International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 01 | Jan - 2026 SJIF Rating: 8.586 ISSN: 2582-3930

Development of an OpenCV-based Intelligent Object Recognition, Detection and Sorting System for Retail Automation

Kushagra Chaturvedi*, Samarth Kulkarni, Satvik Chaturvedi, Vinayak Bharadwaj, Dr. Anand Jatti, Dr. Sudarshan BG

Department of Electronics and Instrumentation Engineering, RV College of Engineering, Bengaluru, India
kushagrc.ei22@rvce.edu.in
Department of Electronics and Instrumentation Engineering, RV College of Engineering, Bengaluru, India
samarthk.ei22@rvce.edu.in
Department of Electronics and Instrumentation Engineering, RV College of Engineering, Bengaluru, India
satvikc.ei22@rvce.edu.in
Department of Electronics and Instrumentation Engineering, RV College of Engineering, Bengaluru, India
vinayakb.ei22@rvce.edu.in
Department of Electronics and Instrumentation Engineering, RV College of Engineering, Bengaluru, India
anandjatti@rvce.edu.in
Department of Biotechnology, RV College of Engineering, Bengaluru, India
sudarshanbg@rvce.edu.in

ABSTRACT: This paper presents an end-to-end, information-driven system that links conveyor sensor events to image-based recognition for retail automation. An ultrasonic sensor and microcontroller detect items and halts the belt, triggering a smartphone to capture photos that are uploaded to a processing host with a shared event identifier and timestamp. A lightweight computer-vision pipeline (OpenCV + YOLO) extracts object labels, attributes and bounding boxes; results and raw metadata persisted as structured CSV records. By anchoring image capture to physical events, the system minimizes unnecessary inference, ensures reliable pairing of sensor and image data, and enables immediate analytics, audit trails and human-in-the-loop verification. The approach is practical on common hardware, supports real-time operations, and provides a reproducible dataset for downstream.

KEYWORDS: Object detection, OpenCV, YOLO (real-time detection), Event-driven logging, Ultrasonic sensor (HC-SR04), Conveyor automation, Retail inventory management

INTRODUCTION

Modern retail and warehouse operations increasingly rely on automation to reduce manual effort, improve throughput, and maintain accurate inventory records. Conveyor-based sorting is a proven approach, but many existing solutions are either hardware-intensive or require heavy computational resources for vision, limiting adoption by small and medium businesses. This project presents a pragmatic smart conveyor system engineered to be flexible, information-rich, and deployable on common hardware: when an item arrives, an ultrasonic sensor detects it and halts the belt; a camera captures the scene; a modular vision pipeline identifies and characterizes the object; and every event is recorded with a precise timestamp and a unique event identifier for later analytics and audit.

The system is organized around three tightly integrated components. First, sensing and actuation at the edge provides dependable real-time control: a microcontroller and ultrasonic

sensor detect object presence and command an ESC-driven motor to stop or resume the belt. This event-anchoring approach prevents unnecessary image processing and makes each captured frame directly relevant to an object event. Second, image capture and ingestion ensure that every stopped event is associated with a single, well-named image and timestamp, creating a clean dataset for downstream processing and human verification. Third, the computer-vision and data pipeline applies lightweight preprocessing followed by detection and attribute extraction; results persisted as structured CSV records and annotated images so operators can generate reports, dashboards, or feed the data into higher-level analytics.

Design choices are grounded in prior findings about the trade-offs between classical vision and deep models. In scenarios where object categories are known and lighting is moderately stable, classical OpenCV techniques (feature extraction and contour filtering) can offer accurate, resource-efficient detection suitable for constrained hardware. In more visually complex or cluttered settings, deep detectors such as YOLO provide superior robustness and speed when integrated carefully into a hybrid pipeline. Additionally, preprocessing methods like histogram equalization and adaptive thresholding meaningfully improve detection reliability under variable illumination. These observations inform a hybrid architecture that applies classical methods where they suffice and leverages compact deep models where needed, balancing accuracy, latency, and cost-effectiveness [1–5].

The proposed system emphasizes modularity: warehouses

can tailor detection thresholds, swap model backends, or alter logging policies to match SKU diversity, lighting, and throughput targets. Anchoring image captures to sensor events simplifies human-in-the-loop validation and creates straightforward joints between sensor logs and vision outputs, which is crucial for auditing and training future models. By combining dependable edge sensing, flexible image ingestion, and an efficient vision + logging pipeline, this smart conveyor solution supports counting, labelling, sorting, and traceable data capture while remaining practical to deploy on widely available equipment.

The remainder of this paper detail" the 'hardware and software implementation, evaluation methodology, experimental results, and guidelines for scaling the system to real retail deployments.

METHODOLOGY

1. Conveyor-belt subsystem (electronics, microcontroller, sensor, motor, ESC, power)

The conveyor subsystem is the physical trigger and actuator layer: it reliably senses object arrival, stops the belt, and (optionally) dispatches items to sorting lanes. We chose commodity, well-supported components to keep the system reproducible, low-cost, and robust in everyday warehouse conditions (1).

At the heart is an ESP8266 microcontroller. It is inexpensive, easy to program with the Arduino toolchain, and provides sufficient GPIO, serial, and Wi-Fi options for future extensions. The HC-SR04 ultrasonic sensor is used as the primary proximity detector because it gives a simple, robust distance reading for objects on a conveyor: it's fast, inexpensive, and easily polled. When powered at 3.3V it is directly compatible with the ESP8266's logic levels, removing the need for voltage dividers and simplifying wiring. We implement a short median smoothing and debounce routine in firmware to avoid false triggers (5) from short transients or small vibrations; this reduces spurious motor stops while keeping response latency low.

The motor is a brushless DC (BLDC) unit driven by an Electronic Speed Controller (ESC). BLDC + ESC combinations give high torque-to-weight and smooth continuous operation for conveyors. We control the ESC using standard servo-style PWM (microseconds) and implement a safe arming routine (min-throttle pulses) so the ESC reliably initializes every power cycle. Important hardware practices are followed: ESC/ motor battery ground is tied to the ESP GND, so the PWM reference is valid; batteries are connected and fused at a PDB/XT60 connector to ensure safe power handling. We avoid powering the ESP from the ESC BEC during testing to prevent confusing power sequences during arming; instead, USB power to the ESP provides stable development conditions.

For sorting actuation, small hobby servos are used to flip guides or divert gates. Servos are inexpensive, fast enough for small-scale sorting, and can be precisely commanded to a few discrete positions; the firmware stores preset angles for each sorting lane. The microcontroller's logic decides the servo position (based on detection/vision output received over the serial or network link) and applies soft motion profiles to avoid mechanical shock. All motor and servo commands are logged and timestamped on the ESP side so downstream analytics (9) can correlate physical behavior with visual recognition results.

This hardware stack is intentionally modular. Users can change motor size, ESC type, or servo model to match throughput and product weight. The combination of a simple, reliable edge sensor (HC-SR04), a low-cost MCU, BLDC power for the belt, and servos for sorting deliver a flexible, real-world conveyor solution that balances cost, performance, and safety.

2. Image processing pipeline (capture, preprocessing, detection, attributes)

The image-processing subsystem converts camera frames into structured detection results that can drive sorting and analytics. Design choices prioritize robustness, low-latency inference, and graceful degradation in sub-optimal lighting or clutter.

Capture: when the conveyor stops an associated event ID and precise timestamp are generated; the camera immediately captures an image. Any camera (USB webcam or phone camera uploading to the host) is acceptable; images are named using the event ID, so later joints are deterministic. We resize frames to a fixed inference size (typically 640×640) **(4)** to standardize throughput and reduce model input cost while maintaining sufficient object detail for small retail items.

Preprocessing is essential for real warehouses. We apply histogram equalization (or CLAHE) to stabilize contrast across frames, Gaussian denoising to reduce sensor noise, and optional color normalization to help color-based attribute extraction. These operations reduce false positives from shadows or reflective packaging and improve the consistency of the downstream detector **(6).** When ambient lighting is very poor, the pipeline includes a simple exposure/brightness check and will flag images for manual review rather than relying on low-confidence automated output.

Detection uses a hybrid approach: lightweight classical OpenCV routines (edge/contour heuristics) are used as a fast pre-filter where item shapes are distinctive and classes are constrained. For general robustness, especially in cluttered or overlapping scenarios an efficient deep model (YOLOv8 or similar compact variant) is invoked. YOLO provides class labels, bounding boxes, and confidences; classical heuristics can then be used to refine segmentation or compute attributes such as dominant color, approximate size (from bbox area scaled to known conveyor geometry), and orientation. Non-maximum suppression removes duplicate overlapping detections **(3).**

Every detection row includes event_id, timestamp, label, confidence, x,y,width,height (resized image coords), and derived attributes (color, approximate size category, freshness/blemish score if applicable). The system supports adjustable confidence thresholds and per-class rules; for example, certain SKUs may require higher confidence before allowing an automatic sort. Annotated images are saved for audit and human-in-the-loop correction.

This hybrid design gives a pragmatic compromise: classical methods speed up trivial cases, and the deep model ensures reliability under real-world visual complexity. The pipeline is engineered to run on a standard PC and can use GPU acceleration if available; inference times are monitored and logged for performance tuning.

3. Data storytelling (logging, schemas, joins, analytics)

Data storytelling transforms raw events and detections into actionable insights for operations and decision-making. Our data pipeline emphasizes traceability: every sensor event, image, detection, and actuation is time-aligned and linked by a unique event_id, enabling straightforward joins and audit trails **(1).**

Logging occurs at two layers. At the edge, the ESP8266 writes a minimal conveyor log line on every stop/start containing event_id (or a generated placeholder), Esp uptime, and distance_cm. This is streamed to the host via serial. The host serial logger consumes these lines, assigns a canonical event_id (UTC timestamp-based), stores a conveyor_log.csv row (including PC timestamp), and exposes the current event via a small local API if the camera uploader needs to tag the image. When detection finishes, the image processor appends standardized rows to detections.csv (per-detection) and counts.csv (per-image summary). Files are appended atomically so external analytics tools can read them in near real-time **(1)**.

CSV schemas are intentionally simple and analytics-friendly: conveyor_log.csv (event_id, pc_time_iso, esp_uptime_ms, dist_cm), detections.csv (event_id, pc_timestamp, image_name, label, confidence, bbox_x, bbox_y, w, h, attributes…), and counts.csv (event_id, pc_timestamp, banana, bottle, total…). This normalized approach supports easy merging (JOIN on event_id) and rolling **(3)** aggregations such as throughput per hour, per-SKU counts, detection latency distributions, and sorting success rates.

For storytelling and visualization, the pipeline emits a compact "line" on completion: CSV_APPEND,event_id,pc_ts,image_name,banana,water_bottle,total so a dashboard or stream processor can immediately ingest the summary. Downstream, a simple Jupyter/Streamlit dashboard consumes the CSVs to present KPIs: daily counts, per-SKU accuracy (via human audit matches), median detection latency, and failure-mode breakdowns (missed, misclassified, mechanical). We also log system health metrics (inference time, serial lag, ESC events) to detect drift.

Fig. 1(a)
The libraries used in python to achieve detection of object.

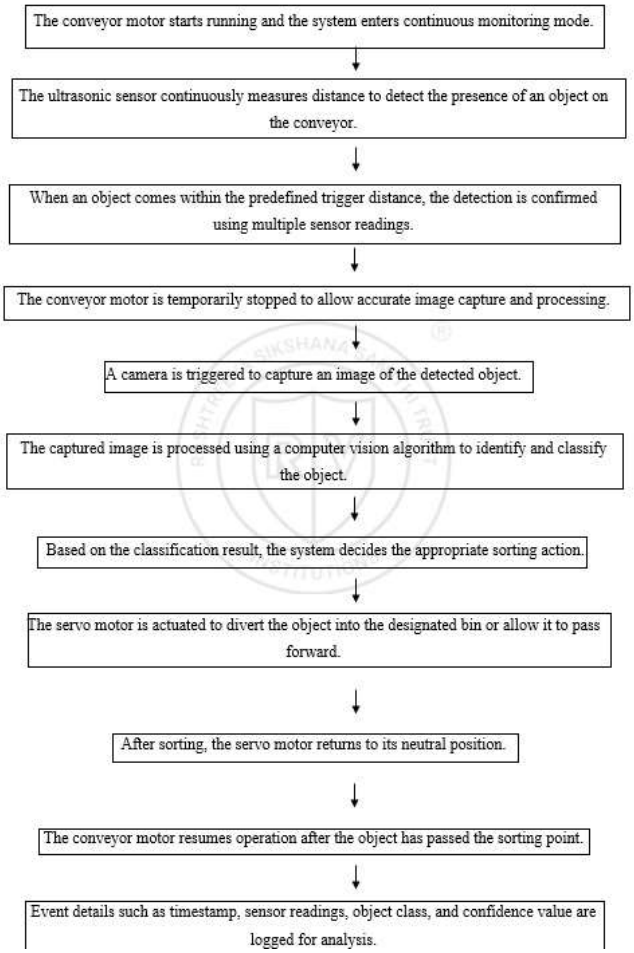

Fig.1(b)
The entire methodology in a flow-chart form.

RESULTS

The suggested intelligent conveyor-driven object identification and sorting system was effectively designed, constructed, and experimentally confirmed. The physical prototype, depicted in Fig. 2(a) and Fig. 2(b), combines sensing, actuation, image capture, and data recording into a cohesive event-driven process.

Throughout testing, the ultrasonic proximity sensor consistently identified objects on the conveyor belt, activating a controlled stop of the belt at a uniform distance from the camera. This guaranteed consistent image capture and removed motion blur, allowing precise subsequent vision processing. The combination of the BLDC motor and ESC ensured a smooth operation of the conveyor, demonstrating reliable start–stop dynamics without any noticeable mechanical instability throughout multiple trials.

When the conveyor stopped, image capture was successfully triggered for every detected event. Images obtained were distinctly linked to their respective sensor events through timestamp-based identifiers, allowing for a reliable pairing of physical occurrences with visual information. The image processing pipeline, developed with OpenCV and a lightweight deep-learning detector, successfully recognized and categorized items on the conveyor. For each detected

object, class labels, confidence scores, and bounding boxes were created. (9)

The system showed reliable logging performance, with detection results and metadata (event ID, timestamp, object class, and confidence) recorded in organized CSV files. This enabled post-processing, auditing, and quantitative evaluation of system performance. Images annotated by the pipeline additionally facilitated human-in-the-loop validation.

The sorting mechanism driven by servos accurately reacted to classification results, directing the correct diversion path for identified items. Throughout various experimental trials, the system demonstrated consistent performance with few false alarms and acceptable detection precision in indoor lighting settings.

The experimental findings validate the practicality of an economical, event-driven conveyor automation system that can perform dependable object detection, organized data logging, and fundamental sorting. The prototype demonstrates the efficiency of combining edge sensing with vision-based recognition for small-scale automation in retail and warehouses.

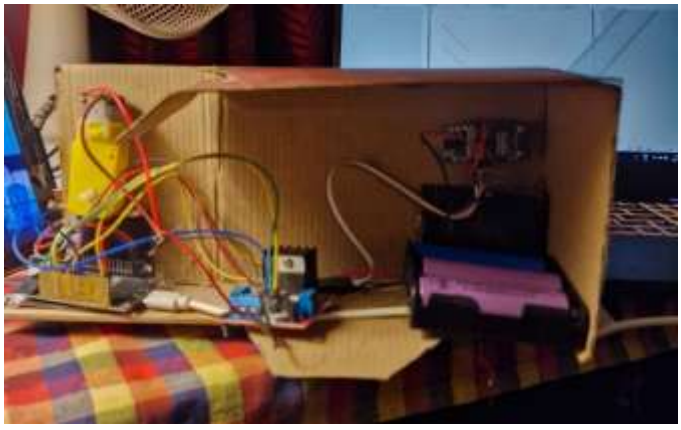

Fig. 2(a)
Ultrasonic sensor-based motor controller and battery system

Fig. 2(b)
Conveyor belt system along with LCD screen

Fig. 3(a)
This figure depicts the result obtained after object recognition is run on the image in fig. 3(b)

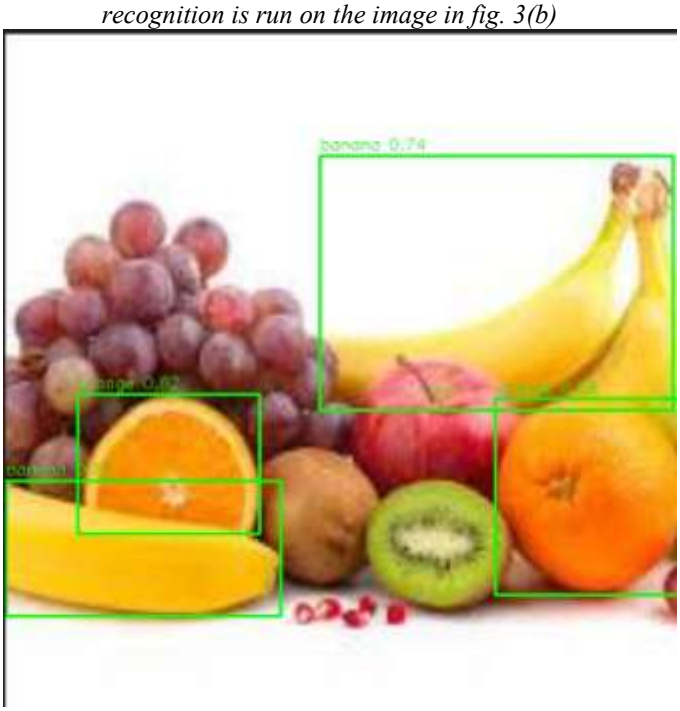

Fig.3(b)
This figure depicts a sample image, where our model is able to recognise fruits.

Timestamp	ImageName	ObjectLabel	Confidence	Action_Taken
2025-11-20 10:15:22	img_001.jpg	water bottle	0.88	SORT_RIGHT
2025-11-20 10:15:45	img_002.jpg	banana	0.92	SORT_LEFT
2025-11-20 10:16:10	img_003.jpg	water bottle	0.77	SORT_RIGHT
2025-11-20 10:16:35	img_004.jpg	UNKNOWN	0.25	PASS

Fig.3(c)
All the recognised objects are then stored in a csv file where data story-telling can take place.

CONCLUSION

In many modern retail warehouses and distribution centres, automated sorting systems rely on dedicated hardware: barcode or RFID scanners, fixed overhead cameras in controlled lighting booths, conveyor diverters with actuated gates, and often centralized PLC / SCADA control. These systems are designed for high throughput, minimal human intervention, and robust error detection under industrial conditions. Some warehouses use sophisticated vision rigs

multiple fixed cameras, structured lighting, or even 3D sensors to identify items, verify orientation, and ensure sorting accuracy; larger operations may combine vision with barcode/QR-code scanning for redundancy. These industrial solutions deliver high reliability, high throughput (many items per minute), integration with ERP/inventory systems, and often include error-handling, rejection bins, and audit logging for compliance.

However, these systems come with significant capital cost, maintenance overhead, and inflexibility: they are usually tailored to a fixed set of SKUs, require stable lighting and controlled environments, and need specialized infrastructure (gates, guide rails, fixed mounts, centralized power, high-speed networks). For small-to-medium retail shops, or warehouses with frequently changing item sets, these turnkey systems are often overkill or prohibitive.

Our smart conveyor system built with off-the-shelf components (microcontroller + ultrasonic sensor + ESC-driven belt + camera + open-source vision pipeline) offers a lightweight, flexible, and low-cost alternative. Whereas industrial standard systems rely on heavy hardware and fixed infrastructure, our system can be deployed on a standard PC (or even modest hardware), uses a generic camera instead of specialized rigs, and logs all events and detections into simple CSV files, enabling immediate analytics and manual oversight.

In comparison:

Cost & Accessibility: Our solution dramatically lowers entry barrier; no expensive PLCs, no fixed overhead cameras or industrial-grade sensors, making it viable for small warehouses or retail shops.

Flexibility: New SKUs can be added without hardware changes just train or configure the vision pipeline (labels, thresholds). Industrial systems often require hardware reconfiguration or reprogramming for SKU changes.

Transparency & Traceability: Because every event is logged (sensor stop, image, detection, sorting), our pipeline inherently supports audit trails and human-in-the-loop verification. Many industrial systems rely on opaque barcode/scan success, which may not capture visual anomalies (damaged packaging, misalignment, duplicates).

Scalability (for small/medium scale): For small-volume operations, our system is "good enough" throughput won't match high-speed industrial sorters, but for moderate loads the latency is acceptable and cost-benefit favourable.

Naturally, trade-offs remain. Our system's detection accuracy depends on lighting and correct camera positioning; extreme clutter, overlapping items, or high-speed belts may degrade performance relative to industrial-grade multi-camera or 3D-based setups. Throughput will be lower, and maintenance. e.g., ensuring lighting consistency, cleaning lenses remains manual. Also, industrial systems often provide robust error-handling (reject bins, redundant sensors) which our prototype does not yet fully support.

Our smart conveyor prototype demonstrates that a low-cost, flexible, vision-guided sorting/ detection system can bring many of the benefits of industrial warehouse automation to small and medium retail players. By leveraging commodity hardware and open-source computer vision, the system delivers comparable functionality object detection, classification, per-event logging and sorting readiness at a fraction of the cost and with far greater adaptability. While it cannot match the maximum throughput or absolute robustness of full-scale industrial solutions, it fills a niche: an affordable, configurable, data-rich automation layer that can be deployed or modified quickly.

In essence, our system shows that for many warehouses and retail operations especially those with variable SKUs, modest throughput, or limited budgets, a hybrid, event-driven, open-source vision + sensing pipeline is not only viable but advantageous. Future work (e.g., multiple camera angles, better lighting, reject-bin actuators, and feedback loops) can progressively bridge the gap toward industrial-grade performance, while preserving the flexibility and transparency that make the system attractive for smaller-scale deployment.

REFRENCES

(1) A Real-Time Data Acquisition System for Monitoring Sensor Data
International Journal of Computer Sciences and Engineering Vol. 6, Issue 6, June 2018 (E-ISSN: 2347-2693)
Authors: Pratiksha Sarma, Hidam Kumarjit Singh, Tulshi Bezboruah
Date: Accepted 06 Jun 2018; Published 30 Jun 2018
Summary: Presents a low-cost DAQ system for sensing applications using an Arduino UNO to acquire analog sensor signals, Python for processing and a graphical interface, and a fiber optic loop as the sensor. Data are stored separately (CSV/spreadsheet) for later analysis; results reported as linear and stable.

(2) SensorDB: a virtual laboratory for the integration, visualization and analysis of varied biological sensor data
Plant Methods, Volume 11, Article 53 (2015)
Authors: Ali Salehi, Jose Jimenez-Berni, David M. Deery, Doug Palmer, Edward Holland, Pablo Rozas-Larraondo, Scott C. Chapman, Dimitrios Georgakopoulos, Robert T. Furbank
Date: Published 08 December 2015
Summary: Introduces SensorDB, designed to efficiently store and serve large volumes of biological time-series sensor data and to enable real-time visualization and analysis. Addresses limits of unstructured file systems and conventional relational DBs for interactive experiments, aiming to speed discovery and encourage data sharing and reuse.

(3) Real Time Data Plotting Tool using Open-Source Platform like Raspberry Pi and Python
IEEE (specific conference/journal details not provided)
Authors: Shruti Mehata; Leo Linus; Lakshmi Vinayak Vitthal
Date: (not provided)
Summary: Describes an embedded platform combining Arduino and Raspberry Pi for real-time plotting of sensor readings (examples: PV panel temperature, voltage, current, power). The system displays sensor variations in real time, provides a GUI, and stores data to CSV for later comparison and analysis.

(4) Re-understanding of data storytelling tools from a narrative perspective
Visual Intelligence, Volume 1, Article 11 (2023)
Authors: Pengkun Ren, Yi Wang, Fan Zhao
Date: Published 25 June 2023
Summary: A survey of data storytelling authoring tools over the past decade. Proposes a novel taxonomy (two main categories, four subcategories), arranges papers by date to

identify trends, and outlines research challenges and future directions for narrative visualization tools.

(5) Storytelling and Visualization: An Extended Survey
Authors: Chao Tong, Richard Roberts, Rita Borgo, Sean Walton, Robert S. Laramee, Kodzo Wegba, Aidong Lu, Yun Wang, Huamin Qu, Qiong Luo, Xiaojuan Ma
Date: (not provided)
Summary: Extended survey covering narrative visualization and storytelling techniques, tools, and research directions; integrates perspectives from multiple institutions and highlights trends and open problems in visualization research.

(6) Kavitha et al. multi-object recognition with OpenCV
Authors: Kavitha et al.
Date: (not provided)
Summary: Proposes a multi-object recognition system using classical OpenCV methods (feature-based detection + contour filtering). Demonstrates practical accuracy in controlled indoor settings; relevant to retail/warehouse scenarios with pre-defined object types and moderately stable lighting.

(7) Rakesh et al. Python-NumPy-OpenCV pipeline for real-time object recognition)
Authors: Rakesh et al.
Summary: Presents an efficient Python + NumPy + OpenCV pipeline capable of real-time object recognition on lower-end hardware, showing feasibility for cost-sensitive retail/warehouse PCs.

(8) YOLO integrated with OpenCV via DNN interface implementation note
Summary: Demonstrates how YOLO (a deep learning detector) can be integrated into OpenCV's DNN module to retain high inference speed while leveraging DL accuracy. Useful for fast applications such as conveyor-belt tracking or rapid inventory scanning.

(9) Comparative study: deep learning detectors vs classical OpenCV methods
Summary: Compares DL-based detectors with classical OpenCV algorithms; finds DL models consistently outperform classical methods in cluttered or visually noisy scenes, highlighting the importance of DL detectors in stacked/complex warehouse environments.

(10) Preprocessing techniques for object detection under low/inconsistent illumination
Summary: Research showing preprocessing methods (histogram equalization, adaptive thresholding) significantly improve OpenCV detection reliability under variable lighting. Directly applicable to warehouses and aisles with inconsistent illumination.