driver motor will actuate the DC motor so that the line follower can move accordingly.

A. Raspberry Pi Model B Pinout

Raspberry Pi is a single-board mini PC which provides a set of GPIOs to control electronic components. The board is equipped with a Broadcom BCM2711 processor (1.5 GHz quad-core Arm Cortex-A72 CPU), 4 GB of RAM, and microSD card reader for storage. In this study, Raspberry Pi is used to receive data from the webcam, process the data, and provide output to the motor driver. Table 1 shows the pinout of the Raspberry Pi 4 Model B. Whereas Fig. 2 shows the schematic of the Raspberry Pi and the motor driver.

B. Jetson Nano Pinout

Jetson Nano is a minicomputer device created by NVIDIA. Despite its small size, the computing performance of Jetson Nano is up to 472 GFLOPS (one million data processing per second) and is capable of running AI modern workloads and also very power efficient, absorbing only 5 watts of power. The device is equipped with a 128-core Maxwell GPU, Quad-core ARM A57 @ 1.43 GHz, 4 GB of RAM and microSD card reader for storage. In this study, the OpenCV is compiled with Cuda support to make the most of the GPU in Jetson Nano. Jetson Nano is used to receive data from the webcam, process the data with Arduino to generate PWM. The Arduino is used to convert Jetson Nano digital signals into analog signals since the Jetson Nano only has very limited number of analog pins. The Arduino then forwards the output to the motor driver

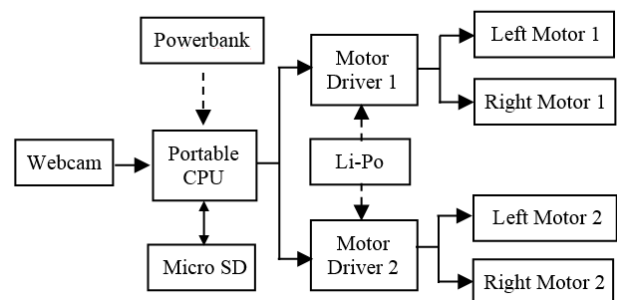


Fig. 1. Hardware block diagram

and the DC motor will move accordingly. Table 2 shows the pinout used on the Jetson Nano, while Fig. 3 shows the schematic of the Jetson Nano, Arduino, and motor drivers.

C. Edge Detection

The boundaries of the black and white image are determined by means of the edge detection. In this study, the method used is Sobel algorithm because of its simplicity [16]. Sobel filter is a straightforward approach to the notion of gradient with smoothing. The gradients in X and Y axes are detected by means of the 3x3 convolution mask. The operator comprises a pair of 3x3 convolution kernels, with one being the other rotated by 90°, as shown by Fig. 4.

One kernel is employed for each of the two perpendicular orientations (vertical and horizontal) separately to give different measurements of the gradient component in each orientation (G_x and G_y). In order to find the absolute magnitude at any point, the G_x and G_y can be employed according to Equation (1) [17].

$$|G| = |G_x| + |G_y|. \tag{1}$$

Table 1. Raspberry Pi 4 Model B Pinout

Raspberry Pi Pin	Motor Driver 1 Pin	Motor Driver 2 Pin
GPIO16	ENA	
GPIO20	IN1	
GPIO21	IN2	
GPIO5	IN3	
GPIO6	IN4	
GPIO13	ENB	
5V	5V	
GND	GND	
GPIO25		ENA
GPIO23		IN1
GPIO24		IN2
GPIO27		IN3
GPIO17		IN4
GPIO22		ENB
5V		5V
GND		GND

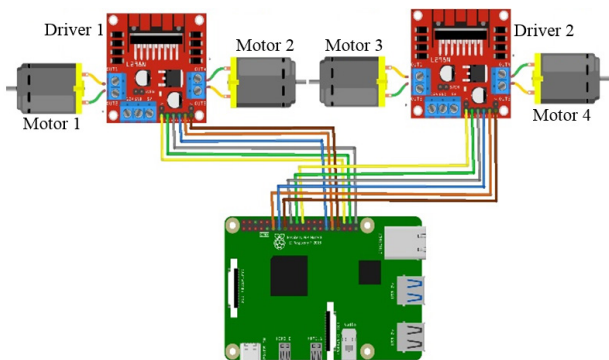


Fig. 2. Schematic of the Raspberry Pi and motor drivers

As for the angle of orientation of the boundary, q , relative to the pixel grid is given by Equation 2.

$$q = \arctan\left(\frac{G_y}{G_x}\right). \tag{2}$$

Table 2. Jetson Nano Pinout

Jetson Nano Pin	Arduino Pin	Motor Driver 1 Pin	Motor Driver 2 Pin
GPIO 22	PIN08		
GPIO 18	PIN06		
GPIO 23		IN 1	
GPIO 24		IN2	
GPIO 25		IN3	
GPIO 08		IN4	
5V		5V	
GND		GND	
	PIN03	ENA	
	PIN09	ENB	
GPIO 11	PIN07		
GPIO 07	PIN05		
GPIO 27			IN1
GPIO 17			IN2
GPIO 10			IN3
GPIO 09			IN4
5V			5V
GND			GND
	PIN10		ENA
	PIN11		ENB

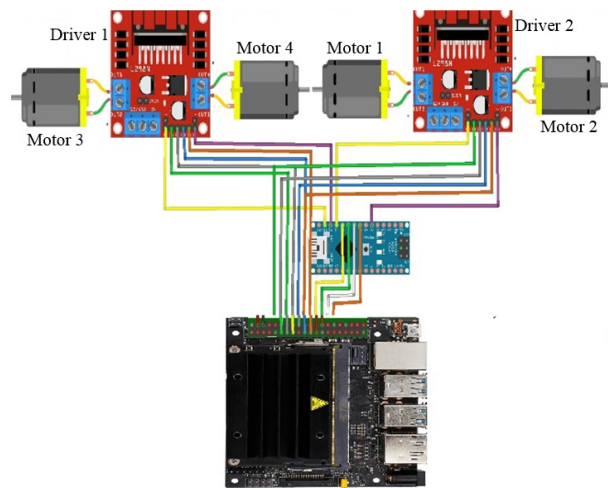


Fig. 3. Schematic of Jetson Nano, Arduino and motor driver

+1	0	-1		+1	+2	+1
+2	0	-2		0	0	0
+1	0	-1		-1	-2	-1
G _x				G _y		

Fig. 4. Mask used by Sobel operator

D. Software

The flow diagram in this study is shown in Fig. 5. The image processing used the OpenCV library which execute the image processing program on the mini CPU to support the computer vision. The first step is the image-capturing process by the webcam. If the camera does not take a picture, the process will be repeated in the mini CPU until an image is captured. Then, the mini CPU conducts the image pre-processing which includes color conversion and color filtering from RGB to grayscale. Equation (3) shows the relationship between the RGB pixel values and the corresponding grayscale value.

$$G = \frac{r + g + b}{3}, \tag{3}$$

G denotes the grayscale value, r denotes the red component of pixel, g denotes the green component of pixel, and b denotes the blue component of pixel.

Having been converted into grayscale, the image is then converted to a binary image with the threshold being the mean of the grayscale value. After obtaining the binary image, all black pixels are assigned as object and white pixels as the background. Therefore, the robot is able to detect the edge of the map line easily [18]. Then, the coordinates of each part of the object are collected and thus the robot's motion can be appropriately determined.

The image processing process begins with the initial reading of the track image which is then forwarded to the motor driver whose task is to run the motor, both adjusting the rotational direction and speed of each motor. When the robot detects the black line, the robot will follow the line according to the line type (straight line, dashed line, turns, and junctions). If an image matches to a predefined category of lines, the mini CPU will display the string output at the terminal which represents the motion

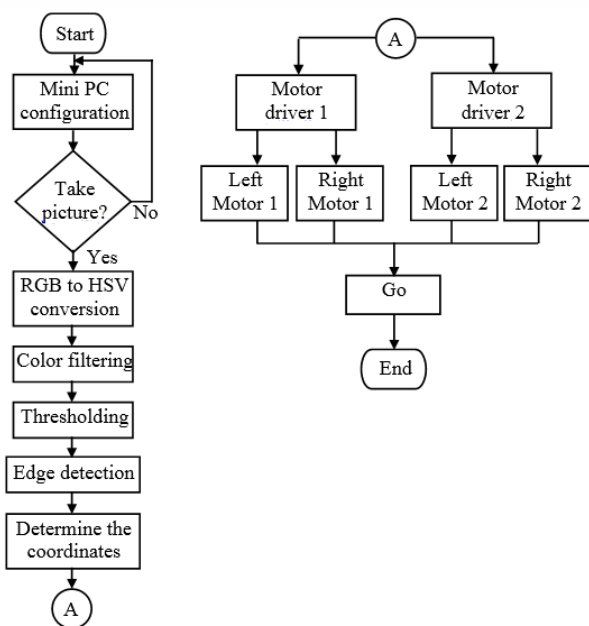
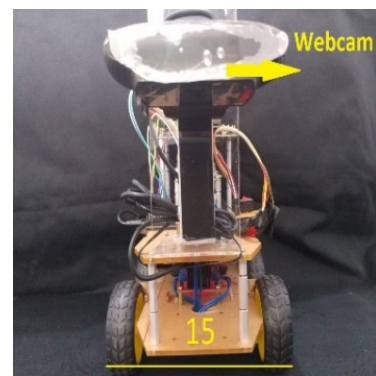


Fig. 5. System flowchart

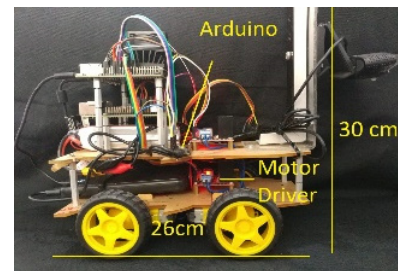
command. The motion commands used in this study are: go straight (lurus), turn right (belok kanan), and turn left (belok kiri).

E. Testing Arrangement

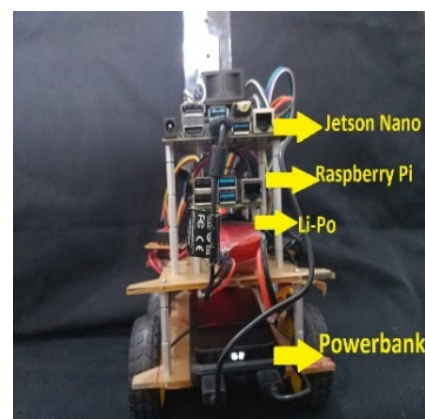
In order to assess the performance of both controllers, some metrics are used, each of which has a specific purpose. First, the recognition accuracy is measured to evaluate the correctness of the controllers in recognizing a range of lines on the map. Ten trials are conducted for each type of line, and based on the results, the accuracy can be calculated in percentage. Secondly, the effect of illumination is taken into account by varying the brightness of the indoor lighting. Two of ten trials are conducted in the open space so that the sunlight is incident on the map. Lastly, the travel time is considered to measure the speed at which the robot traverse on the map. The time elapsed is recorded for one lap, meaning that the start point is the



(a)




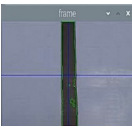

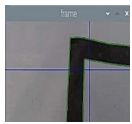

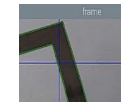



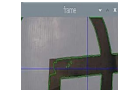

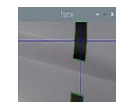

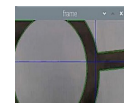


(b)



(c)

Fig. 6. Robot realization (a) Front view (b) Side view (c) Back view

Table 3. The results of path recognition

No	Line on the Map	Output on Terminal	Display on the Image Frame	Remark
1		<pre>Lurus 187 Lurus 176 Lurus 167 Lurus </pre>		Detected
2		<pre>223 Belok kanan </pre>		Detected
3		<pre>72 Belok Kiri 78 Belok Kiri </pre>		Detected
4		<pre>89 Belok Kiri 101 Lurus </pre>		Detected
5		<pre>267 Belok kanan 263 Belok kanan </pre>		Detected
6		<pre>154 Lurus 153 Lurus </pre>		Detected
7		<pre>232 Belok kanan </pre>		Detected
8		<pre>151 Lurus 152 Lurus </pre>		Detected

same as the finish point.

III. RESULTS AND DISCUSSION

The robot frame consists of two levels, where the lower level contains four DC motors, power banks, Li-Po battery and motor drivers. The upper level contains the mini CPU, motor driver and webcam. This robot has dimensions of 26 cm × 15 cm × 30 cm. The realization of the robot shown in Fig. 6. In order for the robot to predict the type of line, the webcam must fully capture the line in full width. Once the line is in the camera's field of view, the robot should be able to determine the motion that must be taken afterward [19]. The geometrical shape of the robot also constraints the minimum distance perceived by the camera, which is 8 cm. Because of that, the camera is mounted 25 cm above the base floor and directed downward.

Table 4. Path recognition testing using Raspberry Pi

Test No.	Type of Line							
	1	2	3	4	5	6	7	8
1	✓	✓	✓	✓	×	✓	×	✓
2	✓	✓	✓	✓	✓	✓	✓	✓
3	✓	✓	✓	✓	✓	✓	✓	✓
4	✓	✓	✓	✓	✓	✓	×	✓
5	✓	✓	✓	✓	✓	✓	×	✓
6	✓	✓	✓	✓	✓	✓	✓	✓
7	✓	✓	✓	✓	✓	✓	✓	✓
8	✓	✓	✓	✓	✓	✓	✓	✓
9	✓	✓	✓	✓	✓	✓	✓	✓
10	✓	✓	✓	✓	✓	✓	✓	✓

Description:
 ✓: Successfully detected
 ×: Fail to detect

A. Path Recognition Testing

This testing aims to determine the success rate of path recognition. Therefore, the actual image on the map, the motion command on the terminal and the captured image are compared, as shown in Table 3.

Raspberry Pi and Jetson Nano were used as mini CPUs and able to execute the whole algorithm without any trouble. Of all kinds of lines, the program could generate motion command that matches the actual track on the map. In this study, if the webcam detected white area, the mini CPU would drive the robot backwards command so that the robot could return to the black line on the map.

The key factor of successful path recognition lies on the accuracy of edge detection. In order to find out how accurate the edge detection is, ten experiments were carried out on each type of line to see whether or not the algorithm can recognize the type of line. The testing was done using Raspberry Pi and Jetson Nano as mini CPU separately, as shown in Table 4 and 5.

The accuracy metric is calculated using the Equation (4) as follows:

$$Accuracy = \frac{N_s}{N_T} \times 100\%, \quad (4)$$

with N_s being the number of successful detections and N_T being the number of trials.

Based on Table 4, the Raspberry Pi can detect all types of lines with an accuracy rate of 100% except for line types 5 and 7, which contain circle shapes, with an accuracy rate of 90% and 70%, respectively. Meanwhile, based on Table 5, the Jetson Nano can detect all type of lines with an accuracy rate of 100%, except for line types 3 and 5, which are left and right turns, each obtaining an accuracy rate of 90%. These errors in detecting several types of lines was caused by lighting factor as the robot vision was obstructed by the robot's own shadow while moving across the map.

B. Effect of Light Intensity

Light intensity is one of the important factors that affect the success of path recognition. Light intensity testing is carried out to assess the webcam’s capabilities in various light intensities and their effect to path recognition, as shown in Table 6. Based on the results in Table 6, the robot will not work if the light intensity is lower than 0.7 lumens / m². Therefore, it can be inferred that the success of the webcam in detecting path on the map is influenced by the light intensity over the map area.

Table 5. Path recognition testing using Jetson Nano

Test No.	Type of Line							
	1	2	3	4	5	6	7	8
1	✓	✓	✓	✓	✓	✓	✓	✓
2	✓	✓	✓	✓	✓	✓	✓	✓
3	✓	✓	✓	✓	✓	✓	✓	✓
4	✓	✓	✓	✓	✓	✓	✓	✓
5	✓	✓	✓	✓	✓	✓	✓	✓
6	✓	✓	✗	✓	✗	✓	✓	✓
7	✓	✓	✓	✓	✓	✓	✓	✓
8	✓	✓	✓	✓	✓	✓	✓	✓
9	✓	✓	✓	✓	✓	✓	✓	✓
10	✓	✓	✓	✓	✓	✓	✓	✓

Description:

✓: Successfully detected

✗: Fail to detect

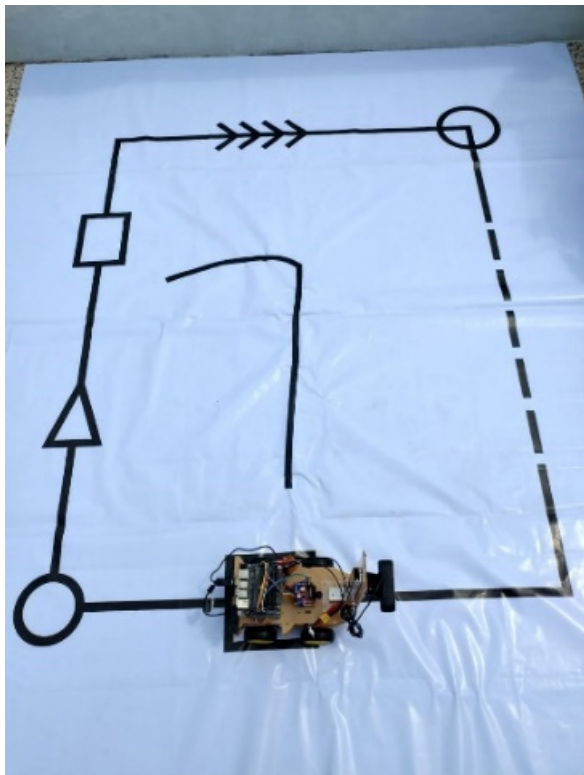


Fig. 7. The robot on the test map

It is because the light might change the color of the detected object. For example, when the light intensity is too low, the color of the object become darker (pixel value is close to 0) due to the poor illumination. On the contrary, under bright lighting condition, the color of the object become brighter (pixel value is close to 255) due to the increased illumination. In this study, the robot is able to detect objects with a minimum light intensity of 0.7 lumens / m² and up to 91,600 lumens / m². The motion control for each trial includes Go, Stop, Turn Left, and Turn Right.

C. Robot Travel Time on the Map

The robot travel time is the time elapsed to complete six laps. The testing was carried out with a light intensity of 88,990 lumens/m² using the Raspberry Pi and Jetson Nano as the mini CPU, and conducted ten times for each mini CPU. Fig. 7 shows the map used when testing the robot’s travel time, and the results are shown in Table 7.

D. Performance Comparison

Table 8 summarizes the performance comparison of line follower robots using Raspberry Pi and Jetson Nano. The results in Table 8 implies that the Jetson Nano outperforms Raspberry Pi in conducting the line following tasks.

Particularly, the Jetson Nano has a fastest lap of 20 seconds with a success percentage of 100% to complete the laps. Meanwhile the Raspberry Pi has a fastest lap 38.16 seconds with a success percentage of 98% to complete the laps. As for the overall accuracy of path

Table 6. Effect of light intensity

No.	Light Intensity (lumens/m ²)	Robot Motion Control							
		Raspberry Pi				Jetson Nano			
		Go	Stop	R	L	Go	Stop	R	L
1	0	✗	✗	✗	✗	✗	✗	✗	✗
2	0.7	✓	✓	✓	✓	✓	✓	✗	✗
3	4.2	✓	✓	✓	✓	✓	✓	✓	✓
4	30	✓	✓	✓	✓	✓	✓	✓	✓
5	79	✓	✓	✓	✓	✓	✓	✓	✓
6	487	✓	✓	✓	✓	✓	✓	✓	✓
7	546	✓	✓	✓	✓	✓	✓	✓	✓
8	790	✓	✓	✓	✓	✓	✓	✓	✓
9	88,990	✓	✓	✓	✓	✓	✓	✓	✓
10	91,600	✓	✓	✓	✓	✓	✓	✓	✓

Description:

✓: Moving

✗: Not moving

Table 7. Travel time of the robot as many as 6 laps

No.	Raspberry Pi (second)	Jetson Nano (second)
1	234	123
2	230	120
3	237	150
4	235	127
5	240	120
6	232	120
7	235	140
8	241	124
9	229	121
10	235	120

Table 8. Performance comparison

	Raspberry Pi	Jetson Nano
Fastest Lap	38.16 seconds	20 seconds
Success Percentage	98%	100%
Path Recognition Accuracy	96%	98%

recognition, the Jetson Nano and the Raspberry yield 98% and 96%, respectively. The superiority of Jetson Nano is because it employs an NVIDIA Maxwell GPU (Graphics Processing Unit) with 128 CUDA cores @ 921 Mhz, while the Raspberry Pi uses 32-bit Broadcom VideoCore VI. The GPU embedded in the Jetson Nano significantly affect the performance processing the video and graphics, and thus robots employing the Jetson Nano can perform faster image processing.

IV. CONCLUSION

A comparative study of computer vision based line follower was done. It can be concluded that the Raspberry Pi and Jetson Nano work well as mini CPUs to conduct line following tasks. The Jetson Nano has a fastest lap of 20 seconds with a success percentage of 100% to complete the laps. Meanwhile the Raspberry Pi has a fastest lap 38.16 seconds with a success percentage of 98% to complete the laps. As for the overall accuracy of path recognition, the Jetson Nano and the Raspberry yield 98% and 96%, respectively. Hence, it is obvious that the Jetson Nano outperforms Raspberry Pi for all aspects. In near future, a vision based line follower can be built by means of machine learning or artificial intelligence to see trade-off with the image processing.

REFERENCES

- [1] S. G. Tzafestas, "Mobile robot control and navigation: A global overview," *Journal of Intelligent & Robotic Systems*, vol. 91, pp. 35–58, 2018.

- [2] M. A. Kader, M. Z. Islam, J. A. Rafi, M. R. Islam, dan F. S. Hossain, "Line following autonomous office assistant robot with PID algorithm," in *Proc. of International Conference on Innovations in Science, Engineering and Technology*, 2018, pp. 109-114.
- [3] V. V. Vivek, E. L. Pradeesh, N. Natarajan, M. C. Pravin, and S. R. Prabhu, "Development of a system to assist in library to automate the book handling process," in *Proc. of IOP Conference Series: Materials Science and Engineering*, vol. 995, 2020, pp. 1-8.
- [4] M. F. Hasan, I. Rahaman, M. S. Sayem, M. M. Hasan, B. K. Tamal, and M. A. R. Rubel, "Designing of cost-effective human assistive robot and performance enhancement," in *Proc. of 11th International Conference on Computing, Communication and Networking Technologies*, 2020, pp. 1-5.
- [5] Y. Oktarina, M. Nawawi, and W. G. Tulak, "Analysis of the sensor line on line follower robot as an alternative transport the tub trash in the shopping center," *Volt: Jurnal Ilmiah Pendidikan Teknik Elektro*, vol. 2, no. 2, pp. 101-108, 2017.
- [6] A. Latif, H. A. Widodo, R. Rahim, and K. Kunal, "Implementation of line follower robot-based microcontroller ATmega32A," *Journal of Robotic and Control*, vol. 1, no. 2, pp. 70-74, 2020.
- [7] S. Shetty, A. Bhavsar, M. Mergu, and A. Talekar, "Line following robot using image processing," in *Proc. of the 3rd International Conference on Computing Methodologies and Communication*, 2019, pp. 1174-1179.
- [8] W. E. Elhady, H. A. Elnemr, and G. Selim, "Implementation and evaluation of image processing techniques on a vision navigation line follower robot," *Journal of Computer Science*, vol. 10, no. 6, pp. 1036-1044, 2014.
- [9] M. V. Gomes, L. A. Bassora, O. Morandin, and K. C. T. Vivaldini, "PID control applied on a line-follower AGV using a RGB camera," in *Proc. of 19th International Conference on Intelligent Transportation Systems*, 2016, pp. 194-198.
- [10] K. D Setyanto, I. Fibriani, and S. Sumardi, "Control of a visionary mobile robot using a webcam on an object with a Raspberry Pi-based arrow," *Jurnal Arus Elektro Indonesia*, vol. 2, no. 1, pp. 27-32, 2016.
- [11] B. A. Asfora, "Embedded computer vision system applied to a four-legged line follower robot," in *Proc. of 23rd ABCM International Congress of Mechanical Engineering*, 2015, pp. 1-7.
- [12] B. K. Sahu, B. K. Sahu, J. Choudhury, and A. Nag, "Development of hardware setup of an autonomous robotic vehicle based on computer vision using Raspberry Pi," in *Proc. of Innovations in Power and Advanced Computing Technologies*, 2019, pp. 1-5.
- [13] L. A. P. Sanchez, C. A. A. Moreno, and H. A. B. Dazza, "Design and construction of a line follower robot guided by pixels values of a camera connected to an FPGA," in *Proc. of 20th Symposium on Signal Processing, Images and Computer Vision*, 2015, pp. 1-5.
- [14] A. Kondakor, Z. Törösvári, A. Nagy, and I. Vajk, "A line tracking algorithm based on image processing," *Proc. of International Symposium on Applied Computational Intelligence and Informatics*, 2018, pp. 39-44.
- [15] M. Yunus, "Comparison of edge detection methods for digital image segmentation processes," *International Journal of Information Technology*, vol. 3, no. 2, pp. 146-160, 2013.
- [16] G. M. H. Amer and A. M. Abushaala, "Edge detection methods," *Proc. of 2nd World Symposium on Web Applications and Networking (WSWAN)*, 2015, pp. 1-7.
- [17] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679-698, 1989.

- [18] A. A Nurcahyani, and R. Saptono, "Identifying the quality of rice using digital images." *Scientific Journal of Informatics*, vol. 2, no.1, pp. 63- 72, 2015.
- [19] A. Kondákor, Z. Töröcsvari, A. Nagy, and I Vajk, "A line tracking algorithm based on image processing," Proc. of IEEE 12th International Symposium on Applied Computational Intelligence and Informatics, 2018, pp. 39-44.