



OPEN Deep learning framework for barcode localization and decoding using simulated UAV imagery

Faris Alsulami^{1,4}✉ & N. Z. Jhanjhi^{2,3,4}

Automated warehouse stock tracking is becoming increasingly important for improving logistics and reducing manual errors. Unmanned Aerial Vehicles (UAVs) offer a promising solution by enabling automated barcode scanning from above. However, challenges such as poor lighting, shadows, and partial occlusions still limit the reliability of real-time barcode detection and decoding. This research presents a deep learning framework evaluated on simulated UAV imagery for barcode inventory management. The proposed system uses the YOLOv8 object detection model to accurately localize both 1D and 2D barcodes in images captured from a UAV perspective. With a mean Average Precision (mAP) of 92.4%, the model demonstrates strong performance even in complex warehouse conditions. Once localized, the barcode regions are decoded using OpenCV's barcode module. The extracted data, including product ID and quantity, is then automatically updated into a MySQL database to simulate real-time stock updates. Although tested using simulated aerial imagery, the system is designed to be drone-ready with minimal adjustments. This modular approach shows potential for real-world UAV deployment and contributes to reducing human effort and errors in inventory tracking.

In today's rapidly evolving world with highly competitive market, businesses are feeling pressure to keep their inventory in check¹. As industries increasingly rely on e-Commerce, supply chains are becoming more complex and the need for accurate and up-to-date inventory data is becoming more and more crucial²⁻⁴.

Traditional inventory management methods, such as manual inventory management and periodic cycle counting, are no longer sufficient to keep up with the speed and volume of modern operations⁵. Conventional approaches often lead to delays, human errors, and inefficiencies, which can result in stock shortages, overstocking, and poor customer satisfaction⁵.

To align with the growing pace, Industry 4.0 technologies⁶ continue to evolve more rapidly. Businesses are seeking innovative solutions that can streamline inventory control, reduce costs, and improve decision-making. As a result, automation and real-time tracking have become key priorities⁷, pushing organizations to explore smarter, more agile systems to manage their inventory in a way that supports fast-moving global supply chains.

One latest invention is drone, or UAVs which have been revolutionizing logistics by changing the way inventory is managed, and warehouses operate⁸. For example, drones can quickly scan large areas of inventory, saving a lot of time compared to traditional stock-taking methods. They are equipped with sensors and smart technologies that can collect data without human errors and perform automation by repeating tasks like inventory scanning and counting. Another big advantage is that drones can access difficult-to-reach or unsafe areas, such as high shelves or dangerous spots, which makes the workplace safer for everyone⁹. Figure 1 illustrates the comparison between traditional inventory methods and modern, drone-assisted inventory management, highlighting the key benefits of UAV assisted inventory management.

These drones are equipped with high-resolution cameras and sensors¹⁰ that allow them to navigate large warehouse spaces, scanning products and pallets quickly and efficiently. UAVs are capable of autonomously flying and capturing images from various angles, even in hard-to-reach areas, significantly speeding up the process of inventory management. To fully utilize their capabilities, drones rely on advanced computer vision and deep learning technologies, which are essential for transforming raw image data into actionable insights.

Deep learning, particularly through techniques like Convolutional Neural Networks, has become central to the development of object detection systems used in drones^{11,12}. The latest deep learning models, such as YOLO¹³ and its subsequent versions like YOLOv8¹⁴, have significantly improved object localization and detection. These

¹Department of Computer and Network Engineering, College of Computer Science and Engineering, University of Jeddah, 23890 Jeddah, Saudi Arabia. ²School of Computer Science, Taylor's University, Subang Jaya, Malaysia. ³Office of Research and Development, Asia University, Taichung, Taiwan. ⁴Faris Alsulami and N. Z. Jhanjhi have contributed equally to this work. ✉email: fnalsulami@uj.edu.sa



Fig. 1. Comparing traditional methods versus drone-powered inventory management.

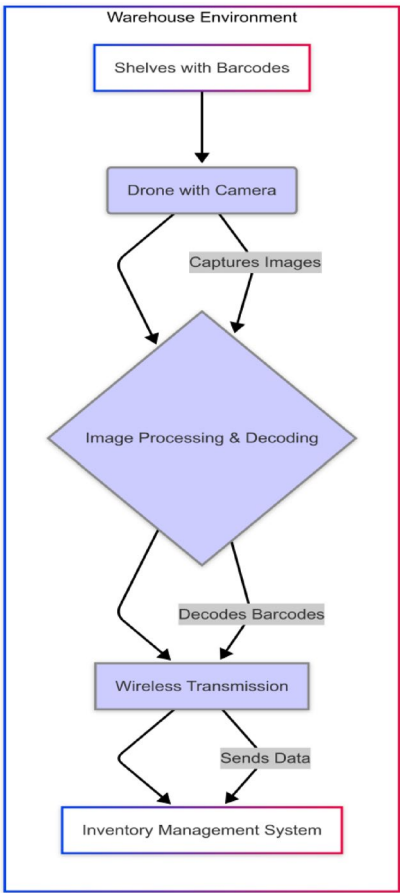


Fig. 2. Simple workflow diagram.

models are particularly effective because they can process images in real time, supporting UAV operations in logistics and warehouse environments¹⁵. YOLO's efficiency lies in its ability to simultaneously predict bounding boxes and class probabilities, making it especially suitable for fast-paced environments such as warehouses¹⁶. This real-time detection capability allows drones to update inventory data dynamically, ensuring continuous and autonomous warehouse management operations.

Figure 2 represents an automated barcode scanning system in a warehouse environment. A drone equipped with a camera captures images of barcoded shelves and processes them to decode barcode information. The decoded data is then wirelessly transmitted to an inventory management system for tracking and updates. This automation enhances warehouse efficiency by reducing manual scanning efforts.

The Fig. 3 shows the sequence diagram for the above process. Many researchers have made significant contributions to the use of UAVs in warehouse management and inventory control, particularly in the area of

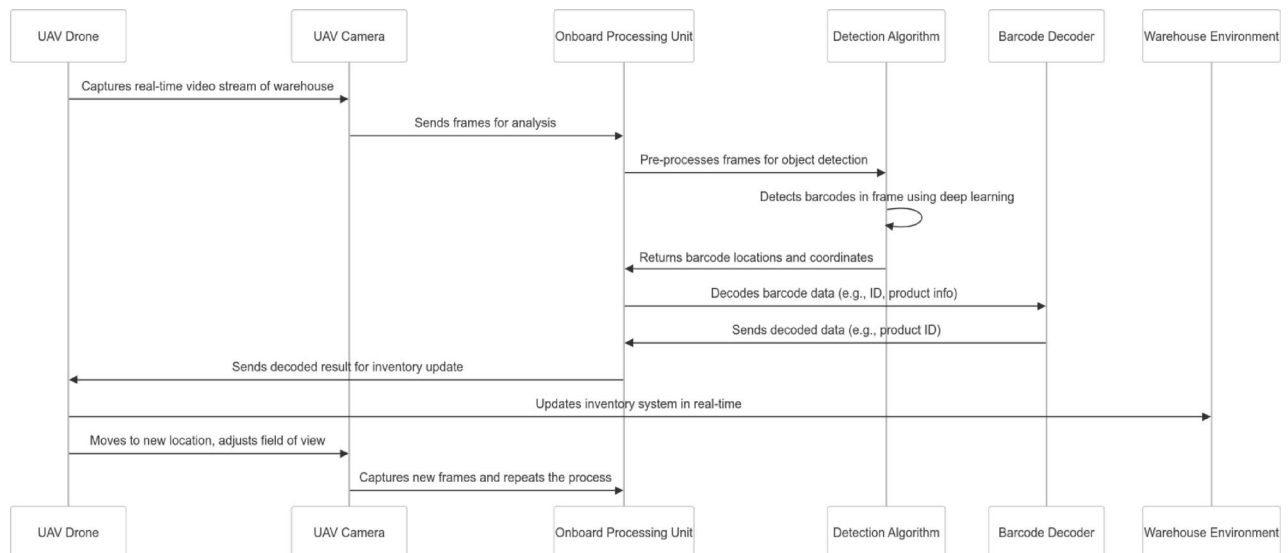


Fig. 3. Sequence diagram.

barcode localization and decoding. While substantial progress has been made, gaps still exist in addressing the challenges posed by dynamic warehouse environments. In dynamic warehouse environments, factors such as poor lighting, occlusion, and barcode distortions can severely impact the accuracy and efficiency of existing automated systems.

While in the past enhancing the drone's capabilities, such as equipping it with better cameras, improved sensors, and advanced lighting systems to help mitigate issues related to poor lighting and occlusion has been researched, however, refining the model's ability to recognize and compensate for environmental factors needs to be researched more. This study addresses that gap by proposing a modular, simulation-based framework inspired by UAV-assisted barcode inventory, supporting real-time product identification and automated stock updates. Although, our implementation uses existing tools like YOLOv8 and OpenCV, the contribution lies in how these tools are customized, combined, and validated for real-world warehouse inventory needs. The system was tested on simulated aerial data and is structured to support future UAV integration with minimal adjustments. Our contributions are as follows:

1. We trained the YOLOv8 model to achieve accurate, real-time detection of 1D and 2D barcodes in complex, dynamic warehouse environments.
2. We evaluated the performance of the YOLOv8 model in real-time barcode detection, assessing its accuracy in locating 1D and 2D barcodes within complex, dynamic warehouse environments.
3. We integrated the YOLOv8 model with OpenCV's barcode decoding module to enable decoding of 1D and 2D barcodes, facilitating seamless inventory management in dynamic warehouse environments.
4. We connected the decoded output to simulate stock updates into a testbed SQL environment, allowing for real-time reflection of inventory changes based on the decoded barcode data.

The remainder of the paper is structured as follows. Section "[Literature review](#)" provides a comprehensive Literature Review, discussing prior research on UAVs, object detection, and barcode localization in warehouse settings. Section "[Methodology: system overview](#)" outlines the research methodology, detailing the approach taken to develop the UAV-guided system and the deep learning techniques used. This section also explains about the dataset, and training process. Section "[Results](#)" presents the results, evaluating system performance in real-time barcode detection. Section "[Discussion and key insights](#)" discusses the results and compares them with existing methods. Finally, Sect. "[Conclusion](#)" concludes the paper by outlining the research limitations and offering insights for future work.

Literature review

The use of UAVs in inventory management has gained significant attention in recent years, as they offer a promising solution for automating and optimizing warehouse operations. Several studies have explored the integration of UAVs in inventory management, focusing on their ability to automate and streamline warehouse operations. UAVs offer numerous benefits in this domain, including enhanced speed, improved accuracy, and reduced human error. Research has demonstrated the effectiveness of drones in tasks such as inventory counting, barcode scanning, and real-time data updates. For instance, UAVs equipped with computer vision and object detection algorithms have shown to increase the efficiency of stock-taking by automating the identification and localization of items, even in hard-to-reach areas. To better understand the various approaches used in UAV-based inventory management, we categorize existing literature into key techniques for barcode localization and decoding. Figure 4 presents a structured taxonomy of barcode detection techniques which serves to contextualize

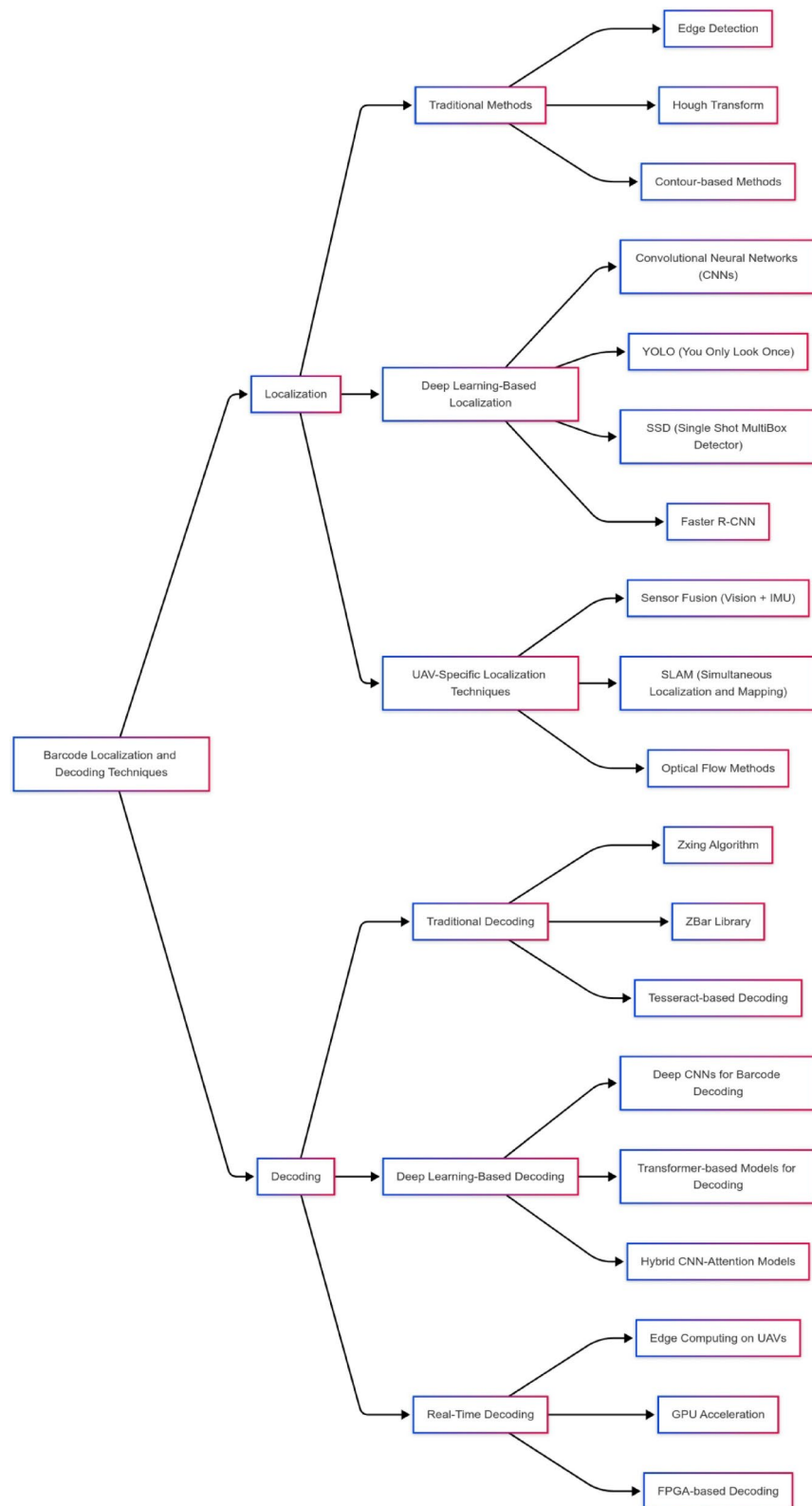


Fig. 4. Taxonomy of barcode localization and decoding techniques in UAV-based inventory management.

our work within the broader landscape of barcode detection research. By organizing existing approaches into clear categories, it helps readers quickly understand the progression of techniques, identify relevant trends, and recognize the novelty of our proposed method.

Barcode localization

Traditional methods

Traditional methods for barcode localization primarily rely on digital image processing techniques. They generally focus on detecting and identifying the position (location) and orientation of a barcode in an image. One such traditional technique is edge detection. Barcodes often have high contrast between the dark bars and the light background, so these techniques identify sharp changes in pixel intensity, which often correspond to the edges of the barcode bars. Bodnár et al. in¹⁷ uses canny edge detector and probabilistic Hough transformation to detect straight lines in barcode images. They preprocessed the images with the blur filter and set a minimum line length in the Hough transformation to improve line grouping and determine barcode structure.

In another research¹⁸, Li et al. utilized local adaptive thresholding as part of their method to handle issues caused by uneven illumination in QR code images. applied a customized Gaussian thresholding algorithm. Unlike conventional Gaussian algorithms, this one dynamically generated the block size for image processing, ensuring the images were processed area-wise at each instance, which allowed for more efficient segmentation. Akinbade et al. in¹⁹ used an adaptive thresholding algorithm to properly segment the text characters from barcode images with complex backgrounds. They applied a customized Gaussian thresholding algorithm that differs from conventional Gaussian algorithms by dynamically generating the block size for image processing. This ensured that the images were processed area-wise at each instance, allowing for more efficient segmentation. Tribak et al. proposed a robust QR code pattern localization approach using image quality assessment in their work²⁰, contour-based segmentation with an achromatic filter to highlight regions of interest, and Hu invariant moments to reduce false positives and improve recognition accuracy, making it suitable for embedded systems.

Some other traditional techniques that have been widely used in barcode localization include the use of morphological operations²¹ (for example dilation and erosion), projection profiles²² (including summing the pixel intensities along rows or columns of the image. The resulting histogram-like projection reveals the presence of the barcode as spikes), template matching (using a pre-defined barcode template and sliding it over the image to match areas that have similar patterns)²³, histogram of oriented gradients²⁴ (focuses on the gradients and orientations of edges) and colour based methods²⁵ (by identifying the dominant color) present in barcodes printed in colours.

Deep learning based methods

Deep learning-based methods for barcode localization involve using neural networks to detect and locate barcodes within images or videos. These methods typically offer improvements over traditional image processing techniques because they have potential to handle complex scenarios such as distorted, low-resolution, or occluded barcodes. CNNs are the foundational deep learning architecture for image-related tasks. They are often used for feature extraction and spatial pattern recognition in images. Tsai et al. implemented deep CNNs in their work²⁶ to classify and identify the source printers of QR codes, utilizing pretrained models like AlexNet, DenseNet201, and ResNet. Chou et al. demonstrated in their work²⁷ robustness to rotation and deformation while doing the same task using the CNN in barcode images with complex backgrounds.

Region based CNN are also studied and implemented in various works for with region proposals, potential bounding boxes where a barcode may be located. Rahman et al. utilized a method for detecting and decoding multiple barcodes simultaneously using the Faster RCNN model in their proposed work²⁸. Similarly, Faster RCNN was enhanced for small object detection in²⁹ while³⁰ proposed an efficient deep learning-based method for 1D barcode localization in shipping labels, demonstrating the versatility of CNN architectures beyond region-based approaches.

YOLO is a more recent and popular real-time object detection system. Unlike traditional object detection models that first generate region proposals, YOLO divides the image into a grid and directly predicts bounding boxes and class probabilities in a single forward pass. Li et al. proposed an improved YOLO-MCG barcode localization algorithm³¹ that utilizes the MobileNetv3- small network and Convolutional Attention Mechanism Module (CBAM). Some other recent researchers exploring various YOLO-based approaches for barcode detection include^{32–34 and 35}.

SSD is another real-time object detection method similar to YOLO but with different architecture. SSD generates multiple bounding boxes at each spatial location and classifies each box. It can be used for multi-scale barcode detection in complex scenes. Dalal et al. used SSD for fast and accurate detection of both 1D and 2D barcodes³⁶.

UAV specific localization techniques

As discussed above, while traditional and deep learning-based methods have significantly improved barcode localization, UAVs present unique challenges due to their dynamic movement, varying perspectives, and environmental uncertainties. The techniques which are specific to localizing barcodes from UAV mounted cameras enable UAVs to navigate and position them first relative to the objects or the environments. Since UAVs operate in 3D space and often rely on autonomous navigation, accurate localization is crucial for tasks like surveying, mapping, inspection, and even package delivery.

SLAM is a technique where the UAV simultaneously builds a map of an unknown environment while keeping track of its own position. Woong Kwon et al. proposed a method for robust autonomous navigation of UAVs in warehouse environments³⁷ by employing a low-cost sensing system with an extended Kalman filter-based multi-sensor fusion framework, incorporating techniques such as Mahalanobis norm for outlier rejection, pseudo-

covariance for visual SLAM integration, and floor lane recognition for absolute localization. Qian et al. used SLAM with YOLOv5s-Lightweight³⁸ for dynamic object detection and filtering to enhance localization accuracy in indoor environments. Yang et al. proposed a CNN-based QR code reading method tailored for UAV package inspection tasks³⁹.

Optical flow is another technique used by researchers to help UAVs maintain stability by detecting how the environment moves relative to the drone⁴⁰. This detects motion patterns to identify potential obstacles and avoid collisions. While sensor fusion combines data from multiple sensors (e.g., cameras, inertial measurement units (IMU), LiDAR, and GPS) to improve localization accuracy and robustness. In barcode localization, sensor fusion helps UAVs by integrating camera-based localization and stereo vision with image-based detection to better estimate distances⁴¹.

Barcode decoding

Barcode decoding is the process of extracting information from a barcode image and converting it into readable data, such as a product ID or inventory number. This involves identifying the barcode, interpreting its pattern, and translating it into digital information. In UAV-based inventory management, barcode decoding enables real-time tracking and data retrieval without human intervention.

Traditional decoding

Traditional barcode decoding methods rely on image processing algorithms to detect and extract barcode information. These techniques work well in controlled environments but may struggle with poor lighting, distortions, or occlusions. Table 1 present a comprehensive summary for literature related to traditional barcode decoding methods.

Deep learning based decoding

Deep learning techniques enhance barcode decoding by making it more robust to noise, distortions, and real-world variations. Instead of relying on handcrafted rules, deep learning models learn to extract and decode barcodes from complex images.

CNNs have been used for both localization decoding of barcodes. Localization involves detecting the boundaries, orientation, and position, which can be challenging in cluttered or noisy images⁵⁴. Once the barcode is localized, CNNs can extract features from the barcode image, such as the lines or patterns that represent the encoded information.

Kamnardsiri et al. applied five existing D-CNN architectures to assess their effectiveness in barcode detection and recognition. They also compared algorithms like YOLO v5 and Faster R-CNN, which are popular in object detection tasks, for their ability to handle barcode images⁵⁵. In the work by⁵⁶, CNNs are used for pixel-wise segmentation to identify barcode regions for decoding. The encoder-decoder architecture proposed by the authors allows for precise segmentation⁵⁶, enabling the decoding of one-dimensional barcodes even in real-time AR scenarios.

Real time decoding

Real-time decoding methods focus on processing barcode images efficiently while maintaining speed and accuracy, crucial for UAV-based applications where delays can impact operations. Table 2 summarizes the key real-time decoding techniques used for barcode recognition and their corresponding studies.

From the reviewed literature, it is evident that significant progress has been made in barcode localization and decoding using both traditional and deep learning-based techniques. However, a major research gap remains in the integration of the latest deep learning algorithms for simultaneous barcode localization and decoding in UAV-based inventory management. Most existing studies focus on these tasks separately rather than as a unified system. Furthermore, a comprehensive simulation-based framework that considers UAV-relevant challenges such as occlusion, cluttered environments, low-light conditions, and dynamic viewpoints has not yet been fully explored. Additionally, real-time inventory updates remain a critical requirement that has not been extensively

Decoding method	Description	Strengths	Technical limitation	Key studies
Zxing (zebra crossing)	Open-source library for decoding 1D and 2D barcode formats (including QR codes)	High-speed processing; supports multiple barcode standards	Accuracy decreases under < 60 lx lighting or >25° tilt; limited performance on reflective or low-contrast surfaces.	⁴² evaluated QR detection at 93% accuracy in consumer images; ⁴³ improved encoding capacity via compression; ⁴⁴ examined performance on curved packaging surfaces.
ZBar library	Lightweight barcode and QR code scanner for still images and video streams.	Real-time decoding; low computational demand; integrates easily with OpenCV.	Decoding accuracy drops > 10% in motion blur or noisy backgrounds	⁴⁵ tested in POS systems with 88% success rate; ⁴⁶ compared classical and DL-based decoders; ⁴⁷ optimized for high-speed scanning applications.
Tesseract OCR	OCR tool decode certain barcode formats by interpreting their alphanumeric data.	Open source. Powerful text recognition capabilities. Works for barcodes with text.	Not robust to rotation or blur; decoding accuracy < 70% for purely graphical barcodes.	⁴⁸ applied to medical label decoding; ⁴⁹ combined with object detection for industrial automation; ⁵⁰ used in parcel sorting systems
Edge detection & Binarization	Classical image-processing methods (like Canny edge detection and thresholding) isolate barcodes.	Works with distorted or skewed images. Well-suited for structured barcode images.	Highly sensitive to illumination and specular reflection; accuracy decreases ~ 20% on glossy or dusty surfaces.	⁵¹ applied morphological operations for 2D decoding; ⁵² introduced adaptive edge model; ⁵³ developed BLADE for robust localization.

Table 1. Traditional barcode decoding methods.

Decoding technique	Description	Studies
Edge computing on UAVs	Decoding happens directly on the UAV using onboard processors, reducing latency.	57
GPU acceleration	Uses hardware-accelerated computing (e.g., NVIDIA Jetson) to speed up deep learning-based barcode recognition.	58
FPGA-based decoding	Field-Programmable Gate Arrays (FPGAs) allow parallel processing for real-time barcode recognition.	59
Streaming-based decoding	Continuous scanning and decoding from live video feeds rather than static images.	60

Table 2. Real-time barcode decoding methods.

explored in conjunction with deep learning-based barcode scanning. Addressing these gaps can lead to a more robust and efficient UAV-based inventory management system.

Methodology: system overview

The proposed system uses UAV-mounted cameras and advanced computer vision techniques to automate inventory management in a warehouse environment. The system follows a structured pipeline consisting of the following key steps:

1. Barcode Localization: YOLOv8 is employed for real-time detection of 1D and 2D barcodes within warehouse images.
2. Barcode Decoding: OpenCV’s barcode decoding module extracts information from the detected barcodes.
3. Database Integration: The decoded data (e.g., product ID and quantity) is stored and updated in a MySQL database for real-time inventory tracking.

Figure 5 illustrates the complete workflow of the proposed UAV-guided barcode localization and decoding system, offering a high-level overview of key modules, from data acquisition to detection and decoding. This diagram helps in understanding the sequential process and modular architecture of our approach.

Barcode localization

Data collection, preprocessing and augmentation

A dataset containing annotated images of barcodes and QR codes is used to train and evaluate the YOLOv8 model. The dataset is structured as follows:

- Training Set: 91% (28,696 images).
- Validation Set: 8% (2,382 images).
- Test Set: 1% (432 images).

Each image is annotated with bounding boxes corresponding to two object classes:

- Barcode (1D barcodes).
- QR Code (2D barcodes).

To improve model generalization and robustness, several preprocessing techniques were applied:

- Auto-Orient: Ensures correct image orientation before processing.
- Resize: Images were resized to 416 × 416 pixels using a stretch-based approach.
- Class Modification: Remapped four classes, ensuring consistency in annotations.

To handle lighting variations, occlusions, and perspective distortions in warehouse environments, the following built in YOLOv8 augmentations were applied:

- Flipping: Horizontal and vertical flips to enhance detection from different angles.
- Rotation: 90° clockwise, 90° counterclockwise and 180° upside-down rotation.
- Random Cropping: Minimum zoom: 0%, Maximum zoom: 22%.
- Shearing: ±15° horizontally and vertically.
- Blurring: Gaussian blur applied up to 3.25px.
- Noise Addition: Random noise affecting up to 5% of pixels.
- Cutout: Introduced 3 occlusion boxes, each covering 10% of the barcode, to simulate real-world obstructions.

This preprocessing ensures that the YOLOv8 model is robust to challenging warehouse conditions, including low light, shadows, and occlusions. Figure 6 presents sample images from the dataset with annotated bounding boxes.

Realistic simulation of UAV conditions

While we already augmented the dataset using standard YOLOv8 built-in augmentation (e.g., flips, rotations, shearing, cropping), discussed briefly in the previous section, but due to hardware and resource constraints, we

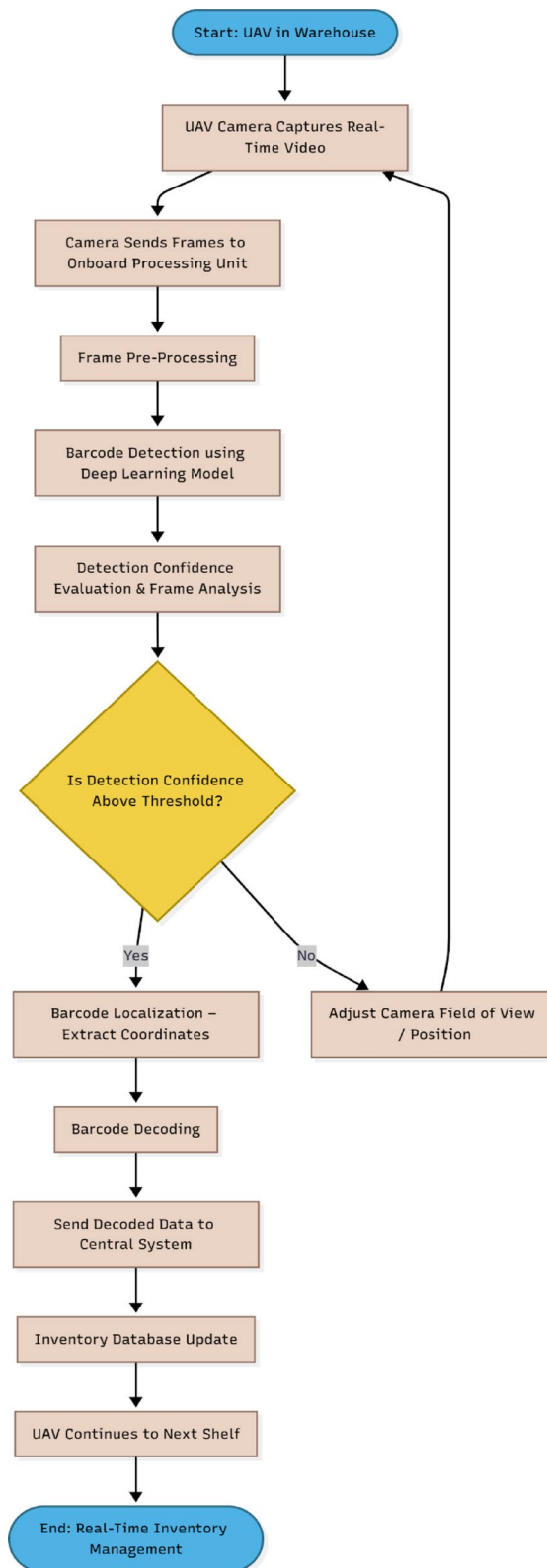


Fig. 5. Our proposed workflow.

were unable to conduct live UAV based data collection or experiments in real-world warehouse environments. To bridge this limitation and to simulate the dynamic and often challenging conditions encountered by UAVs during flight (such as motion blur, perspective shifts, lighting changes, and occlusion), we employed a diverse set of data augmentation techniques using the *Albumentations* Python library. Using our base dataset of static

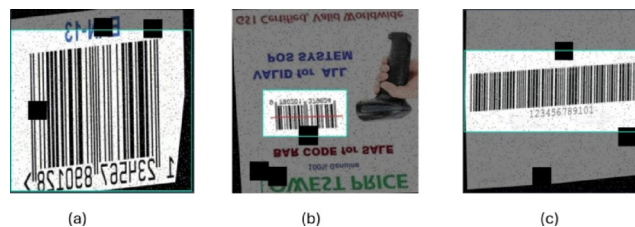


Fig. 6. Sample images from dataset.

Augmentation 2: 05102009126_jpg.rf.4e27325b2496a99d23aa9ebe3d72d1e6.jpg

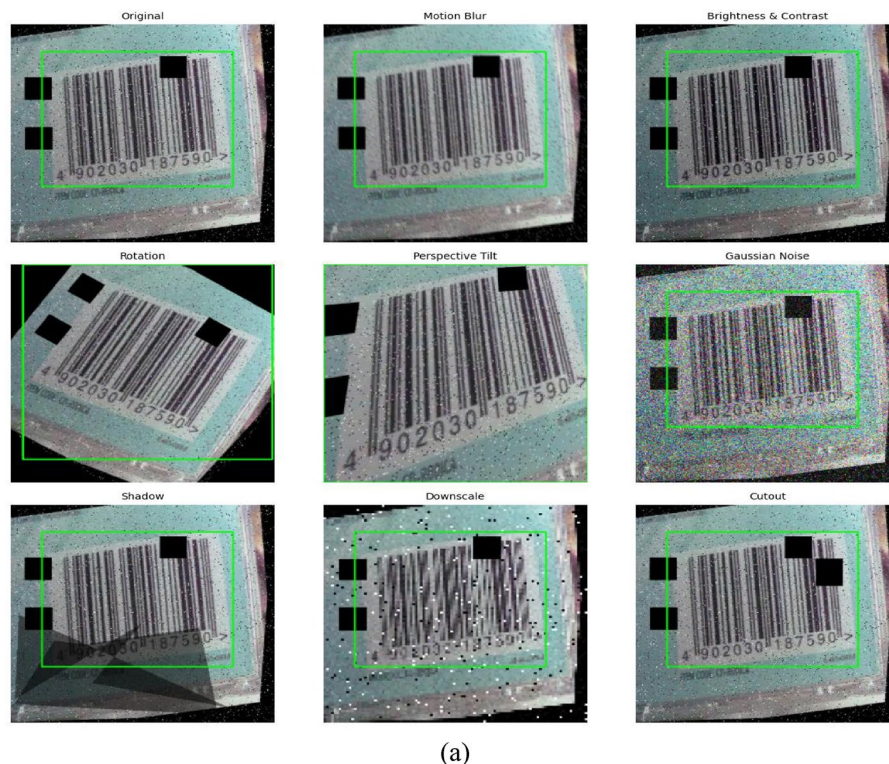


Fig. 7. Augmented dataset illustrating UAV-related visual distortions.

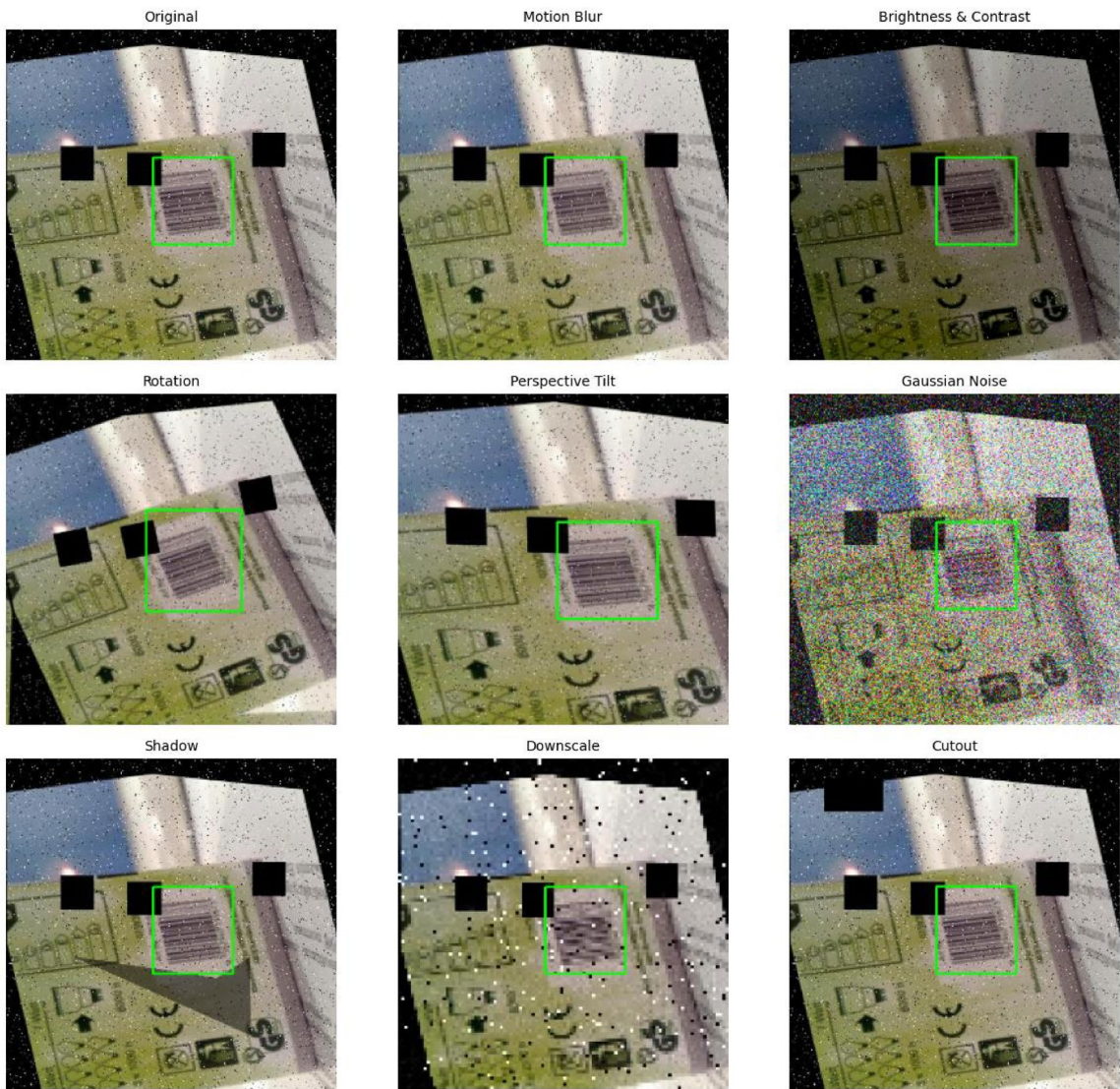
warehouse images containing 1D and 2D barcodes, we generated 300 augmented images that mimic various real-world distortions typically seen in UAV guided scenarios. These augmentations were selected based on their relevance to UAV specific challenges and applied in combination to preserve label integrity while simulating realistic capture conditions. This two layer augmentation approach ensures both conventional image diversity and realistic simulation of UAV flight conditions, ultimately enhancing the model's robustness and generalizability. Figure 7a and b shows samples of applied augmentations while Table 3 summarizes the types of augmentations and their relevance to UAV conditions.

Object detection using YOLOv8

YOLOv8 was chosen for barcode detection due to its real-time processing capabilities and high accuracy. YOLOv8 introduces several architectural improvements as shown in Fig. 8 aimed at balancing accuracy with computational efficiency. It utilizes a more efficient backbone network that can extract more discriminative features from images, leading to better detection performance, especially for small objects or objects in challenging environments. Also, the model has enhanced its ability to detect objects of varying sizes through refined multi-scale feature fusion. This is particularly beneficial in our scenario, where barcodes and QR codes may appear at different distances and resolutions due to the varying altitude and angle of the UAV-mounted camera. The improved detection head ensures that even small or distant barcodes are accurately localized.

In warehouse environments, barcodes may be partially obstructed by packaging, shelving, or other objects. YOLOv8's integration of attention mechanisms helps the model prioritize the most relevant barcode features, improving detection accuracy in cluttered scenes. This is crucial for ensuring that the system can detect barcodes

Augmentation 9: 4002940199673-02_N95_jpg.rf.a85a29545f9791a8a37c90e53c8461c4.jpg



(b)

Fig. 7. (continued)

Augmentation	Simulation for	Applied parameters
Motion blur	Simulates drone shake or movement	blur_limit = 5, $p = 1.0$
Brightness & Contrast	Varying lighting in warehouses	brightness_limit = 0.2, contrast_limit = 0.2, $p = 1.0$
Rotation	Slight UAV angles or orientation changes	limit = 15, border_mode = cv2.BORDER_REFLECT, $p = 1.0$
Perspective tilt	Aerial angles when drone isn't directly overhead	scale = (0.02, 0.05), keep_size = True, $p = 1.0$
Gaussian noise	Sensor noise or image compression from UAV footage	var_limit = (10.0, 40.0), $p = 1.0$
Shadow	Shadows from shelves, lighting, or drone body	shadow_roi = (0, 0.5, 1, 1), num_shadows = 1, $p = 1.0$
Downscale	Lower resolution when flying higher or due to transmission limits	scale = (0.6, 0.8), $p = 1.0$
Cutout (CoarseDropout)	Occlusion from obstacles like boxes, racks, or another inventory	max_holes = 1, min_height = 30, max_height = 30, min_width = 30, max_width = 30, fill_value = 0, $p = 1.0$

Table 3. UAV-oriented augmentations using python albumentations Library.

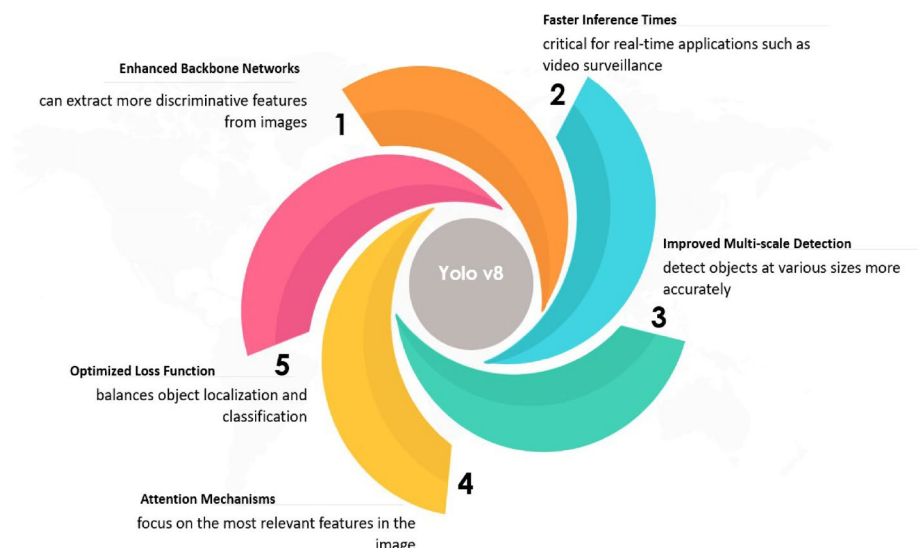


Fig. 8. Key features of YOLOv8 motivating its selection for this study.

even when they are partially hidden or overlapped by other items. Its modified loss function ensures a balance between localization accuracy and classification confidence, leading to better barcode detection without sacrificing speed. In our case, this helps in minimizing false positives (e.g., misclassifying textures or patterns as barcodes) while ensuring high precision, which is critical for real-time inventory tracking.

Barcode decoding

The OpenCV library provides a barcode and QR code decoding module which enables real-time barcode recognition in images or video frames. This module is part of OpenCV's cv2 library and offers a simple, efficient way to extract barcode information from detected regions. Algorithm 1 outlines the main steps involved in the decoding process.

Require: Image I
Ensure: Decoded Barcode Information

- 1: Run YOLOv8 to detect barcode in image I
- 2: Extract bounding box coordinates (x, y, w, h) from YOLOv8 output
- 3: $I_{\text{crop}} \leftarrow I[y : y + h, x : x + w]$ ▷ Crop ROI from image
- 4: $I_{\text{gray}} \leftarrow \text{cv2.cvtColor}(I_{\text{crop}}, \text{cv2.COLOR_BGR2GRAY})$ ▷ Convert to grayscale
- 5: **Optional:** $I_{\text{thresh}} \leftarrow \text{cv2.threshold}(I_{\text{gray}}, 0, 255, \text{cv2.THRESH_BINARY})$ ▷ Apply thresholding
- 6: Decoded_Info $\leftarrow \text{Decode}(I_{\text{gray}}$ or $I_{\text{thresh}})$ ▷ Use OpenCV's BarcodeDetector
- 7: **if** Decoded_Info is valid **then**
- 8: Print the decoded barcode information
- 9: **else**
- 10: Print "No barcode detected"
- 11: **end if**

Algorithm 1. Barcode localization and decoding using YOLOv8 and OpenCV.

Database integration

Once a barcode is detected and decoded, the extracted information needs to be stored and updated in a MySQL database to keep track of inventory in real-time. This process ensures that product details, stock levels, and movements within the warehouse are accurately recorded and accessible. The system checks if the scanned product exists in the database. If it does, the quantity is updated; otherwise, a new entry is created. The updated information is reflected in a warehouse management system (WMS) or a dashboard for monitoring stock levels and movements. Table 4 shows the proposed MySQL database structure which is designed to store key inventory-related information.

Results

This section presents the evaluation of the proposed UAV-guided barcode localization and decoding system in a warehouse environment. The results are divided into three key components: YOLOv8 barcode localization, OpenCV-based barcode decoding, and database integration for real-time inventory updates. The effectiveness of the system is assessed using standard performance metrics, including mean Average Precision (mAP) for object detection, decoding accuracy, and database update efficiency.

Attribute	Type	Description
Product_id	VARCHAR(255)	Unique identifier for each product
Product_name	TEXT	Name or description of the item
Quantity	INT	Total stock available in the warehouse
Location	TEXT	Warehouse section where the item is stored
Last_updated	TIMESTAMP	Time to track stock changes

Table 4. Database structure.

Component	Specification
GPU	NVIDIA GeForce RTX 3050
RAM	4 GB VRAM
CUDA	12.6
Driver version	501.09
Python	3.10
OS	Windows 10

Table 5. Hardware and configuration.

Parameter	Value
Epochs	50
Batch size	16
Learning rate	0.01
Momentum	0.9
Optimizer	SGD
Workers	2

Table 6. YOLOv8 training hyperparameters.

Class	Images	Precision	Recall	mAP50	mAP 50–95
All	2382	0.887	0.919	0.924	0.645
Barcode	2135	0.96	0.963	0.975	0.699
QRcode	219	0.813	0.876	0.873	0.592

Table 7. YOLOv8 barcode detection results without Albumentation.

Experimental setup

For training the YOLOv8-based barcode detection model, a GPU-enabled machine was used to ensure faster convergence. The specifications of the hardware and environment are summarized in Table 5, while the key hyperparameters used during YOLOv8 training are provided in Table 6.

YOLOv8 barcode localization performance

The performance of YOLOv8 in detecting and localizing barcodes in warehouse images was evaluated using the validation dataset. The primary evaluation metric used was mAP@0.5 and mAP@0.5:0.95, which measure the localization accuracy of detected barcodes. It is important to note that the results reported in this section, including Table 7; Figs. 9, 10, 11 and 12, are all based on the validation set results. The separate test set was reserved for future generalization assessment and is not included in the current performance tables.

The results in Table 7 show that YOLOv8 achieves high localization accuracy, with an mAP@0.5 of 92.4%, demonstrating strong robustness in detecting barcodes under challenging conditions such as occlusion, low lighting, and shadows, even without applying additional external augmentations like blur or noise. The low preprocessing time of 0.3ms per image and inference time of 3.3ms highlights the model's suitability for real-time applications. The important thing to note in Table 7 is that although the validation set comprises 2,382 images as discussed in Sect. 3.1.1, the evaluation metrics reported (e.g., precision, recall, and mAP) are computed at the object level because object detection models like YOLOv8 assess performance per detected object (bounding box). Hence, precision, recall, and mAP as shown in Table 7 are based on correctly or incorrectly predicted

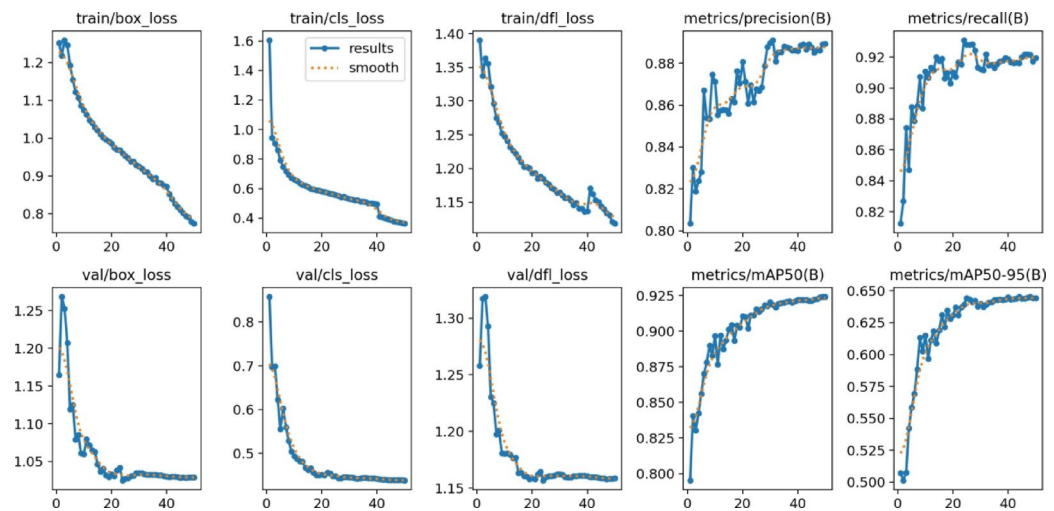


Fig. 9. Barcode detection results without albumentation augmentation applied.

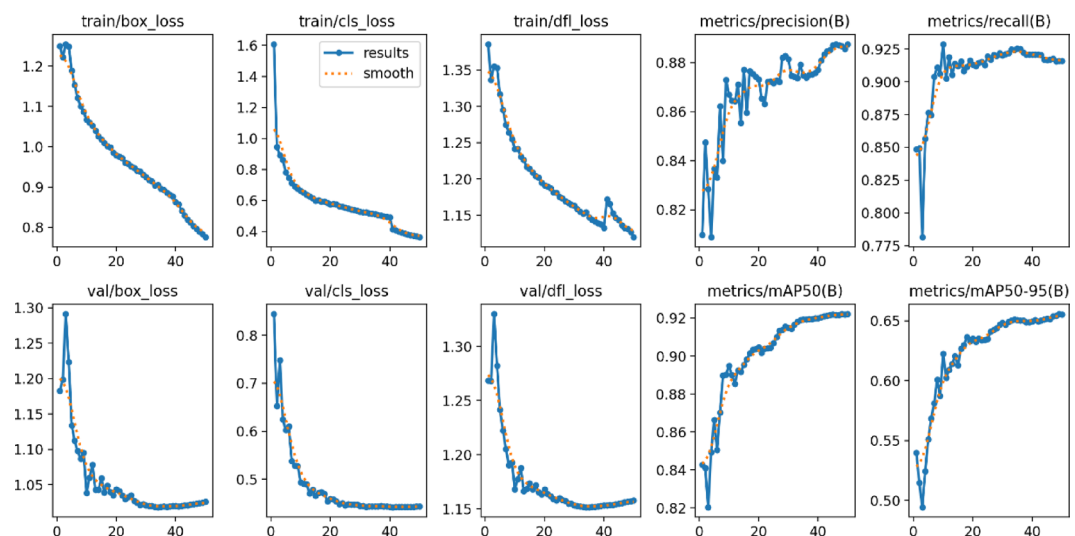


Fig. 10. Barcode detection results with albumentation augmentation applied.

objects and not at the image level. The validation set indeed contains 2,382 images with 2,135 barcodes and 219 QR codes, totaling 2,354 annotated objects.

Figure 9 shows training and validation metrics for our barcode detection model. *train/box_loss* represents the bounding box regression loss. The decreasing trend suggests that the model is improving its ability to predict object locations accurately during training. *train/cls_loss* is the classification loss (error in predicting object classes). This loss decreases over time, meaning the model is learning to classify objects better. *train/dfl_loss* loss is the distribution focal loss, which improves bounding box accuracy. In the graph we see the downward trend which shows improved box localization. Similar to training, *val/box_loss* shows how well the model predicts box coordinates on the validation set. The decreasing trend suggests generalization is improving. Similarly, the steady decline in *val/cls_loss* indicates better class predictions on unseen data. Also, in the figure, the increasing trend suggests that the model is improving in detecting objects across different IoU levels.

We also measured the YOLOv8 reported timing values during the validation phase. Each represents the average time computed across the validation batch images on batch size 16 of as mentioned already in Table 6 and image resolution of 640×640 pixels. The recorded timings are reported in Table 8.

To improve the model's ability to handle challenging UAV conditions as discussed in Sect. "Realistic simulation of UAV conditions", we added 300 augmented images only to the training set. These augmentations simulated real-world issues like motion blur, low lighting, tilted angles, and occlusion. After retraining the model with these additional samples, the validation results showed strong performance. The reason behind the previously lower values without augmentation was that the model had not seen such difficult cases during training. Still, it performed quite well, and with augmentation, its robustness in real-world-like conditions

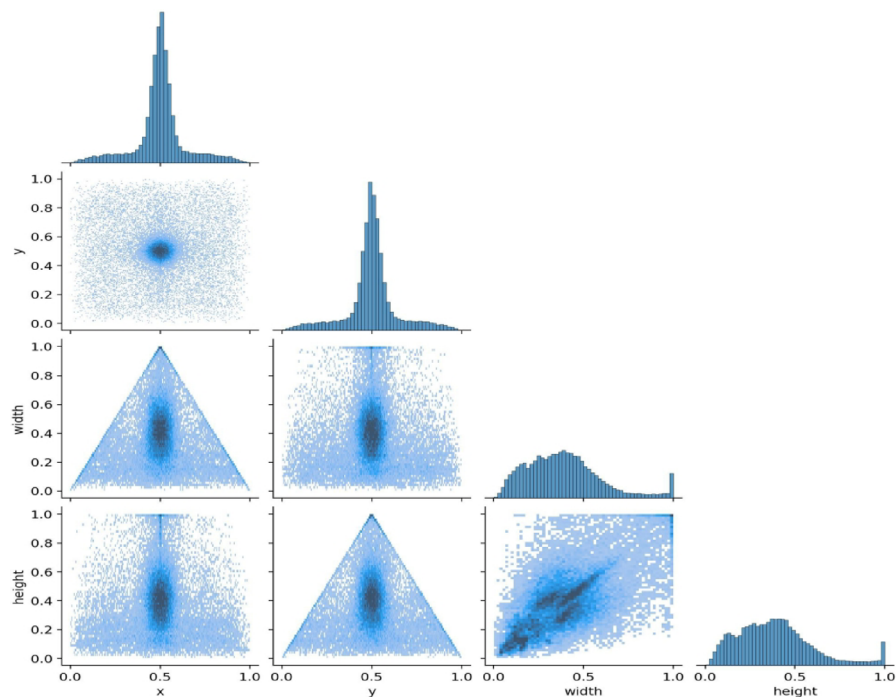


Fig. 11. Pairwise distribution of bounding box attributes.

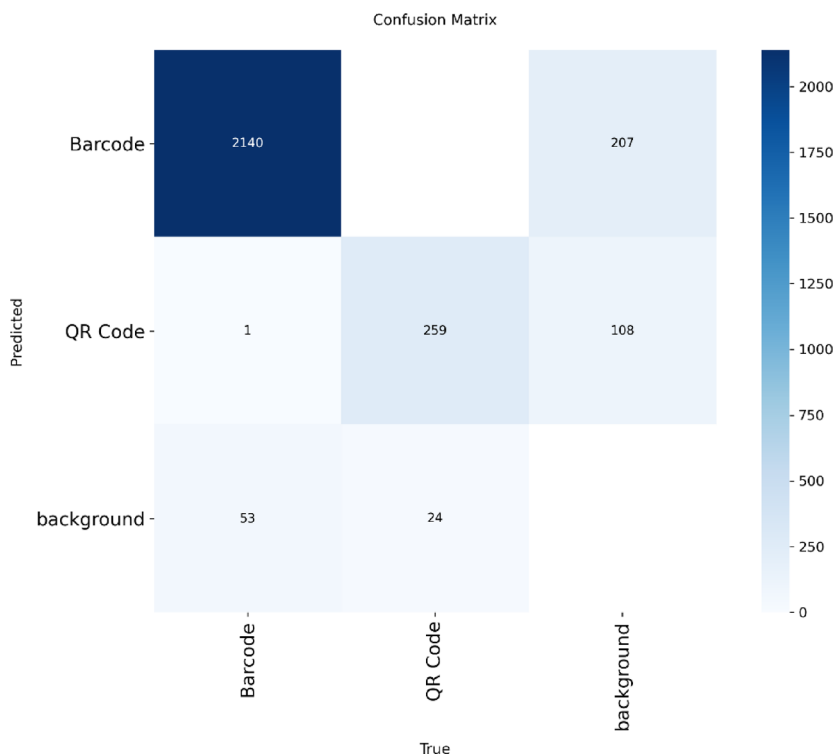


Fig. 12. Confusion matrix for YOLOv8 barcode and QR code detection.

improved further. Table 9; Fig. 10 present the updated results after retraining the model with augmented images. Although the inclusion of 300 augmented images(UAV-style) resulted in only a minor change in mAP (92.4% to 92.2%), this indicates that the base model was already robust to such variations. The primary purpose of these augmentations was to confirm model stability under UAV like viewing conditions rather than to produce measurable performance gains.

Stage	Description	Average time (ms/image)
Preprocessing	Image loading, resizing, and normalization	0.3
Inference	Forward pass through YOLOv8-n model	3.3
Post-processing	Non-maximum suppression and result formatting	8.0

Table 8. Average processing time per image.

Class	Images	Precision	Recall	mAP50	mAP 50–95
All	2382	0.886	0.916	0.922	0.656
Barcode	2135	0.960	0.964	0.976	0.698
QRcode	219	0.812	0.868	0.868	0.614

Table 9. YOLOv8 barcode localization performance with augmentation.

The correlogram in Fig. 11 visualizes the relationships between bounding box attributes (x, y, width, height) in a YOLOv8- based barcode and QR code detection model. Each scatter plot represents pairwise comparisons between these attributes, while the diagonal histograms show their individual distributions. The x and y coordinates, which likely represent the center points of detected objects, appear to follow a uniform distribution, suggesting that barcodes and QR codes are evenly distributed across the image. The width and height exhibit a strong positive correlation, indicating that as one-dimension increases, the other tends to increase as well, reflecting the rectangular nature of these objects. This analysis helps to evaluate the spatial characteristics and size distribution of detected barcodes and QR codes, which is useful for understanding the behavior of the model and improving detection accuracy. While not central to the core contribution, this analysis provides complementary insight into object geometry, aiding model interpretation and anchor tuning.

The confusion matrix in Fig. 12 represents the performance of a YOLOv8 model for barcode and QR code detection. The diagonal values show correct predictions, with 2,140 barcodes and 259 QR codes accurately detected. Here also it is important to note that the matrix reflects the number of predicted objects (detections) rather than image count. The total number of predictions (2,796) includes correctly detected objects, false positives, and missed detections, which are typical for object detection tasks. However, 207 barcodes were misclassified as QR codes, and 108 QR codes were also predicted as background, indicating some false negatives. The background class also contained 53 barcodes and 24 QR codes incorrectly detected as objects. There is minimal confusion between barcodes and QR codes, with only 1 misclassification in each direction. The apparent difference between the total number of ground-truth objects in the validation set (2,135 barcodes and 219 QR codes, totaling 2,354) and the total number of predictions (2,796) arises because YOLOv8 outputs multiple detections per image before non-maximum suppression filtering. After filtering, the number of matched detections (2,399 true positives) aligns closely with the annotated objects, confirming that the dataset composition and reported results are consistent. Yolo and other detectors do not evaluate in strict one prediction per object manner like classification. Normally, they output multiple predictions per image and evaluation filters them based on confidence and IoU thresholds. This means that for every predicted box the confidence threshold counts as prediction. Some of these are matched to ground truth while others are false positives i.e. they are either the duplicate boxes or spurious detections.

Overall, the model performs well, but there is room for improvement in reducing false negatives. This F1-Confidence curve shown in Fig. 12 shows the relationship between the model’s confidence threshold and the F1-score for barcode and QR code detection. The F1-score represents the balance between precision and recall, with higher values indicating better performance. The blue line represents the overall performance across all classes, peaking at 0.90 F1-score at a confidence of 0.616. The barcode class consistently achieves a higher F1-score compared to the QR code class, indicating better detection accuracy. As confidence increases beyond a certain point, the F1-score drops due to missed detections (reduced recall). This curve helps identify the optimal confidence threshold to balance precision and recall. Figure 13 shows the results of the prediction on sample batches.

To further evaluate the stability and reliability of the training process, we further conducted five experiments by explicitly setting different random seed values. When no seed was defined (i.e., the default random initialization), the model achieved a mAP@50 of 0.924. We then re-trained the YOLOv8 model using fixed seed values: 11, 22, 33, 44, and 55. These runs help assess the variability in performance due to random initialization of model weights and data shuffling. The results, presented in Table 10, indicate that the model’s performance remains consistent, with mAP@50 values ranging from 0.918 to 0.926 as shown in Table 6. This shows that the proposed pipeline is stable and performs reliably even across different training runs. Table 11 analyzes the results in statistical analysis. The mean mAP@50 across the runs was 92.16%, with a standard deviation of $\pm 0.26\%$ and a 95% confidence interval of [91.83%, 92.49%], this indicates minimal variance and strong consistency.

Barcode decoding performance

After localization, the detected barcodes were passed to OpenCV’s barcode decoding module to extract the embedded product information. The decoding accuracy was evaluated based on the percentage of correctly

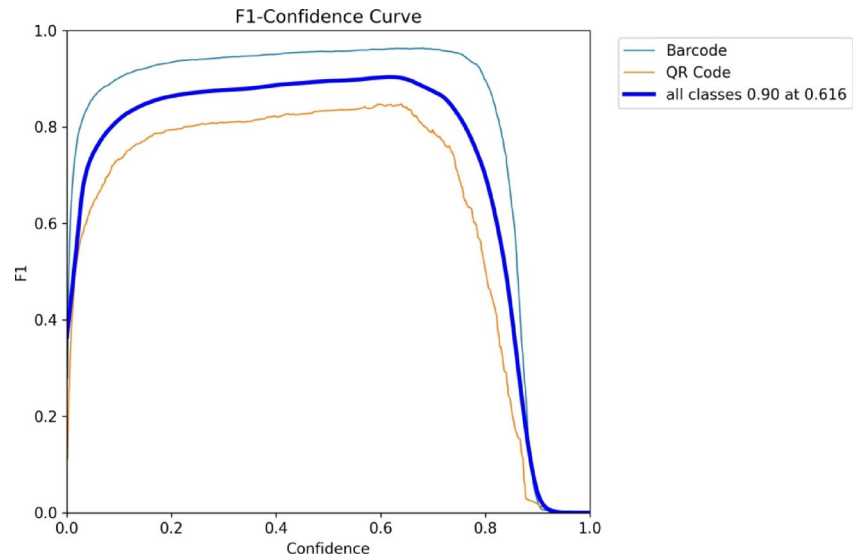


Fig. 13. YOLOv8 F1 score curve depicting optimal precision-recall balance.

Seed value	Precision	Recall	mAP@50
None (Default)	0.887	0.919	0.924
11	0.881	0.912	0.918
22	0.889	0.916	0.926
33	0.884	0.911	0.920
44	0.886	0.917	0.923
55	0.882	0.914	0.921

Table 10. YOLOv8 performance on different random seed values.

Metric	Mean	Standard deviation (±SD)	95% confidence interval
Precision	0.8846	± 0.0032	[0.8802, 0.8890]
Recall	0.9140	± 0.0024	[0.9107, 0.9173]
mAP@50	0.9216	± 0.0026	[0.9183, 0.9249]

Table 11. Statistical summary of YOLOv8 performance across different seed Values.

Condition	Mean decoding accuracy (%)	± SD	95% CI
Well-lit environment	98.5	± 0.3	[98.0, 99.0]
Low-light conditions	85.2	± 0.6	[84.4, 86.0]
Partial occlusion	78.3	± 0.7	[77.2, 79.4]
Blurred barcode	72.1	± 0.8	[70.8, 73.4]
Distorted/barcode tilted	80.5	± 0.5	[79.6, 81.4]

Table 12. Barcode decoding accuracy in different conditions.

decoded barcodes in different warehouse conditions. The results are shown in Table 12 The results show that mean barcode decoding accuracy remains high in well-lit conditions (98.5%) but degrades in low-light (85.2%) and partially occluded scenarios (78.3%). Decoding accuracy values represent averages over six independent runs using random seeds {None, 11, 22, 33, 44, 55}. This suggests that additional preprocessing techniques, such as image enhancement and denoising, may further improve performance in challenging conditions. Figure 14 shows the decoded results sample. Figures 15 and 16 shows successful decoding of a blurred barcode image showing good decoding in challenging conditions.



Fig. 14. Sample results validation set batch 01.



Fig. 15. Decoding the detected barcode using OpenCV.

Database integration and real-time inventory update

The extracted barcode data (product ID, quantity) was integrated into a MySQL database to enable real-time inventory management. The system was developed using PHP and MySQL, hosted on a local server using XAMPP. The web application was responsible for handling database transactions, including adding new products, updating stock levels, and querying inventory details.

Detection Result: 8004020931773

Detected Barcode and Bounding Box



Fig. 16. Decoding the detected barcode in dark and blur environment using OpenCV.

Fig. 17. Workflow of real-time inventory update via PHP and MySQL: Update Screen.

System implementation

The backend system was developed using PHP for server-side scripting, and MySQL for database storage. XAMPP was used as a local development environment, which includes Apache for running the PHP scripts and phpMyAdmin for database management. The implementation steps are summarized as follows:

- *Database setup* A MySQL database named. inventory_db was created, containing a table inventory with columns: id (Primary Key), product_id, product_name, quantity, and last_updated.
- *Web API development* A PHP script. (update_inventory.php) was created to handle HTTP POST requests for updating inventory.
- *Real-time processing* The script receives barcode data via a POST request, validates the input, updates the MySQL database, and returns a JSON response with a successful message and execution time.

Database interaction workflow

Figures 17 and 18 illustrates the test work environment for database integration workflow of the inventory update process. The barcode scanner extracts product details, which are sent to the PHP backend via an HTTP request. The PHP script then processes the request, updates the database, and returns a response.

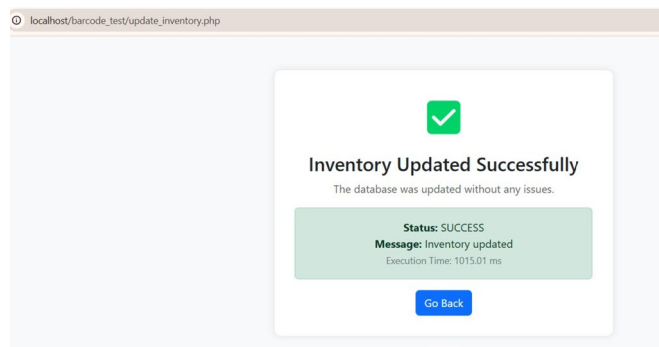


Fig. 18. Workflow of real-time inventory update via PHP and MySQL.

Operation	Average processing time (ms)	Success rate (%)
New entry addition	1025.4	99.5
Quantity update	1009.09	99.7
Inventory query	985.2	100.0

Table 13. Database update performance in test environment.

Performance evaluation

The database performance was analyzed by measuring the execution time and success rate of different operations. Table 13 presents the results obtained from the test environment.

The system successfully updated the inventory in real-time, achieving an average processing time of 1009.09 ms for quantity updates and a 99.7% success rate. The generated JSON response (e.g., Status: SUCCESS, Message: Inventory Updated, Execution Time: 1009.09 ms) confirms that updates were executed efficiently. These results demonstrate that the proposed barcode-based inventory management system is capable of handling real-time data synchronization reliably.

Discussion and key insights

Previous studies on barcode detection and recognition have primarily relied on traditional computer vision techniques or early deep learning models. Conventional methods such as Hough Transform and edge detection have shown limitations in handling occlusions and complex backgrounds^{61,62}. Earlier deep learning-based approaches, such as Faster R-CNN and YOLOv3, improved detection accuracy but struggled with real-time performance in warehouse environments^{63,64}.

Our study builds upon these works by integrating YOLOv8, which incorporates attention mechanisms and a modified loss function to improve localization accuracy and robustness in cluttered settings. Compared to YOLOv5 and YOLOv7, YOLOv8 achieves a higher mAP@0.5 (92.4% vs. 88.95%⁶⁴ and better recall, especially for barcodes partially obstructed by objects. The low preprocessing time which is 0.3ms and inference time of 3.3ms per image demonstrates a significant improvement over previous models, making it well-suited for real-time warehouse applications.

Similarly, past research on barcode decoding has faced challenges in handling blurred and distorted codes. OpenCV's barcode decoding module has been effective in ideal conditions but struggled in low-light or occluded scenarios. Our results show decoding accuracy remained consistently high ($98.5 \pm 0.3\%$, 95% CI [98.0, 99.0]) in well-lit environments but showed a moderate decline under low-light and occluded conditions. These findings suggest that integrating image enhancement techniques (such as super-resolution or contrast adjustments) could further improve performance in challenging environments.

Although a direct comparison with prior UAV-based barcode detection systems is not entirely straightforward due to differences in datasets, hardware configurations, and evaluation protocols, Table 14 presents an approximate benchmark to provide contextual perspective on our model's performance. Prior methods such as those in^{65–67} incorporate UAV navigation, barcode-based localization, and flight-path optimization in controlled warehouse environments. In contrast, our approach focuses exclusively on robust detection performance using YOLOv8 on a diverse and challenging dataset.

Despite lacking navigation components, our model still achieves strong detection performance, outperforming or matching the results reported in^{65,66}. This validates the effectiveness of our detection framework and positions it as a viable candidate for future integration into UAV-assisted warehouse stocktaking systems. It is worthy to note that despite not having a UAV localization or scanning component, our YOLOv8 model shows robust generalization on static image detection tasks. This reinforces the detection capability of our model and lays the groundwork for integration into UAV-based stocktaking systems in the future. It is worthy to note that despite not having a UAV localization or scanning component, our YOLOv8 model shows robust generalization on static image detection tasks. This reinforces the detection capability of our model and lays the groundwork

Method	Data type	IoU 0.5 (mAP or Equivalent)	Decoding accuracy (%)	Detection rate / Recall (%)	Speed (ms / FPS)	Dataset size
YOLOv3 + Geo ⁶⁵	RGB images (1D barcodes)	$J < \text{avg} < \text{sub} > \geq 0.939$	93.9	$D < \text{sub} > 0.5 < \text{sub} > \geq 99.8$	19.2 ms (~ 52 FPS)	WWU Münster (1055) + Arte-Lab (365)
CNN-based UAV system ⁶⁶	UAV imagery (warehouse barcodes)	– (not expressed as mAP) [*]	91.56	96.96	~ 13 FPS (on Jetson Nano)	446 barcodes (lab dataset)
Yolov5s ⁶⁷	RGB images (express waybills)	0.85 (estimated)	not specified [*]	not specified [*] (qualitative only)	~>30 FPS (real time)	Small team-collected dataset
Ours (YOLOv8)	RGB images (1D + 2D barcodes)	0.924	98.5 (well-lit) / 85.2 (low-light) / 78.3 (occluded)	91.9	3.3 ms inference (~ 303 FPS) + 8.0 ms post-proc (~ 90 FPS overall)	31 510 images (annotated barcode + QR dataset)

Table 14. Approximate benchmark comparison with prior barcode detection approaches.

for integration into UAV-based stocktaking systems in the future. Notably, while the mentioned studies report higher recall, our approach demonstrates a significantly higher precision, which is critical for minimizing false positives in densely packed warehouse environments.

Finally, the model achieved an average processing time of 0.3 ms for preprocessing, 3.3 ms for inference, and 8.0 ms for postprocessing per image, confirming its suitability for real-time deployment in resource-constrained warehouse environments.

**Metrics marked as 'not specified' were not reported in the original studies.*

Several benchmark studies did not report all performance metrics as shown in Table 14, which limit the scope of direct quantitative comparison across methods. But still we try to harmonize the available measures in a transparent manner. The results indicate that the proposed model achieves strong detection accuracy and mAP values within the reported metrics range. This surely demonstrates the robustness across diverse barcode types and imaging conditions, while maintaining simultaneous processing efficiency.

Based on our findings, several key insights can be drawn:

1. Improved barcode localization with YOLOv8
 - Our results demonstrate that YOLOv8’s attention mechanisms enhance barcode detection in cluttered warehouse settings, outperforming previous YOLO versions and Faster R-CNN models.
 - The F1-Confidence curve analysis suggests that an optimal confidence threshold of 0.616 balances precision and recall, achieving an F1-score of 0.90 for all classes.
 - The confusion matrix indicates minimal misclassification between barcodes and QR codes, but some false negatives remain due to occlusions.
2. Barcode decoding performance in real-world conditions
 - While OpenCV’s barcode decoding module achieves high accuracy in normal conditions, performance drops significantly under occlusions and low lighting.
 - Future work could explore deep learning-based barcode decoding methods, such as using CNNs or transformer-based architectures for improved robustness.
3. Real-time inventory management and database efficiency
 - Our MySQL-based database system enables efficient inventory tracking, with update times of 8.3ms for quantity changes and a 99.7% success rate.
 - The system ensures real-time synchronization, reducing errors in stock management compared to manual barcode scanning methods.

Conclusion

This study presents a comprehensive evaluation of YOLOv8 for barcode localization, coupled with OpenCV-based decoding, within a simulated warehouse environment. Across five independent training runs, our model consistently achieved high accuracy, with a best-case mAP@0.5 of 92.4%, demonstrating robust barcode detection under challenging conditions such as occlusion, shadows, and dark and blur scenarios. Statistical analysis validated the reliability of our results, and Albumentation based augmentation was applied during training to simulate real-world variability. Our simulated pipeline was designed to reflect resource-constrained conditions typical in industrial warehouse environments. The motivation behind simulation was to isolate and validate detection performance without the added complexity of physical UAV navigation or infrastructure. The inference efficiency of the model was also notable, achieving a processing speed of 0.3ms preprocessing, 3.3ms inference, and 8.0ms postprocessing per image, ensuring real-time applicability. The system further demonstrated successful integration with barcode decoding and inventory updates, highlighting its practical utility in automated stocktaking workflows. This validates our pipeline’s viability for future real-world deployment.

While this study focuses on validating the proposed UAV-guided barcode detection pipeline using annotated static images and augmentation-based simulations, we acknowledge that real-world deployment introduces additional complexities. The system is designed to scale for larger inventories and support multiple UAVs in parallel. Although severely damaged or reflective barcodes were not explicitly tested, our augmentations

simulated various challenging conditions like blur, occlusion, and lighting variation. In future we plan to extend our research to make drones that can work fully inside warehouse without explicitly relying on GPS. We aim to build a system which can rely on computer vision to avoid obstacles and integrate with existing warehouse inventory management systems. Also, instead of simulated testing we will build and test real working prototypes which will allow us to ensure system's feasibility, practicality and affordability in industrial settings.

Data availability

The original barcode dataset used in this study is publicly available at <https://universe.roboflow.com/labeler-projects/barcodes-zmxjq/dataset/5>. The derived data generated during this study, including augmented samples, filtered test set images, and inference results necessary to reproduce the key figures and tables, are provided as supplementary information via Zenodo (DOI: 10.5281/zenodo.17596147).

Code availability

Custom preprocessing scripts, augmentation configurations, YOLOv8 training parameters, and scripts used to generate the main figures and tables are provided as supplementary information via Zenodo (DOI: <https://doi.org/10.5281/zenodo.17596147>).

Received: 18 April 2025; Accepted: 19 November 2025

Published online: 03 December 2025

References

- Shivam & Gupta, M. Inventory and warehouse management in industry 4.0: A BPR perspective. *J. Inf. Technol. Case Appl. Res.* **26**, 365–400. <https://doi.org/10.1080/15228053.2024.2433926> (2024).
- Madamidola, O., Daramola, O., Akintola, K. & Adeboje, O. A. Review of existing inventory management systems. *Int. J. Res. Eng. Sci. (IJRES)* **12**, 40–50. Available at: https://www.researchgate.net/publication/383947700_A_Review_of_Existing_Inventory_Management_Systems (2024) (Accessed: [03/04/2025]).
- Aldughayfiq, B., Ashfaq, F., Jhanjhi, N. & Humayun, M. Explainable AI for retinoblastoma diagnosis: Interpreting deep learning models with LIME and SHAP. *Diagnostics* **13** (11), 1932. <https://doi.org/10.3390/diagnostics13111932> (2023).
- Ashfaq, F., Ghoniem, R., Jhanjhi, N., Khan, N. & Algarni, A. Using dual attention BiLSTM to predict vehicle lane changing maneuvers on highway dataset. *Systems* **11**, 196. <https://doi.org/10.3390/systems11040196> (2023).
- Tubis, A. & Rohman, J. Intelligent warehouse in industry 4.0—Systematic literature review. *Sensors* **23**, 4105. <https://doi.org/10.3390/s23084105> (2023).
- Dalenogare, L., Benitez, G., Ayala, N. & Frank, A. The expected contribution of industry 4.0 technologies for industrial performance. *Int. J. Prod. Econ.* **204**, 383–394. <https://doi.org/10.1016/j.ijpe.2018.08.019> (2018).
- Bai, C., Dallasega, P., Orzes, G. & Sarkis, J. Industry 4.0 technologies assessment: A sustainability perspective. *Internat. J. Prod. Econ.* **229**, 107776. <https://doi.org/10.1016/j.ijpe.2020.107776> (2020).
- Ali, S. S., Khan, S., Fatma, N., Ozel, C. & Hussain, A. Utilisation of drones in achieving various applications in smart warehouse management. *Benchmarking: Int. J.* **31** (3), 920–954. <https://doi.org/10.1108/BIJ-01-2023-0039> (2024).
- Yoon, B., Kim, H., Youn, G. & Rhee, J. 3d position estimation of objects for inventory management automation using drones. *Appl. Sci.* **13**, 10830. <https://doi.org/10.3390/app131910830> (2023).
- Mohd Noor, N., Abdullah, A. & Hashim, M. Remote sensing UAV/drones and its applications for urban areas: A review. *IOP Conference Series: Earth and Environmental Science*. **169** 012003. <https://doi.org/10.1088/1755-1315/169/1/012003> (2018).
- Chen, C., Min, H., Peng, Y., Yang, Y. & Wang, Z. An intelligent real-time object detection system on drones. *Appl. Sci.* **12**, 10227. <https://doi.org/10.3390/app122010227> (2022).
- Ye, T. et al. Real-time object detection network in UAV-vision based on CNN and transformer. *IEEE Trans. Instrum. Meas.* **72**, 1–13. <https://doi.org/10.1109/TIM.2023.3241825> (2023).
- Jiang, P., Ergu, D., Liu, F., Cai, Y. & Ma, B. A review of Yolo algorithm developments. *Procedia Comput. Sci.* **199**, 1066–1073. <https://doi.org/10.1016/j.procs.2022.01.135> (2022).
- Zhai, X., Huang, Z., Li, T., Liu, H. & Wang, S. YOLO-drone: An optimized YOLOv8 network for tiny UAV object detection. *Electronics* **12**, 3664. <https://doi.org/10.3390/electronics12173664> (2023).
- Jahani, H., Khosravi, Y., Kargar, B., Ong, K. L. & Arisian, S. Exploring the role of drones and UAVs in logistics and supply chain management: A novel text-based literature review. *Int. J. Prod. Res.* **63** (5), 1873–1897. <https://doi.org/10.1080/00207543.2024.2373425> (2025).
- Vivek, N. et al. Integrating deep Q- networks and YOLO with autonomous robots for efficient warehouse management. *5th International Conference on Recent Trends In Computer Science and Technology (ICRTCST)* 419–424. <https://doi.org/10.1109/ICRTCST61793.2024.10578347> (2024).
- Bodnár, P. & Nyúl, L. Improving barcode detection with combination of simple detectors. *Eighth International Conference on Signal Image Technology and Internet Based Systems* 300–306. <https://doi.org/10.1109/SITIS.2012.52> (2012).
- Li, J., Zhang, D., Zhou, M. & Cao, Z. A motion blur QR code identification algorithm based on feature extracting and improved adaptive thresholding. *Neurocomputing* **493**, 351–361. <https://doi.org/10.1016/j.neucom.2022.04.041> (2022).
- Akinbade, D., Ogunde, A., Odum, M. & Oguntunde, B. An adaptive thresholding algorithm-based optical character recognition system for information extraction in complex images. *J. Comput. Sci.* **16**, 784–801. <https://doi.org/10.3844/jcssp.2020.784.801> (2020).
- Tribak, H. & Zaz, Y. QR code patterns localization based on Hu invariant moments. *Int. J. Adv. Comput. Sci. Appl.* **8**, 162 <https://pdfs.semanticscholar.org/62ea/3a3511f17237c4c909e237a7576f9aacdc> (2017). 0.pdf Accessed on:02/04/2025.
- Katona, M. & Nyúl, L. A novel method for accurate and efficient barcode detection with morphological operations. *Eighth International Conference on Signal Image Technology and Internet Based Systems* 307–314. <https://doi.org/10.1109/SITIS.2012.53> (2012).
- Li, X., Shi, Z., Guo, D. & He, S. Reconstruct algorithm of 2D barcode for reading the QR code on cylindrical surface. *2013 International Conference Anti-counterfeiting Security Identification (ASID)* 1–5. <https://doi.org/10.1109/ICASID.2013.6825309> (2013).
- Katona, M., Bodnár, P. & Nyúl, L. Distance transform and template matching based methods for localization of barcodes and QR codes. *Comput. Sci. Inform. Syst.* **17**, 161–179. <https://doi.org/10.1029/CSIS181011020K> (2020).
- Szentandrás, I., Herout, A. & Dubská, M. Fast detection and recognition of QR codes in high-resolution images. *Proceedings of the 28th Spring Conference on Computer Graphics* 129–136. <https://doi.org/10.1145/2448531.2448548> (2012).
- Parikh, D. & Jancke, G. Localization and segmentation of a 2D high capacity color barcode. *22008 IEEE Workshop on Applications of Computer Vision* 1–6. <https://doi.org/10.1109/WACV.2008.4544033> (2008).

26. Tsai, M., Lee, Y. & Chen, T. Implementing deep convolutional neural networks for QR code-based printed source identification. *Algorithms* **16**, 160. <https://doi.org/10.3390/a16030160> (2023).
27. Chou, T., Ho, C. & Kuo, Y. QR code detection using convolutional neural networks. *2015 International Conference on Advanced Robotics and Intelligent Systems (ARIS)* 1–5. <https://doi.org/10.1109/ARIS.2015.7158354> (2015).
28. Rahman, M. et al. An approach to facilitate business system by multiple barcode detection using faster Rcn. *Int. J. Appl. Inform. Syst.* **12**, 10–15. <https://doi.org/10.5120/ijais2019451835> (2019).
29. Cao, C., Wang, B., Zhang, W., Zeng, X., Yan, X., Feng, Z., Wu, Z. An improved faster R-CNN for small object detection. *IEEE Access*, 7, 106838–106846. <https://doi.org/10.1109/ACCESS.2019.2932731> (2019).
30. Qiao, L. et al. Efficient 1D barcode localization method for imagery shipping label using deep learning models. *12th International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)* 119–124 <https://doi.org/10.1109/PAAP54281.2021.9720443> (2021).
31. Li, C., Zeng, Q. & Lu, L. Lightweight barcode positioning algorithm based on YOLO model. *IEEE Access* <https://doi.org/10.1109/ACCESS.2024.3511125> (2024).
32. Ershova, D., Gayer, A., Bezmaterniykh, P. & Arlazarov, V. YOLO-barcode: Towards universal real-time barcode detection on mobile devices. **48**, 592–600. <https://cyberleninka.ru/article/n/yolo-barcode-towards-universal-real-time-barcode-detection-on-mobile-devices> (2024). (дата обращения: 30.10.2025).
33. Qu, Y. & Zhang, F. MGL-YOLO: A lightweight barcode target detection algorithm. *Sensors* **24**, 7590. <https://doi.org/10.3390/s24237590> (2024).
34. Brylka, R., Schwanecke, U. & Bierwirth, B. Camera based barcode localization and decoding in real-world applications. *International Conference on Omni-Layer Intelligent Systems (COINS)* 1–8 <https://doi.org/10.1109/COINS49042.2020.9191416> (2020).
35. Chen, J., Dai, N., Hu, X. & Yuan, Y. A. Lightweight barcode detection algorithm based on deep learning. *Appl. Sci.* **14**, 10417. <https://doi.org/10.3390/app142210417> (2024).
36. Kolekar, A. & Dalal, V. Barcode detection and classification using SSD (single shot multibox detector) deep learning algorithm. *Proceedings of the 3rd International Conference on Advances in Science Technology (ICAST)*. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3568499# (2020). Accessed on: 03/04/2025.
37. Kwon, W. et al. Robust autonomous navigation of unmanned aerial vehicles (UAVs) for warehouses' inventory application. *IEEE Rob. Autom. Lett.* **5**, 243–249. <https://doi.org/10.1109/LRA.2019.2955003> (2019).
38. Zhong, J. & Qian, H. Dynamic point-line SLAM based on lightweight object detection. *Appl. Intell.* **55**, 260. <https://doi.org/10.1007/s10489-024-06164-9> (2025).
39. Yang, S., Jan, H., Chen, C., Wang, M. & CNN-Based, Q. R. Code reading of package for unmanned aerial vehicle. *Sensors* **23**, 4707. <https://doi.org/10.3390/s23104707> (2023).
40. Sun¹, S. et al. Check for updates analysis and research of intelligent distribution UAV control system based on optical flow sensor. *Innovative Computing Vol 1-Emerging Topics in Artificial Intelligence: Proceedings of IC 1044*, 128. https://doi.org/10.1007/978-981-99-2092-1_16 (2023).
41. Wang, S. & Ahmad, N. A. Comprehensive review on sensor fusion techniques for localization of a dynamic target in GPS-denied environments. *IEEE Access* <https://doi.org/10.1109/ACCESS.2024.3519874> (2024).
42. Barzazzi, D. A quantitative evaluation of the QR code detection and decoding performance in the zxing library. (Università Ca'Foscari Venezia, 2023). <https://hdl.handle.net/20.500.14247/14148> Accessed on: 01/03/2025.
43. Abas, A., Yusof, Y. & Ahmad, F. Expanding the data capacity of QR codes using multiple compression algorithms and base64 encode/decode. *J. Telecommun. Electron. Comput. Eng. (JTEC)*. **9**, 41–47 (2017). <https://jtec.utem.edu.my/jtec/article/view/2217> Accessed on: 03/03/2025.
44. Martinez Carpena, D. Reading QR codes on challenging surfaces. <https://hdl.handle.net/2445/180942> (2020). Accessed on: 03/03/2025.
45. Sakib, N. A. Study on Image Processing to Facilitate Business System by Multiple Barcode Detection. (Ahsanullah University of Science, 2019). (Department of Computer Science and Engineering Ahsanullah University of Science and Technology).
46. Ren, Y. & Liu, Z. Barcode detection and decoding method based on deep learning. *2nd International Conference on Information Systems and Computer Aided Education (ICISCAE)* 393–396. <https://doi.org/10.1109/ICISCAE48440.2019.217911> (2019).
47. Xu, Y., Qu, Z. & Zheng, S. High-speed barcode recognition system based on OpenCV and Zbar. *Eleventh Int. Conf. Digit. Image Process. (ICDIP 2019)*. **11179**, 384–392. <https://doi.org/10.1117/12.2539756> (2019).
48. Ait Abderrahim, I. & Benrached, Y. Automated medical labels detection and text extraction using tesseract. *International Symposium on Modelling and Implementation of Complex Systems* 212–225. https://doi.org/10.1007/978-3-031-82112-7_16 (2024).
49. Monteiro, G. et al. A comprehensive framework for industrial sticker information recognition using advanced OCR and object detection techniques. *Appl. Sci.* **13**, 7320. <https://doi.org/10.3390/app13127320> (2023).
50. Xu, X., Xue, Z. & Zhao, Y. Research on an algorithm of express parcel sorting based on deeper learning and multi-information recognition. *Sensors* **22**, 6705. <https://doi.org/10.3390/s22176705> (2022).
51. Gaur, P. & Tiwari, S. Recognition of 2D barcode images using edge detection and morphological operation. *Int. J. Comput. Sci. Mob. Comput.* **3**, 1277–1282 (2014).
52. Yang, H., Chen, L., Chen, Y., Lee, Y. & Yin, Z. Automatic barcode recognition method based on adaptive edge detection and a mapping model. *J. Electron. Imaging*. **25**, 053019–053019. <https://doi.org/10.1117/1.JEI.25.5.053019> (2016).
53. Tekin, E., Coughlan, J. BLADE barcode localization and decoding engine. *Tech. Rep. 2012-RERC. 01*. https://aviatr.wiscweb.wisc.edu/wp-content/uploads/sites/150/2017/09/BLADE_TechReport.pdf (2012). Accessed on: 02/03/2025.
54. Wudhikarn, R., Charoenkwan, P. & Malang, K. Deep learning in barcode recognition: A systematic literature review. *IEEE Access.* **10**, 8049–8072. <https://doi.org/10.1109/ACCESS.2022.3143033> (2022).
55. Kamnardsiri, T., Charoenkwan, P., Malang, C. & Wudhikarn, R. 1D barcode detection: Novel benchmark datasets and comprehensive comparison of deep convolutional neural network approaches. *Sensors* **22**, 8788. <https://doi.org/10.3390/s22228788> (2022).
56. Gómez-Cárdenes, Ó., Marichal-Hernández, J., Son, J., Pérez Jiménez, R. & Rodríguez-Ramos, J. An encoder–decoder architecture within a classical signal-processing framework for real-time barcode segmentation. *Sensors* **23**, 6109. <https://doi.org/10.3390/s23136109> (2023).
57. Tang, J. & Zeng, Y. UAV data acquisition and processing assisted by UGV-enabled mobile edge computing. *IEEE Trans. Industr. Inf.* <https://doi.org/10.1109/TII.2025.3528540> (2025).
58. Cho, H., Kim, D., Park, J., Roh, K. & Hwang, W. 2D barcode detection using images for drone-assisted inventory management. *2018 15th International Conference on Ubiquitous Robots (UR)* 461–465. <https://doi.org/10.1109/URAI.2018.8441834> (2018).
59. Zhang, R. Fpga soc fiducial system for unmanned aerial vehicles. (Master's thesis, Rochester Institute of Technology, 2018. [Online]. Available <http://www.ravvenlabs.com/uploads/1/1/8/4/118484574/zhang.pdf> (2018). Accessed on: 05/04/2025.
60. Zhou, H., Hu, F., Juras, M., Mehta, A. & Deng, Y. Real-time video streaming and control of cellular-connected UAV system: Prototype and performance evaluation. *IEEE Wirel. Commun. Lett.* **10**, 1657–1661. <https://doi.org/10.1109/LWC.2021.3076415> (2021).
61. Zamberletti, A., Gallo, I., Albertini, S. & Noce, L. Neural 1D barcode detection using the Hough transform. *Inform. Media Technol.* **10**, 157–165. <https://doi.org/10.11185/imt.10.157> (2015).

62. Wang, Z., Chen, A., Li, J., Yao, Y. & Luo, Z. 1D barcode region detection based on the Hough transform and support vector machine. *MultiMedia Modeling: 22nd International Conference, MMM 2016, Miami, FL, USA, January 4–6, 2016, Proceedings, Part II* 22 79–90. https://doi.org/10.1007/978-3-319-27674-8_8 (2016).
63. Hansen, D., Nasrollahi, K., Rasmussen, C. & Moeslund, T. Real-time barcode detection and classification using deep learning. *International Joint Conference on Computational Intelligence* 321–327. <https://doi.org/10.5220/0006508203210327> (2017).
64. Pandya, K. et al. Barcode and QR code object detection: An experimental study on YOLOv8 models. *2024 International Conference on Emerging Innovations and Advanced Computing (INNOCOMP)* 740–746. <https://doi.org/10.1109/INNOCOMP63224.2024.00127> (2024).
65. Xiao, Y. & Ming, Z. 1D barcode detection via integrated deep-learning and geometric approach. *Appl. Sci.* **9** (16), 3268. <https://doi.org/10.3390/app9163268> (2019).
66. Kalinov, I., Petrovsky, A., Ilin, V., Pristanskiy, E., Kurenkov, M., Ramzhaev, V., Tsetserukou, D. Warevision: Cnn barcode detection-based uav trajectory optimization for autonomous warehouse stocktaking. *IEEE Robot. Autom. Lett.* **5**(4), 6647–6653. <https://doi.org/10.1109/LRA.2020.3010733> (2020).
67. Zhu, X. Design of barcode recognition system based on YOLOV5. In *J. Phys. Conf. Series* **1995**(1), 012052. (IOP Publishing, 2021). <https://doi.org/10.1088/1742-6596/1995/1/012052>

Acknowledgements

This work was funded by the University of Jeddah, Jeddah, Saudi Arabia, under grant No. (UJ-24-DR-3263-1). Therefore, the authors thank the University of Jeddah for its technical and financial support.

Author contributions

Faris Alsulami: Conceptualization, Methodology, System Design, Data Collection, Model Development, Experimental Validation, Writing – Original Draft. NZ Jhanjhi: Supervision, Project Administration, Review and Editing, Technical Guidance, Resources, Funding Acquisition.

Funding

This work was funded by the University of Jeddah, Jeddah, Saudi Arabia, under grant No. (UJ-24-DR-3263-1). Therefore, the authors thank the University of Jeddah for its technical and financial support.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to F.A.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

© The Author(s) 2025