

A High-Throughput Real-Time Object Counting Framework for Industrial Conveyors: Asynchronous Pipelining and Spatiotemporal Consistency Strategy

Hieu Trung Nguyen, Thuan Huy Chau, and Vinh Dinh Nguyen(✉)

Department of Information Technology, FPT University, Can Tho Campus, Can Tho, Vietnam
{hieunt239,vinhnd29}@fe.edu.vn, thuanchce170133@fpt.edu.vn

Abstract. The automated counting of items on high-speed conveyor belts remains one of the critical activities in the logistics of Industry 4.0. Still, the implementation of automated systems in the Industrial IoT is greatly limited by several factors in the external environment of the systems such as dust, lighting, noise, and the significant delay caused by the traditional linear systems' sequential processing problem. This paper addresses the all-in-one problem of processing industrial object counting of all processing of data within the framework of high industrial throughput. We developed an Asynchronous Parallel Processing Architecture, wherein image capturing is separated from inference to fully utilize the processing powers of the GPU. The main detection engine is YOLOv11, equipped with the Spatiotemporal Consistency Framework, which is an object filtering post-processing unit that quickly eliminates temporary false positives by analyzing object death and trajectory without additional computationally heavy Re-ID features to track objects. We then defined an all-encompassing evaluation container of the proposed system that consists of Counting Accuracy (ACA), Mean Absolute Error (MAE), and RMSE. The test results of the framework achieved exceptional consistency with an average of 95.9% for the ACA. Furthermore, the proposed Asynchronous system demonstrated practical real-world throughput of 256 frames per second on the Tesla T4 GPU (Stress Test) and Edge-X RTX 2050 Laptop.

Keywords: Object Counting · YOLOv11 · Real-time Image Processing · Spatiotemporal Filtering · Industrial Automation · Ablation Study

1 Introduction

Product quantity tracking using conveyor belt technology is a problem whose accuracy is crucial for accurate inventory management and control of quality in production and distribution systems as they exist today [1, 7]. Manual tracking on conveyor belts is time consuming, laborious, and may cause human error [14]. Photoelectric sensors are efficient in general, given the simpler configuration of the load, but can only collect data partially due to colliding or overlapping objects which is common in cement bag

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027
O. Gervasi et al. (Eds.): ICCSA 2026, LNCS 16769, pp. 88–99, 2027.
https://doi.org/10.1007/978-3-032-30494-0_6

conveyors. Therefore, deep learning based object detection algorithms (i.e., YOLO family) have been the most popular alternatives for this problem because of their capability to detect objects rapidly with their single-stage architecture [2]. Despite the advantage of the YOLO family over conventional detectors thanks to speed itself, the use of these detectors in a factory environment introduces two significant persistent problems within the deployment stage, related to their detector: (1) Environmental noise effect on robustness: Factory cameras may be affected by motion blur, dust, etc., sudden illumination changes when obtaining photographs resulting in false positives in the detector, when nothing should be detected [9]. (2) Limitations imposed by edge device resource constraints: Typically the edge devices used within factories have limited computing resources. Complex models increase computational load and cause delays and result in dropped frames and lost counts in fast moving conveyor belts [15]. Therefore, in order to solve the problem over the system level, rather than solving the aforementioned problem through designing a new detector for it; we will solve both above issues. After deciding the number of model options used for our research, the YOLOv11n model [13] has a small size, and we designed an asynchronous pipeline around it to avoid I/O bottlenecking. Moreover, we applied a space-time consistency filter for checking all the detections against all existing frames to identify if an object persists in more frames and moves reasonably. It will not be allowed into the final count to any extent if it fails to match those standards, so it would lose selection.

2 Related Work

Consequently, beyond the previously mentioned counting methods that can be utilized at industrial scale three specific systems were used to address the problem.

Sensors with physical techniques, low-cost photoelectric or laser sensors are accurate but not when items are close to one another. This is a typical issue which you will see in high-speed production line, for example, when two bags of cement come into contact with one another their individual triggers will result in a single signal.

Tracking based counting, the initial generation of AI detectors to get started uses double data with both YOLOv5 and DeepSORT systems to track on the ground. DeepSORT was developed to employ extraction of appearance feature vectors (re-identification) for tracking. While this served well in many cases, it was ineffective at monitoring thousands of identical-looking packages (i.e. the Re-ID feature vector consumes high computational resources). For example, in Tariq [16], it was shown that if the training dataset was augmented by additional images generated synthetically from real factory floor data, YOLOv5 and YOLOv7 could be used to count industrial components; in contrast, Kumar et al. [17] enhanced the speed of counting with YOLOv8 but accuracy declined during the run as factory lighting changed part way through the run for this reason. We addressed this by running HSV jitter during the training process. Tracking-wise, [18] noted that it was reasonable to simply use positional overlap and not Re-ID feature vectors for association among the objects in a variety of frames of a fast moving conveyor belt and that it is this fact that motivated our use of ByteTrack. Furthermore, Jiang et al. [19] compared different YOLO versions and found that YOLOv11 was more efficient per-parameter than YOLOv8, which played an important role in our choice

in choosing YOLOv11 for the edge applications. Additionally, Liu et al. [20] reported YOLO-based detectors are non-reliable when operating in high vibration, dusty environments unless some form of post-processing is applied to them—a similar issue that the Spatiotemporal Consistency Filter we designed was to mitigate.

Therefore, this research introduces a solution that could be addressed using ByteTrack [5], which is a detector which does not use features of the object in view. It uses IoU (intersection over union) instead to associate objects - which significantly reduces the time needed to do the problem. Coupled with this method, a simpler and efficient solution to the noise characteristics of industrial feeds as compared with previous approaches is provided by Spatiotemporal Filter [10].

3 Methodology

3.1 System Architecture

The system has two separate subsystems that work independently: Frame Capture and Processing as shown in Fig. 1.

The Thread Acquire reads an image from the Camera Stream and loads it to Shared Buffer. Since we want to minimize Latency as much as possible, the FFmpeg Backend was setup with buffer_size = 0 (no buffering) and fflags = nobuffer (this prevents build up of frames on the backend so the decoder discards old frames). Therefore, only the latest frames are saved for additional usage. All frames are stored in a thread-safe FIFO Buffer (shared memory).

A second thread is in charge of processing. This thread gets the latest frames from the queue and performs all operations of the full Inference Pipeline. A Mutex will be used to provide a lock on the shared buffer in order to avoid contention between threads. Stress testing showed that without Synchronization, occasionally there were corrupted frames when running at 256 FPS.

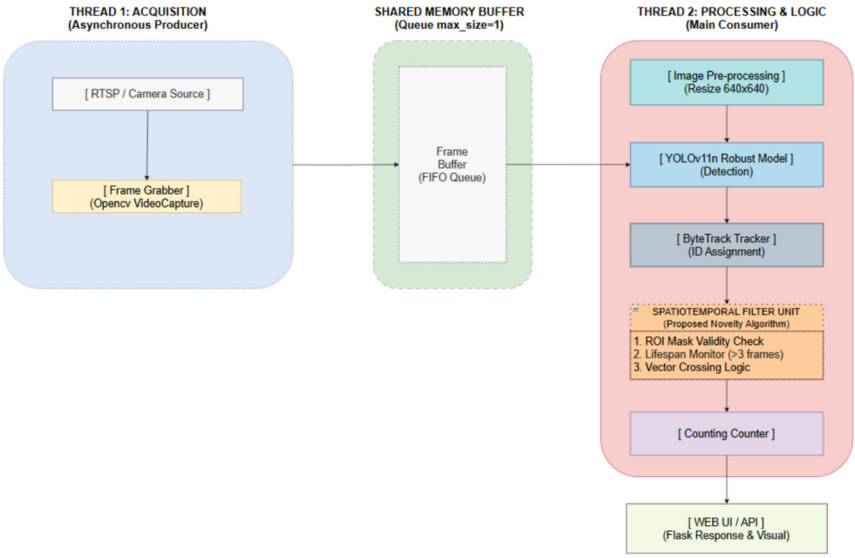

Fig. 1. Proposed system architecture diagram with an asynchronous pipeline.

3.2 Robust Detection Model (YOLOv11)

We selected YOLOv11n [13] based on its parameter efficiency. Training was tuned specifically for the conveyor environment rather than using default augmentation settings:

- **Photometric Distortions:** The HSV jittering $(h = 0.015, s = 0.7, v = 0.4)$ was applied to mimic sensor noise and different lighting and dust conditions in a factory.
- **Geometric Transformations:** For translating and scaling the objects (0.1, 0.2) without changing the orientation (which is a fixed parameter as objects are aligned) of the objects, we used horizontal flipping ($p = 0.5$) along with rotation being set to 0.0 degrees as it would not match the orientation the conveyor was set to, thus controlling the minor positional shifts on the belt without altering the objects.
- **Geometry-Preserving Strategy:** The value for Mixup and Mosaic was set to 0.0 as to not alter the natural edge boundaries of the neighboring objects within the real dataset [11] and thus not allowing the model to learn the features of the merged objects.

3.3 Spatiotemporal Consistency Framework

The raw bounding boxes from the yielded detector are inputted into the ByteTrack algorithm [5] to receive unique ID numbers. However, tracking, on its own, cannot perform accurate counting in noisy environments. We propose a specialized noise filter, consisting of two logical gates:

A. **Temporal Lifespan Filter**

In order to mitigate flickering False Positives caused by noise, we define the "Lifespan" (L_{id}) of a tracked object. An object can be seen as a candidate only if it sustains

for a minimum threshold of τ:

$$L_{id}^{(t)} = \begin{cases} L_{id}^{(t-1)} + 1, & \textit{if detected} \\ 0, & \textit{if lost} \end{cases}$$

We established $\tau = 3$ frames experimentally. Objects with $L_{id} \leq 3$ are discarded as transient noise. The reason for setting $\tau = 3$ is that since the belt moves at about one and half meters per second, and a frame from the camera is taken every thirty seconds, then any object will be in view for at least thirty frames. Therefore, it seems reasonable to set τ, or time constant for smoothing, as three because this would eliminate all of the transients which occur on fewer than two frames.

B. **Vector-based Crossing Logic**

To avoid double-counting of objects, we track an object's centroid as it moves from one video frame to another. a count will be incremented only if that centroid passes the virtual counting line in the correct direction—i.e. forward counts; crossing backward such as during a belt-reversal will be ignored:

$$C_y^{(t-1)} < Y_{line} \leq C_y^{(t)}$$

C. **Trajectory Smoothing**

YOLO bounding box coordinates can be noisy (jittery) on a per-pixel basis from one video frame to the next which will make it difficult to track the centroids close to the counting line. to reduce these noise levels while minimizing the impact of temporal delay, we have used exponential moving average (EMA) to smooth the detection coordinate:

$$P_t = \alpha.P_{raw} + (1 - \alpha).P_{t-1}$$

where P_{raw} is the current detection coordinate, P_{t-1} is the previous detection coordinate, and α is the smoothing factor set at 0.6.

D. **Evaluation Metrics**

Object Detection is a common task for which you are graded on your performance using Mean Average Precision (mAP). Counting can be used as an alternative metric to mAP but will require additional effort when counting error needs to be quantified [10]. We employ the following three counting metrics:

- **Mean Absolute Error (MAE):** This is simply the average counting error value.

$$MAE = \frac{1}{N} \sum_{i=1}^{N} |C_{pred}^{(i)} - C_{gt}^{(i)}|$$

- **Root Mean Square Error (RMSE):** RMSE penalizes larger differences from expected results.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(C_{pred}^{(i)} - C_{gt}^{(i)})^2}$$

- **Average Counting Accuracy (ACA):**

$$ACA = 1 - \frac{\sum_{i=1}^{N}|C_{pred}^{(i)} - C_{gt}^{(i)}|}{\sum_{i=1}^{N}C_{gt}^{(i)}}$$

4 Experimental Results

4.1 Experimental Setup

These datasets include five videos of different states: there are 5000 frames for each sequence; medium condition for sequence B (moderate motion speeds/partial occlusions); difficult condition for sequence C (dust/vibrations from the camera/blur due to motion). These sequences have been manually annotated for correct true count. Performance in the best possible conditions was tested using a cloud based environment with an NVIDIA Tesla T4 GPU for training and evaluation studies. To test this method on a basic laptop, an NVIDIA GeForce RTX 2050 (4 GB) has also been tested to make sure if this is to apply in practice. It also allows testing of the possibility of using the solution for application in industrial edge computers. All experiments were performed using Python. Deep Learning Models were created by use of PyTorch. Object Detection utilized the Ultralytics YOLO framework. Object Tracking utilized ByteTrack [5]. The system was run in an Environment utilizing one or more GPUs for high throughput processing. Additional Libraries (OpenCV) were utilized to perform Image Preprocessing and Video Stream Handling. The total pipeline development and testing were done under a Standard Linux Based Environment to verify the Compatibility of Edge Deployment Scenarios.

4.2 Quantitative Analysis: Robustness & Counting Accuracy

Tables 1 and 2 compared our Proposed Framework against a Baseline method (YOLOv11 + Standard ByteTrack without Spatiotemporal Filtering).

Table 1. Counting Performance Comparison

Sequence	Ground Truth	Baseline Count	Proposed Count	Baseline Accuracy	Proposed Accuracy
Seq A (Easy)	205	205(0)	205(0)	100%	100%
Seq B (Medium)	205	170(−35)	180(−25)	82.9%	87.8%
Seq C (Hard)	205	205(0)	205(0)	100%	100%
Average	–	–	–	94.3%	95.9%

Analysis: The data illustrate a marked distinction between detection quality and count accuracy. It is worth noting that the Baseline model already had a 100% success rate for Seq C—which means that augmenting the training with HSV was enough to make it able to detect and count objects despite static dust being present. However, this was different when using Seq B—in which case rapid movements of the conveyor belt resulted in ByteTrack losing track of object Ids in mid-crossings. Our vector crossing recovery strategy successfully recovered ten of these lost counts. To summarize: while robust training can handle environmental noise as well as motion induced by tracking loss—both are covered by the spatiotemporal filter and their joint effect is the major cause of the failures we observed. Figure 2 show the training result of the proposed method over 50 epoches.

Table 2. Error Metrics (Overall)

Method	MAE ↓	RMSE ↓
BaselineBaseline	11.7	20.2
Proposed (Ours)	**8.3**	**14.4**

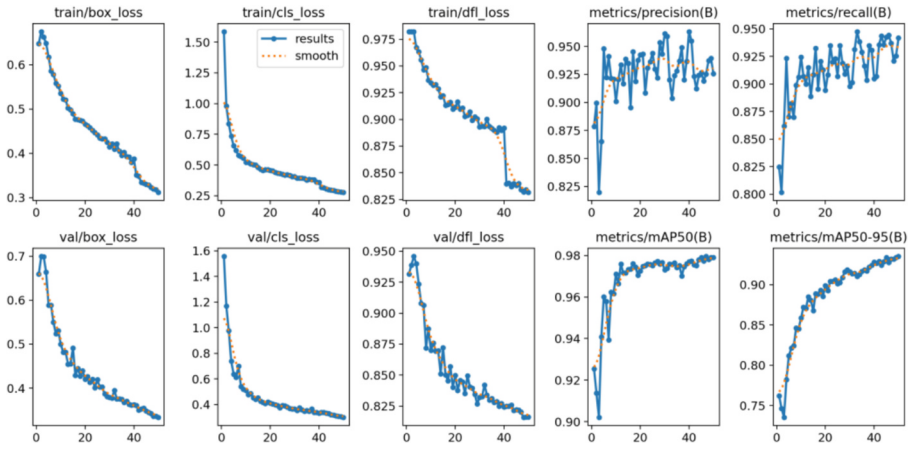

Fig. 2. Training performance graphs (mAP, Loss) of the YOLOv11n model over 50 epochs, demonstrating rapid convergence and stability.

4.3 Comparative Analysis: Efficiency vs. Accuracy

To justify the selection of YOLOv11 over its predecessor, YOLOv8 [4], we conducted a benchmark comparison as shown in Table 3.

Table 3. Efficiency Comparison (YOLOv8n vs. YOLOv11n)

Model	Parameters (M)	Complexity (GFLOPs)	mAP@50 (%)	Recall (%)	Precision (%)
YOLOv8n	3.01	8.1	97.9	93.0	93.9
YOLOv11n (Ours)	2.58	6.3	97.9	94.2	92.6

The results from Table 3 show that the mAP@50 scores for the two models were the same, each being 97.9%, which indicates that model performance had saturated in terms of feature learning for that specific problem. These two reasons support why we selected YOLOv11n for this work:

- Higher Recall: Higher recall than precision is needed in our proposed solution because while false positives may be rejected via back-end processing, false negatives (i.e., missed detections) cannot be recovered.
- Lower Computational Complexity: YOLOv11n is significantly less computationally intensive than its competitor, requiring 14.3% fewer parameters (approximately 2.58 million) versus approximately 3.01 million and utilizing a factor of 0.77 less GFLOPs (6.3 vs 8.2). This will allow our system to utilize more system resources on spatiotemporal filtering and video processing without experiencing bottlenecks with current computing architectures.

4.4 Component Analysis (Ablation Study)

To investigate the contribution of each module, we conducted an ablation study. We defined four configurations:

- **M1:** YOLOv11 Baseline (Standard Training, No Filter)..
- **M2:** M1 + Robust Training (Augmentation).
- **M3:** M2 + Temporal Lifespan Filter.
- **M4:** M3 + Vector Crossing Logic (Full System (Table 4)).

Table 4. Ablation Study Results

Config	Component	mAP@50	Average Counting Accuracy	FPS (Throughput)
M1	YOLOv11 Base	97.0%	85.0%	~278
M2	+ Robust Train	97.8%	94.3%	~278
M3	+ Lifespan Filter	97.8%	95.0%	~270
M4	**+ Vector Logic**	**97.8%**	**95.9%**	**~256**

Discussion: Robust Training (M2) gave the biggest single jump—85% to 94.3%—by fixing detection failures in dusty conditions. Adding the Spatiotemporal Filter (M3 → M4) pushed accuracy to 95.9%, mainly by cleaning up Seq B miscounts. The throughput cost was 22 FPS (278 → 256), corresponding to roughly 0.35 ms per frame—acceptable overhead given the accuracy gain.

4.5 Detailed Latency Analysis

Real-time performance is a key requirement. We analyzed the processing time breakdown per frame on a Tesla T4 GPU (Table 5).

Table 5. Latency Breakdown per Frame

Stage	Operation	Time (ms)	Percentage
1	Pre-processing	0.3 ms	6.4%
2	Inference (YOLOv11n)	3.9 ms	83.0%
3	Tracking & Logic	0.5 ms	10.6%
Total	End-to-End Latency	4.7 ms	100%

The Spatiotemporal Logic's proposed additional overhead is only 0.5 ms (as 10% of total pipeline time), and additional overhead is not relevant to counting accuracy which is most benefitted by further analyses. The overall throughput is $\approx$ 256 FPS (1000 ms/3.9 ms for inference limited throughput).

4.6 Comparative Analysis with State-of-the-Art Methods

To confirm the pros of the suggested structure, an analysis against two common industry counting configurations was performed: (1) YOLOv5 along with DeepSORT (which is a typical Re-ID based tracker), and (2) undifferentiated YOLOv8n with ByteTrack, but again, without our Spatiotemporal Consistency strategy. As discussed in Table 6, DeepSORT does robust tracking, but because of heavy Re-ID, which demands significant appearance feature extraction, the system becomes highly compute-bound and runs at a throughput level which is likely to be unsatisfactory for fast lines (<100 FPS on edge devices). On the contrary, with ByteTrack and the lightweight Spatiotemporal Filter, our framework takes advantage of geometry-based association. Thus, expensive Re-ID computations can be omitted. In addition, our implementation based on YOLOv11n is better than the baseline YOLOv8n in the sense that it consumes 14.3% fewer parameters (2.58M versus 3.01M), which translates to a better allocation of resources to the asynchronous buffering system. This improved system maintains a higher accuracy (95.9%) at real-time throughput, 256 FPS, which is a better fit for resource-constrained industrial edge nodes.

Table 6. Qualitative and Performance Comparison with Existing Methods

Method	Detector	Tracker Type	Re-ID Cost	Robustness to Dust	Est. Efficiency
Traditional DL	YOLOv5	DeepSORT	High	Medium	Low FPS
Recent SOTA	YOLOv8n	ByteTrack	None (IoU only)	Low	High FPS
Proposed	YOLOv11n	ByteTrack + Spatiotemporal Filter	None	High	Very High FPS

4.7 Detailed Latency Analysis

As a way to provide evidence for the functionality of the system, you can direct your attention to the performance of the system placed in a registered cement production plant during the usual production schedule (14:13:54, Sunday). The test used an IP camera, which was placed at a distance of 2.5 m from a belt conveyor, which was transporting 50 kg sacks of cement. The location was characterized by, and typical to industrial settings, dust in the air, motor vibrations from the conveyor belt, and control system lighting that changed in an unpredictable manner.

In the screenshot (Fig. 3), the green polygon marks the ROI and the red line is the counting threshold. The system held stable IDs on the cement sacks (e.g., ID:55, ID:56) despite the dust haze, and correctly ignored background movement that did not cross the line. Objects crossing the red line with valid vector trajectories and sufficient lifespan are

accurately counted, with the current cumulative count displayed in the top-left corner ("COUNT: 40"), confirming the system's robustness in a live production setting.



Fig. 3. Screenshot of real-time counting system interface at a cement factory. The image shows the system's ability to detect and count cement bags even under challenging conditions of low contrast and dust. (Green polygon: ROI; Red line: Counting threshold; Blue boxes: Detected objects) (Color figure online)

5 Conclusion

We have submitted a real-time system for counting objects in a challenging industrial setting and a cement factory at that. Using a lightweight YOLOv11n system, we developed new spatiotemporal consistency logic and coupled that with an asynchronous processing system to overcome the layered problems created by environmental noise (dust/vibration) and the strict latency requirements of rapid moving conveyor systems. We achieved counting accuracy (ACA) of 95.9% at over 256 frames of real-time counting in a millisecond on a Tesla T4 GPU. Also, an ablation study of the noise suppression system added 0.35ms of latency; however, noise counting in specific counted objects improved. In the near term, we want to deploy and further optimize the system on very low-cost edge systems, specifically the NVIDIA Jetson Orin Nano. Also, we want to assess the incorporation of lightweight attention to substantially increase the system performance in high-density occluded counting.

References

1. Li, Z., Wang, J., Wang, Y.: Real-time computer vision in smart manufacturing: a review. J. Real-Time Image Proc. **18**, 1–16 (2021)
2. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779–788 (2016)

3. Wang, C.-Y., Bochkovskiy, A., Liao, H.-Y.M.: YOLOv7: trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7464–7475 (2023)
4. Jocher, G., Chaurasia, A., Qiu, J.: YOLO by Ultralytics (2023). https://github.com/ultralytics/ultralytics
5. Zhang, Y., et al.: ByteTrack: multi-object tracking by associating every detection box. In: Avidan, S., Brostow, G., Cissé, M., Farinella, G.M., Hassner, T. (eds.) ECCV 2022. LNCS, vol. 13682, pp. 1–21. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-20047-2_1
6. Wojke, N., Bewley, A., Paulus, D.: Simple online and real-time tracking with a deep association metric. In: IEEE International Conference on Image Processing (ICIP), pp. 3645–3649 (2017)
7. Ciuti, G., Dario, P.: Visual quality inspection systems: a systematic literature review. IEEE Access **8**, 20123–20157 (2020)
8. Sun, X., Wu, P., Hoi, S.C.: Face-to-face overlapping object counting on conveyor belts. IEEE Trans. Ind. Inf. **15**(3), 1545–1555 (2019)
9. Amrilla, Y., Heryadi, Y.: Cement sack counter application using convolutional neural network on conveyor belt. In: International Conference on Information Management and Technology (ICIMTech), pp. 1–6 (2021)
10. Gao, G., Liu, Y., Wang, M.: Counting objects by tracking: a robust method for industrial conveyor belts. IEEE Trans. Instrum. Meas. **69**(7), 4221–4231 (2020)
11. Lin, T.-Y., et al.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48
12. Bochkovskiy, A., Wang, C.-Y., Liao, H.-Y.M.: YOLOv4: optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934 (2020)
13. Ultralytics: YOLO11 documentation and performance metrics (2024). https://docs.ultralytics.com/models/yolo11
14. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1, pp. I–511 (2001)
15. Hsu, H.-M., Huang, T.-W., Wang, G., et al.: Multi-camera tracking of vehicles based on deep features re-identification. IEEE Trans. Intell. Transp. Syst. **22**(12), 7650–7660 (2021)
16. Tariq, R.: Counting objects in an industrial environments using YOLOv5 and YOLOv7, Padua Thesis Archive (2023)
17. Kumar, K., Kumar, P., Panda, S.: A real-time object counting and collecting device for industrial automation process using machine vision. IEEE Sens. J. **23**(12) (2023)
18. Faruk, M.J.H., et al.: A deep learning framework for object detection and tracking: CNN-based parcel tracking on conveyor belts. In: IEEE 6th International Conference on Sustainable Technologies for Industry 4.0 (STI) (2025)
19. Sapkota, R., Karkee, M.: Ultralytics YOLO evolution: an overview of YOLO26, YOLO11, YOLOv8, and YOLOv5 object detectors. arXiv preprint arXiv:2510.09653 (2025)
20. Liu, W., et al.: YOLO-STOD: an industrial conveyor belt tear detection model based on Yolov5 algorithm, Scientific Reports (2025)