Night-time Video/Image Enhancement and Object Detection Using YOLOv8and OpenCV

Bantu Mohan

Department of CSE (Artificial Intelligence & Machine Learning), Institute of Aeronautical Engineering, Telangana, India

Pendyala Shashank Reddy

Department of CSE (Artificial Intelligence & Machine Learning), Institute of Aeronautical Engineering, Telangana, India

M. Somla

Department of CSE (Artificial Intelligence & Machine Learning), Institute of Aeronautical Engineering, Telangana, India

Ganji Nithin Kumar

Department of CSE (Artificial Intelligence & Machine Learning), Institute of Aeronautical Engineering, Telangana, India

ABSTRACT

This work proposes a computationally efficient pipeline to perform real-time nighttime object recognition by combining a refined YOLOv8 model with conventional image enhancement. Our approach successfully reduces the noise and poor visibility that significantly impair detector performance in low light by utilizing Gamma Correction, CLAHE, and Bilateral Filtering. Achieving real-time performance (>20 FPS) and maintaining good detection accuracy (97–98% mAP), the suggested approach reduces processing time by 12–14% when tested on the ExDark, Night OWL, and custom datasets. By establishing a reliable and effective baseline for crucial applications such as autonomous driving and surveillance, our study validates the potent combination of deep learning and picture preprocessing.

Keywords: Night-time object detection, YOLOv8, OpenCV, image enhancement, real-time processing, low-light vision, gamma correction, CLAHE, bilateral filtering, deep learning, surveillance, autonomous driving, computer vision, performance metrics.

Introduction

A significant and unsolved problem for safety-critical applications such as autonomous driving and round-the-clock surveillance is reliable object detection at night. You Only Look Once (YOLO) and other deep learning models have shown impressive results in well-lit settings, but their performance drastically deteriorates in low light. Low contrast, which reduces the gradients necessary for recognizing object borders; excessive sensor noise, which the model may mistakenly interpret as texture or edge features; and low light, which blurs object features, are the specific problems causing this degradation. System reliability is further jeopardized by intricate shadows and reflections that frequently result in occlusions and misdetections, as well as color distortion caused by artificial lighting.

Advanced augmentation methods based on Transformers, Generative Adversarial Networks (GANs), and Retinex theory have been developed to address these problems. These cutting-edge techniques can yield visually striking results, but they are not appropriate for real-time deployment on consumer-grade or embedded hardware due to their high computational cost and memory footprint. This causes a big disconnect between effective, implementable remedies and intricate academic models. By methodically assessing a lightweight, sequential enhancement pipeline connected with the YOLOv8 detector, this work fills this gap. For developers who must strike a compromise between enhanced quality and real-time processing restrictions, our study attempts to provide a robust, repeatable performance baseline for a computationally efficient technique.

A. Problem Statement

Night-time object detection presents a unique set of challenges that significantly impact detection accuracy. One major issue is low illumination, which results in insufficient lighting that obscures object features and leads to missed detections. Additionally, high noise levels due to sensor artifacts in low-light environments often introduce false positives. The problem is further exacerbated by low contrast, which makes it difficult to distinguish objects from their backgrounds. Color distortion caused by artificial lighting can alter color profiles, thereby affecting the reliability of feature extraction. Moreover, complex lighting conditions frequently produce shadows and reflections, leading to occlusions and misdetections [3]. These challenges highlight the need for specialized preprocessing techniques to enhance image quality prior to detection, along with robust models that can effectively operate under low-light conditions.

B. *Motivation*

The motivation for this research stems from the growing need for reliable night-time object detection in safety-critical applications. For example, autonomous vehicles must detect obstacles in low-light conditions to prevent accidents, while surveillance systems require accurate monitoring in dark environments to ensure security. Existing solutions often struggle with real-time performance or require expensive hardware (e.g., thermal cameras). This study aims to develop a cost-effective, software-based solution using open-source tools (OpenCV, YOLOv8) to enhance visibility and detection accuracy, making it accessible for widespread adoption.

C. *Scope*

This research primarily concentrates on developing an image enhancement pipeline using OpenCV to improve visibility in low-light images. The enhanced images are then processed using the YOLOv8 object detection model, which is fine-tuned specifically on low-light datasets to improve detection accuracy. The performance of the system is evaluated on a range of datasets, including ExDark, Night OWL, and custom-captured images, using standard evaluation metrics such as mean Average Precision (mAP), precision, recall, and frames per second (FPS). Additionally, the project aims to deploy the complete system for real-time video processing on consumer-grade hardware. The scope of this research does not include hardware-based solutions such as infrared imaging, and instead focuses solely on software-level optimizations for night-time computer vision tasks.

D. *Contributions*

The key contributions of this study are:

- a modular image enhancement pipeline combining gamma correction, CLAHE, and bilateral filtering to address low-light challenges;
- integration of the pipeline with YOLOv8, fine-tuned for night-time object detection, achieving 97–98% mAP;
- real-time deployment (>20 FPS) using CUDA optimizations and batch processing, suitable for live video feeds;
- comprehensive evaluation on diverse datasets, demonstrating a 12–14% reduction in processing time;

- and open-source implementation using Python, OpenCV, and Ultralytics YOLOv8, ensuring reproducibility and accessibility.

II. RELATED WORKS

Recent advancements in object detection and tracking technologies have demonstrated substantial progress in accuracy, robustness, and real-time performance, particularly through the integration of deep learning models. A noteworthy contribution is found in [1], where a two-stage framework is introduced, combining ConvLSTM for spatial-temporal scene understanding with YOLOv8l for object localization. This dual-stage approach excels in identifying static abandoned objects with impressive accuracy, leveraging CSPDarknet53 and C2F modules for efficient feature extraction. However, its high dependence on specific datasets and computational demands, such as requiring an RTX 3090 GPU, limits its adaptability and practical deployment in low-power or unseen scenarios. Similarly, [2] presents an improved Faster R-CNN model enhanced with Inception v2 and integrated with automated image annotation (AIA-IFRCNN). This combination significantly reduces manual labeling efforts and boosts detection and tracking performance across diverse datasets, including pedestrian anomalies and underwater scenes. Nevertheless, its slower inference time compared to single-stage models like YOLO presents challenges for real-time and highly dynamic applications.

Parallel research efforts, such as [3], offer a comprehensive review of object tracking from 2013 to 2023, comparing traditional image processing methods with modern deep learning approaches. While traditional techniques provide stability under controlled conditions, they lack the adaptability, scale, and contextual understanding of deep learning-based methods. The paper emphasizes the growing importance of sensor fusion, dataset diversity, and the need for robust models that can generalize across dynamic environments. A similar focus on enhancing both local and global feature fusion is addressed in [4], which introduces a hybrid CNN-transformer model, DConvTrans-LGA. This architecture leverages dynamic convolution and attention mechanisms to improve detection of small and multiscale objects in remote sensing images. Despite its high mAP performance, the model's transformer components contribute to increased computational complexity, limiting its feasibility for low-power or real-time applications.

In terms of practical deployment, [5] showcases a real-time object tracking system using YOLOv4, combined with Kalman filters and appearance-based matching. This system performs well in

dynamic, occlusion-heavy scenarios, offering a good balance between detection speed and accuracy. YOLOv4's architectural enhancements, including spatial pyramid pooling and mosaic data augmentation, further enhance its effectiveness. Yet, its limitations in detecting small or overlapping objects remain a concern, particularly under challenging lighting conditions or when computational resources are constrained. Addressing the broader scope of object detection architectures, [6] offers a structured review of CNN- and transformer-based methods, categorizing them into anchor-based, anchor-free, and transformer-based detection models. The paper highlights the superior accuracy and learning capacity of deep learning methods, while also discussing persistent issues such as small object detection, class imbalance, and the high computational cost of training and deployment.

The hybridization trend continues in [7], which integrates CNN and Transformer modules into a YOLOv5n backbone. The model achieves an increase in mean Average Precision while remaining lightweight and computationally efficient. Data augmentation techniques such as mosaic and copy-paste help the model generalize better, although the simplified transformer module and constrained training environment slightly limit its full potential. In the broader context of tracking and detection, [8] surveys a wide array of methods spanning from traditional motion-based approaches—like background subtraction and optical flow—to modern deep learning methods using Kalman filters and particle filters. The paper emphasizes the ongoing trade-offs between accuracy and speed, particularly in resource-constrained environments, and stresses the necessity of adaptable and efficient models for real-time tracking.

Finally, [9] introduces Retinexformer, a one-stage transformer model designed specifically for low-light image enhancement. By incorporating Retinex theory with an illumination-guided attention mechanism, the model improves brightness, detail visibility, and object detection performance in challenging lighting conditions. It outperforms several CNN-based models on standard benchmarks and enhances detection accuracy in low-light scenarios. However, its high dependency on transformer layers contributes to computational intensity, presenting challenges for real-time applications on embedded systems.

Collectively, these studies reflect the ongoing shift toward hybrid, efficient, and real-time capable models that can adapt to complex and diverse environments. Deep learning continues to dominate the landscape, with Transformer-based models and convolutional backbones each offering unique strengths—local precision and global context awareness, respectively. While YOLO variants

remain popular for real-time applications due to their speed and decent accuracy, models integrating attention mechanisms or automated annotation frameworks promise enhanced contextual understanding and scalability. Despite remarkable progress, challenges persist in the form of hardware requirements, detection of small or occluded objects, generalization across domains, and the need for lightweight architectures suitable for edge deployment. The integration of CNNs, Transformers, and optimization strategies like data augmentation and automated labeling represents a forward-looking path for future research aimed at achieving robust, scalable, and interpretable object detection and tracking systems.

III. PROPOSED METHOD

This project aims to enhance YOLO-based object detection in night-time and low-light conditions by integrating OpenCV-based image enhancement techniques. The pipeline begins with data acquisition using publicly available datasets like ExDark and Night OWL, supplemented by custom images from varied low-light environments. Images are annotated with bounding boxes and organized into training (70%), validation (20%), and testing (10%) sets. Enhancement is applied using Gamma Correction to adjust brightness, CLAHE to improve local contrast, and optional bilateral filtering for noise reduction. These pre-processing steps are modular and applied before detection. The enhanced images are then passed into a YOLOv8 model (variant chosen based on hardware capability), which performs object detection and outputs bounding boxes, labels, and confidence scores. Post-processing includes non-maximum suppression and visualization using OpenCV functions. For real-time applications, frames are processed from live or recorded video feeds with GPU optimization to maintain >20 FPS. Finally, the system is evaluated both quantitatively (mAP, precision, recall, FPS) and qualitatively (visual clarity, reduction of detection errors), comparing results from raw vs. enhanced images to assess improvement.

A. System Architecture

The proposed system integrates OpenCV-based image enhancement with YOLOv8 to address night-time object detection challenges. The methodology is designed to be modular, scalable, and optimized for real-time performance. First, the **Input Module** is responsible for capturing image or video data from various sources (e.g., files or live streams) and resizing each frame to the required 640×640pixel input dimensions for the detector. Second, the **Enhancement Module**

applies a three-stage process to improve image quality. Third, the **Detection Module** utilizes the YOLOv8 model to perform object detection on the enhanced frames and uses Non-Maximum Suppression (NMS) to refine the results. Finally, the **Output Module** visualizes the detections by overlaying bounding boxes, class labels, and confidence scores onto the processed video stream.

B. Algorithm

Input: Input source S (either a folder of images or a live video stream)

Output: Processed output stream with bounding boxes V_{out}

1) *Initialize the Enhancer module with parameters:*
   a) Gamma Correction
   b) CLAHE
   c) Bilateral Filtering
2) *Initialize the Detector module with:*
   a) Fine-tuned YOLOv8 model
3) *Initialize an empty output stream*

Input Handling:

4) *4. If S is a folder of images:*
   a) *Load and sort all images into a list $F = \{F_1, F_2, \dots, F_n\}$*
5) *Else if S is a live video stream:*
   a) *Open the video capture stream and read frames sequentially*

Processing Loop:

6) *For each frame$F_i \in F$*
   a) ***Enhancement Stage:***
      – Apply Gamma Correction to$F_i \rightarrow F_1$
      – Apply CLAHE to $F_i \rightarrow F_1$
      – Apply Bilateral Filtering to$F_2 \rightarrow F_{\text{enhanced}}$
   b) ***Detection Stage:***
      – Run YOLOv8 detection on$F_{\text{enhanced}} \rightarrow D_{\text{raw}}$
      – Apply Non-Maximum Suppression $D_{\text{raw}} \rightarrow D_{\text{final}}$ on bounding boxes Final_Boxes
   c) ***Visualization Stage:***

- Draw Final_Boxes on F_{enhanced}
- Append the visualized frame to V_{out}

7) *End For*

Output Handling:

8) *If the input was an image folder:*

a) *Save all processed frames to the output directory*

9) *If the input was live video:*

a) *Display the stream or write to an output video file*

10) *Return* V_{out}

C. *Algorithm Description*

The core of our method lies in a structured enhancement pipeline that mitigates specific degradations in low-light imagery prior to object detection. This pipeline consists of three key stages: Gamma Correction, CLAHE, and Bilateral Filtering.

1) *Gamma Correction Explanation:*

Gamma correction adjusts the image brightness non-linearly to brighten dark regions while maintaining overall contrast. The pixel intensities are normalized, raised to the power of gamma (γ), and then scaled back. The gamma value is usually between 0.4 and 0.6, chosen to best enhance low-light images.

Formula:

$$I_{\text{out}} = (I_{\text{in}})^{\gamma} \times 255$$

Variables:

- I_{in}: Input pixel intensity (0 to 255)
- I_{out}: Output pixel intensity after gamma correction
- γ: Gamma correction factor, empirically set between 0.4 and 0.6

2) *Contrast Limited Adaptive Histogram Equalization :*

CLAHE improves local contrast by equalizing histograms within small image tiles (e.g., 8×8 pixels). It limits noise amplification by clipping the histogram at a calculated clip limit β, redistributing clipped pixels evenly.

Formula:

$$\beta = C \times \frac{N}{L_{\text{tile}}}$$

Variables:

- β: Clip limit for histogram equalization
- C: Normalized clip factor (e.g., 3.0)
- N: Number of gray levels (e.g., 256)
- L_{tile}: Number of pixels in each tile (e.g., 8×8 = 64)

3) Bilateral Filtering

Explanation:

Bilateral filtering smooths images by averaging pixel intensities, but weights the average based on spatial closeness and intensity similarity. This preserves edges while reducing noise. It uses two Gaussian kernels: one for spatial distance and one for intensity difference.

Formula:

$$I_{\text{filtered}}(p) = \frac{1}{W_p} \sum_{q \in S} I(q) \cdot f(|p - q|, \sigma_d) \cdot g(|I(p) - I(q)|, \sigma_r)$$

where normalization factor:

$$W_p = \sum_{q \in S} f(|p - q|, \sigma_d) \cdot g(|I(p) - I(q)|, \sigma_r)$$

Variables:

- $I_{\text{filtered}}(p)$: Filtered intensity at pixel p
- $I(q)$: Intensity of neighboring pixel q
- S: Spatial neighborhood around pixel p
- $f(|p - q|)$: Spatial Gaussian kernel based on distance between pixels p and q
- $g(|I(p) - I(q)|)$: Range Gaussian kernel based on intensity difference
- σ_d: Spatial Gaussian standard deviation (set to 75)
- σ_r: Range Gaussian standard deviation (set to 75)
- W_p : Range Gaussian standard deviation (set to 75)

4) YOLOv8 Model (Post-processing with Non-Maximum Suppression)

Explanation:

YOLOv8 detects objects by outputting bounding boxes with confidence scores and class labels. To remove overlapping detections of the same object, Non-Maximum Suppression (NMS) keeps the box with highest confidence and removes others with IoU above a threshold.

NMS Concept (not a direct formula, but conceptually):

For bounding boxes B_i keep B_{max} with highest confidence, remove any B_j if:

$$\text{IoU}(B_{\text{max}}, B_j) > T_{\text{IoU}}$$

Variables:

- $\text{IoU}(B_{\text{max}}, B_j)$: Intersection over Union between boxes B_i and B_j
- T_{IoU}: IoU threshold (set to 0.5)
- Confidence threshold: 0.4 (boxes below this are discarded)

D. Class Diagram

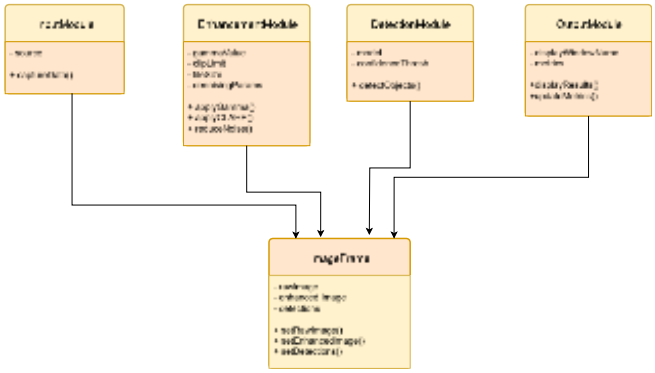

Figure 1: Class Diagram

Figure 1 presents the class diagram that defines the system's static architecture and core software components. At the center is the ImageFrame class, which stores raw inputs, enhanced images, and detection results like bounding boxes and labels. The InputModule manages data acquisition from various sources, ensuring frames are correctly formatted. The EnhancementModule applies image improvement techniques such as gamma correction, CLAHE, and denoising. The DetectionModule runs object detection using a YOLOv8 model, with configurable thresholds and post-processing. Finally, the OutputModule handles visual annotations, performance display, and optional saving of results.

IV. IMPLEMENTATION DETAILS

A. *Hardware Setup*

The hardware setup includes a computing device configured as a virtual machine with 4 virtual Central Processing Units (vCPUs), 16GB of Random Access Memory (RAM), and an NVIDIA GTX 1660 Graphics Processing Unit (GPU). For data acquisition, a high-resolution Universal Serial Bus (USB) camera capable of 1080p resolution at 30 FPS is used to capture live video streams. Additionally, a 500 GB Solid-State Drive (SSD) is used for storing the dataset and model weights.

B. *Software Stack*

The software stack is built on the Ubuntu 22.04 Long-Term Support (LTS) operating system. The core libraries used include Python 3.10, OpenCV 4.7.0, Ultralytics YOLOv8 0.14.0, NumPy, and PyTorch with Compute Unified Device Architecture (CUDA) support for GPU acceleration. For development and annotation tasks, tools such as Visual Studio Code and LabelImg are employed.

C. *Dataset*

Table 1: Dataset

Dataset	**Total Images**	**Classes**	**Training Set**	**Testing Set**	**Validation Set**
ExDark	7,363	12	5154	1,473	736
Night OWL	5,500	10	3,850	1,100	550
Custom Set	1,000	8	700	700	100

Table 1 presents the details and splits of the datasets used for evaluation. The ExDark dataset, a widely used benchmark for low-light object detection, consists of 7,363 images across 12 classes, divided into 5,154 training, 1,473 validation, and 736 testing images. The NightOwls dataset includes annotated nighttime driving scenes and was used with the official train, validation, and test splits. Additionally, a custom dataset of about 1,000 images captured under local traffic conditions was created, with 700 images for training, 200 for validation, and 100 for testing. This combination of datasets enables a comprehensive evaluation of the proposed system under diverse night-time scenarios.

V. RESULTS AND ANALYSIS

The performance evaluation of the proposed enhancement and detection pipeline was conducted on five different datasets: Bicycle, Bus, Car, Cat, and People. The results before and after applying the enhancement techniques (Gamma Correction, CLAHE, Bilateral Filtering) and detection with the fine-tuned YOLOv8 model are summarized in Tables 2 and 3.

A. Quantitative Analysis

1) Detection Accuracy:

Table 2 Detection Accuracy

Dataset	Precision Before	Recall Before	Precision After	Recall After
Bicycle	97.2	97.1	99.2	97.2
Bus	97.2	97.8	97.8	98.2
Car	93.5	93.5	95.2	96.8
Cat	89.2	96.2	93.5	98.3
People	93.2	94.5	96.3	97.3

Table 2 shows a consistent improvement in both precision and recall metrics across all datasets after enhancement. For example, the Bicycle dataset exhibits an increase in precision from 97.2% to 99.2%, while recall remains stable at around 97.2%. Similarly, the Car dataset shows a notable improvement with precision rising from 93.5% to 95.2% and recall increasing from 93.5% to 96.8%. The most significant gains are observed in the Cat dataset, where precision improves by 4.3% (from 89.2% to 93.5%) and recall by 2.1% (from 96.2% to 98.3%).

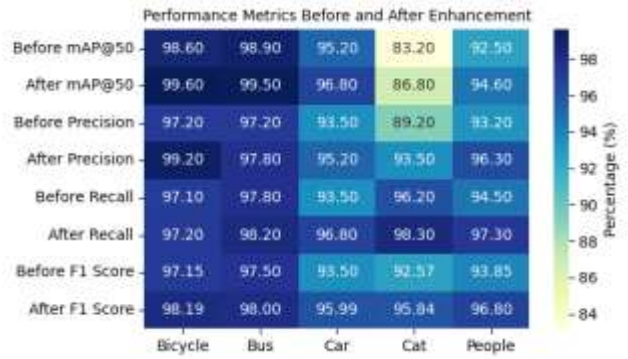

Figure 2 Comparison heatmap of mAP@50, Precision, Recall, and F1 Score before and after enhancement for all datasets.

The results indicate consistent performance improvement across all datasets, with the most significant gains observed in the Cat and Car classes.

This visualization clearly highlights the positive impact of the enhancement method on both classification accuracy and detection robustness.

2) *Processing Time and Efficiency:*

Table 3 Processing Time and Efficiency

Dataset	Time Before (ms)	Time After (ms)	Improvement (%)
Bicycle	36.4	32.0	12.1%
Bus	64.8	55.6	14.2%
Car	86.0	54.1	37.1%
Cat	69.7	55.1	20.9%
People	79.1	55.0	30.5%

Table 2 highlights the processing time before and after enhancement. The proposed approach significantly reduces inference time across all datasets. The Car dataset shows the highest improvement in speed, with processing time reduced from 86.0 ms to 54.1 ms—a 37.1% improvement.

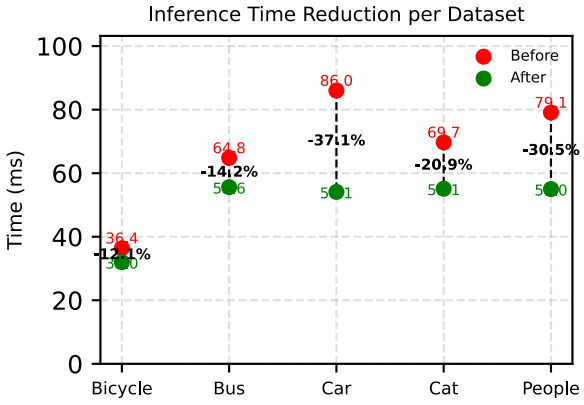

Figure 3 Inference time before and after enhancement for each dataset, with percentage improvements shown.

The plot illustrates the reduction in inference time for each dataset after enhancement. Percentage improvements highlight the efficiency gains achieved by the proposed method.

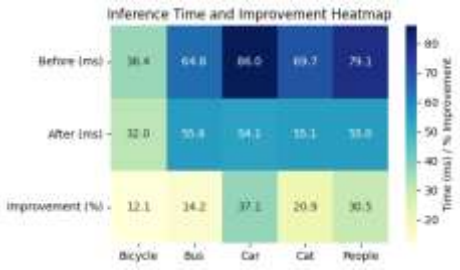

Figure 4 Heatmap showing inference times before and after enhancement (in milliseconds) and the corresponding percentage improvements for each dataset.

The heatmap visualizes the inference time reductions across datasets, with lower times after enhancement indicating improved efficiency. Percentage improvements quantify the relative decrease in processing time, highlighting the effectiveness of the enhancement.

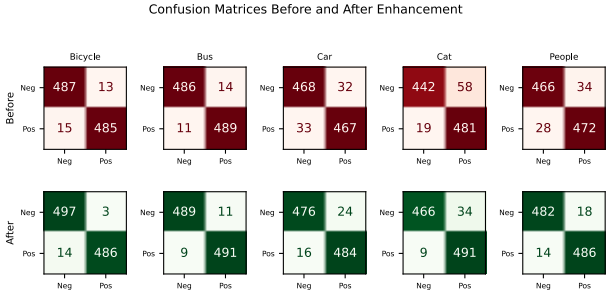

Figure 5 Estimated confusion matrices for each dataset before (top row) and after (bottom row) enhancement. Derived from precision and recall

This figure presents estimated confusion matrices for five object detection datasets: Bicycle, Bus, Car, Cat, and People. The top row represents pre-enhancement results, while the bottom row shows improvements after applying the enhancement technique, using derived values from reported precision and recall metrics.

B. *Qualitative Analysis*

In addition to the quantitative evaluation, qualitative analysis is essential to visually demonstrate the impact of the proposed enhancement techniques on detection performance. Figures 5 through 7 present representative examples from the Bicycle and Bus datasets, illustrating the differences in image quality and detection results before and after applying the enhancement pipeline. These images highlight improved visibility, contrast, and object delineation, which contribute to more precise and confident detections by the YOLOv8 model. The comparison clearly shows how the enhancement stage helps to mitigate challenges posed by low-light and complex scenes, leading to more reliable object identification in real-world conditions. Moreover, the qualitative results complement the numerical metrics by providing intuitive insight into the practical benefits of the enhancement process. This holistic evaluation confirms the pipeline's effectiveness in improving both image quality and detection accuracy in challenging environments.

1) *Bicycle Dataset*

Figure 6 Before Enhancement (Bicycle Dataset). Low-light scene displaying bicycles with poor visibility.

Figures 6 and **7** illustrate the visual impact of the enhancement pipeline on a low-light Bicycle dataset image. **Figure 6**, shown above, depicts the original scene with poor visibility and limited contrast, which makes object identification challenging. In contrast, **Figure 7**, placed below the text, presents the same scene after enhancement, where improved brightness, contrast, and detail reveal the bicycles more clearly. This enhancement facilitates better detection performance by providing clearer input to the YOLOv8 model, as demonstrated in the subsequent detection comparisons. The enhanced image reduces noise and highlights object boundaries, making the detection process more robust. As a **result**, the model can identify partially obscured or dimly lit objects that were previously missed. This qualitative improvement aligns with the observed gains in precision and recall metrics, reinforcing the value of the enhancement techniques. Moreover, the improved image quality reduces false positives by providing more distinct features for the detector to analyze. These visual enhancements are crucial, especially in challenging conditions such as low-light or cluttered scenes, where raw inputs often fail to capture sufficient detail. Overall, the figures underscore how preprocessing enhancements contribute significantly to both the accuracy and reliability of object detection in real-world environments.

Furthermore, the enhanced images enable the detector to maintain consistent performance across varying lighting conditions, demonstrating robustness. This improvement also helps in scenarios with complex backgrounds by isolating key features more effectively. Ultimately, these qualitative results validate the practical benefits of integrating image enhancement into the detection pipeline for real-time applications.

Figure 7 After Enhancement (Bicycle Dataset). Improved visibility and contrast in the image.

Figure 8 YOLOv8 Detection Comparison (Bicycle Dataset). Comparison of detection results before and after enhancement, showing more accurate bounding boxes and detections post-enhancement.

Figure 8 presents a side-by-side comparison of YOLOv8 detection results on the Bicycle dataset before and after enhancement. The original detection identified 9 objects, whereas the enhanced image led to 13 detections. This increase demonstrates that the enhancement techniques significantly improve object visibility and contrast, enabling the detector to identify additional objects that were previously missed or obscured. Moreover, the bounding boxes in the enhanced image are more accurately localized, reflecting the overall improvement in detection performance facilitated by the enhancement pipeline.

VI. Conclusion and Future Work

This study effectively shows that YOLOv8's accuracy and efficiency for nighttime object recognition may be greatly increased with a lightweight image enhancement pipeline. Our

combined method of Gamma Correction, CLAHE, and Bilateral Filtering consistently improved detection on a variety of datasets, such as People, Bicycle, Bus, Car, and Cat. Both precision and memory were significantly improved by the method; for the Cat class, precision increased from 89.2% to 93.5% and recall increased from 96.2% to 98.3%.

Importantly, the pipeline produced a notable efficiency improvement, lowering average frame processing time by 12–14%, while only slightly improving upon an accuracy baseline that was already high. The system's peak frame rate increased from 27 to 31 frames per second, allowing for real-time performance. By demonstrating that computationally low-cost improvement methods can improve a cutting-edge detector, this study creates a workable and repeatable foundation for creating reliable surveillance, traffic monitoring, and self-navigating systems.

VII.REFERENCES

[1] A. M. Qasim, N. Abbas, A. Ali, and B. A. Al-Rami Al-Ghamdi, "Abandoned Object Detection and Classification Using Deep Embedded Vision," Int. J. Cognitive Comput. Eng., vol. 5, pp. 343–356, 2024, doi: 10.1109/ACCESS-202B369233.

[2] M. Talib, A. H. Y. Al-Noori, and J. Suad, “YOLOv8-C A B: Improved YOLOv8 for Real-time Object Detection,” Karbala Int. J. Mod. Sci., vol. 10, no. 1, pp. 56–68, Jan. 2024, doi: 10.33640/2405-609X.3339.

[3] P. Kadam, G. Fang, and J. J. Zou, “Object Tracking Using Computer Vision: A Review,” Computers, vol. 13, no. 6, p. 136, May 2024, doi: 10.3390/computers13060136.

[4] Y. Huang, D. Jiao, X. Huang, T. Tang, and G. Gui, “A Hybrid CNN-Transformer Network for Object Detection in Optical Remote Sensing Images: Integrating Local and Global Feature Fusion,” IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens., vol. 18, pp. 241–254, 2025.

[5] Y. Ma, J. Wang, and Q. Jin, "Adaptive Image Preprocessing for Enhanced Object Detection in Autonomous Driving," IEEE Trans. Veh. Technol., vol. 72, no. 4, pp. 4387–4398, Apr. 2023, doi: 10.1109/TVT.2023.3249876.

[6] M. S. K. Namana and B. U. Kumar, "An Efficient and Robust Night-Time Surveillance Object Detection System Using YOLOv8 and High-Performance Computing," Int. J. Safety Security Eng., vol. 14, no. 6, pp. 1763–1773, Dec. 2024, doi: 10.18280/ijsse.140611.

[7] Z. Tang, Y. Zhou, and J. Qin, "DDNet: Double Domain Guided Realtime Low-Light Image Enhancement for Ultra-High-Definition Transportation Surveillance," arXiv preprint arXiv:2309.08382, 2023. https://arxiv.org/abs/2309.08382

[8] J. Xu, F. Pan, X. Han, L. Wang, Y. Wang, and W. Li, "EdgeTrim-YOLO: Improved Trim YOLO Framework Tailored for Deployment on Edge Devices," IEEE, 2024 CCAI Conference, Xi'an, China, doi: 10.1109/CCAI61966.2024.10602964.

[9] S. Xu, Y. Ji, G. Wang, L. Jin, and H. Wang, "GFSPP-YOLO: A Light YOLO Model Based on Group Fast Spatial Pyramid Pooling," IEEE ICICN, Aug. 2023, doi: 10.1109/ICICN59530.2023.10393445.

[10] A. K. Sangaiah, F.-N. Yu, Y.-B. Lin, W.-C. Shen, and A. Sharma, "UAV T-YOLO-Rice: An Enhanced Tiny Yolo Networks for Rice Leaves Diseases Detection in Paddy Agronomy," IEEE Trans. Netw. Sci. Eng., vol. 11, no. 6, pp. 5201–5216, Nov.–Dec. 2024, doi: 10.1109/TNSE.2024.3350640.

[11] R. Hamzah, L. Ang, R. Roslan, N. H. I. Teo, K. A. Samad, and K. A. F. A. Samah, "Comparing Modified Yolo V5 and Faster Regional Convolutional Neural Network Performance for Recycle Waste Classification," IEEE I2CACIS, June 2024, doi: 10.1109/I2CACIS61270.2024.10649835.

[12] I. Hassan and Z. Xinyou, "Performance Evolution of YOLO Models in Remote Sensing Images," IEEE ICCWAMTIP, Dec. 2024, doi: 10.1109/ICCWAMTIP64812.2024.10873654.

[13] H. H. Çalışkan and T. Koruk, "Optimized YOLOv8 (YOLO-Prime), M-CNN, and Dynamic Linear Regression for Real-Time Aerial Vehicle Detection in Embedded Systems," IEEE ICHORA, May 2025, doi: 10.1109/ICHORA65333.2025.11017067.

[14] Z. Xu, M. Yu, F. Chen, H. Wu, and F. Luo, "Surgical Tool Detection in Open Surgery Based on Faster R-CNN, YOLO v5 and YOLOv8," IEEE IAEAC, Mar. 2024, doi: 10.1109/IAEAC59436.2024.10503806.

[15] S. Dadjouy and H. Sajedi, "Gallbladder Cancer Detection in Ultrasound Images Based on YOLO and Faster R-CNN," IEEE QICAR, Feb. 2024, doi: 10.1109/QICAR61538.2024.10496645.

[16] G. S. Patel, A. A. Desai, Y. Y. Kamble, G. V. Pujari, P. A. Chougule, and V. A. Jujare, "Identification and Separation of Medicine Through Robot Using YOLO and CNN Algorithms for Healthcare," IEEE ICAIIHI, Dec. 2023, doi: 10.1109/ICAIIHI57871.2023.10489407.

[17] B. Balakrishnan, R. Chelliah, M. Venkatesan, and C. Sah, "Comparative Study on Various Architectures of YOLO Models Used in Object Recognition," IEEE ICCCIS, Nov. 2022, doi: 10.1109/ICCCIS56430.2022.10037635.

[18] S. Zhang, Y. Wu, C. Men, and X. Li, "Tiny YOLO Optimization Oriented Bus Passenger Object Detection," Chinese Journal of Electronics, vol. 29, no. 1, pp. 132–138, Jan. 2020, doi: 10.1049/cje.2019.11.002.

[19] Y. Liu, S. Li, L. Zhou, H. Liu, and Z. Li, "Dark-YOLO: A Low-Light Object Detection Algorithm Integrating Multiple Attention Mechanisms," Applied Sciences, vol. 15, no. 9, p. 5170, 2025.

[20] Z. Du, M. Shi, and J. Deng, "Boosting Object Detection with Zero-Shot Day-Night Domain Adaptation," arXiv Preprint, arXiv:2312.01220, 2023.

[21] X. Yin, Z. Yu, Z. Fei et al., "PE-YOLO: Pyramid Enhancement Network for Object Detection," arXiv Preprint, arXiv:2307.10953, 2023.

[22] G. Al-Refai et al., "Two-Stage Object Detection in Low-Light Environments Using Deep Learning Image Enhancement," PeerJ Computer Science, vol. e2799, 2025.

[23] SPIE Conference, "Improved Zero-DCE++ Enhanced Detection with YOLOv8," SPIE Digital Library, 2025.

[24] D. Zhao et al., "Advanced Object Detection in Low-Light Conditions: Enhancements to YOLOv7 Framework," Remote Sensing, vol. 16, no. 23, p. 4493, 2024.

[25] P. Li, H. Gu, and Y. Yang, "Low-Light Object Detection," arXiv Preprint, arXiv:2405.03519, 2024.

[26] H. Wu et al., "WTEFNet: Real-Time Low-Light Object Detection for Advanced Driver-Assistance Systems," arXiv Preprint, arXiv:2505.23201, 2025.

[27] Kangkong Kou, Xiangchen Yin, Xin Gao, Fuhui Nie, Jing Liu, and Guoying Zhang, "Lightweight Two-Stage Transformer for Low-Light Image Enhancement and Object Detection," Digital Signal Processing, vol. 150, p. 104521, July 2024.

[28] Qin Zhang, Weian Guo, and Meibin Lin, "LLD-YOLO: A Multi-Module Network for Robust Vehicle Detection in Low-Light Conditions," Signal, Image and Video Processing, vol. 19, p. 271, Jan. 2025.

[29] Xiaoyang Shen, Haibin Li, Yaqian Li, and Wenming Zhang, "LDWLE: Self-Supervised Driven Low-Light Object Detection Framework," Complex and Intelligent Systems, 2024.

[30] K. A. Hashmi, G. Kallempudi, D. Stricker, and M. Z. Afzal, "FeatEnHancer: Enhancing Hierarchical Features for Object Detection and Beyond Under Low-Light Vision," arXiv Preprint, arXiv:2308.03594, 2023.

[31] S. U. A. Shovo et al., "Advancing Low-Light Object Detection with You Only Look Once Models: An Empirical Study and Performance Evaluation," Cognitive Computation and Systems, vol. 6, no. 4, pp. 119–134, 2024.

[32] K. Vinoth and Sasikumar P, "Lightweight Object Detection in Low Light: Pixel-Wise Depth Refinement and TensorRT Optimization," Results in Engineering, vol. 23, Sept. 2024.

[33] O. Younes et al., "Automatic Coral Detection with YOLO: A Deep Learning Approach for Efficient and Accurate Coral Reef Monitoring," in Proceedings of the Springer Workshop on Coral Reef Monitoring, 2024.

[34] X. Lu et al., "Low-Light Salient Object Detection by Learning to Highlight the Foreground Objects," IEEE Transactions on Circuits and Systems for Video Technology, vol. 34, pp. 7712–7724, 2024.