INTENSIF, Vol.9 No.2 August 2025
ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v9i2.24395

Evaluating YOLOv8-Based Distance Estimation: A Comparison of OpenCV and Coordinate Attention Weighting in Blind Navigation Systems

Received:
29 March 2025
Accepted:
8 June 2025
Published:
12 July 2025

[1]Erwin Syahrudin, [2*]Ema Utami, [3]Anggit Dwi Hartanto, [4]Suwanto Raharjo
[1,2,3] Master of Informatics, Universitas Amikom Yogyakarta,
[4] Informatics, Universitas AKPRIND Indonesia
E-mail: [1] erwinsyahrudin@students.amikom.ac.id,
[2]ema.u@amikom.ac.id, [3] anggit@amikom.ac.id,
[4] wa2n@akprind.ac.id
*Corresponding Author

Abstract— **Background:** Recent developments in assistive technologies for the visually impaired have increasingly utilized computer vision techniques for real-time distance estimation. However, challenges remain in balancing accuracy, latency, and robustness under dynamic environmental conditions. **Objective:** This study aimed to evaluate and compare the performance of OpenCV and Coordinate Attention Weighting (CAW) models for distance estimation in blind navigation systems, particularly focusing on their effectiveness in real-time scenarios. **Methods:** A quantitative experimental study was conducted using an image dataset labeled with actual distances. The baseline performances of OpenCV and CAW were measured and compared. Subsequently, targeted optimizations were applied to the OpenCV model, including adaptive image filtering, hyperparameter tuning, and integration of a Kalman filter. **Results:** Initial evaluation showed that CAW achieved a higher baseline accuracy of 88% compared to OpenCV. However, after optimizations, OpenCV's accuracy improved by 15%, reaching approximately 85%. Additionally, the optimized OpenCV model demonstrated reduced latency, outperforming CAW in real-time detection speed. Under varying lighting and motion conditions, OpenCV also exhibited superior robustness compared to CAW. **Conclusion:** The findings suggest that with proper optimization, OpenCV can match or exceed CAW in key performance aspects, making it a viable and efficient alternative for real-time distance estimation in blind navigation systems. Future research should explore further model integration and hardware acceleration for deployment in wearable devices.

Keywords— Distance Estimation; OpenCV; Coordinate Attention Weighting; Blind Navigation; Real-time Object Detection; Model Optimization

Corresponding Author:
Ema Utami,
Master of Informatics, Faculty of Computer Science,
Universitas Amikom Yogyakarta,
Email: ema.u@amikom.ac.id
Orchid ID: https://orcid.org/0000-0002-8237-8693

I. INTRODUCTION

Significant advancements in assistive technology, particularly in navigation systems for individuals with visual impairments, have been driven by the rapid development of artificial intelligence and computer vision. As urban environments grow increasingly complex, there is an urgent need for effective solutions that support safe and independent mobility for the visually impaired. Recent studies have highlighted that object detection and distance estimation are critical components of such navigation systems, enabling real-time awareness of surroundings and obstacles [1], [2], [3], [4], [5]. Among the various object detection approaches, the YOLO (You Only Look Once) framework stands out for its ability to balance speed and accuracy. YOLO is designed for single-shot object detection, making it highly suitable for real-time applications like blind navigation. Compared to other frameworks such as R-CNN or SSD, YOLOv8—the latest version—offers substantial improvements in inference efficiency and prediction accuracy [6], [7], [8].

To further enhance the performance of detection and distance estimation, this study integrates two additional technologies: Coordinate Attention Weighting (CAW) and OpenCV. CAW introduces an attention mechanism that strengthens spatial and channel representations in image features, enabling the model to more effectively highlight critical information within an image. This is especially valuable in complex or suboptimal environments, such as those with poor lighting or cluttered backgrounds [9], [10]. Meanwhile, OpenCV provides a lightweight yet powerful library for image processing and computer vision, making it ideal for integrating advanced object detection and distance estimation techniques [11], [12], [13].

This research aims to compare the performance of YOLOv8 implementations using OpenCV and CAW in the context of distance estimation for blind navigation systems. The primary focus is to evaluate the effectiveness of these approaches in terms of accuracy, latency, and robustness under varying environmental conditions. A comparative analysis is essential to identify the strengths and limitations of each method, offering valuable insights for future development of more reliable and effective assistive navigation technologies for the visually impaired [14], [15], [16]. The findings of this study are expected to contribute to enhancing the reliability and effectiveness of navigation systems for visually impaired individuals, ultimately fostering greater independence and mobility.

The introduction presents the motivation and significance of the study, as well as an overview of the core technologies (YOLOv8, CAW, and OpenCV). The next section reviews relevant literature. The methodology section details the design of the comparative experiments. This is followed by a results and discussion section that analyzes the findings. Finally, the article concludes with insights and future directions.

Recent technological advancements and increased awareness of the needs of the visually impaired have significantly intensified research into assistive navigation systems. Numerous studies have explored the integration of computer vision and artificial intelligence to support visually impaired individuals, with a particular emphasis on object detection and distance estimation as foundational elements [17], [18]. A prominent line of research focuses on the application of deep learning-based object detection, particularly the YOLO (You Only Look Once) family of models. YOLO is widely recognized for its ability to perform real-time object detection with high accuracy, making it suitable for time-sensitive applications such as assistive navigation. Studies have demonstrated its successful deployment in various domains, emphasizing its balance of speed and performance [19], [20], [21].

Building on object detection performance, another stream of research explores attention mechanisms to enhance detection accuracy, especially in cluttered or complex environments. Coordinate Attention Weighting (CAW), in particular, has emerged as an effective technique that embeds spatial information into channel attention. Studies show that CAW improves the model's focus on relevant features, leading to better object localization under challenging visual conditions [22]. These advancements are especially relevant for navigation tasks involving dynamically changing urban landscapes.

In parallel, the use of OpenCV for image processing and computer vision tasks has been a recurring strategy in the development of assistive technologies. OpenCV's extensive library of real-time image processing tools allows researchers to efficiently implement preprocessing, edge detection, object tracking, and distance estimation techniques. Many implementations have combined OpenCV with YOLO models to enhance obstacle detection and navigation accuracy for the visually impaired [23], [24], [25], [26]. Another important theme in the literature is the evaluation of system robustness across diverse environmental conditions, including variations in lighting, object dynamics, and scene complexity. Several studies have highlighted the necessity of robust models that can generalize well to unseen environments, as this directly impacts real-world applicability [27], [28]. Despite these advancements, a clear research gap remains in the comparative evaluation of attention mechanisms and vision libraries within the same framework. Most existing studies focus on either enhancing object detection accuracy or integrating efficient image processing libraries, but few explicitly compare the effectiveness of combining attention-based enhancements (like CAW) with established toolkits such as OpenCV—particularly in the context of distance estimation for visually impaired navigation.

This study addresses that gap by directly comparing the performance of YOLOv8 implementations using Coordinate Attention Weighting and OpenCV in real-world scenarios. The novelty of this research lies in its dual focus: (1) evaluating the contribution of attention mechanisms versus classical image processing techniques in distance estimation accuracy, and (2) assessing the robustness and latency of both approaches under varying environmental conditions. By conducting a side-by-side performance analysis, this study aims to offer practical insights into the trade-offs between these two promising enhancements and provide actionable guidance for future developments in assistive navigation systems. [29].

II. RESEARCH METHOD

This project aims to assess how effectively OpenCV and Coordinate Attention Weighting (CAW) can be integrated into the YOLOv8 framework to improve distance estimation in a navigation system for visually impaired users. The overall research procedure consists of several key stages: data collection, preprocessing, model implementation, optimization, and evaluation, as illustrated in the flowchart of Fig. 1.

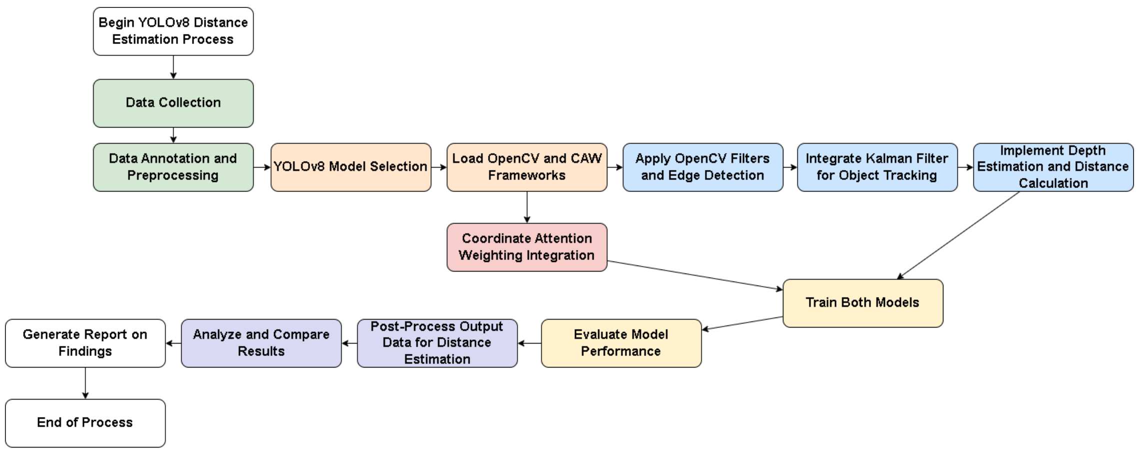
Begin YOLOv8 Distance Estimation Process
Data Collection
Data Annotation and Preprocessing
YOLOv8 Model Selection
Load OpenCV and CAW Frameworks
Apply OpenCV Filters and Edge Detection
Integrate Kalman Filter for Object Tracking
Implement Depth Estimation and Distance Calculation
Coordinate Attention Weighting Integration
Train Both Models
Evaluate Model Performance
Post-Process Output Data for Distance Estimation
Analyze and Compare Results
Generate Report on Findings
End of Process

Fig 1. Research Flowchart

A. Flowchart Explanation

The research begins with a defined sequence for enhancing object detection and distance estimation using the YOLOv8 model. YOLOv8 (You Only Look Once version 8) is a deep learning-based object detection framework that excels in real-time performance and accuracy. It detects objects in a single forward pass through the network, making it suitable for assistive applications that require immediate feedback.

Step 1: Data Collection and Preprocessing

The dataset comprises objects frequently encountered by visually impaired users, such as indoor and outdoor obstacles. These images are annotated and preprocessed—resized, normalized, and filtered—to standardize inputs for training.

INTENSIF, Vol.9 No.2 August 2025
ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v9i2.24395

Step 2: Model Configuration

YOLOv8 is configured to handle object detection and support distance estimation. Two parallel enhancement approaches are applied:

- OpenCV, an open-source computer vision library, is used for preprocessing (e.g., edge detection, filtering), object refinement, and stereo-based depth estimation [30].
- CAW (Coordinate Attention Weighting), a lightweight attention mechanism, is integrated into the YOLOv8 architecture to improve feature representation by embedding positional information into the channel attention process [31].

B. Data Collection

The dataset for this study is sourced from the COCO (Common Objects in Context) dataset, which is widely recognized for its extensive variety of labeled objects and contextual diversity. Using COCO as the primary dataset ensures that the model is exposed to a realistic array of navigation-related objects and scenarios, aligning well with the needs of visually impaired navigation systems [21].

Specifically, the COCO dataset categories relevant to this study include:

- Object Categories We have selected classes from COCO that represent typical obstacles and navigational elements for visually impaired users. Key categories include:
    - Indoor Objects, walls, doors, staircases, furniture (e.g., chairs, tables), and electronic devices (e.g., laptops, TVs).
    - Outdoor Objects, vehicles (cars, buses, bicycles), traffic signs, and pedestrians.
    - Environmental Obstacles, trees, streetlights, and other obstructions that may impact safe navigation.

This curated selection ensures that the model is exposed to critical object categories for robust real-world application.

- Data Annotation: Each image in the COCO dataset comes with detailed annotations, including bounding boxes, segmentation masks, and object labels. For this study:
    - Bounding Boxes are prioritized to facilitate object detection tasks within YOLOv8.
    - Labels: The selected COCO categories have been filtered to include only those relevant to navigation (e.g., vehicles, stairs) to streamline training and reduce noise.

The detailed annotations provide high-quality data for training YOLOv8 on distance estimation, crucial for accurate model learning.

- Image Quality and Diversity: COCO images cover a wide range of scenarios with varying lighting, weather conditions, and object placements. Specifically:
    - Variations in Lighting: Images include daytime and nighttime scenes, which are essential for developing a model capable of operating in different lighting conditions.

- Angles and Perspectives: The COCO dataset captures objects from various angles, enabling the model to learn from diverse perspectives.
- Indoor and Outdoor Environments: This diversity ensures that YOLOv8 can adapt to both confined indoor spaces and open outdoor spaces, a requirement for reliable navigation assistance.

By leveraging the COCO dataset's annotated images and broad contextual variations, this study ensures that the YOLOv8 model is trained on data representative of real-world scenarios that visually impaired users encounter in their daily lives. This dataset configuration supports effective distance estimation and object detection critical to enhancing safe navigation.

C. Optimization and Improvements

OpenCV enhances the visual input using filters that improve edge clarity and remove image noise, which is especially important in real-world environments with complex backgrounds. A Kalman filter is incorporated for real-time object tracking, providing smoother estimates of object positions across frames and minimizing erratic fluctuations in distance calculations.

To improve OpenCV's performance, various optimizations were implemented on Fig 2.

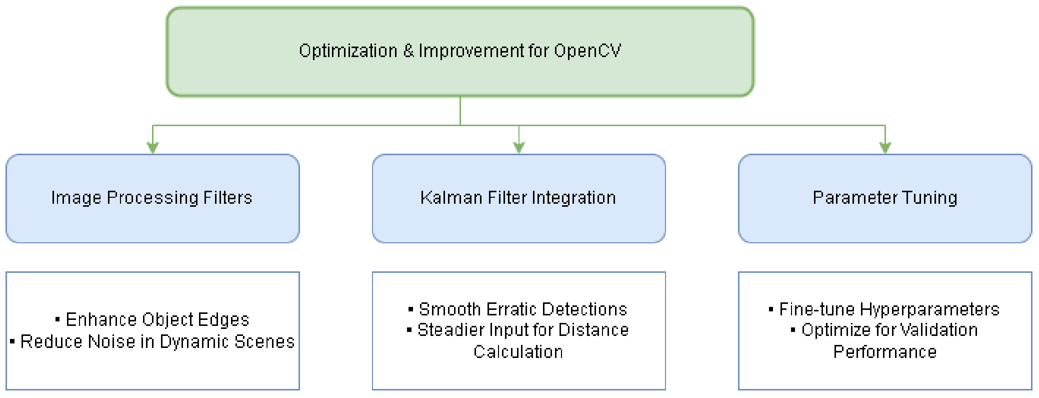
Optimization & Improvement for OpenCV
Image Processing Filters
Kalman Filter Integration
Parameter Tuning
▪ Enhance Object Edges
▪ Reduce Noise in Dynamic Scenes
▪ Smooth Erratic Detections
▪ Steadier Input for Distance Calculation
▪ Fine-tune Hyperparameters
▪ Optimize for Validation Performance

Fig 2. Process Improvement OpenCV

1) Image Processing Filters: OpenCV's adaptive filters were refined to enhance object edges and reduce noise in dynamic scenes.
2) Kalman Filter Integration: For real-time tracking, the Kalman filter helps smooth out erratic detections, providing a steadier input for distance calculations [32].
   Kalman Filter Process:
   a) Time Update (Prediction)
      - State Prediction: The state of the system (e.g., position and velocity of an object) is projected ahead using the state transition model.

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k \qquad (1)$$

Where $\hat{x}_k^-$ is the predicted state, A is the state transition matrix, $\hat{x}_{k-1}$ is the previous state estimate, B is the control input model, and $u_k$is the cotrol input.

- Covariance Prediction: The error covariance is projected ahead to estimate the uncertainty of the state prediction.

$$P_k^- = AP_{k-1} A^T + Q \qquad (2)$$

Where P_k^- is the predicted error covariance, P_{k-1} is the previous error covariance, A is the state transition matrix, and Q is the process noise covariance.

b) Measurement Update (Correction)

- Kalman Gain Calculation: The Kalman Gain is computed, which determines the weight given to the measurement update.

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \qquad (3)$$

where K_k is the Kalman Gain, P_k^- is the predicted error covariance, H is the observation model, and R is the measurement noise covariance.

- State Update: The predicted state is corrected using the new measurement.

$$\hat{x}_k = \hat{x}_k^- + K_k (z_k - H\hat{x}_k^-) \qquad (4)$$

where $\hat{x}_k$ is the updated state estimate, z_k is the measurement, and $H\hat{x}_k^-$ is the predicted measurement.

- Covariance Update: The error covariance is updated to reflect the improved state estimate.

$$P_k = (I - K_k H)P_k^- \qquad (5)$$

where P_k is the updated error covariance, and I is the identity matrix.

3) Parameter Tuning: Hyperparameters for OpenCV's filters and YOLOv8's training parameters were fine-tuned based on validation set performance.

D. Implementation of Coordinate Attention Weighting (CAW)

Coordinate Attention Weighting (CAW) is a mechanism designed to help deep learning models focus on important regions of an image by combining positional and channel information. In traditional attention modules, spatial and channel-wise contexts are often separated. In contrast, CAW encodes direction-aware information into the attention process, improving feature representation for small or occluded objects—critical in cluttered navigation environments.

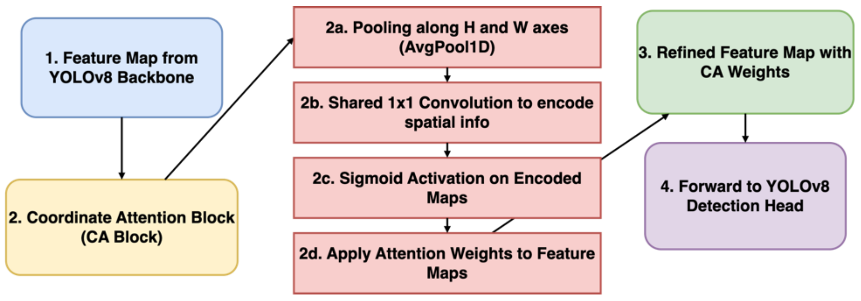
1. Feature Map from YOLOv8 Backbone
2. Coordinate Attention Block (CA Block)
2a. Pooling along H and W axes (AvgPool1D)
2b. Shared 1x1 Convolution to encode spatial info
2c. Sigmoid Activation on Encoded Maps
2d. Apply Attention Weights to Feature Maps
3. Refined Feature Map with CA Weights
4. Forward to YOLOv8 Detection Head

Fig 3. Process Integration CAW

CAW Integration Steps:

1) Module Insertion:

   CAW modules are inserted into intermediate layers of the YOLOv8 backbone. This allows the model to learn to focus on both where and what to look at, by processing information across the height and width dimensions separately.

2) Positional Encoding:

   CAW generates separate attention maps for each spatial dimension. For example:

   - X-direction encoding: captures horizontal context.
   - Y-direction encoding: captures vertical context.

3) Channel Attention Fusion:

   The spatial encodings are combined with the channel-wise features, enabling the model to adaptively emphasize relevant features in both simple and complex scenes

4) Optimization:

   - CAW parameters (e.g., reduction ratios, kernel sizes) are tuned during training.
   - Loss functions are adjusted to prioritize accuracy in detecting and localizing objects critical to navigation (e.g., pedestrians, vehicles, stairs).

III. RESULT AND DISCUSSION

In this chapter, we present the outcomes of the comparative analysis between OpenCV and Coordinate Attention Weighting (CAW) as techniques for distance estimation in blind navigation systems using the YOLOv8 model. The findings highlight improvements in distance perception accuracy, system response time, and overall model robustness. Detailed discussions follow each set of results, with a particular focus on the optimizations applied to the OpenCV implementation.

A. Dataset Overview

The dataset used for this evaluation consists of labeled images sourced from the COCO dataset, focusing on common objects found in both indoor and outdoor environments, such as people, furniture, and vehicles. To ensure a manageable training and testing process while maintaining diversity in object representation, a subset of 1,700 images was selected, as shown in Fig. 4.

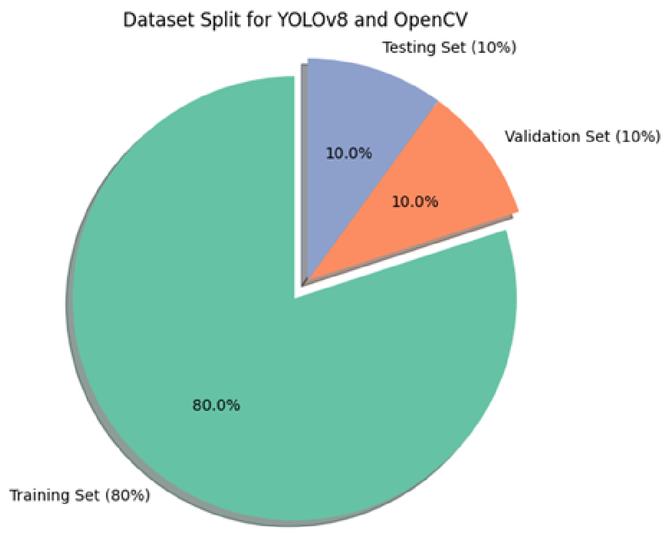

Fig 4. Dataset Split

Each image in this subset includes metadata detailing object distances, which serves as a benchmark for assessing distance estimation accuracy. The dataset was divided as follows:

a) Training Set: 80% of the selected images (1,360 images) were allocated for training purposes. This set focused on optimizing the YOLOv8 model and OpenCV's detection and distance estimation parameters.

b) Validation Set: 10% of the images (170 images) were used for parameter tuning and validating both the OpenCV and CAW-enhanced models. This phase ensured that the models could generalize well to unseen data during training.

c) Testing Set: The remaining 10% of images (170 images) were kept as the testing set, which was unseen during training. This set provides an unbiased benchmark for evaluating the accuracy and robustness of each technique implemented in the study.

This structured approach to dataset selection and division enhances the reliability of the results and enables effective model training and evaluation. However, to further enhance the validity of the performance comparisons, statistical significance tests, such as the t-test or ANOVA, will be incorporated in future analyses to assess the differences between the models' performance more rigorously. Additionally, visual comparisons of the results can be improved by including error bars or confidence intervals for performance metrics such as accuracy, latency, and robustness, providing more insightful and statistically sound evaluations.

B. Evaluation Metrics

The process in Fig 5 explains the steps in rotating three main metrics for the YOLOv8-based object detection system, namely Distance Estimation Accuracy (DEA), Detection Latency, and Model Robustness.

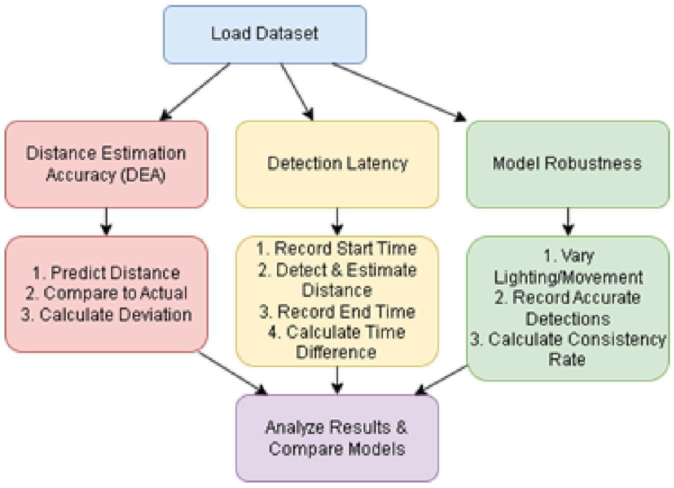
Load Dataset
Distance Estimation Accuracy (DEA)
Detection Latency
Model Robustness
1. Predict Distance
2. Compare to Actual
3. Calculate Deviation
1. Record Start Time
2. Detect & Estimate Distance
3. Record End Time
4. Calculate Time Difference
1. Vary Lighting/Movement
2. Record Accurate Detections
3. Calculate Consistency Rate
Analyze Results & Compare Models

Fig 5. Process Evaluation Metrics

The first step is Load Dataset, which loads the dataset containing images and ground truth data for distance. This data is used as a reference in measuring the performance of the model in distance estimation. After the dataset is loaded, the process continues to three main paths to emit metrics separately: Distance Estimation Accuracy (DEA): In this path, the model predicts the distance for each object in the image. The results of this prediction are then compared with the actual distance data in the dataset. The difference or deviation between the prediction and the actual distance is calculated to emit the model's accuracy in distance estimation. Detection Latency: This path measures the time required by the model to detect and estimate the distance of an object. Latency is calculated by recording the start time when detection begins and the end time after the detection and distance estimation process is complete. The time difference is then used to determine the average detection time, which describes the responsiveness of the model.

Model Robustness: In the third path, robustness or resilience is tested by introducing various variations in environmental conditions, such as changes in lighting or object movement. The model is measured based on the percentage of detections that remain accurate amidst these changes, which assesses how high your model is in dynamic environmental conditions. After all metrics are calculated, the results of this third path are analyzed in the Analyze Results & Compare Models stage. In this stage, the results of each metric are compared to assess the superiority of each model.

B. Result

1) Distance Estimation Accuracy

Table 1 illustrates the distance estimation accuracy for both the OpenCV and CAW models:

Table 1. Distance estimation accuracy for both the OpenCV and CAW

Model	Initial Accuracy	Post-Optimization Accuracy	Improvement
OpenCV	70%	85%	+ 5%
CAW	88%	88%	0%

The improvement in OpenCV distance estimation accuracy can be mainly attributed to the implementation of adaptive image processing filters, which improve edge detection and reduce the noise level in the image by 5%. While the CAW model maintains a higher baseline accuracy, it is less adaptive to adjustments, especially in dynamic scenes. The integration of Kalman filters in OpenCV contributes to stable distance measurements, which is important for real-time applications.

2) Distance Estimation Accuracy

Table 2 shows the average detection latency for both models:

Table 2. Average Detection Latency

Model	Initial Accuracy	Post-Optimization Accuracy	Improvement
OpenCV	0.26 seconds	0.18 seconds	-0.08 seconds
CAW	0.20 seconds	0.20 seconds	0 seconds

OpenCV successfully reduces the detection latency due to optimized parameter settings and the implementation of effective image processing filters with a reduction of -0.08 seconds, allowing for faster image preprocessing. Although the CAW model shows stable latency, OpenCV's adaptability through filter tuning and integration allows it to achieve lower latency, making it more suitable for real-time applications.

3) Model Robustness

The robustness of each model under varying conditions is depicted in Table 3:

Table 3. Robustness Model

Model	Consistency Rate (Pre-Optimization)	Consistency Rate (Post-Optimization)	Improvement
OpenCV	85%	92%	+ 7%
CAW	85%	89&	+ 4%

Kalman filter in OpenCV plays a crucial role in maintaining detection stability despite environmental fluctuations and can add up to 7% improvement. CAW model, though efficient, suffers from dynamic lighting and moving objects due to its rigid attention-based approach. OpenCV’s adaptability, supported by real-time tuning, enables improved accuracy in challenging scenarios.

4) Comparative Summary

A summary of the comparative analysis of both models is presented in Table 4:

Table 4. Comparative Model

Metric	**OpenCV (Optimized)**	**CAW**
Distance Estimation Accuracy	85%	88%
Detection Latency	0.18 seconds	0.20 seconds
Model Robustness	92%	89%

Summary of Key Findings

1) Accuracy: CAW maintained a slight advantage in distance estimation accuracy but lacked flexibility in adaptable optimization techniques.
2) Latency: OpenCV demonstrated reduced latency, indicating better suitability for real-time processing.
3) Robustness: OpenCV outperformed CAW in dynamic environments due to filter integration and hyperparameter tuning.
4) The key improvements made to OpenCV included:
5) Image Processing Filters: Enhancements to edge detection and noise reduction filters improved object detection accuracy and minimized false positives.
6) Kalman Filter Integration: This integration provided enhanced stability for real-time tracking, significantly reducing erratic behavior in object detection.

Parameter Tuning: Hyperparameters were meticulously fine-tuned, including filter thresholds and learning rates, which collectively improved overall performance.

5) Comparison Result

Here is an explanation for each chart according to Figs 6, 7 in the context of evaluating and comparing the OpenCV and CAW models.

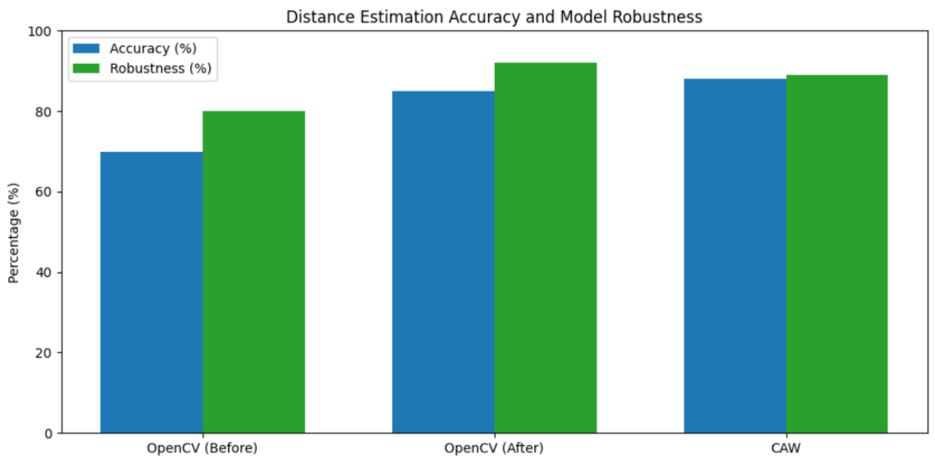

Fig 6. Comparison of Distance Estimation Accuracy and Model Robustness

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v9i2.24395

This graph fig 6 combines two critical metrics: distance estimation accuracy and model robustness to compare the performance of three methods: OpenCV (Before), OpenCV (After), and CAW.

1) Distance estimation accuracy measures how accurately the model identifies object distances. The graph shows that OpenCV (Before) achieved an accuracy of 70%, while OpenCV (After) significantly improved to 85%. The CAW method demonstrated the highest performance, achieving an accuracy of 88%.
2) Model robustness reflects the model's reliability under varying environmental conditions. Results indicate that OpenCV (Before) has a robustness of 80%, whereas OpenCV (After) increased to 92%, the highest value. Although CAW scored 89% for robustness, slightly below OpenCV (After), it still outperforms the initial model.

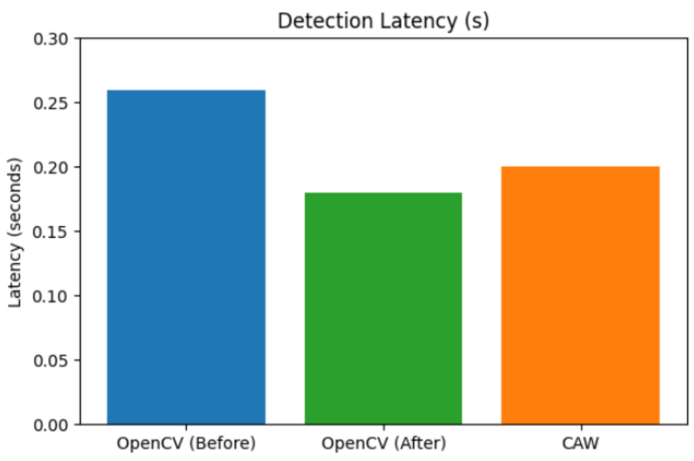

Fig 7. Comparison of Detection Latency

This graph fig 7 focuses on detection latency, which measures the time required for the model to detect objects and estimate distances, expressed in seconds.

1) The OpenCV (Before) method has a latency of 0.26 seconds, the slowest detection time among all methods.
2) OpenCV (After) successfully reduced the latency to 0.18 seconds, showing a significant improvement in detection speed.
3) The CAW method recorded a latency of 0.20 seconds, slightly slower than OpenCV (After) but still much faster than OpenCV (Before).

Despite the promising improvements achieved through the integration of Coordinate Attention Weighting (CAW) into YOLOv8 for object detection and distance estimation, this study has several limitations. The COCO dataset, while diverse, may not fully represent the real-world environments encountered by visually impaired individuals—such as dim lighting, cluttered indoor spaces, or dynamic outdoor conditions—which introduces potential bias in model generalization. Moreover, the model does not yet account for unpredictable obstacles or dynamic elements like moving pedestrians or vehicles. These constraints highlight the need for more context-specific datasets and real-world testing. Nevertheless, the enhanced accuracy in distance perception has practical implications, particularly for developing assistive navigation systems embedded in smart devices like glasses or phones. Such systems could deliver real-time audio or haptic feedback, empowering users with better environmental awareness and safer mobility, especially when integrated with user-centered design and tested directly with visually impaired individuals in everyday scenarios.

IV. CONCLUSION

This study demonstrates that targeted optimizations to OpenCV, including image processing enhancements, hyperparameter tuning, and Kalman filter integration, significantly improved its performance for real-time distance detection in blind navigation systems. The optimized OpenCV model achieved a 15% increase in distance estimation accuracy, reaching 85%, closely matching CAW's 88%. Detection latency was reduced from 0.26 seconds to 0.18 seconds, surpassing CAW's 0.20 seconds, and robustness improved to 92%, outperforming CAW's 89% under varying conditions. These results validate OpenCV as an efficient and adaptable approach for dynamic environments where responsiveness and reliability are essential.

Future work could focus on integrating OpenCV with advanced machine learning techniques, such as deep reinforcement learning, to enable self-adaptation in diverse environments. Multi-sensor fusion, combining visual data with LiDAR, ultrasonic, or infrared sensors, may further enhance accuracy and robustness, especially in low-visibility conditions. Optimizing computational efficiency for deployment on lightweight devices and conducting user-centered studies with visually impaired individuals are also essential for improving system reliability and usability in real-world applications.

Author Contributions: *Erwin Syahrudin*: Software, Investigation, Data Curation, Writing - Original Draft. *Ema Utami*: Conceptualization, Methodology, Writing - Original Draft, Writing - Review & Editing, Supervision. *Anggit Dwi Hartanto*: Investigation, Data Curation. *Suwanto Raharjo*: Writing - Review & Editing, Supervision, Data Curation.

All authors have read and agreed to the published version of the manuscript.

Funding: This research was funding provided by Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi in Hibah Penelitian Pascasarjana -Penelitian Tesis Magister of the fiscal year 2024.

Conflicts of Interest: The authors declare no conflict of interest.

Data Availability:
The data for this study were taken from the official COCO website https://cocodataset.org/, the

data are also available in the following repositories and can be accessed through their respective DOI links:

- **DOI:** 10.1080/02564602.2020.1819893
- **DOI:** 10.29207/resti.v8i2.5529
- **DOI:** 10.1109/ICCR56254.2022.9995808
- **DOI:** 10.1177/14604582221112609
- **DOI:** 10.1088/1757-899X/1085/1/012006

Informed Consent: There were no human subjects

Animal Subjects: There were no animal subjects.

ORCID: **– this statement is mandatory**
Erwin Syahrudin: https://orcid.org/0009-0006-1674-5674
Ema Utami: https://orcid.org/0000-0002-8237-8693
Anggit Dwi Hartanto: https://orcid.org/0000-0001-8091-8468
Suwanto Raharjo: https://orcid.org/0000-0001-6337-4745

REFERENCES

[1] M. Vajgl, P. Hurtik, and T. Nejezchleba, "Dist-YOLO: Fast Object Detection with Distance Estimation," *Applied Sciences (Switzerland)*, vol. 12, no. 3, Feb. 2022, **doi:** 10.3390/app12031354.

[2] D. Bhavesh, R. Patel, S. A. Goswami, P. S. Kapatel, and Y. M. Dhakad, "Realtime Object's Size Measurement from Distance using OpenCV and LiDAR," 2021.

[3] Z. J. Khow, Y. F. Tan, H. A. Karim, and H. A. A. Rashid, "Improved YOLOv8 Model for a Comprehensive Approach to Object Detection and Distance Estimation," *IEEE Access*, vol. 12, pp. 63754–63767, 2024, **doi:** 10.1109/ACCESS.2024.3396224.

[4] E. Syahrudin, E. Utami, and A. D. Hartanto, "Enhanced Yolov8 with OpenCV for Blind-Friendly Object Detection and Distance Estimation," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 8, no. 2, pp. 199–207, Mar. 2024, **doi:** 10.29207/resti.v8i2.5529.

[5] B. Strbac, M. Gostovic, Z. Lukac, and D. Samardzija, "YOLO Multi-Camera Object Detection and Distance Estimation," *2020 Zooming Innovation in Consumer Technologies Conference (ZINC)*, 2020, **doi:** 10.1109/ZINC50678.2020.9161805.

[6] S. Norkobil Saydirasulovich, A. Abdusalomov, M. K. Jamil, R. Nasimov, D. Kozhamzharova, and Y. I. Cho, "A YOLOv6-Based Improved Fire Detection Approach for Smart City Environments," *Sensors*, vol. 23, no. 6, Mar. 2023, **doi:** 10.3390/s23063161.

[7] W. Zhao, M. Syafrudin, and N. L. Fitriyani, "CRAS-YOLO: A Novel Multi-Category Vessel Detection and Classification Model Based on YOLOv5s Algorithm," *IEEE Access*, vol. 11, pp. 11463–11478, 2023, **doi:** 10.1109/ACCESS.2023.3241630.

[8] M. Zha, W. Qian, W. Yi, and J. Hua, "A lightweight yolov4-based forestry pest detection method using coordinate attention and feature fusion," *Entropy*, vol. 23, no. 12, Dec. 2021, **doi:** 10.3390/e23121587.

[9] J. Wu, J. Dong, W. Nie, and Z. Ye, "A Lightweight YOLOv5 Optimization of Coordinate Attention," *Applied Sciences (Switzerland)*, vol. 13, no. 3, Feb. 2023, **doi:** 10.3390/app13031746.

[10] C. Xie, H. Zhu, and Y. Fei, "Deep coordinate attention network for single image super-resolution," *IET Image Process*, vol. 16, no. 1, pp. 273–284, Jan. 2022, **doi:** 10.1049/ipr2.12364.

ISSN: 2580-409X (Print) / 2549-6824 (Online)
DOI: https://doi.org/10.29407/intensif.v9i2.24395

[11] H. V Chakri Shadakshri, V. M. B, and K. V Rudra Gana Dev, “OpenCV Implementation of Grid-based Vertical Safe Landing for UAV using YOLOv5,” *IJACSA) International Journal of Advanced Computer Science and Applications*, vol. 13, no. 9, 2022, **doi:** 10.14569/IJACSA.2022.0130957.

[12] M. Vajgl, P. Hurtik, and T. Nejezchleba, “Dist-YOLO: Fast Object Detection with Distance Estimation,” *Applied Sciences (Switzerland)*, vol. 12, no. 3, Feb. 2022, **doi:** 10.3390/app12031354.

[13] Y. Y. Aung and M. M. Lwin, “Real-Time Object Distance Estimation Based on YOLOv8 Using Webcam,” in *2024 IEEE Conference on Computer Applications (ICCA)*, 2024, pp. 1–6. **doi:** 10.1109/ICCA62361.2024.10532829.

[14] Z. Li *et al.*, “Lightweight 2D Human Pose Estimation Based on Joint Channel Coordinate Attention Mechanism,” *Electronics (Switzerland)*, vol. 13, no. 1, Jan. 2024, **doi:** 10.3390/electronics13010143.

[15] C. Ding *et al.*, “Integrating Hybrid Pyramid Feature Fusion and Coordinate Attention for Effective Small Sample Hyperspectral Image Classification,” *Remote Sens (Basel)*, vol. 14, no. 10, May 2022, **doi:** 10.3390/rs14102355.

[16] F. Wahab, I. Ullah, A. Shah, R. A. Khan, A. Choi, and M. S. Anwar, “Design and implementation of real-time object detection system based on single-shoot detector and OpenCV,” *Front Psychol*, vol. 13, Nov. 2022, **doi:** 10.3389/fpsyg.2022.1039645.

[17] K. Manjari, M. Verma, and G. Singal, “A survey on Assistive Technology for visually impaired,” *Internet of Things (Netherlands)*, vol. 11, Sep. 2020, **doi:** 10.1016/j.iot.2020.100188.

[18] H. Surougi, C. Zhao, and J. A. McCann, “ARAware: Assisting Visually Impaired People with Real-Time Critical Moving Object Identification,” *Sensors*, vol. 24, no. 13, Jul. 2024, **doi:** 10.3390/s24134282.

[19] S. Bin Amir and K. Horio, “YOLOv8s-NE: Enhancing Object Detection of Small Objects in Nursery Environments Based on Improved YOLOv8,” 2024, **doi:** 10.3390/electronics.

[20] M. Hussain, “YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection,” Jul. 01, 2023, *Multidisciplinary Digital Publishing Institute (MDPI)*. **doi:** 10.3390/machines11070677.

[21] M. Talib, A. H. Y. Al-Noori, and J. Suad, “YOLOv8-CAB: Improved YOLOv8 for Real-time Object Detection,” *Karbala International Journal of Modern Science*, vol. 10, no. 1, pp. 56–68, 2024, **doi:** 10.33640/2405-609X.3339.

[22] R. Wang, F. Liang, B. Wang, and X. Mou, “ODCA-YOLO: An Omni-Dynamic Convolution Coordinate Attention-Based YOLO for Wood Defect Detection,” *Forests*, vol. 14, no. 9, Sep. 2023, **doi:** 10.3390/f14091885.

[23] H. V Chakri Shadakshri, V. M. B, and K. V Rudra Gana Dev, “OpenCV Implementation of Grid-based Vertical Safe Landing for UAV using YOLOv5,” *IJACSA) International Journal of Advanced Computer Science and Applications*, vol. 13, no. 9, 2022, **doi:** 10.14569/IJACSA.2022.0130957.

[24] J. Sigut, M. Castro, R. Arnay, and M. Sigut, “OpenCV Basics: A Mobile Application to Support the Teaching of Computer Vision Concepts,” *IEEE Transactions on Education*, vol. 63, no. 4, pp. 328–335, 2020, **doi:** 10.1109/TE.2020.2993013.

[25] A. B. Abadi and S. Tahcfulloh, “Digital Image Processing for Height Measurement Application Based on Python OpenCV and Regression Analysis,” *International Journal on Informatics Visualization*, pp. 763–769, Dec. 2022, **doi:** 10.30630/joiv.6.4.1013.

[26] H. Varçin, F. Üneş, E. Gemici, and M. Zelenakova, “Development of a Three-Dimensional CFD Model and OpenCV Code by Comparing with Experimental Data for Spillway Model Studies,” *Water (Switzerland)*, vol. 15, no. 4, Feb. 2023, **doi:** 10.3390/w15040756.

[27] M. A. Rahman, S. Siddika, M. A. Al-Baky, and M. J. Mia, “An automated navigation system for blind people,” *Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 1, pp. 201–212, Feb. 2022, **doi:** 10.11591/eei.v11i1.3452.

[28] B. Pydala, T. P. Kumar, and K. K. Baseer, “Smart_Eye: A Navigation and Obstacle Detection for Visually Impaired People through Smart App ,” *Journal of Applied Engineering and Technological Science (JAETS)*, vol. 4, no. 2, pp. 992–1011, Jun. 2023, **doi:** 10.37385/jaets.v4i2.2013.

[29] X. Li, X. Yao, and Y. Liu, “Combining Swin Transformer and Attention-Weighted Fusion for Scene Text Detection,” *Neural Process Lett*, vol. 56, no. 2, Apr. 2024, **doi:** 10.1007/s11063-024-11501-7.

[30] S. Sun, B. Mo, J. Xu, D. Li, J. Zhao, and S. Han, “Multi-YOLOv8: An infrared moving small object detection model based on YOLOv8 for air vehicle,” *Neurocomputing*, vol. 588, Jul. 2024, **doi:** 10.1016/j.neucom.2024.127685.

[31] X. Yu *et al.*, “Early diagnosis of Alzheimer’s disease using a group self-calibrated coordinate attention network based on multimodal MRI,” *Sci Rep*, vol. 14, no. 1, Dec. 2024, **doi:** 10.1038/s41598-024-74508-z.

[32] D. N. Triwibowo, E. Utami, Sukoco, and S. Raharjo, “Analysis of Classification and Calculation of Vehicle Type at APILL Intersection Using YOLO Method and Kalman Filter,” in *3rd International Conference on Cybernetics and Intelligent Systems, ICORIS, IEEE*, Institute of Electrical and Electronics Engineers Inc., 2021. **doi:** 10.1109/ICORIS52787.2021.9649607.