Design and Implementation of a Parking Lot Monitoring System Using the YOLOv5 Method (Case Study in the AH Polinema Building Parking Lot)

Ananda Alif Kemal Firmansyah[1], Rieke Adriati Wijayanti[2*], Hudiono[3]

[1,2,3]Digital Telecommunication Network Study Program, Department of Electrical Engineering, State Polytechnic of Malang, 65141, Indonesia.

[1]anandakemal3@gmail.com, [2]riekeaw@polinema.ac.id, [3]hudiono@polinema.ac.id

***Abstract*— High vehicle productivity and not accompanied by adequate road construction can result in traffic jams on the highway. Apart from causing traffic jams on the roads, the large number of private vehicles, especially four-wheeled vehicles, results in full parking spaces in public service areas. The size of parking spaces in public places cannot keep up with developments in the number of private cars, so that many car parking spaces are full and it is difficult to know which parking positions can still be occupied. The Empty Parking Lot Monitoring System in the AH Polinema Building was carried out based on the level of difficulty in finding empty slots for parking four-wheeled vehicles or cars, so this system was formed as a solution by becoming an information system for the general public to make it easier to find empty slots for parking without visiting the parking lot. This system utilizes two web cameras facing the parking lot of the AH Polinema Building. The captured images from the two web cameras will be processed via a single-board computer (SBC), namely the Raspberry Pi 4B. Image processing was carried out using the YOLOv5 method. This processing results in the calculated value of the total empty slots for the AH Polinema Building parking lot. This output is integrated into the website so that the public can easily access information on vacant parking lots in the AH Polinema Building.**

***Keywords*— Car, Machine Learning, Parking, Parking Lot, YOLOv5.**

I. INTRODUCTION

A. Background of the problem

Private vehicles are the most popular choice among people because of their comfort, effectiveness, and efficiency in moving places, while online transportation is the second most popular choice among people and public transportation is the last choice [1]. According to Dadang's research, high vehicle productivity and not accompanied by adequate road construction can result in traffic jams on the highway [2]. Apart from causing traffic jams on the roads, the large number of private vehicles, especially four-wheeled vehicles, results in full parking spaces in public service areas. The size of parking spaces in public places cannot keep up with developments in the number of private cars, so that many car parking spaces are full and it is difficult to know which parking positions can still be occupied [3].

In this research, one of the case studies of the parking lot at the AH Polinema Building is raised. In the car park at the AH Polinema Building, visitors cannot know which parking position can be occupied without entering the parking area. According to the parking officer at the AH Polinema Building, there are problems in managing the parking area. One example is when a visitor's vehicle exceeds the parking capacity, the officer will direct the car to look for a parking space that is located away from the AH Building. There are two parking lots at the AH Polinema Building, the first is in front of the Polinema Mini Soccer Field, behind the cycling bicycle parking area. The second is behind the AH Electrical Engineering Building, in front of the Lecturer Motorcycle Parking Lot. The AH Polinema Building Parking Lot can be provided with convenience with the Parking Capacity Information System, so that visitors and parking attendants have the facility to monitor empty parking lots [4]. Visitors who enter the parking lot do not need to go around looking for an empty lot

for parking because the system will provide information on empty parking spaces in the AH Polinema Building Parking. Based on the problems, a tool is needed to carry out image processing to detect and calculate the number of parking spaces. With this tool, it is hoped that it will make it easier for visitors to find out the parking position without entering the parking lot first. In previous research, the image processing process in parking lots was carried out using computing power from the GPU. However, this research had shortcomings in conducting training and testing datasets which resulted in reduced accuracy values [3]. Then this research was applied to a prototype, so that in this research parking lot detection was implemented in the AH Polinema Building as a development of previous research.

In this research, a detection process was carried out and counted the number of empty parking spaces. Visitors and

parking attendants can monitor the parking lot at the AH Polinema Building via a website that will be integrated with the tool. Therefore, a method is needed to carry out the process. The method used to solve the problem is using the YOLOv5 method. The main reason for choosing this method is because the YOLOv5 method has varying models (weights), so that various tests can be carried out to get the most optimal performance. Another reason for choosing the YOLOv5 method is based on the speed in processing an image in real-time with an average speed of 30 fps (frames per second) [5]. This research refers to the following objectives:

1. Design and Implement a Parking Lot Monitoring System Using the YOLOv5 Method by utilizing a website as a user interface.
2. To integrate the detection results and calculation of the number of available parking spaces with the website.
3. To monitor the number of empty parking spaces based on the website.

II. METHOD

The type of research that will be carried out is Manufacturing/Development research. This research develops previous research. To create a parking space availability system requires designing tools and systems so that maximum results are obtained. This research consists of three main systems, namely input, process, and finally output. The input and process systems interact with each other via a USB type A cable. Meanwhile, the process and output systems interact with each other via the internet. The following is a block diagram of the system:

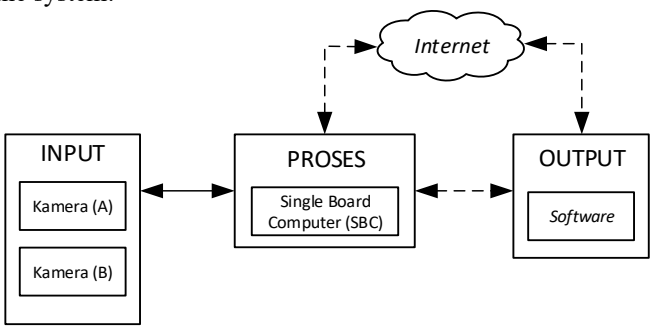

Figure 1. Block Diagram Of System

In Figure 1 it can be seen that there are two camera inputs, with the camera details being a web camera in the parking lot to detect cars so that empty parking slots can be calculated in the AH Polinema Building [6]. The two cameras are connected to a Single Board Computer (SBC) via a USB type A cable. The images taken by the two cameras are processed by the SBC in the form of a Raspberry Pi 4B. The results of image processing are forwarded to software output in the form of a website.

The website is the output of this research system. The site contains the condition of the parking lots that have been detected by the camera, then also displays the number of parking spaces available at the AH Polinema Building Parking. The results of detecting the availability of empty parking spaces on two cameras are displayed on one website page. The internet plays an important role in research as a communication medium between systems, so that the system can run well.

To make it easier for system users to understand the overall system workflow and gain a better understanding of how system components interact and influence each other, a system flowchart was created [7]. The following is an image of the system flowchart:

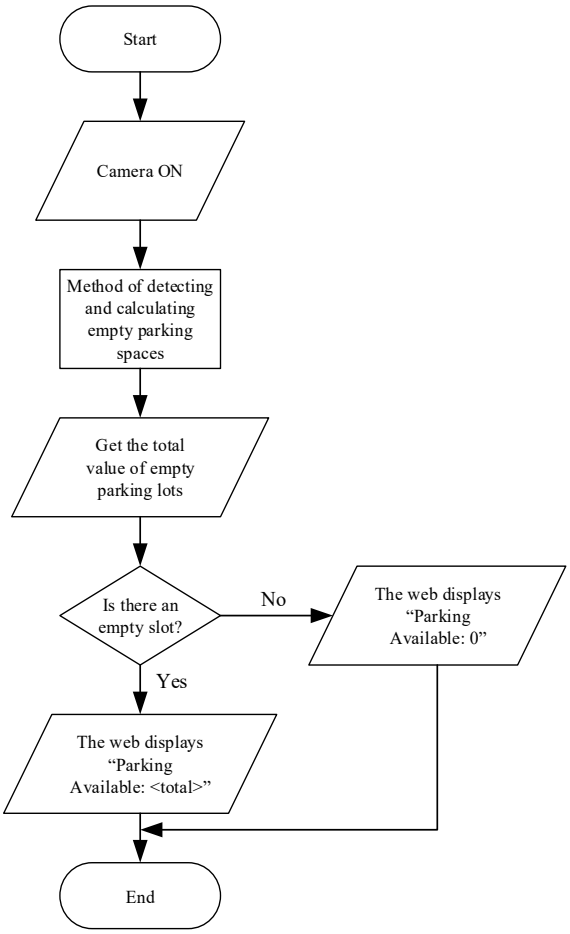

Figure 2. System Flowchart

In Figure 2 which explains the flow of the research system, there is a camera located in the parking lot at the AH Polinema Building to detect and count the number of empty parking slots for cars. The captured images will be processed by a Single Board Computer (SBC) using the YOLOv5 method. The YOLOv5 method is tasked with detecting and counting the number of cars in the AH Polinema Building Parking Lot [8]. Then the system will provide a choice whether there are empty parking slots or not. If there are empty slots then the Website will display "Parking Available: <total>", but if there are no empty slots then the Website will display "Parking Available: 0".

Then in Figure 3 there is a system workflow diagram of the empty parking lot monitoring system at the AH Polinema Building Parking.

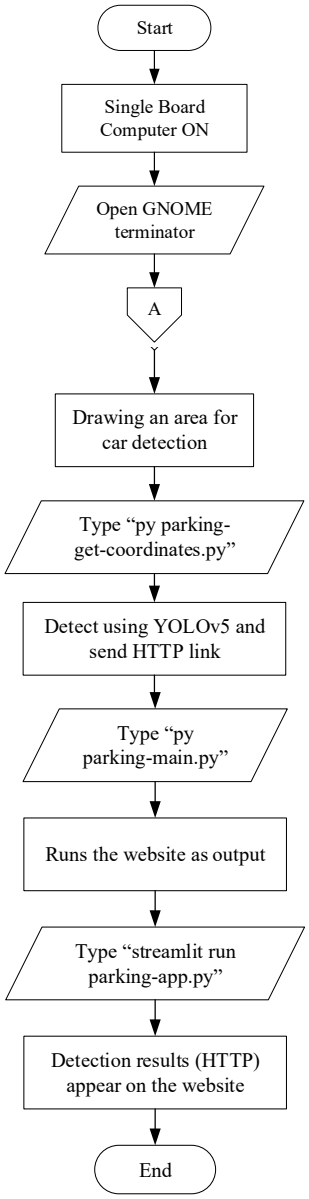

Figure 3. System Workflow Diagram [9]

The following is a description of the diagram:

1. The Single Board Computer (SBC) in the form of a Raspberry Pi 4B is on.
2. Open GNOME Terminator (a Linux terminal emulator programmed in Python). Used to run programs on the Raspberry Pi 4B.
3. Draw an area for car detection, done by typing "py parking-get-coordinates.py" then clicking "Enter". Then a window will appear showing the results captured by two cameras. Double right click on the window to draw a point that is connected to another point. Then left click once to save the coordinate points that have been drawn. Then click “q” to finish and close the window [10].
4. Detect using YOLOv5 and send an HTTP link, used after finishing drawing the coordinate points. Type “py parking-main.py” to start detecting cars and counting the number of empty parking spaces. The results of car detection and calculation of the number of parking spaces will be sent by SBC via a local HTTP link.
5. Run the website as output, by typing "streamlit run parking-app.py" and clicking "Enter" then SBC will integrate the parking-app.py program with the Streamlit website display. A localhost URL link such as localhost:8501 or an IP URL link such as 192.168.3.162:8501 will appear and display the results of integrating the Python language with the Streamlit website.
6. On the website display, images captured by two cameras will appear. There is a bounding box for cars in the AH Polinema Building Parking. The results of calculating the number of empty slots will appear in the form of text on the image captured by the camera.

III. RESULTS AND DISCUSSION

To obtain the results, a test was carried out on the parking space availability detection system at the AH Polinema Building Parking which was carried out by directly monitoring the parking lots. Three types of tests were carried out, namely:

1. Detection accuracy testing, which is carried out repeatedly to obtain a good average accuracy value.
2. Raspberry Pi 4B performance testing was carried out to obtain frame rate (fps) values and CPU usage percentage values.
3. Testing the calculation of the total availability of parking spaces, was carried out to test the system for calculating parking spaces in the AH Polinema Building Parking.

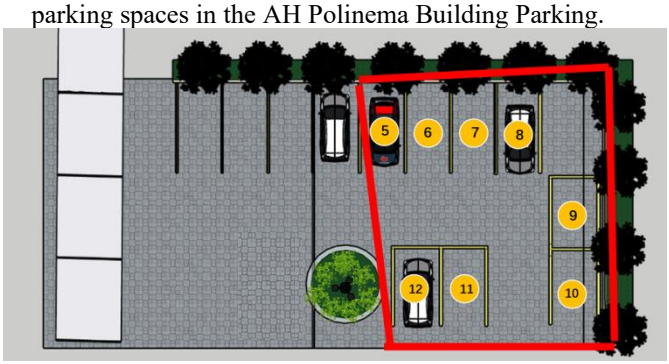

Figure 4. Parking lot slot numbering for Web Camera A

In Figure 4 is an illustrative image for the numbering of parking lot slots used for testing on Web Camera A [11]. And in Figure 5 is an illustrative image for numbering parking lot slots used for testing on Web Camera B.

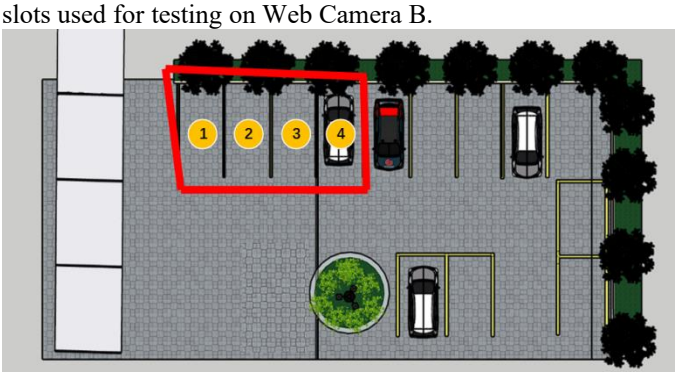

Figure 5. Parking lot slot numbering for Web Camera B

The numbering of parking slots in the AH Polinema Building Parking was carried out to test the accuracy of car detection in the AH Polinema Building parking lot, test the performance of the Raspberry Pi 4B in detecting cars using the YOLOv5 method, and frame rate testing on car detection results in the AH Polinema Building Parking.

A. Detection Accuracy Testing

In this test, car detection accuracy was tested using the YOLOv5 method. This test does not discuss aspects other than accuracy values. This test was carried out in three conditions, namely: morning, afternoon, and evening. These conditions have different intensities of sunlight. With these conditions, it is hoped that there will be no interference with the accuracy of detection [12]. Below is a test of the average accuracy value with three types of conditions using the YOLOv5 method in the AH Polinema Building Parking.

From the Table I, the average value of car detection confidence is obtained in the following conditions: morning. The test was carried out in the morning at 07.30 – 08.00. There are eight cars that fill the parking slots with a total of ten available parking slots. Various accuracy values were obtained in this test. The formula for calculating the average confidence value is as follows [13]:

$$\frac{(The\ sum\ of\ the\ decimal\ values\ for\ all\ tests)}{(total\ testing)} \quad (1)$$

Based on the calculations, the average confidence value for detecting parking space availability in the morning is 65.128%, rounded up to 65.13%. After testing the data in the morning, continue testing the data in the daytime.

From the Table II, the average value of car detection confidence is obtained under conditions: daytime. The tests were carried out during the day at 11.30 – 12.00. There are nine cars that fill the parking slots with a total of twelve affordable parking slots. Various accuracy values were obtained in this test.

Based on the calculations number 1, the average confidence value for detecting parking lot availability during the day is 85.128%, rounded up to 85.13%. After testing the data during the day, continue testing the data in the afternoon.

From the Table III, the average value of car detection confidence is obtained in the conditions: afternoon. The tests were carried out during the day at 15.00 – 15.30. There are three cars that fill the parking slots with a total of twelve affordable parking slots. Various accuracy values were obtained in this test.

Based on the calculations number 1, the average confidence value for detecting parking lot availability in the afternoon is 87.722%, rounded up to 87.7%. From the detection accuracy test, it can be concluded that the YOLOv5 method has sufficient capabilities to overcome the problems in this research.

B. Raspberry Pi 4B Performance Testing

In this sub-chapter, tests are carried out to measure the frame rate and the amount of CPU usage on the computer when detecting cars using YOLOv5. Testing was carried out using two different CPUs, namely BCM2711B0 (Linux) and Ryzen 3 5300U with Radeon Graphics (Windows) [11]. Testing on the Ryzen 3 5300U with Radeon Graphics aims to benchmark the performance of the BCM2711B0. Table IV is test to measure the frame rate (fps) when using one Web Camera and two Web Cameras.

TABLE I
ACCURACY TESTING IN THE MORNING

Car confidence value per slot													Total confidence	Average confidence
1	2	3	4	5	6	7	8	9	10	11	12	13		
-	88 %	-	-	-	88 %	80 %	-	82 %	92 %	93 %	-	-	65,4	65,13%
-	80 %	-	-	-	87 %	89 %	-	81 %	85 %	88 %	-	-	63,75	
-	93 %	-	-	-	86 %	92 %	-	85 %	82 %	88 %	-	-	65,75	
-	89 %	-	-	-	89 %	93 %	-	80 %	90 %	88 %	-	-	66,12	
-	89 %	-	-	-	86 %	83 %	-	86 %	88 %	85 %	-	-	64,62	

TABLE II
ACCURACY TESTING IN THE DAYTIME

Car confidence value per slot													Total confidence	Average confidence
1	2	3	4	5	6	7	8	9	10	11	12	13		
-	95 %	86 %	80 %	-	78 %	84 %	82 %	-	-	83 %	83 %	-	83,9	**85,13%**
-	80 %	86 %	83 %	-	92 %	-	80 %	-	-	85 %	84 %	-	84,2	
-	80 %	83 %	83 %	-	80 %	95 %	82 %	-	-	84 %	82 %	-	83,6	
-	80 %	80 %	82 %	-	86 %	90 %	85 %	-	-	80 %	82 %	-	83,1	
-	89 %	86 %	86 %	-	80 %	85 %	90 %	-	-	85 %	80 %	-	85,1	
-	83 %	80 %	94 %	-	82 %	93 %	86 %	-	-	89 %	85 %	-	86,5	
-	92 %	83 %	84 %	-	95 %	-	87 %	-	-	92 %	93 %	-	89,4	
-	81 %	84 %	93 %	-	86 %	-	89 %	-	-	82 %	88 %	-	86,1	
-	83 %	84 %	92 %	-	90 %	80 %	82 %	-	-	80 %	85 %	-	84,5	
-	80 %	84 %	88 %	-	91 %	83 %	88 %	-	-	93 %	87 %	-	86,7	
-	88 %	83 %	84 %	-	80 %	89 %	83 %	-	-	85 %	88 %	-	85	

TABLE III
ACCURACY TESTING IN THE EVENING

Car confidence value per slot													Total confidence	Average confidence
1	2	3	4	5	6	7	8	9	10	11	12	13		
-	84 %	-	-	-	87 %	-	-	-	-	-	-	-	85,5	**87,7%**
-	80 %	-	-	-	85 %	-	-	-	-	-	-	-	82,5	
-	89 %	-	-	-	86 %	-	-	-	-	-	-	-	87,5	
-	84 %	-	-	-	89 %	-	-	-	-	-	-	-	86,5	
-	85 %	-	-	-	94 %	-	-	-	-	-	-	-	89,5	
-	95 %	-	-	-	84 %	-	-	-	-	-	-	-	89,5	
-	93 %	-	-	-	94 %	-	-	-	-	-	-	-	93,5	
-	94 %	-	-	-	85 %	-	-	-	-	-	-	-	89,5	
-	82 %	-	-	-	89 %	-	-	-	-	-	-	-	85,5	

TABLE IV
FRAME RATE PERFORMANCE TESTING

Total Source	CPU	Condition	FPS
1 camera	Ryzen 3 5300U	Without running YOLOv5	30
1 camera	Ryzen 3 5300U	Running YOLOv5	2
1 camera	Ryzen 3 5300U	When detecting the car	2

Total Source	CPU	Condition	FPS
2 camera	Ryzen 3 5300U	Without running YOLOv5	12 12
2 camera	Ryzen 3 5300U	Running YOLOv5	2 2
2 camera	Ryzen 3 5300U	When detecting the car	2 2
1 camera	BCM2711B0	Without running YOLOv5	12
1 camera	BCM2711B0	Running YOLOv5	1
1 camera	BCM2711B0	When detecting the car	1
2 camera	BCM2711B0	Without running YOLOv5	12 12
2 camera	BCM2711B0	Running YOLOv5	1 1
2 camera	BCM2711B0	When detecting the car	1 1

From the Table IV, twelve tests were carried out to measure the frame rate (fps) with three conditions, namely: first running the camera without running the detection process using YOLOv5, second when running the detection process using YOLOv5, and third when YOLOv5 detected a car. Performance testing was carried out on 1 camera using Ryzen 3 5300U and 1 camera using BCM2711B0. Performance on the Ryzen 3 5300U produces a frame rate of 30 fps when not running YOLOv5, 2 fps when running YOLOv5 and 2 fps when a car is detected. Meanwhile, when using the BCM2711B0 it produces a frame rate of 12 fps without running YOLOv5, 1 fps when running YOLOv5 and 1 fps when a car is detected.

Then the detection was carried out with 2 cameras using a Ryzen 3 5300U producing a frame rate of 12 fps on both cameras when not running YOLOv5, when running YOLOv5 it produced a frame rate of 2 fps on both cameras and when a car was detected it produced a frame rate of 2 fps on both cameras. Detection results with 2 cameras using the BCM2711B0 are no different compared to using 1 camera. The performance results of the BCM2711B0 using 2 cameras, when not running YOLOv5 it is 12 fps, when running YOLOv5 it produces a frame rate of 1 fps on both cameras, and when a car is detected, it produces a frame rate of 1 fps on both cameras.

TABLE V
FRAME RATE PERFORMANCE TESTING

Total Source	CPU	Condition	Peak CPU Usage Value
1 camera	Ryzen 3 5300U	Running YOLOv5	35,8%
		Detects car	37,4%
2 camera	Ryzen 3 5300U	Running YOLOv5	40,8%
		Detects car	42,2%
1 camera	BCM2711B0	Running YOLOv5	93,05%
		Detects car	94,7%
2 camera	BCM2711B0	Running YOLOv5	94,525%
		Detects car	95,025%

It can be seen that the performance of the Ryzen 3 5300U CPU is indeed better than the BCM2711B0. There is a close relationship between the CPU combination and the operating system. The operating system and CPU work together to manage and execute commands. Even though the CPU and operating system has a relationship to improve performance, the Raspberry Pi 4B does not have a feature to update the CPU components.

Then the next test was carried out to get the CPU usage value when running YOLOv5 on BCM2711B0 and Ryzen 3 5300U. The resulting CPU usage value will be compared with the frame rate performance results. Table V is a Table of CPU usage testing when running YOLOv5 for the car detection process.

From the Table V, four tests were carried out to measure the peak value of CPU usage under two conditions, namely: first when running the detection process using YOLOv5 on the BCM2711B0 and Ryzen 3 5300U, second when the YOLOv5 method detected a car in the parking lot on the BCM2711B0 and Ryzen 3 5300U. This test was carried out with a variety of sources, namely: one webcam and two webcams.

Based on the performance test, the peak value of BCM2711B0 CPU usage is the result of calculations from quad core CPU performance. By relying on the performance of four CPU units, the CPU usage percentage value can reach a maximum of 400% [14]. Therefore, percentage calculations are carried out in the form of 0-100 percent. The following is the formula used for this calculation:

$$\frac{Uses\ 4\ CPU\ units}{max.usage\ of\ 4\ CPU\ units} \times 100\% \quad (2)$$

by using this formula, the percentage value for BCM2711B0 CPU usage is obtained with a percentage range of 0-100 percent. It can be concluded that the performance on the Ryzen 3 5300U is better than the BCM2711B0 on the Raspberry Pi 4B. Therefore, the small frame rate value when running on the BCM2711B0 is caused by large CPU usage.

C. Testing the Calculation of Total Parking Space Availability

In this sub-chapter, a calculation system is tested by simulating four empty parking lots with two cameras. This test was carried out using a Ryzen 3 5300U CPU and BCM2711B0 running on a Raspberry Pi 4B. Information regarding vacant land is located in the image as in Figure 4.6:

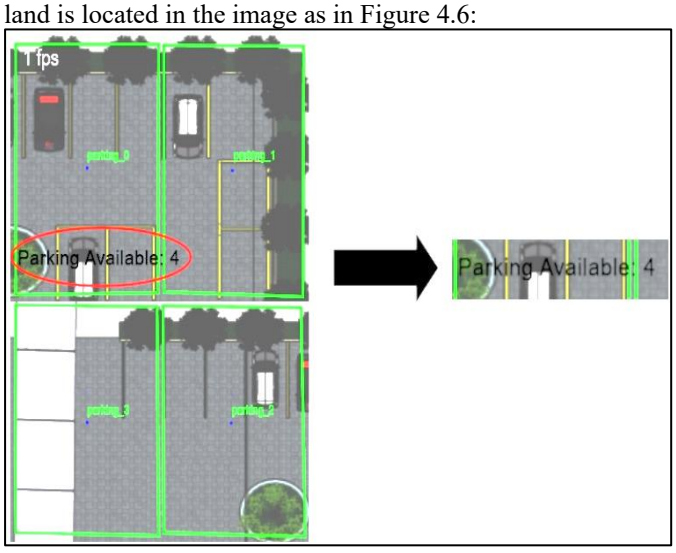

Figure 6. Place Information About Empty Parking Lots

Information regarding the total empty parking spaces is displayed on the website display as system output. The Table

VI is the test results from calculating the total availability of parking spaces:

TABLE VI
FRAME RATE PERFORMANCE TESTING

No.	Total Obejcts	Jumlah Slot (simulasi)	Total Slot Kosong
1	1 car	4 slots	3 slots
2	1 car	4 slots	3 slots
3	2 cars	4 slots	2 slots
4	2 cars	4 slots	2 slots
5	2 cars	4 slots	2 slots
6	3 cars	4 slots	1 slot
7	3 cars	4 slots	1 slot
8	4 cars	4 slots	0 slot

D. Website Testing as an Output System

This test is carried out to test the system output in the form of a website. The website used as system output is the result of integration of the Python language. The tool used to integrate the Python language is Streamlit.

There is system output in the form of a local website display. The website provides an information system regarding the number of parking spaces available. The information system is located at the bottom left of webcam video A. Videos on websites can run in real-time with an average FPS value of one second. The display of parking space availability runs automatically. So, if a car enters the parking area, the availability of parking space will decrease.

E. Data analysis

The results of testing the tool obtained an average confidence value from testing in the morning and afternoon. and in the afternoon, namely 65.13%, 85.13% and 87.7%. When detecting cars in the morning, there is a hardware limitation on one of the webcams. The weakness of the webcam is that it has a small viewing distance or range, so it cannot detect three parking lot slots. Therefore, during the daytime test, a change was made to the webcam, which initially could not detect three parking slots to one parking slot so that twelve parking lots could be seen on two webcams.

Then, in performance testing on the Raspberry Pi 4B, the frame rate and CPU usage values were obtained. In frame rate performance testing, an average value of 1 fps was obtained. This is influenced by CPU usage performance when carrying out the detection process using the YOLOv5 method. In Figure 4.7, you can see that the CPU usage on the Raspberry Pi 4B is 372.3%, equivalent to 93.05% when running the detection process using the YOLOv5 method.

The frame rate value can be optimized by reducing the video resolution that will be detected by the YOLOv5 method. By reducing the video resolution, CPU performance will decrease [15]. However, video resolution that is too small causes a lack of detection accuracy. So, in this research the video resolution used in this research is 560p.

Then, the total availability calculation is displayed in the car detection result frame using the YOLOv5 method as in Figure 4.6. The slot calculation results ran smoothly and did not experience any problems. Assuming the total parking space in the AH Polinema Building Parking is twelve slots, it is adjusted to the range or viewing distance of the webcam. The total parking area will decrease automatically when it detects a car in the parking area [16]. The output of the parking space availability detection system runs smoothly using a Streamlit-based website.

IV. CONCLUSION

Based on the test results of the Parking Space Availability Detection system in the AH Polinema Building Parking, it can be concluded that: This research was carried out in two stages, the first was designing and designing a parking lot monitoring system using the YOLOv5 method, and the second was implementing the system in the AH Polinema Building Parking. The implementation process took approximately one month and was installed on a light pole in the middle of the parking lot. Object detection was successfully carried out in real-time. YOLOv5 can detect objects accurately but does not run optimally on the Raspberry Pi 4B. Changing video resolution for detection using the YOLOv5 method can affect the accuracy value. During the two days of testing, it was found that the average confidence value in the afternoon was better than in the morning and afternoon. The value of the parking area calculation is going well. In the parking space calculation test, the test was carried out by filling the parking space slots. Information regarding the total parking area is located at the bottom left of video display A on the website. The video detection results that appear on the website run well. The video runs according to the CPU computing capabilities of the Raspberry Pi 4B. The website is used as the output of the parking lot monitoring system using the YOLOv5 method in the AH Polinema Building Parking. The framework used to create websites is Streamlit. There were several obstacles during the implementation process, for example the wrong choice of camera hardware, the frame rate value was too small, and limited land to install the equipment in the parking lot of the AH Polinema Building. After various kinds of testing, the system can run and can be implemented on lamp posts in the AH Polinema Building Parking. Limitations were found on the BCM2711B0 CPU used by the Raspberry Pi 4B, this CPU was less than optimal in carrying out the detection process on two cameras using the YOLOv5 method. The processing power of the CPU can be optimized by reducing the video resolution, but at the cost of reducing the accuracy value.

REFERENCES

[1] S. Sugianto dan M. A. & Kurniawan, “Tingkat Ketertarikan Masyarakat terhadap Transportasi Online, Angkutan Pribadi dan Angkutan Umum Berdasarkan Persepsi,” Jurnal Teknologi Transportasi dan Logistik, vol. 1, no. 2, pp. 51-58, 2020.

[2] D. I. Mulyana dan M. A. & Rofik, “Implementasi Deteksi Real Time Klasifikasi Jenis Kendaraan Di Indonesia Menggunakan Metode YOLOV5,” Jurnal Pendidikan Tambusai, vol. 6, no. 3, pp. 13971-13982, 2022.

[3] S. Jupiyandi, F. R. Saniputra, Y. Pratama, M. R. Dharmawan dan I. & Cholissodin, “Pengembangan

deteksi citra mobil untuk mengetahui jumlah tempat parkir menggunakan CUDA dan modified YOLO," Jurnal Teknologi Informasi dan Ilmu Komputer, vol. 6, no. 4, pp. 413-419, 2019.

[4] S. E. Anjarwani, H. I. Illina dan N. & Agitha, "Sistem Informasi Daya Tampung Area Parkir Pada Pusat Perbelanjaan Untuk Meningkatkan Layanan Penggunaan Parkir (Studi Kasus: Lombok Epicentrum Mall)," Journal of Computer Science and Informatics Engineering, vol. 6, no. 1, pp. 1-9, 2922.

[5] A. Amin dan M. W. & Kasrani, "Penerapan Metode Yolo Object Detection V1 Terhadap Proses Pendeteksian Jenis Kendaraan Di Parkiran," Jurnal Teknik Elektro Uniba, vol. 6, no. 1, pp. 194-199, 2021.

[6] Q. Aini, N. Lutfiani, H. Kusumah dan M. S. & Zahran, "Deteksi dan Pengenalan Objek Dengan Model Machine Learning: Model Yolo," CESS (Journal of Computer Engineering, System and Science), vol. 6, no. 2, pp. 192-199, 2021.

[7] X. Cong, S. Li, F. Chen, C. Liu dan Y. & Meng, "A Review of YOLO Object Detection Algorithms based on Deep Learning," Frontiers in Computing and Intelligent Systems, vol. 4, no. 2, pp. 17-20, 2023.

[8] D. Manajang, S. Sompie dan A. Jacobus, "Implementasi Framework Tensorflow Object Detection API Dalam Mengklasifikasi Jenis Kendaraan Bermotor," Jurnal Teknik Informatika, vol. 15, no. 3, pp. 171-178, 2021.

[9] S. Liu, L. Qi, H. Qin, J. Shi dan a. J. Jia, "Path Aggregation Network for Instance Segmentation," dalam IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 2018.

[10] S. Ratna, "Pengolahan Citra Digital Dan Histogram Dengan Phyton Dan Text Editor Phycharm," Technologia: Jurnal Ilmiah, vol. 11, no. 3, pp. 181-186, 2020.

[11] SketchUp, "SketchUp for Web," SketchUp, July 2019 Version 1.3 . [Online]. Available: https://app. sketchup.com/. [Accessed 22 03 2023].

[12] R. Pi, "Operating System Images," Raspberry Pi, [Online]. Available: https://www.raspberrypi.com/ software/operating-systems/. [Accessed 17 09 2023].

[13] Glenn Jocher, "ultralytics/yolov5: v6.0 - YOLOv5n \'Nano\' models, Roboflow integration, TensorFlow export, OpenCV DNN support," dalam Zenodo, 2021.

[14] M. Carranza-García, J. Torres-Mateo, P. Lara-Benítez dan J. & García-Gutiérrez, "On the performance of one-stage and two-stage object detectors in autonomous vehicles using camera data," Remote Sensing, vol. 13, no. 1, p. 89, 2021.

[15] C. -Y. Wang, H. -Y. M. Liao, Y. -H. Wu, P. -Y. Chen, J. -W. Hsieh dan a. I. -H. Yeh, "CSPNet: A New Backbone that can Enhance Learning Capability of CNN," dalam IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Seattle, WA, USA, 2020.

[16] N. Jannah, S. A. Wibowo dan T. S. Siadari, "Eksploitasi Fitur Untuk Peningkatan Kinerja Deteksi Objek Berbasis Pada Pesawat Tanpa Awak," dalam eProceedings of Engineering, 2022.