Accredited SINTA 2 Ranking
Decree of the Director General of Higher Education, Research, and Technology, No. 158/E/KPT/2021
Validity period from Volume 5 Number 2 of 2021 to Volume 10 Number 1 of 2026

Published online at: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 8 No. 3 (2024) 413 - 422 e-ISSN: 2580-0760

Integration of YOLOv5 Algorithm and OpenCV in Innovative Smart Parking Management Approach

Akmal Hidayah[1,6*], Sitti Zuhriyah[2], Billy Eden William Asrul[3], Yuyun[4,5], Esa Prakasa[6]
[1,2,3,4]Dept. of Computer System, Faculty of Computer Science, Handayani University, Makassar, Indonesia
[5]Natural Language Processing, Research Center for Data and Information Sciences (BRIN) Bandung, Indonesia
[6]Human Computer Interaction and Visualization, Research Center for Data and Information Sciences (BRIN) Bandung, Indonesia.
[1]akmalhidayat826@gmail.com, [2]zuhriyah@handayani.ac.id, [5]billy@handayani.ac.id, [4,5]yuyu010.@brin.go.id, [4]esap001@brin.go.id

Abstracts

The problem of automatic parking lot identification and vehicle detection in open areas is becoming increasingly important due to the growth in the number of vehicles in Indonesia, particularly in big cities, resulting in difficulties in finding parking spaces during peak hours. In this condition, the_motorists often have to compete for parking spaces. This research aims to develop a smart parking system that integrates YOLOv5 and OpenCV algorithms. This approach combines both algorithms thoroughly to identify parking spaces and detect vehicles in real-time in diverse parking scenarios. It is conducted in an open area with reference to parking conditions at the BRIN Bandung office. This study collected data from three different parking lot conditions, namely empty, partially occupied, and full. In each condition, the system successfully detected the parking lots and vehicles accurately. The novel contribution of this research is the development of a smart parking system that uses an integrated approach, providing an effective solution for the challenges of parking lot availability and vehicle detection. By utilizing the advantages of both algorithms, we successfully created a system that can identify the parking spaces and detect the vehicles accurately and efficiently under various parking circumstances. Thus, this research makes a significant contribution to the development of smart and adaptive parking management technology.

Keywords: smart parking; YOLOv5; OpenCV; vehicle detection; parking space identification

How to Cite: A. H. Hidayah, Sitti Zuhriyah, Billy Eden William Asrul, Y. Yuyun, and Esa Prakasa, "Integration of YOLOv5 Algorithm and OpenCV in Innovative Smart Parking Management Approach", *J. RESTI (Rekayasa Sist. Teknol. Inf.)* , vol. 8, no. 3, pp. 413 - 422, Jun. 2024.
DOI: https://doi.org/10.29207/resti.v8i3.5728

1. Introduction

Object recognition has advanced significantly in the realm of computer vision, particularly in community-oriented endeavours. Numerous methodologies derived from research outcomes have been extensively utilized to support communal activities. As image data continues to grow, the prompt identification of objects within these images becomes increasingly essential [1]. With the increasing amount of image data, the need for object recognition becomes progressively crucial for facilitating the identification of objects with the images quickly. Object recognition for these images is widely used to recognize various elements, such as texts [2], vehicles [3], fingerprints [4], faces [5], and others. This research was conducted in response to the increasing number of vehicles in Indonesia, more specifically in big cities, which makes it difficult to find parking spaces during peak hours. In Indonesia, as the price of vehicles becomes more affordable, their numbers keep growing and the need for more parking spaces_[6]. For the places, such as shopping malls, schools, airports, and others, competition for parking spaces becomes fierce when parking spaces are available [7]. Currently, most of the parking lots do not have a structured system and are mostly managed manually, which is often less efficient. The most common problem in parking lots is the time-wasting search for available parking spaces. The users often have to go around the parking lot until they find an empty parking space [8]. This situation is particularly evident at the National Innovation Research Institute Bandung office, where the parking system still relies on manual handling by security guards. Inefficient parking management can lead to traffic congestion, prolonged time to find parking spaces and inconvenience for office users.

Received: 22-02-2024 | Accepted: 28-04-2024 | Published Online: 22-06-2024

In addressing these challenges, the adoption of object recognition technologies in computer vision, especially by utilizing deep learning technologies, has emerged as a promising solution to detect vehicles and assess parking space availability. Previous research has emphasized utilizing AlexNet's Convolutional Neural Network (CNN) [9] to identify parking space availability with a high degree of precision. In another approach, several studies have focused on evaluating the performance of the YOLO algorithm in detecting parking lot availability. For example, [10] proposed a method using AlexNet architecture to detect available parking spaces based on vehicle position. They conducted experiments using a large dataset covering various lighting conditions and vehicle types. The testing process was conducted by dividing the dataset into training data and test data, and they recorded a detection accuracy of 98.0%, precision of 98.0%, and recall of 97%. However, this research has limitations in terms of scalability and detection speed, especially when applied in real-time situations in crowded parking areas. In addition, [11] proposed a more sophisticated approach using the Tiny-YOLO method to detect vehicles in streaming videos in real time. They used a well-labeled dataset to train the model and measured the performance of their algorithm using measures of accuracy, detection speed, and computational resource usage. The test results showed high accuracy, reaching 97.5%-98% on the test data, with detection speed reaching 28-38 FPS (Frames per Second). Lastly, research[12] introduced an approach using Blackbox methods that allows the execution of modules on the system and direct observation of the results. They adopted a more experimental approach by conducting a series of tests on an actual parking system to validate the reliability and effectiveness of their approach. The test results show that the Blackbox method has a satisfactory accuracy rate as expected. However, this research has limitations in scalability and integration with wider parking systems.

However, these studies have not delved into the specific process of obtaining parking positions as well as vehicle detection. Therefore, the purpose of this research is to develop an advanced parking system that integrates the YOLOv5 algorithm and OpenCV. Through the application of effective object recognition technology, it is expected that this system can provide real-time parking space availability information with improved accuracy and efficiency. The new contribution of this research is the development of a smart parking system that uses an integrated approach. By utilizing the advantages of the Yolo algorithm and OpenCV library, this research succeeded in creating a system that can identify parking spaces and detect vehicles accurately and efficiently in various parking situations. Thus, this research has made a significant contribution to the development of intelligent and adaptive parking management technology, which has the potential to increase the effectiveness of vehicle detection and automatic parking lot labelling. In addition, it is expected that this research can also provide a greater effective solution in addressing the issue of parking space availability, optimizing resource utilization, and improving the parking experience for office users.

2. Research Methods

This research adopts a quantitative approach to collect and analyze data related to parking space availability at the BRIN Bandung office. The selection of a quantitative method is selected to enable objective measurements and essential statistical analysis in understanding the effectiveness of the smart parking system. The research process was detailed in three stages, encompassing data collection, data analysis, examine the use of images and videos as input for the testing.

Data collection was conducted by taking samples from one of the parking areas at the BRIN Bandung office every day for a month. The volume of the collected data exceeded 50 images in (.jpg) format and 10 videos in (.mp4) format, which included full, empty, and partially occupied parking conditions. The sample size was chosen based on the prior evaluations of parking activities at the BRIN Bandung office, and images were captured using a Redmi Note 10 smartphone camera.

The collected dataset was analyzed using the YOLOv5 algorithms and OpenCV library to achieve accurate vehicle detection and automatic parking space labelling.

Data processing involved utilizing the Torch model from YOLOv5 to detect vehicles in each image. Furthermore, images were converted to grayscale to facilitate further analysis. This process included converting images from colour to grayscale format, ensuring data processing consistency, and minimizing dimensions required for analysis.

2.1 OpenCV

Computer Vision is a concept that mimics the ability of human vision using software and hardware on computers, it integrates knowledge in the fields of computer science, electrical engineering, mathematics, physiology, biology, and cognitive science to understand and simulate the process of human vision [13]. The term "OpenCV" stands for "open-source computer vision." Its architecture consists of software, databases, and pre-programmed plugins with support for integrating computer vision applications [14]. It is one of the most widely used toolkits among a large developer community. OpenCV is renowned for its extensive usage in building real-world applications for industrial purposes. It follows programming languages such as C/C++, Python, and Java and can be employed to develop computer vision software for desktop and mobile platforms like Windows, Linux, macOS, Android, and iOS.

The latest releases are OpenCV-4.5.2 and OpenCV-3.4.14. It is free, open-source, user-friendly, and easy to install. It is designed for numerical productivity with a significant emphasis on real-time applications. While the initial version used the C programming language, its success increased with the release of Version 2.0, which introduced the implementation in C++[15]. C++ is used to introduce new features. The platform includes the latest distribution updates (version: 4.5.2) as well as older iterations. Photos must be in BGR or Grayscale format to be displayed or saved by using OpenCV. Otherwise, undesired results may occur [16]. The use of computer vision technology or OpenCV is more economical because one camera can classify many parking spaces simultaneously [17]. The parking space detection method in this research uses image processing techniques through the OpenCV library.

2.2 YOLOv5

YOLOv5, the fifth-generation object recognition algorithm, was introduced in April 2020. Since it is broadly similar to previous generations of YOLO, YOLOv5 stands out with its ability to be run using only the Python programming language. The advantages of using Python include the ease of its installation and integration on Internet of Things (IoT) devices. Architecturally, YOLOv5 adopts updates that are largely similar to YOLOv4, with few differences found in this study. However, detailed documents on the YOLOv5 architecture are not widely available yet; the developer only publishes the code repository on GitHub and performs updates there. Based on the code analysis in the .yaml file, the YOLOv5 model uses a Focus structure on the backbone and CSP network, supplemented with SPP on the neck and PANet, and uses GIoU-loss on the model head [18]. Glenn Jocher's integration adds an anchor box selection process into YOLOv5, allowing the model to automatically learn the best anchor boxes in the dataset used during the training process [19]. YOLOv5 was chosen for this research because it is equipped with a good balance between accuracy and speed, which surpasses YOLOv3 and MobileNet-SSDV [20].

In terms of network depth and feature map width, YOLOv5 can be categorized into four models: YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x [21]. The performance of these models is assessed and documented on the MS COCO test dataset, as illustrated in Table 1.

The performance comparisons reveal that YOLOv5s exhibits the fastest processing speed, while YOLOv5x achieves the highest detection accuracy among these models. Notably, all four models share an identical network structure, encompassing input, backbone, neck, and prediction components. This paper introduces a method to achieve a lightweight model design, minimizing FLOPs, parameter count, and model size while preserving detection accuracy. Consequently, YOLOv5s is chosen as the benchmark model for further enhancements [21].

Table 1. The evaluation of various YOLOv5 model performance [21]

Model	Size(pixels)	mAP (0.5)	mAP(0.5:0.95)	Speed v100 (ms)	Params (M)	FLOPs (G)
YOLOv5s	640 × 640	55.4	36.7	2.0	7.3	17.0
YOLOv5m	640 × 640	63.3	44.5	2.7	21.4	51.3
YOLOv5l	640 × 640	66.9	48.2	3.8	47.0	115.4
YOLOv5x	640 × 640	68.8	50.4	6.1	87.7	218.8

Figure 1 depicts the structure of YOLOv5s as the representative model, with detailed descriptions of each network component.

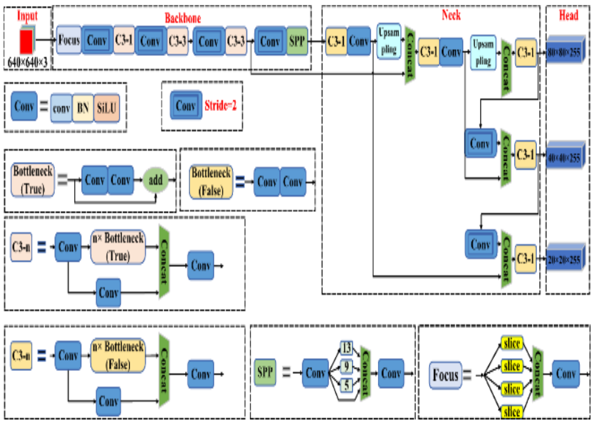


Figure 1. Database Mirroring Architecture[21]

The identical network structure comprises input, backbone, neck, and prediction components. This paper introduces a method to achieve a lightweight model design, minimizing FLOPs, the number of parameters, and the model's size while ensuring detection accuracy. Consequently, YOLOv5s is chosen as the benchmark model and further enhanced. Figure 1 illustrates the structure of YOLOv5s as a representative model, with detailed descriptions of each network component provided[21].

In order to enhance the accuracy and speed of vehicle detection in this study, we proposed a lightweight vehicle detection network model based on YOLOv5s. The schematic diagram of the improved YOLOv5s structure can be seen in Figure 2. The parameters in each box indicate the output size of the feature map of the previous layer as the input size for the feature map of this layer, the output size of the feature map of this layer, the size of the convolution kernel, and the size of the step. In this study, we embed the CBAM module (bright yellow) after the C3-3 and SPP modules in the main YOLOv5s network. We also introduced the C3Ghost module (green) and Ghost module (light grey) into the YOLOv5s neck network [21].

The CBAM module aims to improve the feature extraction capability of the network by highlighting the critical information for vehicle detection, thereby enhancing vehicle accuracy in various scenarios.

Additionally, the C3Ghost and the Ghost modules aim to further reduce the number of parameters and FLOPs of the model. The role of YOLOv5 in this study is to detect facial objects in images, with the model used was the YOLOv5s model. These refinements have been integrated into our innovative smart parking management approach, combining the YOLOv5 algorithm and OpenCV.

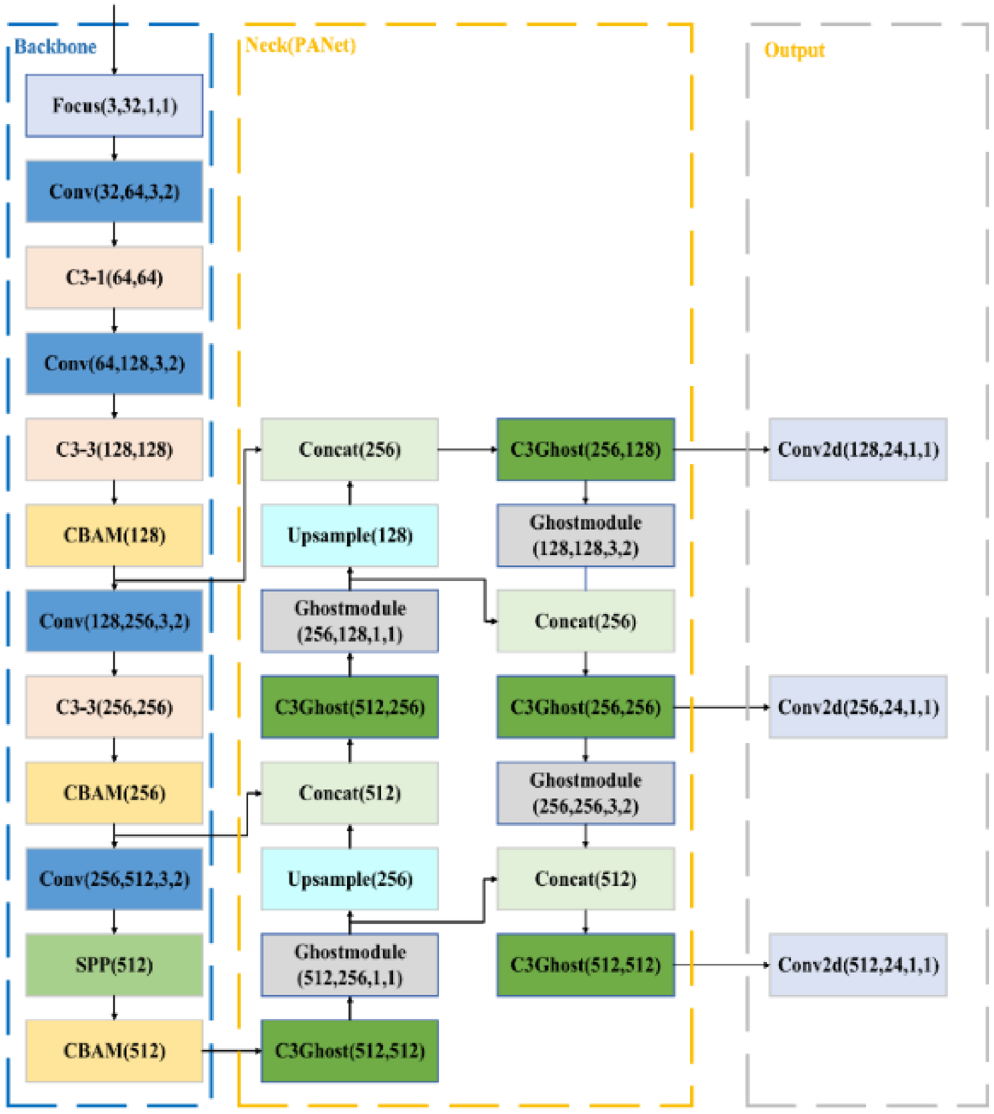


Figure 2. Database Mirroring Architecture[21]

The capturing of images and videos unfolds as a pivotal aspect in our investigation, employing the Redmi Note 10 mobile camera across diverse parking scenarios. Videos have been meticulously recorded to extract dynamic and situational data within the parking area. This approach enables us to comprehensively probe into and analyze a holistic portrayal of varied parking conditions. Manual labelling parking spaces is a crucial step for efficient parking vacancy detection in our innovative smart parking management approach. The dataset provides labelled images with coordinate information about parking slots in XML files. However, in our context, this data is not applicable as our designed system operates in real-time scenarios. Thus, manual marking of parking spaces is performed at each parking location.

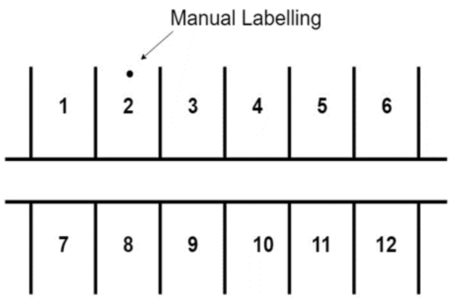


Figure 3. Illustration Rectangle labels are manually defined to represent parking lots.

Initially, resizing is accomplished using the OpenCV library, adjusting the image to the desired dimensions, such as 1020px × 600px, to ensure a focused image.

Resizing is crucial for creating labels on parking slots using rectangles. Subsequently, the identified positions of parking slots are visually represented on the image with magenta-coloured rectangles (255, 0, 255), outlining the boundaries of each parking slot. Figures 3. and 4. depict the process of creating rectangles on a photo of an empty parking lot in our smart parking management approach.

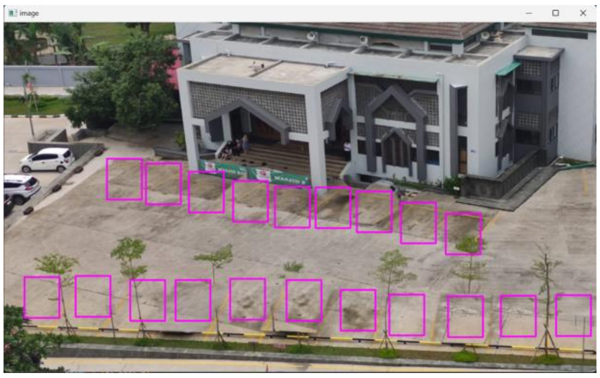


Figure 4. Rectangle labels are manually defined to represent parking lots.

2.3 Parking Area Mapping

In the parking area mapping phase, we applied an innovative approach using the OpenCV library to gain a comprehensive understanding of parking space availability. The process begins by converting the image into grayscale and black-and-white formats. Crucial white line detection is performed using the OpenCV library, specifically the cv2. Adaptive Threshold() function, to generate a binary representation facilitating the identification of empty areas. This adaptive thresholding method allowed us to dynamically calculate the threshold value based on the local neighborhood of each pixel. Two adaptive methods are available: cv2.ADAPTIVE_THRESH_MEAN_C, where the threshold value is the mean of the neighborhood area and cv2.ADAPTIVE_THRESH_GAUSSIAN_C, where the threshold value is the weighted sum of neighborhood values with weights was calculated using a Gaussian window. As part of the evaluation, the system delineates each successfully identified parking space. The cvzone.putTextRect() method is utilized to highlight each parking space by displaying the number of vehicles. The colour and thickness of the text are dynamically adjusted based on the number of detected vehicles. For example, if the number of vehicles is below the critical threshold of 583, the text will be displayed in green with the RGB format (0, 255, 0), indicating parking space availability. Conversely, if the number of vehicles exceeds the threshold, the text colour changes to red with the RGB format (0, 0, 255), representing that the parking space is occupied as Pseudocode 1.

```
function checkParkingSpace(imgPro):
    spaceCounter = 0
    for each pos in posList do:
        x, y = pos
        imgCrop = cropImage(imgPro, y, y+height, x, x+width)
        count = countNonZero(imgCrop)
        putTextRect(img, str(count), (x, y+height-3), scale=1, thickness=2, offset=0, colorR=(0, 255, 0))

        if count < 583 then:
            color = (0, 255, 0)
            thickness = 5
            spaceCounter += 1
        else:
            color = (0, 0, 255)
            thickness = 2
        rectangle(img, pos, (pos[0] + width, pos[1] + height), color, thickness, 2)
    putTextRect(img, 'Empty: ' + spaceCounter + '/' + len(posList), (100, 50), scale=3, thickness=5, offset=20, colorR=(0, 200, 0))
# Parking space detection
imgGray = convertColor(img, COLOR_BGR2GRAY)
imgBlur = gaussianBlur(imgGray, (3, 3), 1)
imgThreshold = adaptiveThreshold(imgBlur, 255, ADAPTIVE_THRESH_GAUSSIAN_C, THRESH_BINARY_INV, 25, 16)
imgMedian = medianBlur(imgThreshold, 5)
kernel = createKernel((3, 3), uint8)
imgDilate = dilate(imgMedian, kernel, iterations=1)
checkParkingSpace(imgDilate)
```

Pseudocode 1. Pseudocode of Parking Area Mapping

With this detailed approach, the system provides a clear and interactive visualization of parking space availability status, leveraging the advantages of the OpenCV algorithm in image analysis and object detection. The flowchart of the parking area mapping process can be seen in Figure 5.

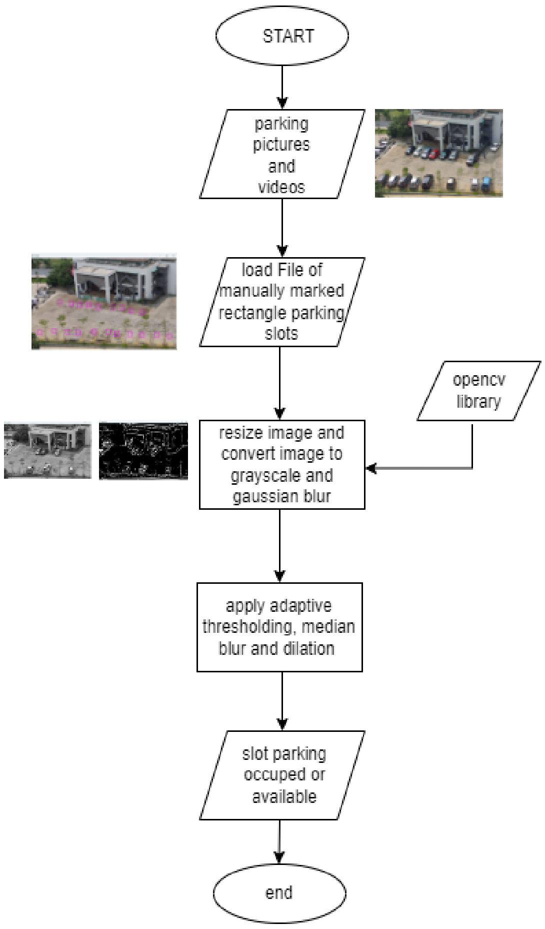

START
parking pictures and videos
load File of manually marked rectangle parking slots
opencv library
resize image and convert image to grayscale and gaussian blur
apply adaptive thresholding, median blur and dilation
slot parking occuped or available
end

Figure 5. Flowchart of the parking area mapping

2.4 Vehicle Detection in Parking Lots

The YOLOv5 model comprises three crucial components: the Backbone, Neck, and Head. Detailed architecture can be seen in Figure 5. The Backbone consists of several Convolutional Cross Stage Partial (CSP) processes for feature extraction from the input image. The advantage of CSPNet lies in its optimal feature extraction capability to enrich information from the image. Additionally, these features will be passed to the Neck to identify objects of the same but different sizes and scales, often referred to as pyramid features. The Neck section of YOLOv5 employs PANet to achieve pyramid features. The final stage is the Head, where the model performs detection and evaluation using the results of object frames and the probability values of each class.

Our innovative approach to vehicle detection using the YOLO (You Only Look Once) algorithm leverages pre-trained models from YOLOv5. We implement advanced configurations to obtain bounding boxes and classify vehicles, including cars. Figure 6 illustrates the workflow of vehicle detection using the YOLOv5s model.

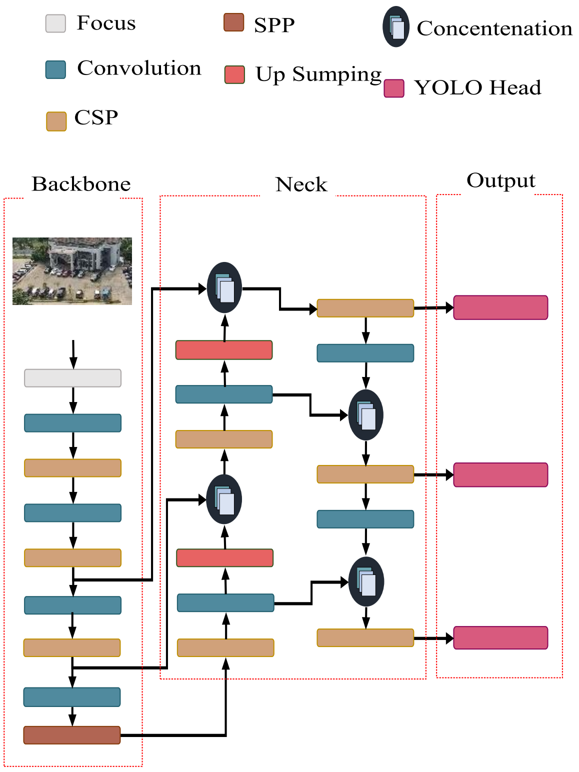


Figure 6. YOLOv5 Architecture for Vehicle Detection in Parking Lots

Subsequently, we utilize this model to detect objects in the input images. The detection results include bounding box coordinates and class labels for each detected object. We iterate over the detection results to retrieve coordinate and class label information for each object. Next, we calculate the centre coordinates of each bounding box for further analysis. If the detected object is a car, we utilize the OpenCV library to draw bounding boxes around it and add a class label (e.g., 'car', 'motorcycle') near the bounding box. Thus, this process enables us to efficiently detect and identify cars in the input images. Through the integration of YOLO for vehicle detection, our system not only serves as an efficient parking management tool but also as a responsive solution to the dynamics of parking space utilization.

2.5 Testing Vehicle and Parking Slot Detection

The testing stage is split into two components: the initial test employs images as input, whereas the subsequent test integrates real-time video. The objective of the image-based testing is to evaluate the accuracy of the YOLOv5 and OpenCV models in detecting vehicles and parking slots. In the initial step of the image-based testing, empty parking lot images are prepared. The program then performs detection, and if no cars are detected, the parking slots are marked in green, indicating that all twenty slots are empty. After the successful detection, the program continues to identify partially occupied parking spaces. The system detects cars with blue bounding boxes and empty ones remain green as seen in Figure 7.

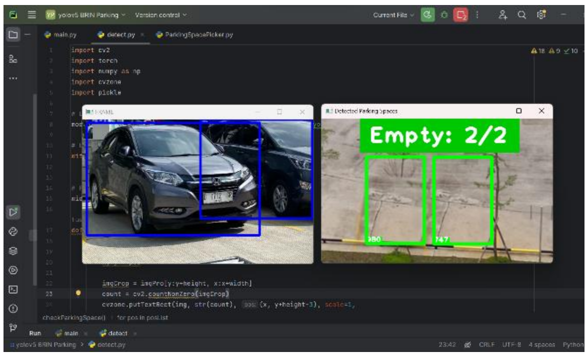

```
import cv2
import torch
import numpy as np
import cvzone
import pickle
```
```
        imgCrop = imgPro[y:y+height, x:x+width]
        count = cv2.countNonZero(imgCrop)
        cvzone.putTextRect(img, str(count), (x, y+height-3), scale=1,
```

Figure 7. Example of bounding box results

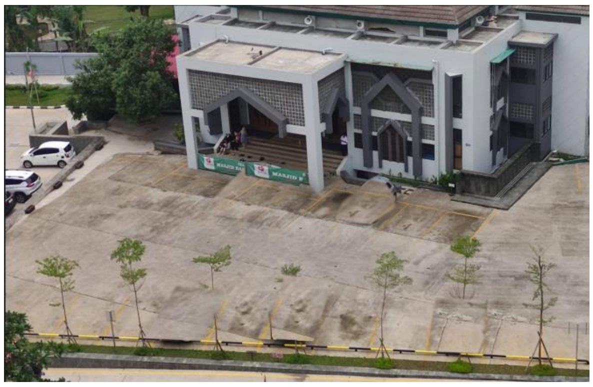
3. Results and Discussions

In this research, the testing of the Innovative Smart Parking Management Approach is divided into 3 stages. The first stage is Parking Area Mapping, the second stage is Vehicle Detection in Parking Lots, and the third stage is the Testing of Vehicle and Parking Slot Detection.

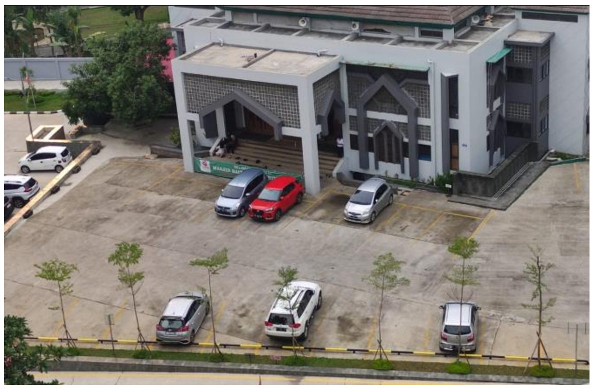
3.1 Results of Parking Area Mapping

In the parking lot mapping testing phase, an image depicting empty, partially occupied, and full parking conditions was involved. This image consists of 20 parking slots identified through the OpenCV rectangle tool for the rectangular slot parking process. The testing data was conducted using the Integrated Development Environment (IDE) PyCharm with the Python programming language.

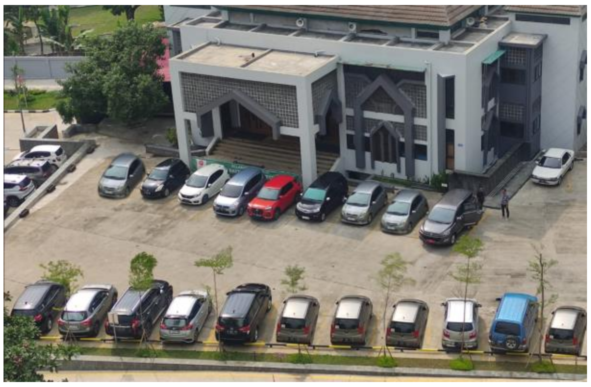
During the parking lot mapping testing, the analysis of results was performed using the checkParkingSpace(imgPro) function, which aims to evaluate the availability of parking spaces in each previously identified slot.



(a)



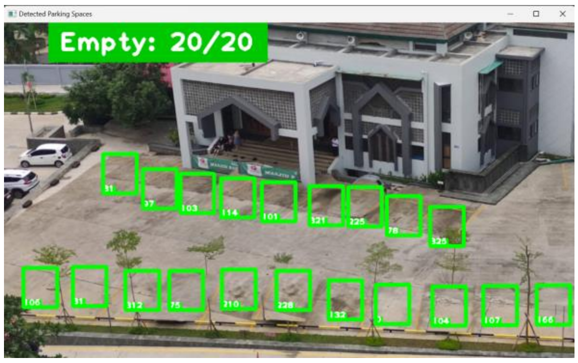
(b)



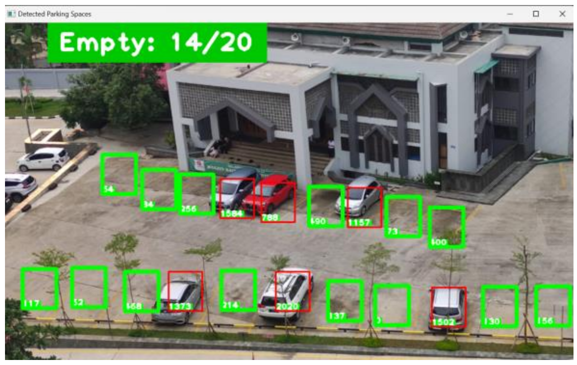
(c)

Figure 8. Evaluation data on the model (a) empty parking lot, (b) partially occupied, and (c) Full

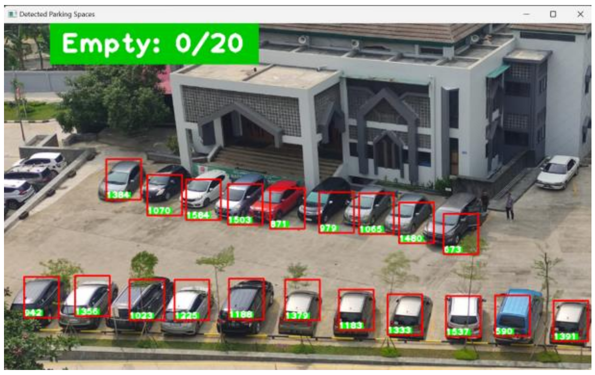
The checkParkingSpace function is responsible for checking the availability of parking spaces in each identified slot. By utilizing the predetermined slot positions, this function analyzes based on the number of pixels detected in each slot. The analysis results provide information on whether a parking slot is empty or occupied. Firstly, it iterates through each predetermined slot position to ensure that each slot is evaluated individually. Then, in each iteration, the input image is cropped according to the boundaries of the respective parking slot. This allows for further analysis of specific parts of the image. Using cv2.countNonZero, the function calculates the number of pixels with non-zero values in the cropped image. This number reflects how much area in the parking slot has been occupied. Each parking slot is visualized on the main image using rectangles. The colour and thickness of the rectangles indicate the availability status of the parking space. If the number of pixels is below 583, it indicates that the parking space is empty, and the rectangle is green. Conversely, if the number of pixels exceeds 583, it indicates that the parking space is occupied, and the rectangle is red. A text is displayed on the image providing statistical information about the number of empty parking spaces out of the total evaluated slots. Figures 8 and 9 illustrate the results of parking lot mapping testing using the OpenCV model.



(a)



(b)



(c)

Figure 9. Results of evaluation data on the model (a) empty parking lot, (b) partially occupied, and (c) Full

3.2 Result of Vehicle Detection in Parking Lots

The vehicle detection results are obtained through training on a dataset of top-down parking lot images, hereafter referred to as the configuration of epochs and batch size. The selection of the best model is based on monitoring the accuracy value at each epoch, and the model with the highest accuracy is chosen. Stochastic Gradient Descent (SGD) is used as the optimizer. In the training process, 14 training data and 2 validation data are used. The training data is utilized to train the model to accurately identify vehicles with occlusion, while the validation data is used to evaluate and validate the generated model during the training process. The performance of the model can be considered good if the recall, mAP0.5, precision, and mAP0.5:0.95 values approach 1, while the box loss, object loss, and class loss values approach 0.

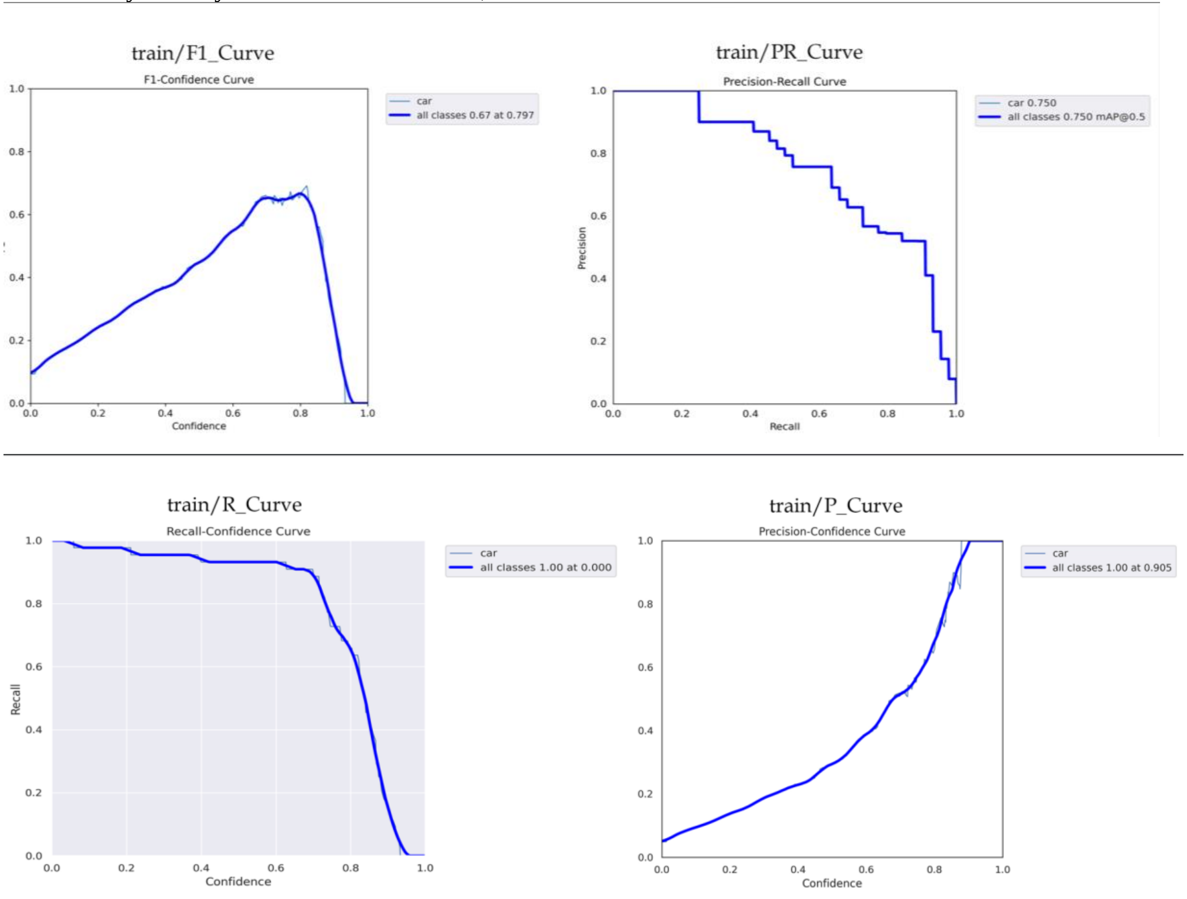


Figure 10. Training Process Results curve

Table 2. Vehicle detection test results

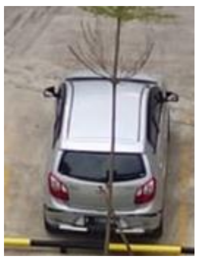
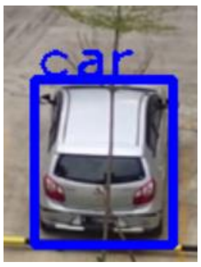
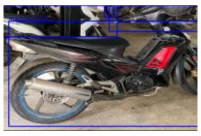
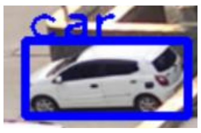
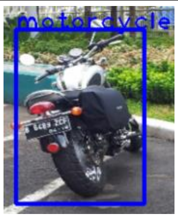
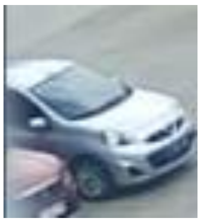
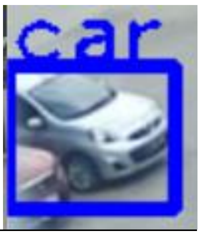
Image Data	Label	Result
	Car	
	Motorcycle	
	Car	
	Motorcycle	
	Car	

Figure 10 illustrates that the Precision of Compilation I increases rapidly and consistently; the compilation results show a faster and more stable increase in recall. Differences in image quality in the Compilation results contribute to variations in the improvement of recall, mAP0.5, precision, and mAP0.5:0.95 values. These differences in image quality can also lead to weighted model performance being worse. Furthermore, in the testing phase, testing was conducted using an AXIOO Slimbook 14 R5 laptop. The testing process was carried out using PyCharm Community Edition 2023.3.1 and utilizing the AMD Ryzen5-3500u GPU. Testing was done by utilizing the best.pt model that has been trained and shows the x and y coordinates where the car is positioned.

The test results present the accuracy of detecting cars with blue labels. Upon closer examination, it was found that the images used to train the real-time system were taken in an open area. This leads to our hypothesis that the accuracy of the system is negatively influenced by the brightness variability commonly encountered in such environments. Table 2. displays examples of still image test results, where the developed software displays the names of the identified labels above the bounding boxes surrounding the recognized individuals. It is important to note that these results are presented using still images.

3.3 Results from Integration Testing between Vehicle Detection and Parking Slots.

The integration testing phase focused on assessing the performance of the vehicle detection system in different parking lot scenarios as seen in Figure 11. Three distinct parking lot conditions were considered: empty, partially occupied, and nearly full. In the case of an empty parking lot, the system marked all parking slots in green if no vehicles were detected. For partially occupied parking lots, detected vehicles were highlighted with blue bounding boxes, while partially occupied parking slots were marked in red. Meanwhile, in scenarios where the parking lot was nearly full, the system identified vehicles and marked partially occupied slots in red, while empty slots remained green. Through these tests, the effectiveness of the vehicle detection system in various parking scenarios was evaluated.

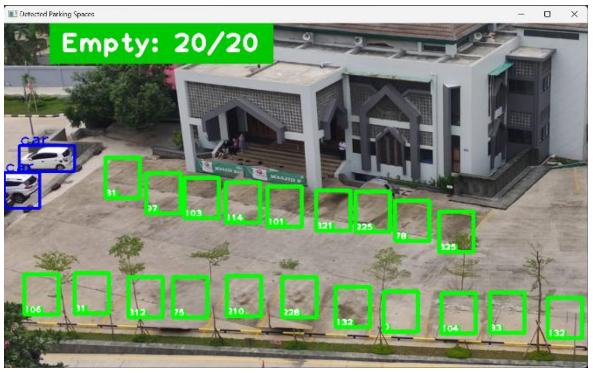
Furthermore, based on the performance results of the parking slot and vehicle counting system shown in Table 4, it can be concluded that the angle selection, data quality, and weather conditions affect the detection of parking slots and vehicles. The results from the proposed algorithm, utilizing the provided data, indicate that 31 vehicles were detected across three parking conditions: empty, partially occupied, and full, along with 20 parking slots being successfully identified. This demonstrates that the proposed algorithm achieves optimal performance, with a vehicle counting calculation percentage of 100%.

Table 3. Test Data Classification Results Based on Integration Results

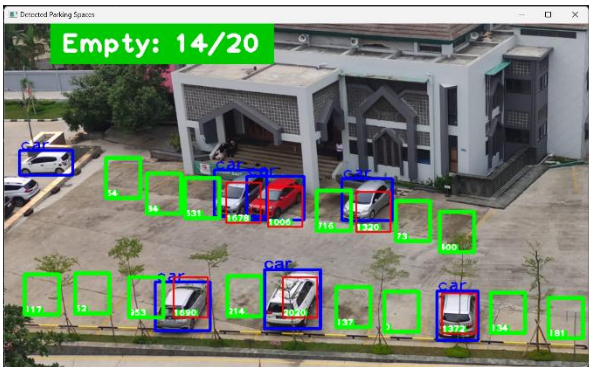
Number of Vehicles Detected	Number of Vehicles Not Detected	Number of detected parking slots	Number of undetected parking slots	Percentage (%)
31	0	20	0	100

Table 4. The calculation results

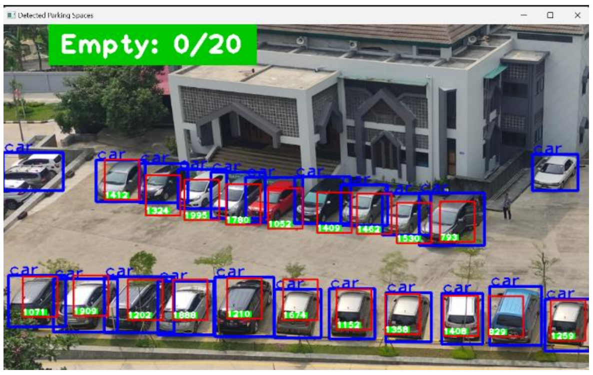
Data Classification Labels	Number of Detections	Detection Results (Actual)	
		Parking Slot	Vehicle
Parking Slot	20	20	31
Vehicle	31	20	31



(a)



(b)



(c)

Figure 11. Results of Integration Testing between Vehicle Detection and Parking Slots (a) empty parking lot, (b) partially occupied, and (c) Full

4. Conclusions

The conclusion drawn from this study is that the integration of YOLOv5 and OpenCV algorithms proves effective in detecting and mapping parking space availability at the BRIN Bandung office. Object recognition technology, particularly YOLOv5, has demonstrated its capability to enhance accuracy and efficiency in parking management, provide real-time and precise information, and thus minimize prolonged time in searching **for** parking **slots**. The alignment between the research objectives and the findings underscores that the utilization of this technology can offer an impressive solution to address parking availability issues.

Acknowledgements

This research was supported by the National Innovation Research Agency Samaun Samadikun, Handayani University Makassar. The authors declare no conflicts of interest. This research did not receive specific grants from any funding agency in the public, commercial, or non-profit sectors.

References

[1] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, “Deep Learning for Computer Vision: A Brief Review,” *Comput Intell Neurosci*, vol. 2018, 2018, doi: 10.1155/2018/7068349.

[2] F. Borisyuk, A. Gordo, and V. Sivakumar, “Rosetta: Large scale system for text detection and recognition in images,” *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 71–79, 2018, doi: 10.1145/3219819.3219861.

[3] J. Liang, X. Chen, M. L. He, L. Chen, T. Cai, and N. Zhu, “Car detection and classification using cascade model,” *IET Intelligent Transport Systems*, vol. 12, no. 10, pp. 1201–1209, 2018, doi: 10.1049/iet-its.2018.5270.

[4] C. Lin and A. Kumar, “Contactless and partial 3D fingerprint recognition using multi-view deep representation,” *Pattern Recognit*, vol. 83, pp. 314–327, 2018, doi: 10.1016/j.patcog.2018.05.004.

[5] N. Sugianto, D. Tjondronegoro, and B. Tydd, “Deep Residual Learning for Analyzing Customer Satisfaction using Video Surveillance,” *Proceedings of AVSS 2018 - 2018 15th IEEE International Conference on Advanced Video and Signal-Based Surveillance*, pp. 1–6, 2018, doi: 10.1109/AVSS.2018.8639478.

[6] F. M. Wibowo and F. T. Syifa, “Rancang Bangun Sistem Pencarian Posisi Kendaraan di Area Parkir,” *Jurnal RESTI*, vol. 4, no. 1, pp. 77–79, 2020.

[7] A. Geitgey, “Snagging Parking Spaces with Mask R-CNN and Python,” Medium.Com. [Online]. Available: https://medium.com/@ageitgey/snagging-parking-spaces-with-mask-r-cnn-and-python-955f2231c400

[8] Jupiyandi Saniputra, F. R. Pratama, and Yoga Dharmawan, “Pengembangan Deteksi Citra Mobil Untuk Mengetahui Jumlah Tempat Parkir Menggunakan Cuda Dan Modified Yolo Development of Car Image Detection To Find Out the Number of Parking Space Using Cuda and Modified Yolo,” *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK)*, vol. 6, no. 4, pp. 413–419, 2019, doi: 10.25126/jtiik.201961275.

[9] S. Bangar, “Arsitektur jaringan saraf konvolusional (CNN) yang dikenal sebagai AlexNet,” Medium.Com. [Online]. Available: https://medium.com/@siddheshb008/alexnet-architecture-explained-b6240c528bd5

[10] E. Tanuwijaya and C. Fatichah, “Penandaan Otomatis Tempat Parkir Menggunakan YOLO untuk,” *BRILIANT: Jurnal Riset dan Konseptual*, vol. 5, pp. 189–198, 2020.

[11] Primasdika Yunia Putra, A. S. Arifianto, Zilvanhisna Emka Fitri, and Trismayanti Dwi Puspitasari, “Deteksi Kendaraan Truk pada Video Menggunakan Metode Tiny-YOLO v4,” *Jurnal Informatika Polinema*, vol. 9, no. 2, pp. 215–222, 2023, doi: 10.33795/jip.v9i2.1243.

[12] A. Muzaki *et al.*, “Deteksi Ketersediaan Lahan Parkir Dengan Menggunakan OpenCV,” vol. 3, pp. 237–244, 2024.

[13] F. P. P. dan I. Susilawati, “Prototipe Sistem Deteksi Ketersediaan Lahan Parkir Menggunakan Metode Algoritma Canny Edge,” *Jurnal Information System & Artificial Intelligence*, vol. 1, no, pp. 94–99, 2021.

[14] P. Adusumalli, H., Kalyani, D., Sri, RK, Pratapteja, M., & Rao, “Deteksi Masker Wajah Menggunakan OpenCV,” in *Konferensi Internasional Ketiga tentang Teknologi Komunikasi Cerdas dan Jaringan Seluler Virtual (ICICV)*, IEEE, 2021, pp. 1304–1309.

[15] R. T. H. Hasan and A. B. Sallow, “Face Detection and Recognition Using OpenCV,” *Journal of Soft Computing and Data Mining*, vol. 2, no. 2, pp. 86–97, 2021, doi: 10.30880/jscdm.2021.02.02.008.

[16] P. Jagtap, AM, Kangale, V., Unune, K., & Gosavi, “Studi fitur mirip LBPH, Eigenface, Fisherface dan Haar untuk Pengenalan wajah menggunakan OpenCV,” in *Sistem Berkelanjutan Cerdas (ICISS)*, IEEE, 2019, p. (hal.219-224).

[17] S. Rahman and H. Dafitri, “Pengembangan Convolutional Neural Network untuk Klasifikasi Ketersediaan Ruang Parkir,” *Explorer (Hayward)*, vol. 2, no. 1, pp. 1–6, 2022, doi: 10.47065/explorer.v2i1.148.

[18] Maulana Nur Hidayah, Febryanti Sthevanie, and Kurniawan Nur Ramadhani, “Deteksi Penggunaan Masker Pada Citra Menggunakan YOLOv5 Dengan CNN,” vol. 10, no. 5, pp. 4903–4909, 2023.

[19] D. Thuan, “Evolution of Yolo Algorithm and Yolov5: the State-of-the-Art Object Detection Algorithm,” *Oulu University of Applied Sciences*, p. 61, 2021.

[20] R. Iyer, P. Shashikant Ringe, R. Varadharajan Iyer, and K. Prabhulal Bhensdadiya, “Comparison of YOLOv3, YOLOv5s and MobileNet-SSD V2 for Real-Time Mask Detection,” *Int J Res Eng Technol*, vol. 8, no. 7, pp. 1156–1160, 2021, [Online]. Available: https://www.researchgate.net/publication/353211011

[21] Y. Yunefri, Sutejo, Y. E. Fadrial, K. Anggraini, M. Ramadhani, and R. Hardianto, “Implementation of Object Detection With You Only Look Once Algorithm in Limited Face-To-Face Times in Pandemic,” *Journal of Applied Engineering and Technological Science*, vol. 4, no. 1, pp. 400–404, 2022, doi: 10.37385/jaets.v4i1.1161.