Journal of Computation Physics and Earth Science
ISSN: 2776-2521 (online)
Volume 5, Number 1, April 2025, Page 89 - 98
https://journal.physan.org/index.php/jocpes/index


# Implementation of Web-based Realtime Monitoring System Using YOLOv8 for Green Box Detection and Automatic Capture in Navigation Missions at the Indonesian Boat Contest

**Yobel Eliezer Mahardika[1], Tonny Wahyu Aji[2], Dimas Aditya Wiranata[3], Antonius Kukuh Prasetyaji[4]**
[1,2,3,4]Undergraduate Program in Applied of Instrumentation Meteorology, Climatology Geophysics (STMKG)

**Article Info**

***Article history:***

Received March 14, 2025
Revised March 19, 2025
Accepted March 20, 2025

***Keywords:***

Real-time monitoring
Web-based visualization
YOLOv8
Flask
MySQL
Object detection
OpenCV

**A B S T R A C T**

This research presents the design and development of a real-time object detection and monitoring system specifically aimed at identifying green box objects using the YOLOv8 model. The system integrates OpenCV for frame-by-frame video processing, MySQL for image storage as Binary Large Objects (BLOBs), and a Flask-based web interface for real-time visualization. Green box objects detected with a confidence score above 0.7 are cropped and stored in the database. A dynamic web interface, updated every 2 seconds using AJAX, enables real-time monitoring and allows users to download the latest detected image for further analysis. Experimental results demonstrate that the YOLOv8 model achieves high detection accuracy, as measured by precision, recall, and mean average precision (mAP). The proposed system effectively combines object detection, data storage, and web-based visualization to provide a robust and scalable solution for real-time monitoring. Tests conducted under real-world conditions confirm the system's efficiency and reliability. Future work may involve hardware acceleration via edge computing, support for multi-object detection, and integration of advanced tracking algorithms to broaden its applicability in autonomous systems and industrial automation.



***Corresponden Author:***

Yobel Eliezer Mahardika,
Undergraduate Program in Applied of Instrumentation Meteorology, Climatology Geophysics (STMKG)
Tangerang City, Banten, Indonesia
Email: yobel.em69@gmail.com

## 1. INTRODUCTION

Real-time object detection is one of the important aspects in the development of navigation technology and autonomous systems [1], especially in a competition environment such as the Indonesian Ship Contest (KKI). In this competition, one of the missions that must be completed is to detect navigation markers in the form of green boxes above the water surface. This marker serves as a reference point for the ship to complete the trajectory according to the race rules. However, dynamic environmental conditions such as water ripples, light reflections, weather changes, and ship movements add challenges to the object detection process. Therefore, a detection system that is fast, accurate, and able to adapt to environmental changes is needed to assist the navigation team in completing the competition mission.

Deep learning technology, especially in the field of computer vision, has become an excellent solution in solving object detection problems with a high degree of accuracy [2]. One popular approach is the You Only Look Once (YOLO) algorithm, which is known for its ability to perform object detection in real time. YOLO algorithm has an advantage over traditional detection methods because it only requires one calculation to predict the location and class of objects in the image, resulting in much higher processing speed [3]. YOLOv8, as the latest iteration of YOLO, has seen significant improvements in terms of accuracy and computational efficiency. YOLOv8 can achieve mAP (mean Average Precision) up to 50-95% on various datasets with faster inference time than its predecessor [4].

The application of YOLO has been tested in various contexts, including object detection in maritime environments. YOLOv5 can detect floating objects on the sea surface with more than 90% accuracy [5], even under varying lighting conditions. YOLO has good adaptability in dynamic environments, making it an ideal choice for applications in the field of autonomous ship navigation [6]. In this study, YOLOv8 was chosen because it is capable of processing video data in real time with a speed rate of up to 30 FPS (frames per second) and a higher level of accuracy [7].

In addition to the detection aspect, this research also includes an automatic shooting system that uses OpenCV to capture photos when a green box is detected. The detected images are then stored in a MySQL database in BLOB (Binary Large Object) format. This format was chosen because of its ability to store binary data such as images in their original size without any loss of quality [8]. Efficient and structured data storage in the database allows the system to quickly access the stored images again [9].

Furthermore, to facilitate the technical team in monitoring the detection results during the competition, the system is designed using a web-based monitoring system built with the Flask framework. Flask is a lightweight and flexible Python backend framework, enabling the development of web-based monitoring systems with AI integration in a simple yet effective manner [10]. The system is equipped with an automatic refresh feature using AJAX every 2 seconds, so that the technical team can view the green box detection results and object photos in rea ltime through the browser. In addition, an Image Download button is provided to allow users to download the latest images as documentary evidence or for further analysis.

This research aims to develop a fast, accurate, and accessible object detection system through a Flask-based web interface to support the completion of navigation missions in the Indonesian Ship Contest. The system is expected to improve monitoring efficiency and provide flexibility for the control team in accessing and analyzing detection data. By utilizing YOLOv8 technology and a web-based monitoring system, this research provides an innovative solution that can be applied not only in the context of competition, but also for the development of autonomous navigation systems in the future.

## 2. LITERATURE REVIEW

Real-time object detection systems have become a major topic in the development of computer vision and machine learning-based technologies. One of the most effective methods in solving this problem is You Only Look Once (YOLO) [11]. YOLO is known for its ability to perform object detection in a single computation step, unlike previous methods such as Region-based CNN (R-CNN) and Fast R-CNN that require multiple computation steps [12]. This approach makes YOLO much faster while maintaining high accuracy. For example YOLOv2 can process up to 40 FPS (frames per second) with sufficient detection accuracy [13].

As technology evolves, the latest version of YOLO, YOLOv8, provides significant improvements in accuracy and computational efficiency. YOLOv8 integrates a lighter architecture and more optimized processing algorithms, enabling the model to achieve higher mAP50-95 on various benchmark datasets [14]. YOLO's implementation in maritime applications, where YOLOv5 successfully detected floating objects on the sea surface with an accuracy rate of 92%, even under varying environmental conditions [15]. These advantages make YOLOv8 an ideal solution for object detection systems in Indonesian Ship Contest (KKI) missions, where speed and accuracy are critical in detecting green boxes as navigation markers.

In addition to the detection aspect, this research also utilizes OpenCV as a supporting library to capture video and take photos automatically when the green box is detected. OpenCV has been widely used in video processing applications due to its flexibility in handling real-time image input [16]. The integration of OpenCV with deep learning algorithms can accelerate the decision-making process in visual-based systems, such as in robotics and automatic navigation [1].

For image data storage, this research relies on MySQL with the use of BLOB (Binary Large Object) format. The BLOB format was chosen because of its ability to store image data in original size without losing quality [17]. Storing data in BLOB allows structured management of images in the database, making data access faster and more efficient [18]. This is relevant to the needs of web-based monitoring systems, where image data must be accessible and updated in a short time.

In presenting the detection results, this system uses Flask as a Python-based backend framework. Flask is widely used in web application development due to its lightweight and flexible nature [10]. Flask facilitates integration between machine learning models and user interfaces by providing efficient API endpoints for presenting data [19]. Flask also supports AJAX (Asynchronous JavaScript and XML) for dynamic data updates on the web interface, making it an ideal solution for web-based monitoring systems [20]. Web-based monitoring systems have proven effective in various studies. Web-based monitoring provides flexibility for users to access data from various browser-enabled devices [21].

Building on prior studies, the integration of YOLOv8, OpenCV, MySQL, and Flask shows strong potential for addressing real-time object detection challenges with efficient data handling and presentation [19][22]. This research develops a system capable of detecting green boxes, capturing images automatically, storing them in a MySQL database, and displaying them via a web-based interface with auto-refresh. Its main contribution is the implementation of an end-to-end solution to support ship navigation missions in dynamic scenarios such as the Indonesian Ship Contest.

## 3. SYSTEM DESIGN AND IMPLEMENTATION

Before entering the core system of monitoring and photographing this green box, we must first build the YOLOv8 model development which is carried out in stages to ensure optimal performance in detecting the desired object. In this yolov8 development, we adjust to the challenges and missions given to the 2024 Indonesian Ship Contest (KKI), which is to detect green boxes.

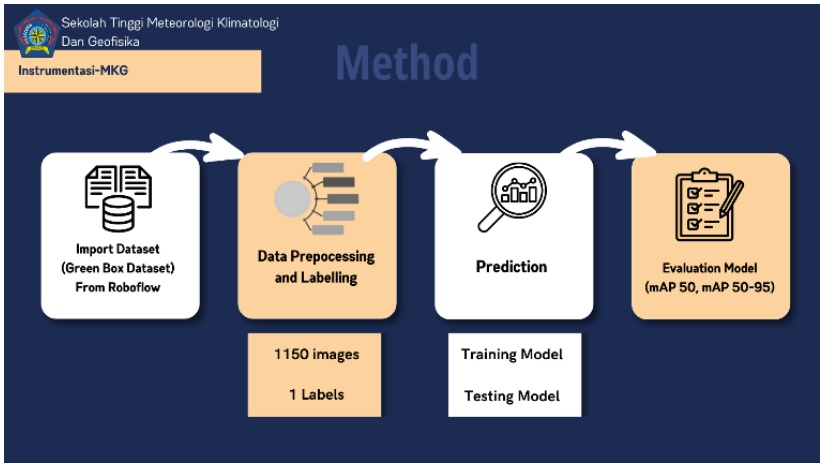


Fig. 1. The Method Steps to Detect the Green Box

The process starts with importing a dataset containing images labeled as green boxes. This dataset consists of 1,150 images with bounding box annotations created using the Roboflow platform. Next, data preprocessing is performed to improve the quality of the dataset, including data augmentation such as flipping, scaling, and normalization. After preprocessing is complete, model configuration is performed, where parameters such as input size (640x640), batch size, learning rate, and number of epochs are set according to the needs of the system. The model is then trained using the training dataset through a forward pass and backward pass process, where the model weights are updated based on the loss function value.

During training, the model is validated on the validation dataset to monitor performance metrics such as loss, precision, recall, and mAP (mean Average Precision). If the model performance does not meet the criteria, the configuration or training process is repeated until the model reaches optimal performance [3]. After the training is complete, the model is evaluated on the test dataset to calculate the final metrics, such as mAP50 and mAP50-95. The results of this evaluation are used to confirm the model's ability to detect objects with high accuracy [23].

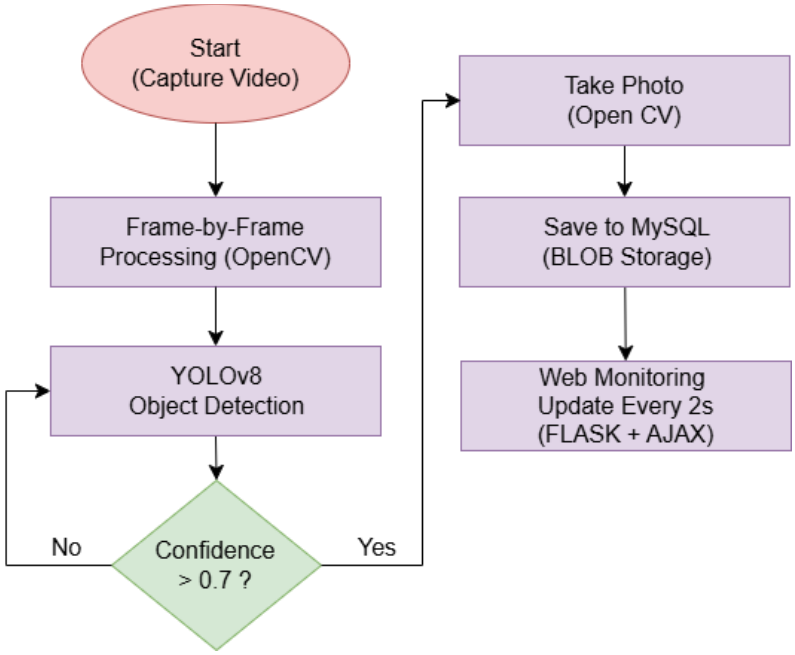


Fig. 2. Flowchart of real time detection system using YOLOv8 and web-based monitoring

The implementation of the real-time detection system using the YOLOv8 model involves several key components that are connected sequentially. The process starts with video capture using a camera connected to the system. The video input is captured in a frame-by-frame format using the OpenCV library. Each generated frame is then sent to the YOLOv8 model for object detection. The model detects the presence of a green box object and provides output in the form of a bounding box and confidence score [24].

In the decision stage, the system checks the confidence score of the detection. If the confidence score is greater than 0.7, the system proceeds to the take-photo process, where the part of the image containing the detected object is cropped based on the bounding box coordinates using OpenCV. If the confidence score does not meet the threshold, the image is ignored and the system moves on to the next image. The captured image is then stored in a MySQL database in Binary Large Object (BLOB) format. The database is designed as a table with id, filename, photo, and timestamp columns. Storing images in BLOB format allows for efficient and organized image management [25].

The final stage of this system is web-based monitoring using Flask as the backend and AJAX to update data in real time. Flask provides an endpoint to retrieve the latest images from the database, while AJAX is used to send periodic requests every 2 seconds. The detected images are displayed on the web interface, allowing users to monitor the results in realtime.

## 4. RESULT AND DISCUSSION

In this study, we conducted a series of trials to evaluate the performance and functionality of the YOLOv8-based real-time detection system. The primary objective was to assess the system's capability to accurately detect green box objects, process video input in real time, and store valid detections into a MySQL database. The trials were carried out using a continuous video stream, where each frame was processed and analyzed to identify objects meeting the defined confidence threshold of 0.7. Detected images were automatically cropped, stored, and displayed through a web-based interface, which updated every 2 seconds. The results from these trials provide insights into the reliability and efficiency of the proposed system in a dynamic environment.

### 4.1. Confusion Matrix Analysis

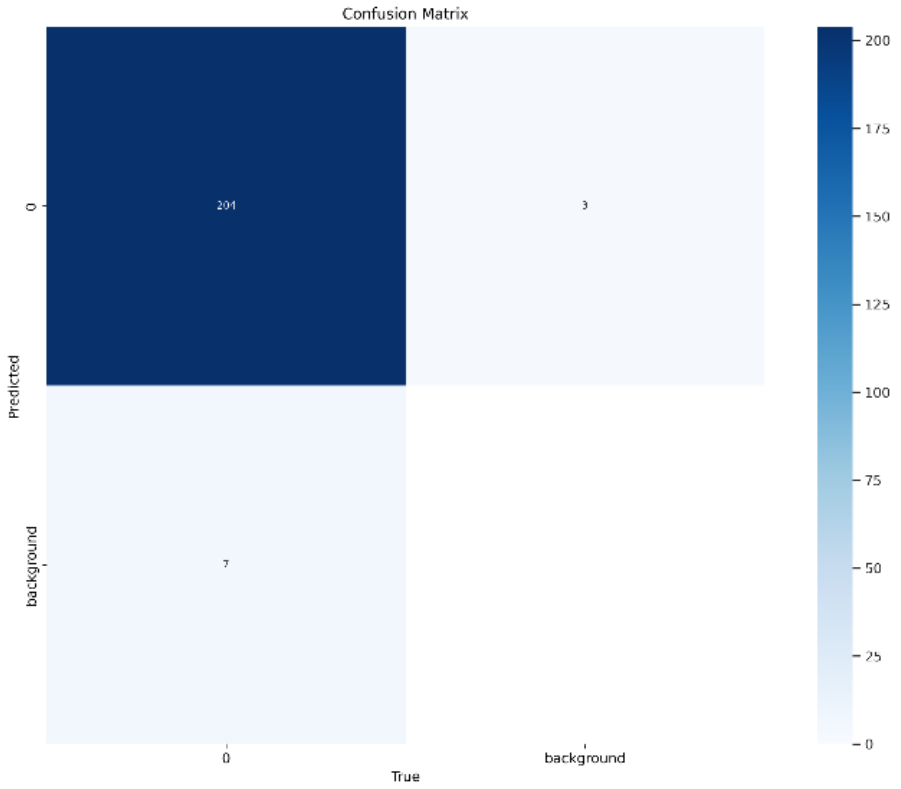


Fig. 3. Matrix performance of the YOLOv8-based real-time detection system in identifying green box objects

The confusion matrix presented in Fig. 3 evaluates the performance of the YOLOv8-based real-time detection system in identifying green box objects. The matrix shows that the system successfully detected 204 true positives (TP), which indicates that the target objects were correctly identified as belonging to the green box class. However, there are 3 false positives (FP), where background elements were misclassified as green boxes. Additionally, 7 false negatives (FN) were observed, representing instances where the system failed to detect green box objects even though they were present in the input frames. These false negatives could be caused by occlusions, low contrast between the target object and the background, or confidence scores below the defined threshold of 0.7. The overall distribution of the matrix highlights the system's high accuracy, with a significant dominance of true positives compared to false predictions. The low false positive and false negative values suggest that the YOLOv8 model performs reliably for real-time detection tasks. Further analysis of precision, recall, and F1-score will provide a more quantitative measure of the system's effectiveness in handling dynamic input data.

### 4.2. Training and Validation Performance of YOLOv8 Model

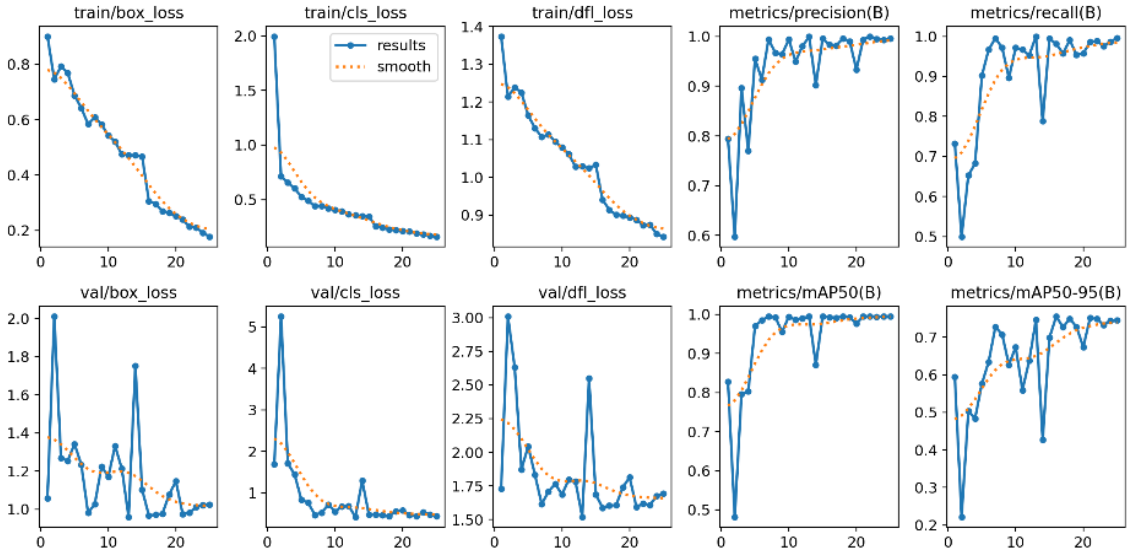


Fig. 4. Matrix performance of the YOLOv8-based real-time detection system in identifying green box objects

The training and validation performance of the YOLOv8 model is illustrated in Fig. 4, which highlights the progression of loss values and evaluation metrics over 25 epochs. The train/box_loss, train/cls_loss, and train/dfl_loss curves show a consistent decline as the number of epochs increases. This trend indicates that the model is learning effectively, as the loss values—representing errors in bounding box predictions, classification, and distribution focal loss—steadily decrease. The training loss curves demonstrate smooth convergence, signifying that the model adapts well to the training dataset.

In the validation phase, the val/box_loss, val/cls_loss, and val/dfl_loss exhibit some fluctuations, particularly in the earlier epochs, which is expected due to variations in the validation dataset. Despite these minor variations, the overall downward trend in validation loss suggests that the model generalizes well and avoids significant overfitting, as indicated by the minimal gap between training and validation losses.

The evaluation metrics provide further insights into the model's performance. The precision and recall metrics converge towards values close to 1.0, reflecting the model's ability to make accurate predictions with minimal false positives and false negatives. Additionally, the mAP50 metric, which measures the mean average precision at a 50% IoU threshold, approaches 1.0, demonstrating excellent detection performance. Meanwhile, the mAP50-95, which evaluates the model's performance across a range of IoU thresholds, stabilizes around 0.7-0.75. This result confirms that the model maintains robust accuracy under varying detection conditions.

In summary, the steady reduction in loss values, combined with the high precision, recall, and mAP scores, indicates that the YOLOv8 model successfully learns to detect green box objects with high accuracy. The small gap between training and validation losses further confirms the model's ability to generalize to unseen data, making it reliable for real-time object detection tasks.

### 4.3. Detection Results Visualization

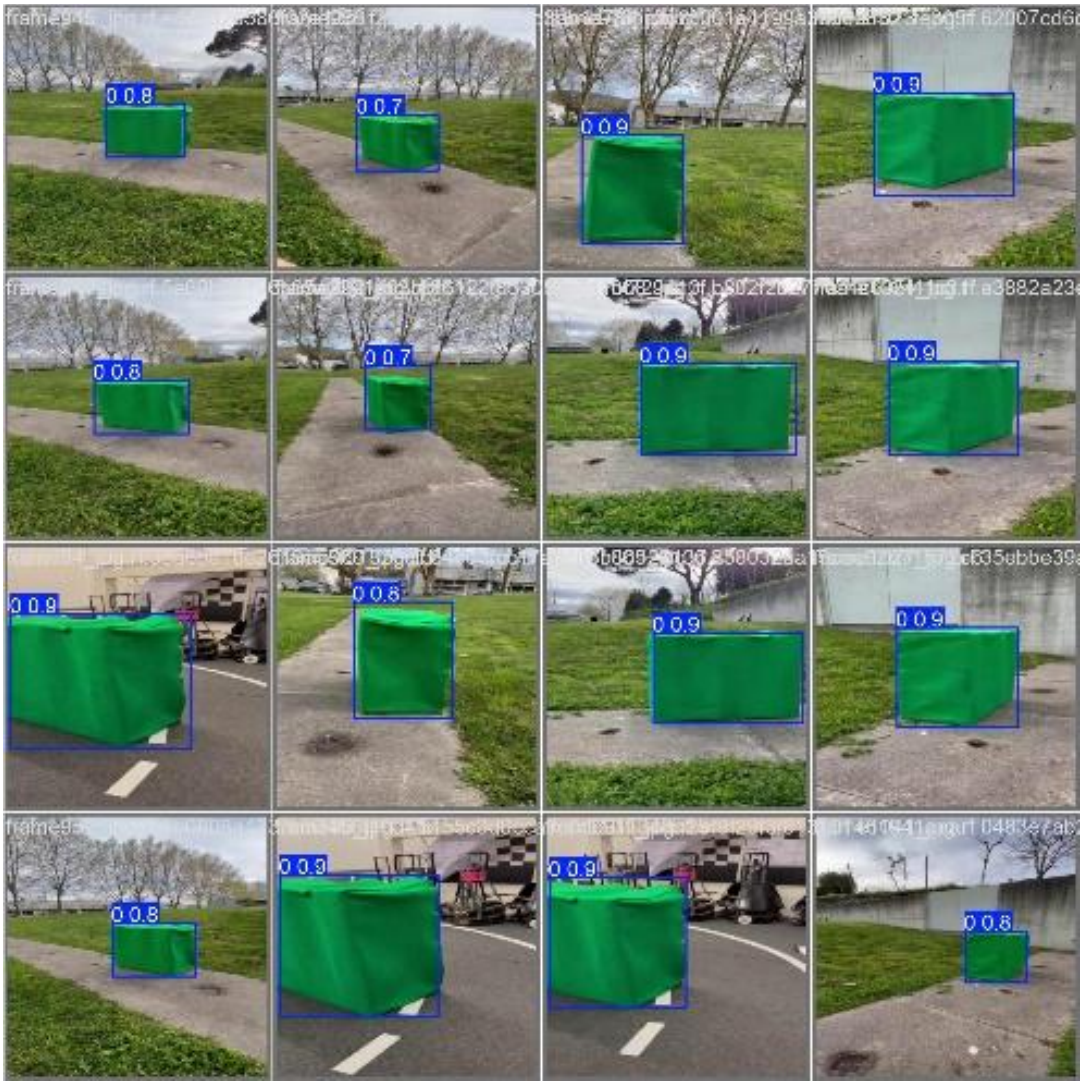


Fig. 5. Visualization of YOLOv8 Detection Results on Green Box Objects (Image-Based Testing)

The visualization of the YOLOv8 detection results demonstrates the system's ability to accurately identify and localize green box objects in image-based testing scenarios. As shown in Fig. 5, the YOLOv8 model successfully detected green box objects across multiple images, displaying confidence scores ranging from 0.7 to 0.9. Each detected object is annotated with a bounding box and its corresponding confidence score, which reflects the model's prediction accuracy. The testing dataset includes various conditions such as changes in angle, lighting, and background, simulating real-world scenarios to evaluate the robustness of the model. Despite these variations, the YOLOv8 model maintains consistent performance, with bounding boxes accurately placed around the green box objects. This indicates the model's reliability in detecting objects with high confidence while minimizing false positives.

The results confirm that YOLOv8 effectively handles object localization and confidence assessment in static image datasets. Such capabilities are essential for real-time applications, as they ensure the detection system can provide accurate and dependable results under varying environmental conditions.

### 4.4. Live Camera Detection Test

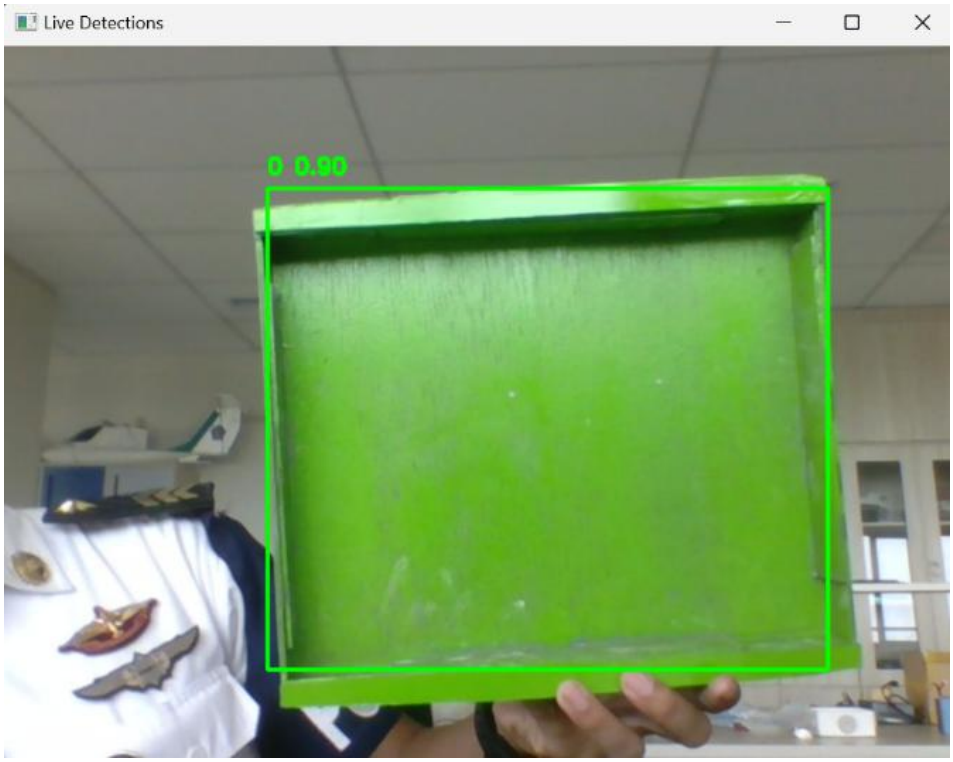


Fig. 6. Live Camera Detection Results Using YOLOv8

The live camera test, as shown in Fig. 6, demonstrates the real-time performance of the YOLOv8-based detection system in identifying green box objects. In this test, video frames were continuously captured using a live camera feed and processed in real time to detect target objects. The system successfully detected the green box in the frame with a confidence score of 0.90, as displayed on top of the bounding box. The bounding box is accurately aligned with the green box object, confirming the model's robustness in handling real-time video input. Unlike the image-based test, the live camera test introduces additional challenges such as motion blur, varying lighting conditions, and dynamic background elements. Despite these factors, the system consistently maintains high detection accuracy and rapid processing speed, enabling smooth real-time performance.

This result validates the system's capability to operate effectively in real-time scenarios, making it suitable for applications requiring live object detection and tracking. The successful detection in the live camera feed further highlights the integration of the YOLOv8 model with OpenCV for frame-by-frame processing and object localization.

### 4.5. Integrated System Testing

The integrated system testing was conducted to evaluate the performance of the end-to-end solution, combining YOLOv8, OpenCV, and a web-based monitoring system. The system encompasses real-time detection of green boxes using a webcam, saving detected images in BLOB format to a MySQL database, and displaying the latest detected image on a web interface. At this stage, the video captured in real-time is processed by YOLOv8 to detect green box objects. If the confidence score of the detection exceeds 0.7, the system uses OpenCV to capture the frame, crop the image based on the bounding box coordinates, and save it to the MySQL database in the photo column as a BLOB. The most recent image stored in the database is then retrieved and displayed on the web interface.

The figure below demonstrates the final output of the web interface, where the most recent green box detection is displayed, along with a button to download the image.


Fig. 7. Web Interface Display of the Latest Detected Image

This figure showcases the user interface for web monitoring, where the most recently detected green box image is displayed in real-time. The system updates the image every 2 seconds using an AJAX mechanism, ensuring a responsive and dynamic interface. Additionally, the Download Image button allows users to retrieve the displayed image in .png format directly from the MySQL database.

## 5. CONCLUSION

This paper presents a real-time green box detection and monitoring system developed using the YOLOv8 object detection model, OpenCV for image processing, MySQL for image storage, and Flask for the web-based monitoring interface. The system integrates these components into a unified pipeline, enabling real-time detection, efficient data management, and dynamic visualization of the detected images.

The results demonstrate that the YOLOv8 model performs accurate object detection in real-time, with a confidence threshold of 0.7 ensuring reliability during testing. Detected images were cropped based on bounding box coordinates and successfully stored in MySQL as Binary Large Object (BLOB) data. The database architecture allows for organized image management and retrieval. The web interface, designed using Flask and updated using AJAX every two seconds, provides a dynamic and responsive platform to display the latest detected image and enables users to download the displayed image seamlessly.

The system successfully meets the objectives of detecting, storing, and displaying green box objects in real-time. Experimental results confirm the robustness of the integration and the system's capability to handle live video feeds while maintaining smooth data flow between components. This makes the system suitable for applications in automated monitoring tasks, such as autonomous ship competitions, where identifying green box objects is part of mission-critical operations.

Further enhancements can focus on improving the system's processing efficiency by leveraging edge computing hardware such as NVIDIA Jetson or Raspberry Pi for real-time deployments. Additionally, incorporating multi-object detection capabilities and implementing tracking algorithms can enhance the system’s versatility in various real-world scenarios. The web interface can also be expanded to include live video streaming, historical data visualization, and advanced analytics for improved monitoring and decision-making.

Overall, this work provides a practical and scalable solution for real-time object detection and monitoring, with potential applications in robotics, autonomous systems, and industrial automation. The findings and implementation serve as a foundation for further research and development in similar fields

## REFERENCE

[1] N. Adiuku, N. P. Avdelidis, G. Tang, and A. Plastropoulos, “Advancements in Learning-Based

Navigation Systems for Robotic Applications in MRO Hangar: Review," *Sensors*, vol. 24, no. 5, pp. 1–22, 2024.
[2] N. Chitraningrum *et al.*, "Comparison Study of Corn Leaf Disease Detection based on Deep Learning YOLO-v5 and YOLO-v8," *J. Eng. Technol. Sci.*, vol. 56, no. 1, pp. 61–70, 2024.
[3] Y. Yin, H. Li, and W. Fu, "Faster-YOLO: An accurate and faster object detection method," *Digit. Signal Process. A Rev. J.*, vol. 102, p. 102756, 2020.
[4] M. R. A. P. and Suharjito, "Comparison of YOLOv8 and EfficientDet4 Algorithms in Detecting the Ripeness of Oil Palm Fruit Bunch," *2023 10th Int. Conf. ICT Smart Soc.*, 2023.
[5] F. Chen *et al.*, "Soft-NMS-Enabled YOLOv5 with SIOU for Small Water Surface Floater Detection in UAV-Captured Images," *Sustain.*, vol. 15, no. 14, pp. 1–18, 2023.
[6] Y. Hovde, Ø. S. Odden, M. Aanning, and M. Holt, "Graph-based slam and navigation in a simulated environment and buoy classification using YOLO for an Autonomous Surface Vessel," no. May, 2020.
[7] C. Fang and X. Yang, "Lightweight YOLOv8 for Wheat Head Detection," *IEEE Access*, vol. 12, no. April, pp. 66214–66222, 2024.
[8] S. Steinhoff, "Evaluation of MySQL 8.0 Spatial Data Features," *Researchgate.Net*, no. July, 2019.
[9] B. Szabó-Takács, T. Takács, and A. Roček, "MeteoRead: Client Software for Inserting Observed Atmospheric Data into MySQL<sub>TM</sub> Database and Downloading them into Different File Format," *J. Softw. Eng. Appl.*, vol. 14, no. 10, pp. 563–574, 2021.
[10] D. Ghimire, "Comparative study on Python web frameworks: Flask and Django," *Metrop. Univ. Appl. Sci.*, no. May, pp. 13–33, 2020.
[11] J. Terven, D. M. Córdova-Esparza, and J. A. Romero-González, "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS," *Mach. Learn. Knowl. Extr.*, vol. 5, no. 4, pp. 1680–1716, 2023.
[12] M. M. Mijwil, K. Aggarwal, R. Doshi, K. K. Hiran, and M. Gök, "The Distinction between R-CNN and Fast R-CNN in Image Analysis: A Performance Comparison," *Asian J. Appl. Sci.*, vol. 10, no. 5, 2022.
[13] J. Redmon and A. Farhadi, "Yolo9000," *Cvpr2017, Xplore*, no. April, pp. 187–213, 2017.
[14] X. Zhou, L. Jiang, C. Hu, S. Lei, T. Zhang, and X. Mou, "YOLO-SASE: An Improved YOLO Algorithm for the Small Targets in Complex Backgrounds," *Sensors*, vol. 22, no. 12, pp. 1–14, 2022.
[15] M. Zhu and Y. He, "Department of Natural Resources of Sichuan Province , Chengdu , Sichuan , PRC 610072 Sichuan Research Institute for Eco-system Restoration & Geo-disaster Prevention , Chengdu , Sichuan ," pp. 1647–1650, 2020.
[16] R. T. H. Hasan and A. B. Sallow, "Face Detection and Recognition Using OpenCV," *J. Soft Comput. Data Min.*, vol. 2, no. 2, pp. 86–97, 2021.
[17] K. T. Ahmed, H. Afzal, M. R. Mufti, A. Mehmood, and G. S. Choi, "Deep Image Sensing and Retrieval Using Suppression, Scale Spacing and Division, Interpolation and Spatial Color Coordinates with Bag of Words for Large and Complex Datasets," *IEEE Access*, vol. 8, pp. 90351–90379, 2020.
[18] S. K. Mainali and W. Zhang, "( 12 ) United States Patent," vol. 2, 2021.
[19] M. Grinberg, *Flask web development*. O'Reilly Media, Inc, 2018.
[20] D. Verma, "A Comparison of Web Framework Efficiency–Performance and network analysis of modern webframeworks," *Bachelor's Thesis Turku Univ. Appl. Sci. Bachelor Eng. Inf. Commun. Technol.*, p. 115, 2022.
[21] R. Nugraha, "Radiosonde System Using ESP32 And LoRa Ra-02 Web-Based for Upper-Air Profile Observation," pp. 2–6.
[22] "Магістерська дисертація," 2024.
[23] A. N. Azhar and M. L. Khodra, "Fine-tuning Pretrained Multilingual BERT Model for Indonesian Aspect-based Sentiment Analysis," *2020 7th Int. Conf. Adv. Informatics Concepts, Theory Appl. ICAICTA 2020*, pp. 2980–2988, 2020.
[24] M. Monika, U. Rajender, A. Tamizhselvi, and A. S. Rumale, "Real-Time Object Detection in Videos Using Deep Learning Models," *Ictact J. Image Video Process.*, no. December, p. 2, 2023.
[25] V. LAZAROVA, "Working With Large Objects in Application Development," *Rev. Econ.*, vol. 75, no. 2, pp. 98–104, 2023.