The International Journal of Advanced Manufacturing Technology (2023) 126:179–195
https://doi.org/10.1007/s00170-023-11098-6

ORIGINAL ARTICLE

MOSPPA: monitoring system for palletised packaging recognition and tracking

Julio Castaño-Amoros[1] · Francisco Fuentes[1] · Pablo Gil[1]

Received: 23 August 2022 / Accepted: 9 February 2023 / Published online: 23 February 2023

Abstract
The paper industry manufactures corrugated cardboard packaging, which is unassembled and stacked on pallets to be supplied to its customers. Human operators usually classify these pallets according to the physical features of the cardboard packaging. This process can be slow, causing congestion on the production line. To optimise the logistics of this process, we propose a visual recognition and tracking pipeline that monitors the palletised packaging while it is moving inside the factory on roller conveyors. Our pipeline has a two-stage architecture composed of Convolutional Neural Networks, one for oriented pallet detection and recognition, and another with which to track identified pallets. We carried out an extensive study using different methods for the pallet detection and tracking tasks and discovered that the oriented object detection approach was the most suitable. Our proposal recognises and tracks different configurations and visual appearance of palletised packaging, providing statistical data in real time with which to assist human operators in decision-making. We tested the precision-performance of the system at the Smurfit Kappa facilities. Our proposal attained an Average Precision (AP) of 0.93 at 14 Frames Per Second (FPS), losing only 1% of detections. Our system is, therefore, able to optimise and speed up the process of logistic distribution.

Keywords Manufacturing automation · Cardboard packaging · Pallets recognition and tracking · Pallets workflow control

1 Introduction

In logistics, the goods transported by road, air and water must be moved using platforms known as pallets. Pallets, which are made of cheap lumber and nails, serve as a solid load base. They are a standard means of moving products in logistics centres but are also used for easy transportation when shipping by road, sea, or air. About 5.1 billion pallets are used around the world [1] and many of them are used to move boxes of goods and products from producers or manufacturers to customers.

✉ Julio Castaño-Amoros
julio.ca@ua.es

✉ Pablo Gil
pablo.gil@ua.es

Francisco Fuentes
ffn1@alu.ua.es

[1] AUtomatics, RObotics, and Artificial Vision Lab, University of Alicante, San Vicente del Raspeig, Carr. de San Vicente del Raspeig, S/N, 03690 Alicante, Spain

Packaging factories usually design and manufacture solutions for consumer, retail, industrial products, e-commerce, etc. Some of these solutions consist of creating corrugated cardboard packaging for fruit, vegetables, and bulk agriculture. Manufacturers make and supply their customers with a wide variety of corrugated board for this kind of packaging, such as solid boards or tailored folding carton sheets. The products are supplied as batches of disassembled packaging vertically stacked on a pallet for easy transportation (Fig. 1).

In factories, the pallets travel in arbitrary positions on a set of roller conveyors controlled by human operators, as shown in Fig. 2. The structure of conveyors creates paths connecting different nodes of the logistics network within the factory. The pallets carry the product from several machines that produce corrugated cardboard and on the roller conveyors to the storage locations in the warehouse or to the shipping gates for road transport. Figure 3 shows a partial view of a roller conveyor map in a Spanish Smurfit Kappa factory in which the proposed system was tested. It illustrates points along the route at which conveyor lines join and the load can be diverted using a roller turntable. The turntables are controlled by a human operator. This

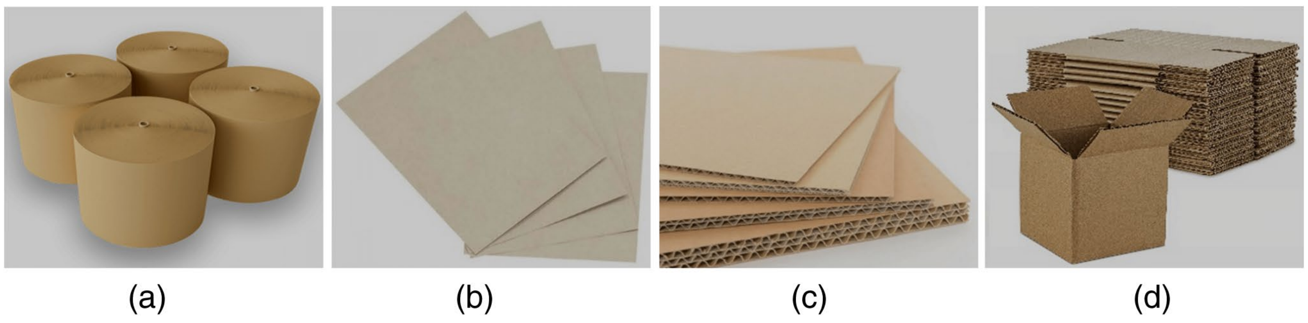


Fig. 1 Manufacturing stages: (**a**) Containerboard roll. (**b**) Softwood fibre sheets obtained by processing different kinds of rolls. (**c**) Corrugated cardboard sheets obtained by mixing fibre sheets. (**d**) Folding boards with which to build cardboard boxes

operator can change the pallet trajectory by rotating it any angle from 90 up to 180°. At this point, the operator identifies the product manually and diverts it by following storage and distribution policies (type of product, warehouse status, delivery date, etc.). The standard speed of the roller conveyor system is 0.08 m/s. This pace was chosen by the factory to avoid the excessive vibration of the load on the pallet.

To improve the tasks that this human operator performs, we implement an automatic visual monitoring system for the pallets. The idea is to achieve flexibility through warehouse automation, as suggested in [2]. Our system assists the operator to sort packages in order to ensure that they can be distributed around the factory more efficiently and without errors.

In a previous work [3], we used low-cost colour cameras to carry out preliminary experiments with which to detect and recognise pallets loaded with different types of disassembled packages. The proposal in question also makes it possible to track pallets along the path of the roller conveyor. However, we found some problems with YOLO-type object detectors when the pallets were partially occluded in the image while the turntable was rotating. This problem forced the tracker to change the ID and generate the wrong statistics. We have, therefore, extended the experimentation in order to solve this issue and thus improve the detection, recognition and tracking processes. We have specifically tested more state-of-the-art tracking methods and now propose the use of an Oriented Bounding Boxes technique with YOLOv5, while also maintaining Mask R-CNN and the original YOLOv5 in the comparison. The goal is to improve the performance of our system by reducing the number of detections lost by our previous Pallet Detector, especially when the pallet is not completely seen by the camera while the turntable is rotating. This new solution, therefore, makes it possible to increase the robustness of our system. In this paper, we carry out extensive and rigorous experimentation on these three detectors and different types of trackers, from traditional to deep learning–based trackers with the objective of optimising the warehouse distribution in order to prevent the dis-placement of forklift trucks and congestion in the node queues. This consequently leads to a reduction in vehicle traffic, which may cause accidents, and to production not being affected by the saturation of manufacturing lines. Moreover, the limited space in the warehouse can be used more efficiently.

This paper is organised as follows. Section 2 presents the visual recognition and computer vision techniques used in industrial logistics, while Section 3 presents our proposal for the implementation of a visual monitoring intelligent system for packaging pallets on conveyor belts. Section 4 shows the results obtained in a real scenario with different kinds of pallets, and finally, Section 5 shows a summary of our findings and contributions, along with the main limitations of this work and some future research lines.

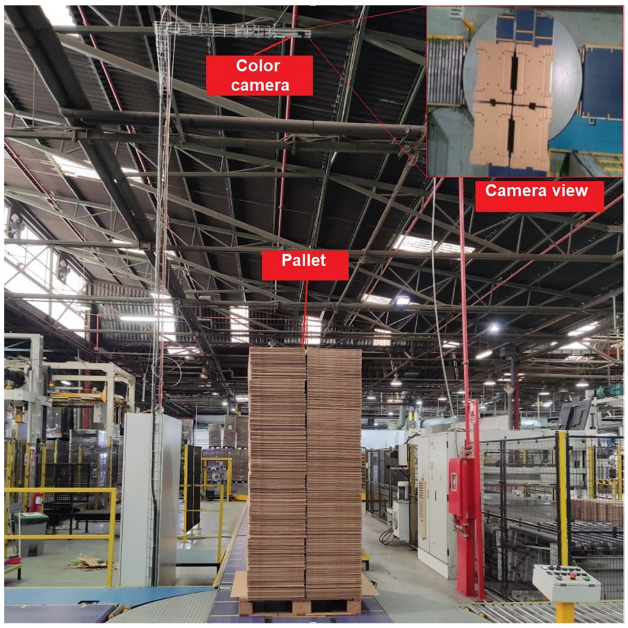


Fig. 2 Pallet transporting stacked disassembled packaging. Camera height ~ 6.5 m, pallet height ~ 2.5 m

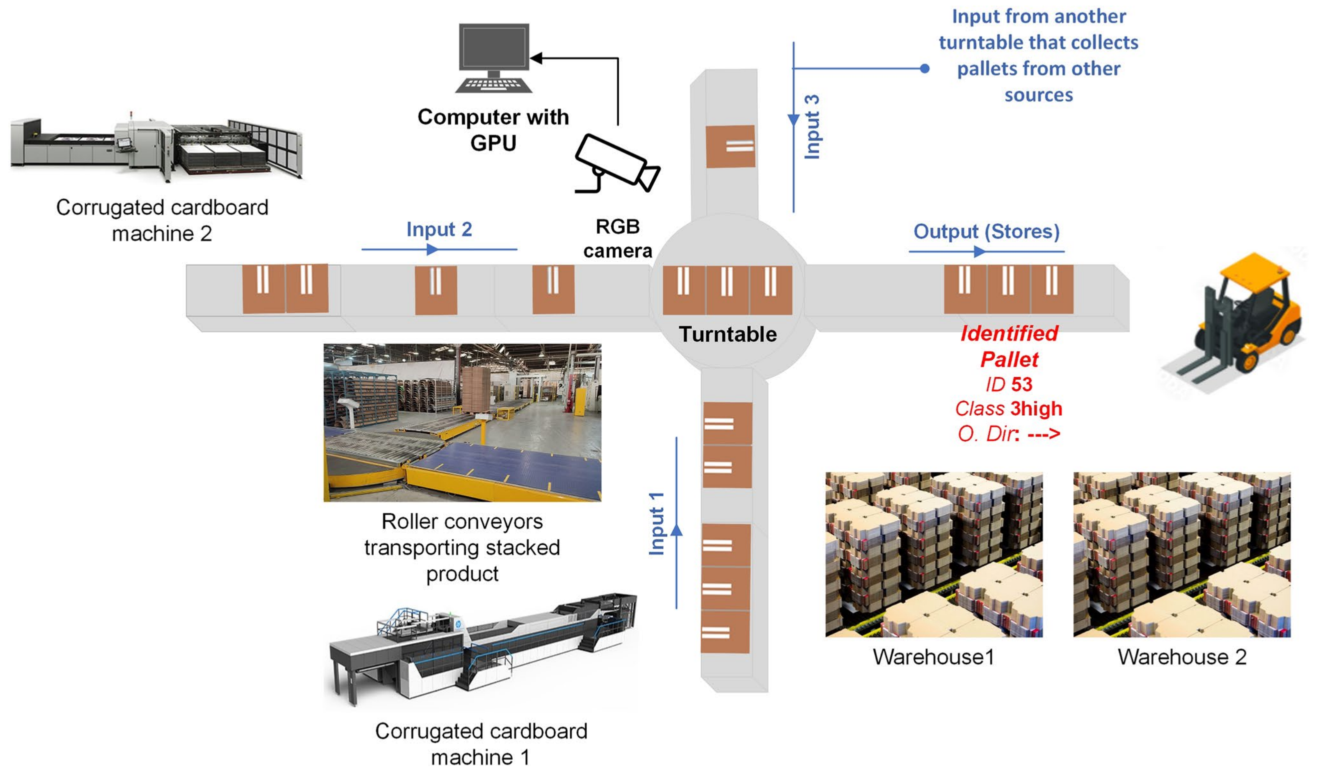

Computer with GPU
Input from another turntable that collects pallets from other sources
Input 3
Corrugated cardboard machine 2
Input 2
RGB camera
Output (Stores)
Turntable
Identified Pallet
ID 53
Class 3high
O. Dir: --->
Roller conveyors transporting stacked product
Input 1
Warehouse1
Warehouse 2
Corrugated cardboard machine 1

Fig. 3 Partial map of a roller conveyor connecting different factory nodes in a Spanish Smurfit Kappa factory

2 Related works

Digital transformation processes have, in the last few years, begun to be used to optimise logistics in factories, and more specifically, machine vision, as reviewed by [4]. The work of [5] and [6] discusses how image processing and computer vision techniques contribute to improving logistics by automating a great variety of operations, such as security and protection, occupancy of storage, traceability and trackability or inspection for quality control, etc.

Specialised algorithms based on machine learning have recently been implemented in order to recognise packaging boxes on pallets. For example, in the study [7] shows a Neural Network (NN) called TetraPackNet that is used to detect corner points from the images of the side faces of pallets. This approach was designed as an extension of a well-known NN called CornerNet [8]. Previously, in [9], the same authors implemented a pipeline consisting of a combination of segmentation algorithms. This pipeline detects the whole pallet unit and its number of visible stacked packaging boxes.

In the aforementioned works, the packages on pallets are recognised but are neither spatially located in the environment nor tracked when they change position. Focusing on this aspect, some work such as that shown in [10] have presented a method with which to detect, localise and track multiple pallets by applying machine learning to the data of a 2D laser rangefinder on board a mobile robot. More specifically, they have combined a segmentation NN such as Faster R-CNN [11] in cascade with a convolutional classifier.

Other approaches are focused on recognising and locating pallets in warehouses or other types of objects/packages on conveyor belts. In the first case, the goal is to discover the position of the pallet pockets in order to insert the forks of a forklift truck correctly, as occurs in [12] or [13]. In this respect, [14] compared state-of-the-art image algorithms for object detection with classic neural detectors such as YOLOv1 [15] in all cases when applied to the recognition of pallets in grayscale images. The results showed a significant improvement in accuracy and response time. In [16], the authors used RGB cameras and compared some known Convolutional Neural Networks (CNNs) such as Faster R-CNN, YOLOv4 [17] and SSD [18], again in order to recognise pallet pockets. 3D cameras and processing algorithms of point clouds (3D-Keypoints, depth segmentation, etc.) were also used in [12, 19, 20] and [21] for this same task. Note that the purpose of all these works is automatic pallet detection, but using only views of the side faces. Moreover, they detect only the lumber platform and not the load of packages.

With respect to the detection and recognition of objects on conveyor belts, in [22–24], for instance, the authors used traditional computer vision methods (contour and ellipse detection), and deep learning based methods (Mask R-CNN and YOLO) to localize and track cardboard and plastic boxes on different types of conveyors belts in the logistics industry. In the same line, but applied to other industries, in [25], an improved tiny version of YOLOV3 is used to detect coal and gangue on a coal conveyor belt. In [26], meanwhile, background subtraction and segmentation is used to detect foreign materials (metal, wood, rubber, etc.), and in [27], ResNet, VGG and DenseNet architectures are used to detect imperfections in thermoforming food packages. Although these works are not focused on pallet detection, the approaches used are similar to that of our proposal, since they address object detection and instance-level recognition in different fields and industries. As occurs in our problem, the objects are homogeneous and it is difficult to distinguish them.

Unlike the works described above, our method is focused on detecting and recognising packaging loaded on pallets, and not on lumber platforms as described in [12, 14, 16]. Moreover, we use video sequences captured from an RGB camera with a bird's eye view configuration rather than a side view to detect the packing, as occurred in [7, 9]. Our proposal is based on 2D techniques rather than 3D processing, as in [12, 19, 20], signifying that we require less computational complexity and there is less dependency on the camera type and its location. Moreover, only a few works, such as [10], are able to carry out trackability. The majority of works do not, therefore, process video but only static images without movement. In our proposal, we add this essential processing in order to not only recognise, but also control roller conveyor congestion. Our system does this by tracking the number of pallets of each type, and estimating the starting point (source machine) from the pallet trajectory. The proposal analyses the characteristics of roller conveyor congestion and allows the factory to make the decisions. Our goal is inspired by [28], but considers a different application scenario and provides a completely different solution.

3 Methods

Deep learning algorithms are in a phase of continuous development and progress, especially when applied to computer vision. However, even when using these algorithms, object detection and segmentation tasks are a challenge yet to be solved in some applications. This development and progress have, over the years, generated significant research and results in the fields of both object detection [29–31] and segmentation [32]. Another important line of research to have emerged in recent years is the image tracking, as shown in [33, 34]. Deep learning algorithms are also undergoing a boom in this area [35] because they facilitate large-scale work in different environments and situations. Taking the study and analysis of some of the aforementioned algorithms as our starting point, we propose a monitoring system for the recognition and tracking of palletised packaging (MOSPPA) in the paper industry.

The general scheme of the proposed method is shown in Fig. 4. It consists of a two-stage architecture composed of CNNs, one for detection and recognition, and the other for tracking. Our system is able to select from among three different detection algorithms: *YOLOv5* [36], *YOLOv5* with Oriented Bounding Box (*YOLOv5OBB*) [37, 38] or *Mask R-CNN* for the detection stage [39] combined with different visual trackers based on correlation filters such as Kernelized Correlation Filters (KCF) [40], sparse representation such as Multiple Instance Learning (MIL) [41], adaptive correlations in Fourier space such as Minimum Output Sum of Squared Error (MOSSE) [42], or deep learning strategies such as *DeepSORT* [43], *StrongSORT* [44] or *ByteTrack* [45] for the tracking stage. These algorithms were trained and tuned for the specific task of monitoring disassembled packaging.

In the first stage, the RGB cameras, in a bird's eye view configuration, start recording the video sequence when a laser sensor located on each turntable detects the presence of a pallet. Each frame in this sequence is the input image of our *Pallet detector*, which processes this input and infers a prediction. This prediction contains three fields: the location of the pallet in the image, the score or confidence of the pallet type, and the predicted class or pallet type. The format of the pallet location differs depending on the CNN used (YOLOv5, YOLOv5OBB or Mask R-CNN), as will be discussed later.

In the second stage, the output obtained previously from the *Pallet detector* acts as the input for this stage. This input allows the selected tracking algorithm to estimate a unique identification number for each pallet and maintain that identifier throughout its trajectory. It is, therefore, possible to calculate the counters for the production statistics and logistics. Moreover, pallet location, confidence and type are used to visualise results and calculate the direction of the pallet on the turntable. This direction is achieved by measuring the centroid displacement of the pallet location in the image.

3.1 Pallet detection

On the one hand, YOLOv5 is an algorithm that was designed to detect instances of known objects by computing bounding boxes in order to mark their regions in the image. This object detector has three main parts: the backbone, the neck and the head. The NN employed as the backbone in YOLOv5 is Cross Stage Partial Network (CSP) [46], which is used to extract visual features from the input images. These features can be low-level features (corners or borders) or high-level features (e.g. the eyes, nose or ears of a cat). The neck used

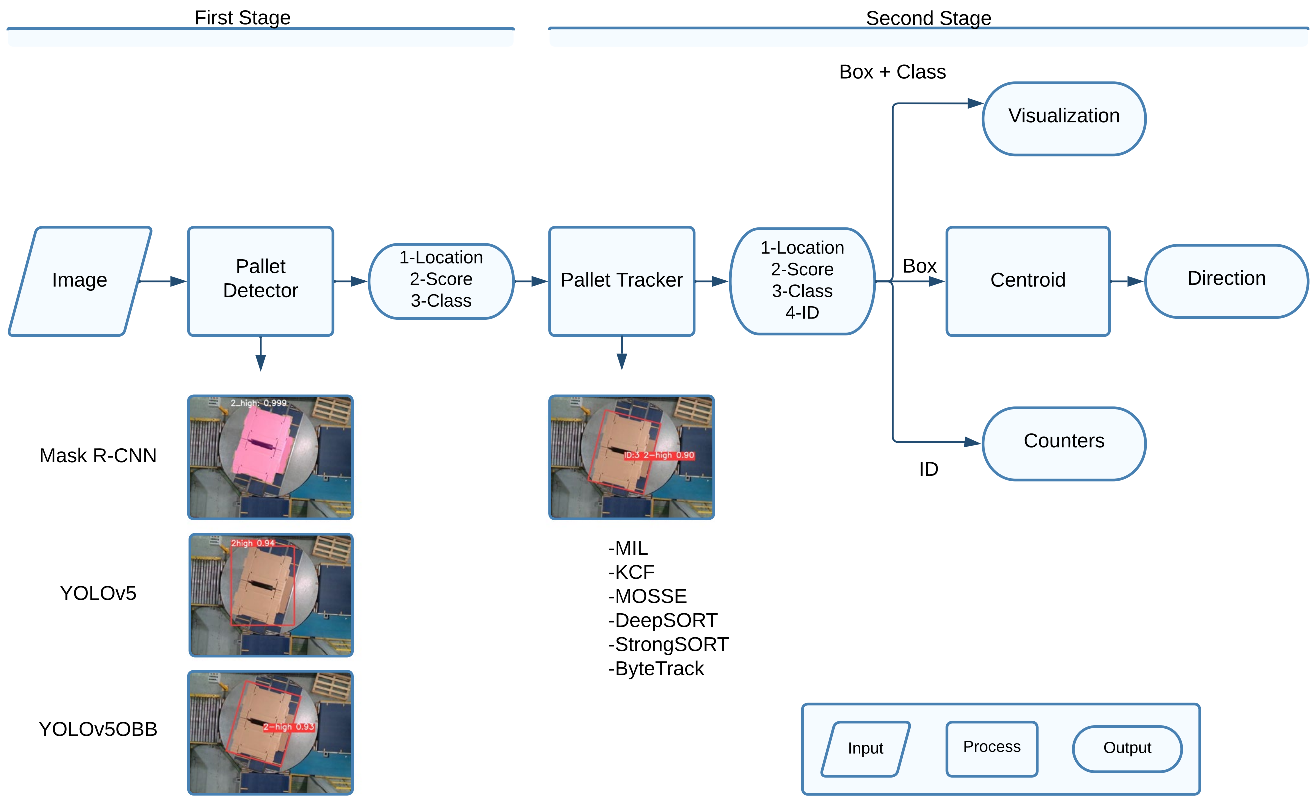


Fig. 4 Pipeline of our implementation

to generate feature pyramids in order to learn features in different sizes and scales is the Path Aggregation Network (PANet) [47] and finally, YOLOv5 uses the same head as in the previous versions. This applies anchor boxes and generates the output bounding box regression and object classification. A bounding box is a rectangle that surrounds the object detected in the image, and most importantly, it is the pallet location estimated by YOLOv5.

However, from a top view perspective, a straight bounding box does not fit rotated pallets correctly. The OBB generated by YOLOv5OBB can, therefore, estimate the orientation of the pallet with greater precision. YOLOv5OBB, therefore, maintains the same structure as YOLOv5, but also includes some new features with which to solve the angle prediction of the pallets detected as shown in Fig. 5. Note that the formulation of the prediction as a classification problem rather than a regression problem is the most important improvement, together with the addition of the Circular Smooth Label (CSL) [38] technique. This new technique allows the resolution of important issues in oriented object detection, such as the periodicity of angles and the exchangeability of edges.

On the other hand, the Mask R-CNN algorithm detects the object instances by segmenting the image in regions, in which each region contains only the object pixels. These regions, which are better known as masks, are the pallet location generated by Mask R-CNN in the image.

This object detector is divided into two stages. The first stage is composed of a Feature Pyramid Network (FPN) that uses a ResNet101 CNN as a backbone in order to extract features from the image. This stage also contains a Region Proposal Network (RPN), which processes the feature map extracted by the FPN in order to propose regions that may contain objects. The second stage obtains the regions proposed by the first stage and assigns them to different areas

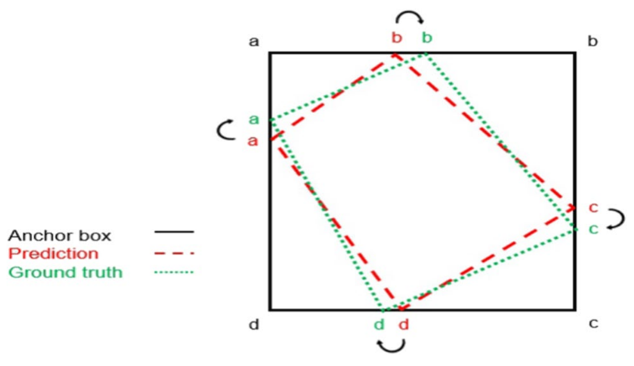


Fig. 5 Calculation of OBB by learning to predict to the angle offset between the predicted OBB and the ground truth OBB

of the feature map, after which the objects, bounding boxes and masks are extracted.

Figure 6 shows several examples of pallet detection using the three aforementioned methods. These examples show that all of these methods are able to detect and recognise pallets of varied shapes, heights, widths and colours under different indoor light conditions (morning and afternoon) with *AP* values higher than 90%.

We chose these three methods from among those shown in [29–32] because they achieved the best performance (in terms of accuracy, precision and time) for similar tasks to ours, without being forced to run on extremely expensive GPUs. These algorithms take an image as input, and follow a supervised learning methodology. This signifies that they have to be taught to recognise the objects by means of a prior training phase. In this phase, the NNs learn to automatically extract image features of objects. These features allow these NNs to perform a full classification (Mask R-CNN) or a classification-regression task (YOLOv5, YOLOv5OBB).

3.2 Pallet tracking

Tracking algorithms were formerly used to predict the location of an object in a video sequence from an initial detection in the first frames. Trackers are now able to track moving objects within the Field of View (FoV) of a camera, developing a dynamic detection. The objects must, therefore, be detected beforehand in static images, after which the temporal information regarding the spatial features that define each detection is analysed.

The state of the art of dynamic detection–based tracking methods consists of training correlation filters (KCF), discriminative classifiers (MIL), adaptative correlations (MOSSE) or deep neural networks (based on SORT and ByteTrack) to distinguish the target from the background. Of all of them, we have selected some of the most representative methods for each type, e.g. KCF, MIL, MOSSE, DeepSORT, StrongSORT and ByteTrack, to be evaluated for our goal. KCF estimates an optimal image filter that, when applied to the input image, produces a desired response. The desired response is typically of a Gaussian shape in the centre of the target location, signifying that the score decreases with the distance. It is highly efficient and is capable of running at hundreds of FPS. MIL trains a logistic classifier in an online manner in order to separate the object from the background. The algorithm obtains the samples by looking in a small neighbourhood around the current location so as to generate several potential positive predictions. MOSSE, our last choice of classical trackers, calculates correlations in Fourier space in order to minimise the sum of squared errors among actual and

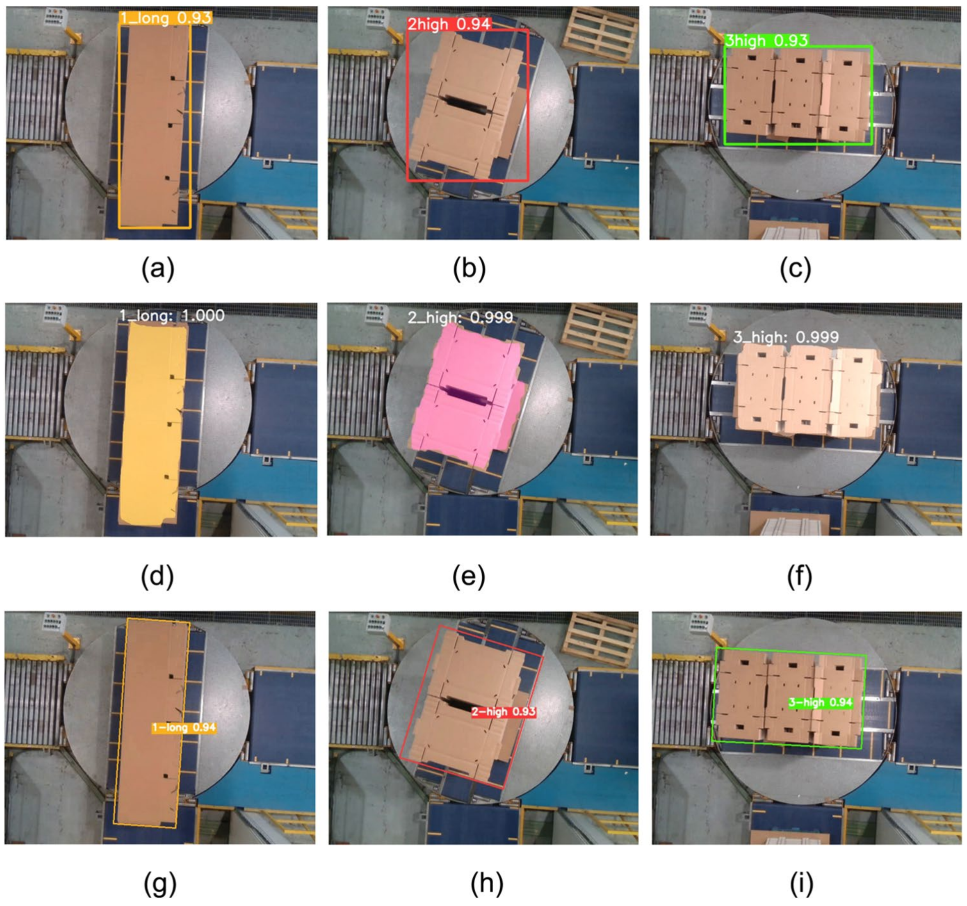


Fig. 6 Examples of detections from our *Pallet detection* module using YOLOv5 (top row), Mask R-CNN (middle row) and YOLOv5OBB (bottom row)

predicted correlations. MOSSE is, therefore, able to track objects rapidly.

With respect to trackers based on deep learning techniques, DeepSORT is composed of four stages: detection, estimation, association and track identification. In our proposal, the original DeepSORT detector is replaced with our Pallet detector, which makes it possible to select between one of the three aforementioned NNs (YOLOv5, YOLOv5OBB or Mask R-CNN). The remaining stages have been maintained. The second stage is accordingly the original, and it propagates the detection from the current frame with a linear constant velocity model and a Kalman filter. The third stage attempts to assign detections to existing targets using the Intersection Over Union (IoU) as a cost matrix, solved by the Hungarian algorithm [48]. The final stage imposes a different identity on each object in the image when objects enter and leave the image.

The use of the StrongSORT tracker with new techniques was proposed in order to update and improve DeepSORT. Its authors first replaced the custom and simple CNN, which is used in DeepSORT to extract features, with a stronger feature extractor [49] which uses ResNet50 [50] CNN as a backbone, thus allowing the extraction of more discriminative features. Second, the feature bank technique, which was used in DeepSORT to store features of short tracks, has been replaced with the feature updating strategy proposed in [51], which is more efficient in terms of time consumption. Finally, StrongSORT has adopted the Enhanced Correlation Coefficient (ECC) technique [52] for camera motion compensation and the vanilla Kalman Filter has been replaced with the NSA Kalman Filter [53], which handles the problem of noise detection.

ByteTrack is a new method in which a new technique with which to perform data association has been proposed. Current deep learning trackers usually carry out this data association by eliminating bounding boxes with scores lower than a threshold. However, occluded objects are discarded, producing fragmented trajectories. This tracker tracks by associating bounding boxes with high or low scores, achieving very good results in several benchmarks.

Figure 7 shows examples of pallet tracking from the six aforementioned trackers. As can be seen, deep learning–based trackers, such as (i, d), (e, k) and (f, l), are more accurate because they take the output from the pallet detector as input, while the traditional trackers run both detection and tracking tasks. Traditional trackers, such as (a, g), (b, h) and (c, i), accordingly have a worse fit for the bounding box.

3.3 Set-up and implementation

Our proposal is designed to be able to use several colour Intel®RealSense™ D415 cameras that provide images with a resolution of 640 × 480 pixels. Note that, although this camera also provides depth information, we have used only RGB data to speed up the detection process. We do not, however, discount the use of the depth channel in future work if it is required in order to improve the precision of new packages. These cameras capture up to 30 FPS, but the FPS has been set to 15 by considering the speed of the roller conveyor in our experimentation. We have mounted a camera on two different turntables (Fig. 3) in order to monitor the traffic at different connection points of roller conveyors in industry. Figure 2 shows an example of a camera capturing a bird's eye view of a turntable. The cameras are connected to a computer with the following components: an i5-7400 CPU 3.00 GHz (4 cores/4 threads), 16 GiB DIMM DDR4 Synchronous 2,

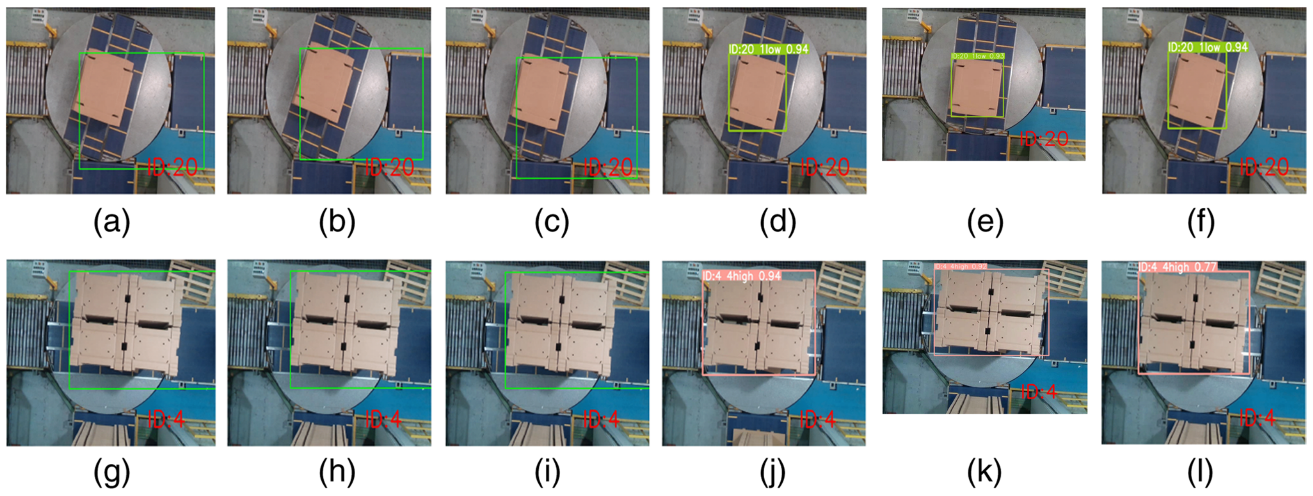


Fig. 7 Tracking examples from our *Pallet tracker* module comparing YOLOv5 with different trackers: MIL (**a**,**g**), KCF (**b**,**h**), MOSSE (**c**,**i**), DeepSORT(**d**,**j**), StrongSort (**e**,**k**) and ByteTrack (**f**,**l**)

a Western Digital WDS500G2B0A WD 500 GB, and a NVIDIA GTX 1070Ti (8 GiB) GPU.

We additionally use a reflective/thrubeam laser displacement sensor for each camera in order to detect targets based on position, i.e. the sensor is used as a camera trigger.

When the camera is triggered, the algorithm shown in Algorithm 1 starts to find an existing Identifier (ID) in the previous camera queue $Queue_{i-1}$ (if there are multiple cameras). If the previous queue is empty, the current *i-camera* algorithm starts to track the pallet, and the assigned ID is queued in its own queue $Queue_i$. If the previous queue is not empty, then the algorithm obtains the ID from that camera and starts to track that ID.

Algorithm 1 Camera control algorithm

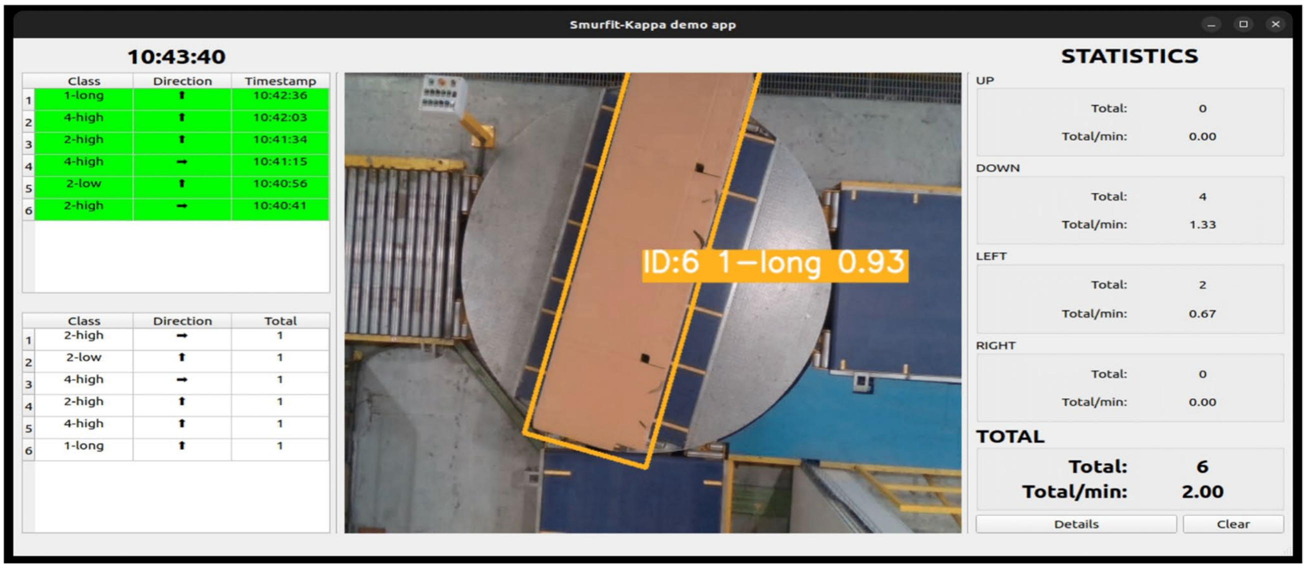
```
1: if $Queue_{i-1}$ is empty then
2:    ID, class, localization ← palletDetection (frame)
3:    Tracker ← ID
4:    Track ID
5:    $Queue_i$.append(ID)
6: else
7:    ID ← $Queue_{i-1}$.pop(0)
8:    Tracker ← ID
9:    Track ID
10:   $Queue_i$.append(ID)
11: end if
```

Our system is, therefore, able to maintain the same ID throughout the roller conveyor map, although it is tracked by different cameras on its way. This issue prevents the pallet ID from being modified during the tracking process performed by other cameras located in the same path. The first camera that detects and tracks a pallet is that which assigns the ID, while those that follow simply check whether there are already-identified pallets in the queues.

The coding was implemented in a conda environment: Python 3.9.12, PyTorch 1.10.1, CUDA 11.3 (with driver version 470.129.06) and OpenCV 4.5.5 over Ubuntu 18.04.6 LTS Bionic Beaver. We also used PySide 6.2.2 to build the Graphical User Interface (GUI) so as to allow the human operator to visualise the inspection tasks (Fig. 8). Our GUI allows the human operator to address the process and interact with devices such as cameras and laser sensors.

The GUI is divided into three columns: left, centre and right. The left-hand column contains two tables showing all the information related to the detection (pallet type, direction and timestamp). Real-time detections are shown in the top table. The colour indicates the reliability of the detection on the basis of the average score/confidence during the trajectory of the pallet on the turntable. Green means that there is confidence in the detection, while red warns the operator to check this pallet. The bottom table is employed to save a record of detections sorted by pallet type. In the centre column, we show the image from the camera when adding the predicted information from our system (Identifier (ID), class, score). Finally, statistical values regarding production are shown in the right-hand column (pallets and pallets per minute in each direction and in total). These values help the operator to anticipate congestion on the roller conveyor.

The procedure is the following. First, packaging machinery manufactures each corrugated cardboard package. A package is finished when it has been cut according to the customer's design (Fig. 9). The customer



10:43:40

	Class	Direction	Timestamp
1	1-long	↑	10:42:36
2	4-high	↑	10:42:03
3	2-high	↑	10:41:34
4	4-high	→	10:41:15
5	2-low	↑	10:40:56
6	2-high	→	10:40:41

	Class	Direction	Total
1	2-high	→	1
2	2-low	↑	1
3	4-high	→	1
4	2-high	↑	1
5	4-high	↑	1
6	1-long	↑	1

STATISTICS

UP
Total: 0
Total/min: 0.00

DOWN
Total: 4
Total/min: 1.33

LEFT
Total: 2
Total/min: 0.67

RIGHT
Total: 0
Total/min: 0.00

TOTAL
Total: 6
Total/min: 2.00

Fig. 8 Interface visualization of our system showing the detection and tracking of one of the cameras

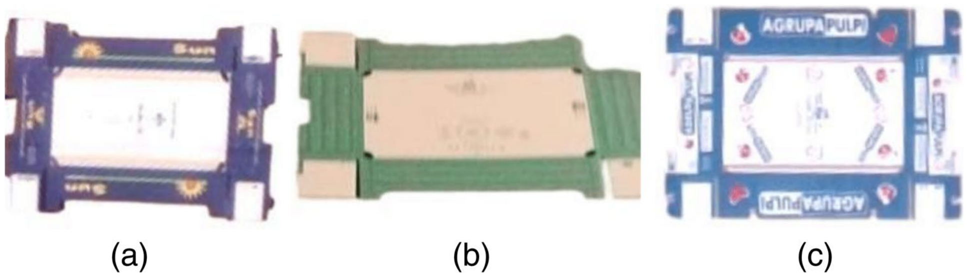


Fig. 9 Three disassembled packages with custom designs. The design is printed on the back of the package

chooses the punched shape and the printing pattern. The disassembled packages are then stacked (with the printed design downwards) on a pallet until it is full, after which the pallet is located on a system of roller conveyors to be transported inside the factory.

Second, the pallet moves along the roller conveyor and passes through some checkpoints. Each of these checkpoints consists of our visual inspection system and a roller turntable, on which different conveyor lines converge (Fig. 3). When a pallet reaches the border of the inspection area, a laser sensor detects its presence and enables the camera to capture a video in real time. The images from the cameras are transmitted to a computer in which we run the pipeline shown in Fig. 4. As mentioned previously, our system recognises and tracks each pallet on the turntable, providing a unique ID for each pallet detected (Fig. 8).

4 Experiments and results

When describing the experimentation, we first present the dataset created to train and test the proposed system. Since the *Pallet detector* and *Pallet tracker* modules were implemented in order to allow the operator to work with different algorithms, it was recommendable to compare the performance of the models. This was done by performing detection experiments by calculating AP values for the validation and test set. We then analysed which tracker best fitted the detector chosen, comparing different trackers in terms of IoU and FPS.

4.1 Dataset

As explained in Section 3, each client's product has a custom design (Fig. 9). However, different clients may order the same type of pallet (number of boxes per layer, height, width, depth, etc.). The number of pallet types can, therefore, be reduced to seven classes, considering that all clients order these types of pallets but with different designs. Note that the printed side is not visible from the camera when the product is stacked. We annotated each class depending on the stack configuration (number of boxes per layer) and the pallet type (low, high or long). For example, a long pallet with a stack configuration of one box was annotated as *1long*, as can be seen in Fig. 8. Figure 10 shows the stack configuration of the pallets in each class.

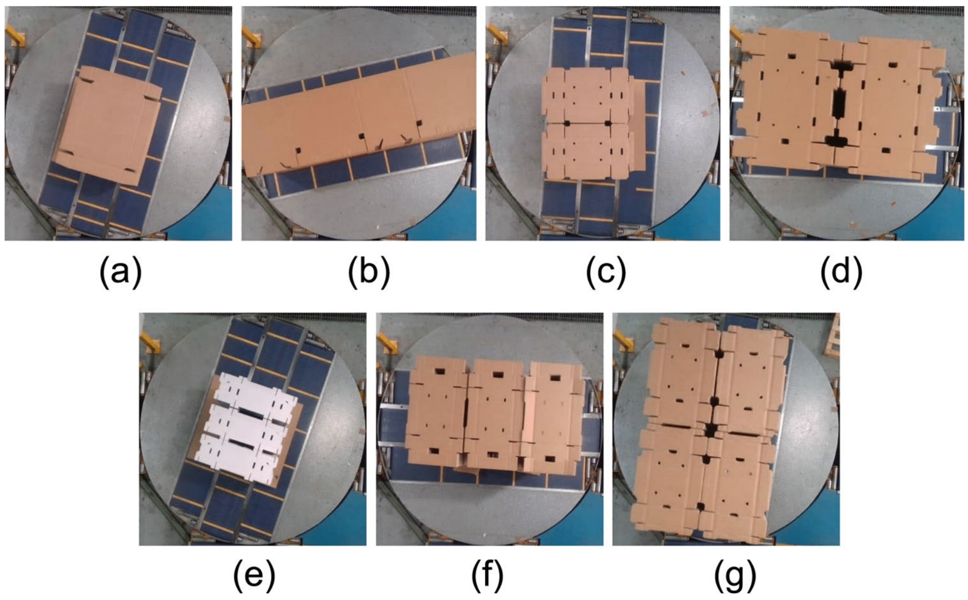


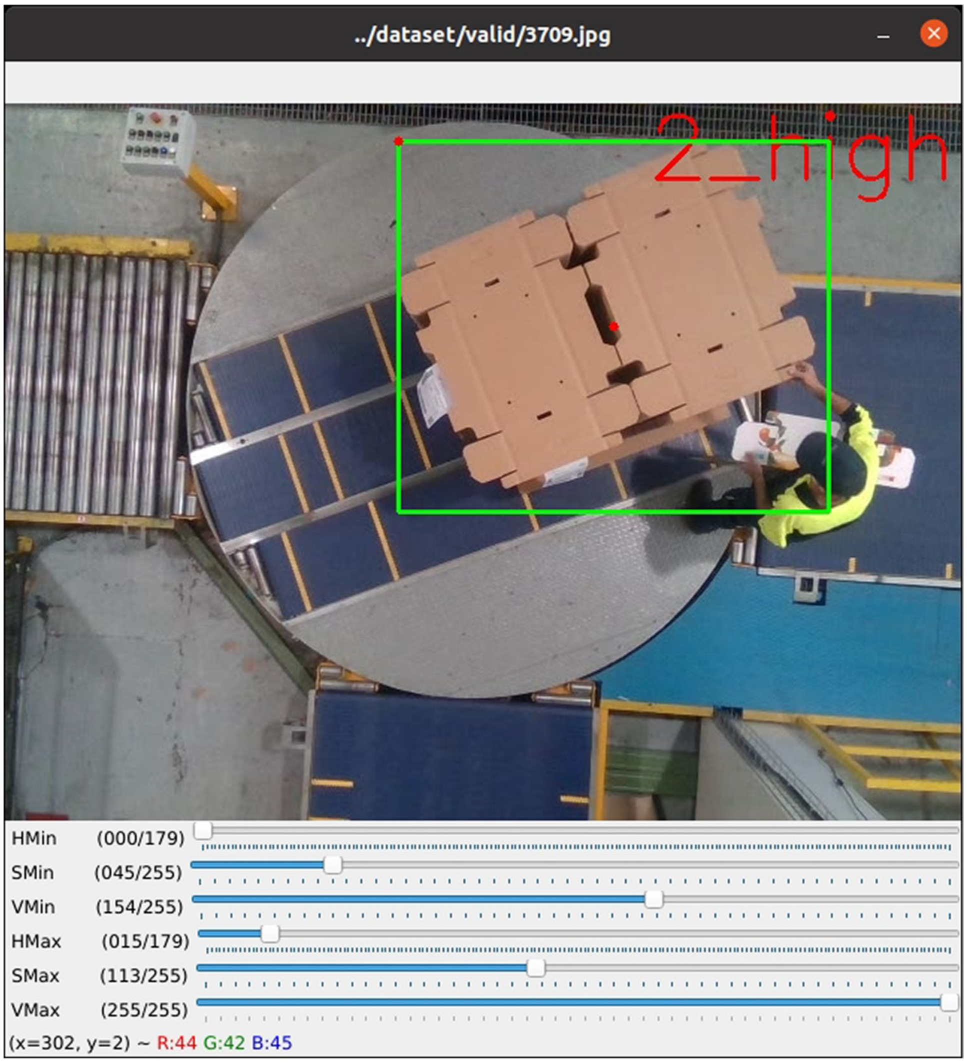
Fig. 10 Stack configuration of each pallet and samples in our dataset. (**a**, **b**, **e**) and (**f**) contain one type of packaging, while (**c**, **d**) and (**g**) contain more than one type of packaging, with different shaped boxes

In order to create our data collection, almost 6 h of video sequences were recorded in one of the Smurfit Kappa Company's factories. We captured image samples on different days and during different work shifts to obtain sufficient samples of each class in different light conditions and workloads. We then extracted 315,527 frames from these videos. However, we eliminated those frames in which there was no pallet or it did not appear entirely in the image. After removing these images, our final dataset contained 30,365 images. The total number of images was distributed in each class as follows: (a) 14.358%, (b) 14.355%, (c) 14.355%, (d) 14.349%, (e) 14.355%, (f) 13.875% and (g) 14.352%. Note that the dataset is well-balanced.

As a common step before training the detectors of our system, we divided the dataset into two sets: training (70%: 21,255 samples) and validation (30%: 9110 samples). Moreover, apart from the training dataset, we recorded other videos only to test the detectors and trackers. In the case of the detectors, the test set was made up of 100 randomly chosen image samples per class. In that of the trackers, the test set was created by grouping consecutive images of the pallet trajectory from its entrance onto the turntable to its exit.

Although the same dataset has been used for the three detectors selected from the state of the art, YOLOv5, Mask R-CNN and YOLOv5OBB calculate the pallet location in different ways. It is, therefore, necessary to annotate the images with bounding boxes for YOLOv5, pixel masks for Mask R-CNN and OBB for YOLOv5OBB. This is because the three detectors are NNs of different categories, as commented on previously: two of them are detection NNs and the other is a segmentation NN.

A GUI (Fig. 11) was used to speed up the annotation process, thus making it semi-automatic. This GUI uses the Hue-Saturation-Value (HSV) colour space in order to segment the cardboard boxes from the background of the image. The HSV colour space was chosen because it achieved the best results when compared to the other colour spaces available.



HMin (000/179)
SMin (045/255)
VMin (154/255)
HMax (015/179)
SMax (113/255)
VMax (255/255)
(x=302, y=2) ~ R:44 G:42 B:45

Fig. 11 GUI based on HSV colour space used to speed up the annotation process

The HSV filter was set using 3 sliders, each of which had a single channel with which to precisely adjust the filter. At the beginning of the execution, the GUI automatically sorted the images available in the input directory. It was, therefore, necessary only to select the desired class (type of pallet) using our keyboard, and establish the minimum and maximum HSV values required to annotate the pallet sample.

Once the cardboard box is located, it is annotated for each of the three chosen NN as can be seen in Fig. 12. On the one hand, for YOLOv5, we label the pallets with a rectangular bounding box, extracting the height, width and the coordinates of its centroid. This labelling method is less accurate because it contains background pixels, but it is faster than the other two labelling methods. For Mask R-CNN, we only label the pixels of the pallet, extracting the contour of it. This method is more accurate because it does not contain background pixels but this labelling method is more time-consuming. On the other hand, for YOLOv5OBB, we label the pallets with an oriented bounding box, extracting the coordinates of its corners. This method achieves a better trade-off between pixel accuracy and labelling time.

4.2 Training, validation and testing of pallet detectors

Before starting the training phase, we set the model and the training hyperparameters for YOLOv5, YOLOv5OBB and Mask R-CNN. As can be seen in Table 1, we chose the small size model for YOLOv5 and YOLOv5OBB and the Resnet50 backbone for Mask R-CNN, seeking a fast detection process but also maintaining a reliable detection. Models with a larger number of parameters were tested but they did not achieve a good trade-off between speed and quality of detection.

Moreover, the three NNs were trained following a transfer learning strategy with COCO pretrained weights [54] because this allows the models to converge faster than using other datasets or training from scratch. Finally, we applied rotation (±90°), horizontal and vertical flip to the training of YOLOv5 and YOLOv5OBB in order to improve their performance, while no transformation improved the training results of Mask R-CNN. Other transformations, such as zoom or a change of colours were discarded to avoid confusing the models of the different types of pallets.

Table 2 shows the training hyperparameters, from which we chose the optimal batch size values that our GPU can hold, an initial learning rate of 0.007 with weight decay of 0.0005, and an SGD with Momentum (0.94) as the optimiser. We tested several learning rates and optimizers, with 0.007 and SGD with Momentum being the optimum for our task. We used an NVIDIA A100 tensor core GPU with 40 GB memory for training and a GTX1070Ti for validation and testing, as mentioned in Section 3.3. The

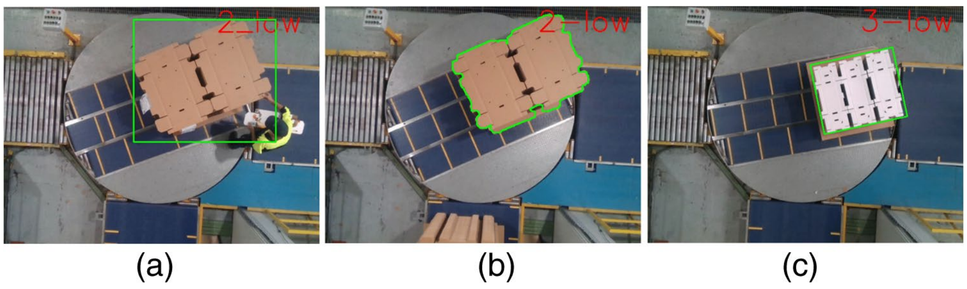


Fig. 12 Example of different annotated outputs. (**a**) YOLOv5. (**b**) Mask R-CNN. (**c**) YOLOv5OBB

Table 1 General information of each NN model

Neural Network	Model	Backbone	Pretrained weights	Data augmentation
YOLOv5	Small	CSPNet	COCO	Horizontal and vertical flip, ±90º rotation
Mask R-CNN	-	Resnet50	COCO	-
YOLOv5OBB	Small	CSPNet	COCO	Horizontal and vertical flip, ±90º rotation

Table 2 Training hyperparameters of each model

Neural Network	Optimizer	Learning rate	Batch size	Loss
YOLOv5	SGD	0.007	256	Binary Cross Entropy
Mask R-CNN	SGD	0.007	8	Binary Cross Entropy (masks) and Softmax (class)
YOLOv5OBB	SGD	0.01	256	Binary Cross Entropy

A100 capabilities facilitate an exploration of parameters that is more effective, whereas GTX1070Ti has a good response between inference time and energy consumption.

The models were trained by applying a divide and conquer strategy in order to achieve an almost optimal training and dataset. This strategy consists of first training our models with only one type of pallet, and then evaluating the performance of the trained model. We then realised that the performance values for our model varied depending on the spatial location of the pallet in the image, which was caused because this first version of our dataset was imbalanced in these cases. After balancing our dataset, we repeated the training process and the success rate improved. This process was iterated by adding more classes and more sample variability to each class in the dataset until it was completed according to the company's requirements.

In order to measure the performance of the pallet detection task, we always used the different AP_{IoU} computed as in Eq. (1) as a score, in which each IoU was obtained according to Eq. (2) by applying a specific threshold of between 0.5 and 0.95, where 1 is the maximum value. IoU measures the similarity between ground-truth and the detected region. The threshold determines the tolerance value that is allowed. We also calculated an average AP, as in Eq. (3), in which AP_{50} denotes an AP using a threshold of 0.5.

$$AP_{IoU} = \sum_{i \in \widetilde{r}} (r_{i+1} - r_i) \cdot \max_{\widetilde{r}:\widetilde{r} \geq r_i+1} \rho(\widetilde{r}) \tag{1}$$

$$IoU = \frac{area(gt \cap pd)}{area(gt \cup pd)} \tag{2}$$

$$AP_{50:95} = \frac{1}{10} * \sum_{k=0}^{9} AP_{50+5*k} \tag{3}$$

where r, p, $\check{r}$, gt and pd denote levels of recall, precision and recall values, ground truth and prediction bounding boxes.

We designed two types of experiments. First, we studied the behaviour of our system using the validation set. Later, we used the test set, which is different from the validation and training sets, to determine the ability to generalise pallet detection from samples never seen before. In addition, for each type of experiment, we evaluated the behaviour without distinguishing between *stack configuration* and *pallet type*, e.g. using the full validation set or the full test set.

The experimental results show that the three detectors obtain significant results for both the validation and the test set, as is shown in Tables 3 and 4. YOLOv5 achieves a higher $AP_{50:95}$ value with the validation set, which means that it better detects the samples that were used in the training phase. Nonetheless, YOLOv5OBB outperforms YOLOv5 and Mask R-CNN in the test set with unseen samples, proving its superior ability to generalise, as shown in the last row of Table 4.

Tables 5 and 6 show the $AP_{50:95}$ values of each detector with each of the 7 classes from the dataset. Although YOLOv5 obtains better results as regards detecting classes independently, YOLOv5OBB and Mask R-CNN detect with a lower standard deviation than YOLOv5 (Table 4), signifying that this last detector produces more variation in its detection scores. We are, therefore, of the opinion that if new pallet classes or new samples are added in the factory, the results will not be worse. In addition, YOLOv5 and YOLOv5OBB run in real time with a very small time inference in a basic GPU, while Mask R-CNN is very slow (Table 3). We, therefore, decided to discard Mask R-CNN owing to the impossibility of achieving an execution in real time at high production rates.

Although YOLOv5OBB outperforms YOLOv5 with the test set in terms of the ability to generalise, according to total $AP_{50:95}$ and its standard deviation, its performance is not as good as anticipated for some types of pallet but is still very close, as shown in Table 6. YOLOv5OBB is, on average, 11% better for c, e and g, while its performance is similar to YOLOv5 for a, d and f, despite being 2.76% lower. Only in the case of b is YOLOv5 clearly better than YOLOv5OBB.

For a more exhaustive comparison of both methods, we added a new metric that consists of calculating the percentage of detections that the detector loses during the trajectory of the pallet along the turntable. This percentage is truly important because it could cause a change in ID within the tracker, thus generating wrong product monitoring statistics, which would also cause errors in logistic management. To illustrate this metric, we have extracted a fragment from the test video. This fragment contains six pallets, each with a different class.

In order to obtain the percentage of lost detections, it is first necessary to count the total number of frames within this video fragment (4600 frames). The elimination of the frames in which there is no pallet is important if a fair value metric is to be attained, and the final number of frames is, therefore, 3070. We then count the number of detections generated by YOLOv5OBB and YOLOv5: 3040 and 2816, which correspond to a lost detection percentage of 1% and 8.3%, respectively. Note that this 8.3% of lost detections in YOLOv5 forces the tracker employed to change the ID, as can be seen on the right-hand side of Fig. 13. The most common cases in which YOLOv5 loses detections are when the pallets are not completely seen by the camera, at the entrances to the turntable or when the turntable is rotating. However, YOLOv5OBB is able to maintain a good performance in these cases. This is because YOLOv5OBB has learned to predict oriented bounding boxes that contain more pixels of the pallets and less of the background, signifying that when the pallets appear incomplete in the

Table 3 Detection results using full validation set

Detection algorithm	AP_{50}	AP_{75}	$AP_{50:95}$	Inference (ms)
YOLOv5	0.995	0.993	0.959	5.5
Mask R-CNN	0.999	0.995	0.887	274
YOLOv5OBB	0.955	0.960	0.914	8.3

Table 4 Detection results using full test set

Detection algorithm	AP_{50}	AP_{75}	$AP_{50:95}$
YOLOv5	0.986	0.984	0.865 ± 0.068
Mask R-CNN	0.984	0.984	0.853 ± 0.047
YOLOv5OBB	0.955	0.972	0.889 ± 0.049

Table 5 $AP_{50:95}$ results for each type of pallet on validation phase

Detection algorithm	a	b	c	d	e	f	g
YOLOv5	0.971	0.960	0.916	0.940	0.991	0.960	0.981
Mask R-CNN	0.887	0.902	0.928	0.880	0.850	0.885	0.878
YOLOv5OBB	0.943	0.891	0.982	0.849	0.905	0.954	0.876

Table 6 $AP_{50:95}$ results for each type of pallet on testing phase

Detection algorithm	a	b	c	d	e	f	g
YOLOv5	0.893	0.958	0.799	0.840	0.764	0.907	0.896
Mask R-CNN	0.776	0.889	0.879	0.796	0.866	0.885	0.880
YOLOv5OBB	0.867	0.873	0.961	0.812	0.900	0.878	0.934

image, YOLOv5OBB is still able to detect them. However, YOLOv5 has more difficulties in distinguishing the pallets from the background because the rectangular bounding boxes contain noise from the background of the image.

Finally, this metric makes it possible to select YOLOv5OBB as the Pallet Detector because its percentage of lost detections is almost negligible, thus helping the tracker to maintain the ID for each pallet sample.

4.3 Tracker comparison

Having chosen YOLOv5OBB as the default recommended detector for the three algorithms implemented in our system, we tested YOLOv5OBB and YOLOv5 in combination with the trackers described in Section 3.2. This experimentation, which consisted of comparing both detectors with each tracker, was divided into two kinds of experiments. First, a comparison of classical trackers (KCF, MIL, MOOSE) and deep learning–based trackers (DeepSORT, StrongSORT and ByteTrack) was carried out using YOLOv5 as a detector. Second, we compared both detectors with deep learning–based trackers in order to eventually select one of the trackers to accompany YOLOv5OBB in the final proposed system.

The default parameters of the trackers were obtained from experimentation with different benchmarks, as described in [41]. They also performed correctly when the detector was YOLOv5OBB owing to its small percentage of lost detections. Nonetheless, if the detector used was YOLOv5, deep learning–based trackers would require parameter tuning in order to ensure a correct detection following manufacturing (roller conveyor speed) and visual conditions (image appearance).

Table 7 shows the results of the parametric tuning for the main parameters. We chose a tolerance of 0.2 as the similarity value that we allowed to accept the identification of each pallet (MAX_DIST). An *IoU* threshold of 0.7 was used to compare detected bounding boxes in order to create a new pallet ID or maintain it (MAX_IOU_DISTANCE). We additionally used a maximum of 30 frames without detection as the default value for a limit to delete an ID from the tracker memory (MAX_AGE). Finally, we saved 3 successful detections (N_INIT) in order to create a new ID and 100 feature vectors from previous detections (NN_BUDGET) in order to identify a pallet as default values for these parameters. This parametric tuning indicated that YOLOv5 needed to increase the number of frames without detection in order to maintain the ID of the pallet because of the 8.3% of missed detections mentioned previously.

The results of both experiments were calculated in terms of *IoU*, as shown in Eq. (2) and its performance in FPS. Table 8 shows the results of the first experiment, in which deep learning–based trackers outperformed the classical methods in *IoU* with very similar FPS values. In the next experiment, it was, therefore, necessary only to compare the deep learning–based trackers. YOLOv5 was always used as the detector in order to obtain an honest comparison. All of the methods based on deep learning provided better results than did the classical trackers.

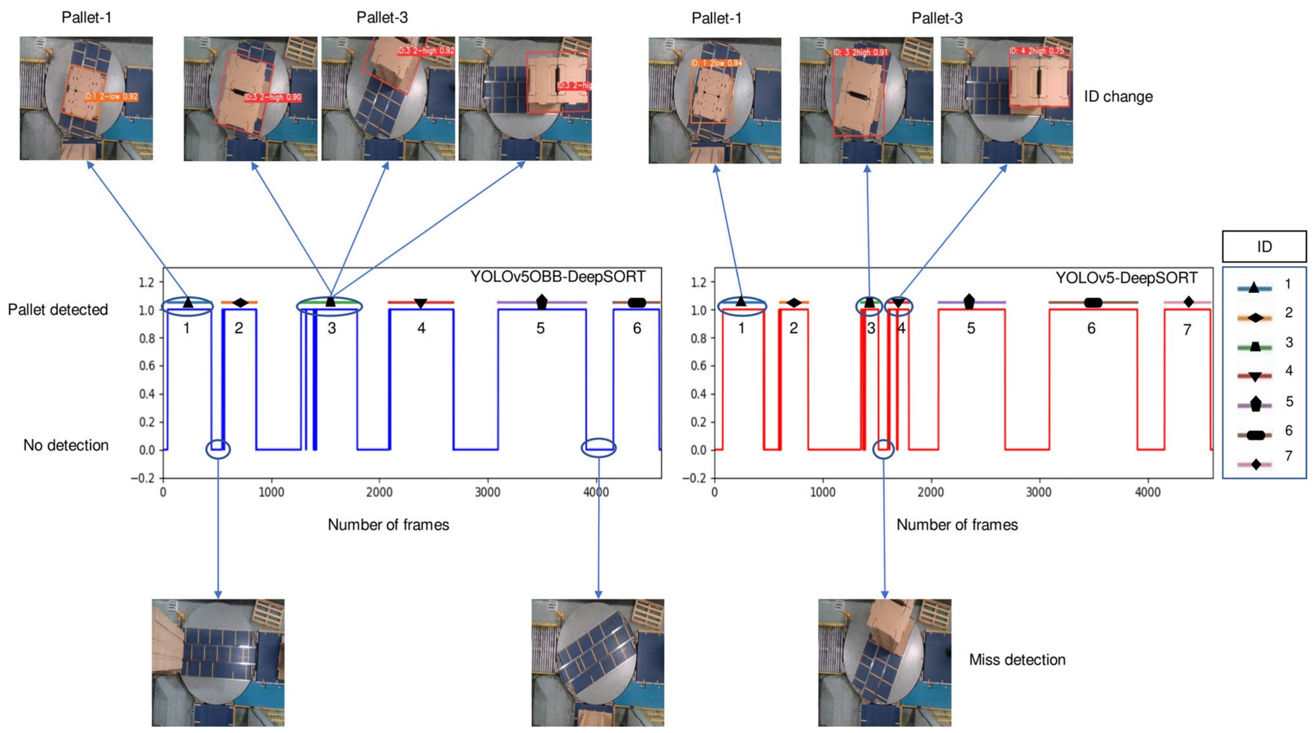


Fig. 13 Detections of YOLOv5OBB (left) and YOLOv5 (right) with the six pallets in a video fragment

More specifically, DeepSORT, StrongSORT and ByteTrack accomplished the same precision in the detection with a similar tracking time, although if we are rigorous, the last tracker was 3% faster than the SORT-based method.

Finally, we tested the best trackers but using YOLOv5OBB as the detector, since it provides more robust solutions, as shown in the previous section. As justified in the results shown in Table 9, any deep learning–based tracker of this nature would be adequate for our task, in which it is necessary only to track one object moving at the same time at a relatively slow speed, which was 0.08 m/s. However, DeepSORT was chosen because it achieved a higher FPS value when compared with the others when YOLOV5OBB was used as detector. Our system by default, therefore, recommends YOLOv5OBB as the Pallet Detector and DeepSORT as the Pallet Tracker. Note that our system loses 5.1% of speed as regards detection and tracking, considering that the camera captures 15 FPS. The proposal consequently ensures that if the roller conveyor speed is increased for production reasons, and more tracking speed is, therefore, required, the selected tracker will continue to provide a good performance. Since the camera can capture up to 30 FPS, if we consider the aforementioned loss then the speed of the tracker should be higher than 27 FPS in this assumption.

Table 7 Parametric tuning of the main parameters in deep learning-based trackers depending on the detector selected. Note that YOLOv5OBB-DeepSORT uses default values from the original DeepSORT paper

Parameters	YOLOv5OBB	YOLOv5
MAX_AGE	30	150
N_INIT	3	7
NN_BUDGET	100	10
MAX_IOU_DISTANCE	0.7	0.7
MAX_DIST	0.2	0.2

5 Conclusions

We propose a visual monitoring system with which to recognise and track different types of palletised packaging throughout the production line of a company pertaining to the corrugated cardboard industry. It is composed principally of two modules, a *Pallet detector* and a *Pallet tracker*. We have evaluated different algorithms to be used as backbones of our system, which is configurable, and our proposal comprises the combination YOLOv5OBB-DeepSORT because this provides the best results. Our system provides a GUI which allows human operators to manage the pallet monitoring process in the roller conveyor network of the factory. The operators do not, therefore, waste time on manual

Table 8 Classical trackers versus deep learning-based trackers using YOLOv5 as a detector in all cases

Tracking algorithm	IoU	FPS
MIL	0.774	14.111
KCF	0.781	14.017
MOSSE	0.749	14.120
DeepSORT	0.931	14.079
StrongSORT	0.931	14.038
ByteTrack	0.931	13.545

Table 9 Comparison of deep learning-based trackers using YOLOv5OBB as a detector

Tracking algorithm	*IoU*	*FPS*
YOLOv5OBB-DeepSORT	0.930	14.234
YOLOv5OBB-StrongSORT	0.930	14.053
YOLOv5OBB-ByteTrack	0.930	14.031

tasks such as picking up the top box of the pallet in order to discover the pallet type or counting the number of pallets of each type. The statistic production values guide the operators to anticipate situations in which the conveyors next to the storage areas may collapse. A demo of our system can be seen from the link to electronic supplementary material.

Moreover, in this paper, we solve the problem of ID changes, which are produced by missed detections of YOLOv5, by training and deploying a YOLOv5OBB model. This model is able to generate OBB, which does not lose any detections at crucial moments such as when the pallets disappear from the image but are still on the turntable. Nevertheless, we are aware that our system may have some limitations, the main one being the necessity to annotate new data and re-train our models when clients order new types of pallets (new stack configuration) or when the camera type or position is changed. However, this issue is very common in deep learning research. Furthermore, as we do not contemplate occlusions or night shifts in our dataset, the hit rate of our system would probably drop significantly. This might, for example, occur when a human operator is manipulating the palletised packages or when the light conditions are very different because the pipeline is working at night, since we have assumed working hours from morning to evening.

Future research lines include collecting statistical feedback in order to evaluate our system during a longer period of time, implementing a closed loop training process so as to automate the re-training of our models and obtaining industrial cameras that are more robust in industrial environments.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s00170-023-11098-6.

Acknowledgements We would like to thank Smurfit Kappa Iberoamericana S.A. (http://www.smurfitkappa.com/), Partida de Canastell, B-393, 03690 San Vicente del Raspeig, Alicante (Spain). Smurfit Kappa Alicante provided know-how regarding corrugated cardboard and packaging.

Author contribution J.C., F.F. and P.G contributed to the design and implementation of the research, to the analysis of the results and to the writing of the manuscript.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. Research work was partially funded by the private project (SMURFITKAPPA-21), supported by Smurfit Kappa Iberoamericana S.A. and University of Alicante.

Data availability The datasets analysed during the current study are not publicly available due confidential company data by Smurfit Kappa Alicante Iberoamericana S.A., but are available from Institutional Repository of the University of Alicante (pablo.gil@ua.es) on reasonable request.

Declarations

Consent for publication The authors give the Publisher the permission to publish the Work in this journal.

Conflict of interest The authors declare no competing interests.

References

1. Kocí V (2019) Comparisons of environmental impacts between wood and plastic transport pallets. Sci Total Environ 686:514–528. https://doi.org/10.1016/j.scitotenv.2019.05.472
2. Custodio L, Machado R (2020) Flexible automated warehouse: a literature review and an innovative framework. Int J Adv Manuf Technol 106:533–558. https://doi.org/10.1007/s00170-019-04588-z
3. Castaño-Amoros J, Fuentes F, Gil P (2022) Visual monitoring intelligent system for cardboard packaging lines. 27th int. conf. on emerging technologies and factory automation 1–8. https://doi.org/10.1109/ETFA52439.2022.9921712
4. Smith ML, Smith LN, Hansen MF (2021) The quiet revolution in machine vision - a state-of-the-art survey paper, including historical review, perspectives, and future directions. Comput Ind 130:103472. https://doi.org/10.1016/j.compind.2021.103472
5. Borstell H (2018) A short survey of image processing in logistics. 11th int. doctoral student workshop on logistics 43–46. https://doi.org/10.13140/RG.2.2.24664.39688

6. Shishira S, Rao V, Sudarsan SD (2019) Proximity contours: vision based detection and tracking of objects in manufacturing plants using industrial control systems. 17th Int Conf Ind Inform 1:1021–1026. https://doi.org/10.1109/INDIN41052.2019.8972032
7. Dörr L, Brandt F, Naumann A, Pouls M (2021) Tetrapacknet: four-corner-based object detection in logistics use-cases. 43th dagm german conf. on pattern recognition 545–558. https://doi.org/10.1007/978-3-030-92659-5_35
8. Law H, Deng J (2020) Cornernet: detecting objects as paired keypoints. Int J Comput Vis 128:642–656. https://doi.org/10.1007/s11263-019-01204-1
9. Dörr L, Brandt F, Pouls M, Naumann A (2020) Fully-automated packaging structure recognition in logistics environments. 25th Int Conf Emerg Technol Fact Autom 1:526–533. https://doi.org/10.1109/ETFA46521.2020.9212152
10. Mohamed IS, Capitanelli A, Mastrogiovanni F, Rovetta S, Zaccaria R (2020) Detection, localisation and tracking of pallets using machine learning techniques and 2d range data. Neural Comput Appl 32(13):8811–8828. https://doi.org/10.1007/s00521-019-04352-0
11. Ren S, He K, Girshick R, Sun J (2017) Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell 39(6):1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031
12. Iinuma R, Hori Y, Onoyama H, Fukao T, Kubo Y (2021) Pallet detection and estimation for fork insertion with RGB-D camera. Int Conf Mechatron Autom 854–859. https://doi.org/10.1109/ICMA52036.2021.9512641
13. Kita Y, Takase R, Komuro T, Kato N, Kita N (2022) Localization of pallets on shelves in a warehouse using a wide-angle camera. 17th Int Conf Adv Motion Control 187–194. https://doi.org/10.1109/AMC51637.2022.9729302
14. Bohacs G, Rozsa Z, Bertalan B (2021) Mono camera based pallet detection and pose estimation for automated guided vehicles. 10th int. conf. on logistics, informatics and service sciences 1–11. Springer. https://doi.org/10.1007/978-981-33-4359-7_1
15. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: unified, real-time object detection. Conf Comput Vis Pattern Recognit 779–788. https://doi.org/10.1109/CVPR.2016.91
16. Zaccaria M, Monica R, Aleotti J (2020) A comparison of deep learning models for pallet detection in industrial warehouses. 16th Int Conf Intel Comput Commun Process 417–422. https://doi.org/10.1109/ICCP51029.2020.9266168
17. Bochkovskiy A, Wang C-Y, Liao H-YM (2020) Yolov4: optimal speed and accuracy of object detection. CoRR, abs/2004.10934, https://doi.org/10.48550/ARXIV.2004.10934
18. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu C-Y, Berg AC (2016) Ssd: Single shot multibox detector. Eur Conf Comput Vis 21–37. https://doi.org/10.1007/978-3-319-46448-0_2
19. Molter B, Fottner J (2018) Real-time pallet localization with 3d camera technology for forklifts in logistic environments. Int Conf Serv Oper Logist Inform 297–302. https://doi.org/10.1109/SOLI.2018.8476740
20. Xiao J, Lu H, Zhang L, Zhang J (2017) Pallet recognition and localization using an RGB-D camera. Int J Adv Robot Syst 14 (6). https://doi.org/10.1177/1729881417737799
21. Hong R-J, Li Y-R, Hung M-H, Chang J-W, Hung JC (2022) Integrating object detection and semantic segmentation into automated pallet forking and picking system in AGV. Int Conf Front Comput Lect Notes Electr Eng 827:121–129. https://doi.org/10.1007/978-981-16-8052-6_13
22. Karim MM, Doell D, Lingard R, Yin Z, Leu MC, Qin R (2019) A region-based deep learning algorithm for detecting and tracking objects in manufacturing plants. Procedia Manuf 39:168–177. https://doi.org/10.1016/j.promfg.2020.01.289
23. El-sayed ME, Youssef AW, Shehata OM, Shihata LA, Azab E (2022) Computer vision for package tracking on omnidirectional wheeled conveyor: case study. Eng Appl Artif Intell 116:105438. https://doi.org/10.1016/j.engappai.2022.105438
24. Cho J, Kang S, Kim K (2022) Real-time precise object segmentation using a pixel-wise coarse-fine method with deep learning for automated manufacturing. J Manuf Syst 62:114–123. https://doi.org/10.1016/j.jmsy.2021.11.004
25. Pan H, Shi Y, LeiWang XZFX (2022) Fast identification model for coal and gangue based on the improved tiny yolov3. J Real-Time Image Proc 19:687–701. https://doi.org/10.1007/s11554-022-01215-1
26. Saran G, Ganguly A, Tripathi V, Kumar AA, Gigie A, Bhaumik C, Chakravarty T (2022) Multi-modal imaging-based foreign particle detection system on coal conveyor belt. Trans Indian Inst Metals 75:2231–2240. https://doi.org/10.1007/s12666-021-02492-3
27. Banús N, Boada I, Xiberta P, Toldrà P, Bustins N (2021) Deep learning for the quality control of thermoforming food packages. Sci Rep 11:21887. https://doi.org/10.1038/s41598-021-01254-x
28. Li Y, Niu Y, Liu Y, Zheng L, Wang Z, Zhe W (2021) Computer vision based conveyor belt congestion recognition in logistics industrial parks. 26th Int Conf Emerg Technol Fact Autom 1–8. https://doi.org/10.1109/ETFA45728.2021.9613245
29. Liu L, Ouyang W, Wang X, Fieguth P, Chen J, LiuPietik¨ainen XM (2020) Deep learning for generic object detection: a survey. Int J Comput Vis 128(2):261–318. https://doi.org/10.1007/s11263-019-01247-4
30. Zaidi SSA, Ansari MS, Aslam A, Kanwal N, Asghar M, Lee B (2022) A survey of modern deep learning based object detection models. Digit Signal Process 126:103514. https://doi.org/10.1016/j.dsp.2022.103514
31. Li K, Wan G, Cheng G, Meng L, Han J (2020) Object detection in optical remote sensing images: a survey and a new benchmark. ISPRS J Photogramm Remote Sens 159:296–307. https://doi.org/10.1016/j.isprsjprs.2019.11.023
32 Minaee S, Boykov YY, Porikli F, Plaza AJ, Kehtarnavaz N, Terzopoulos D (2021) Image segmentation using deep learning: a survey. IEEE Trans Pattern Anal Mach Intell 44(7):3523–3542. https://doi.org/10.1109/TPAMI.2021.3059968
33. Liu S, Liu D, Srivastava G, Polap D, Wo´zniak M (2021) Overview and methods of correlation filter algorithms in object tracking. Complex Intell Syst 7(4):1895–1917. https://doi.org/10.1007/s40747-020-00161-4
34. Wu Y, Lim J, Yang M-H (2013) Online object tracking: a benchmark. Conf Comput Vis Pattern Recog 2411–2418. https://doi.org/10.1109/CVPR.2013.312
35. Marvasti-Zadeh SM, Cheng L, Ghanei-Yakhdan H, Kasaei S (2022) Deep learning for visual tracking: a comprehensive survey. IEEE Trans Intell Transp Syst 23(5):3943–3968. https://doi.org/10.1109/TITS.2020.3046478
36. Jocher G (2021) Yolov5. Retrieved from https://github.com/ultralytics/yolov5
37. Kaixuan H (2022) Yolov5obb. Retrieved from https://github.com/hukaixuan19970627/yolov5obb
38. Yang X, Yan J (2020) Arbitrary-oriented object detection with circular smooth label. Eur Conf Comput Vision-Lect Notes Comput Sci 12353:677–694. https://doi.org/10.1007/978-3-030-58598-3_40
39. He K, Gkioxari G, Doll´ar P, Girshick R (2017) Mask R-CNN. Int Conf Comput Vis 2961–2969. https://doi.org/10.1109/ICCV.2017.322
40 Henriques JF, Caseiro R, Martins P, Batista J (2014) High-speed tracking with kernelized correlation filters. IEEE Trans Pattern

Anal Mach Intell 37(3):583–596. https://doi.org/10.1109/TPAMI.2014.2345390

41. Babenko B, Yang M-H, Belongie S (2010) Robust object tracking with online multiple instance learning. IEEE Trans Pattern Anal Mach Intell 33(8):1619–1632. https://doi.org/10.1109/TPAMI.2010.226
42. Bolme DS, Beveridge JR, Draper BA, Lui YM (2010) Visual object tracking using adaptive correlation filters. Conf Comput Vis Pattern Recogn 2544–2550. https://doi.org/10.1109/CVPR.2010.5539960
43. Wojke N, Bewley A, Paulus D (2017) Simple online and realtime tracking with a deep association metric. Int Conf Image Process 3645– 3649. https://doi.org/10.1109/ICIP.2017.8296962
44. Du Y, Song Y, Yang B, Zhao Y (2022) Strongsort: make deepsort great again. CoRR, abs/2202.13514. https://doi.org/10.48550/arXiv.2202.13514
45. Zhang Y, Sun P, Jiang Y, Yu D, Yuan Z, Luo P et al (2022) Bytetrack: multi-object tracking by associating every detection box,13682, 1–21. https://doi.org/10.1007/978-3-031-20047-2_1
46. Wang C-Y, Liao H-YM, Wu Y-H, Chen P-Y, Hsieh J-W, Yeh I-H (2020) Cspnet: a new backbone that can enhance learning capability of CNN. Conf Comput Vis Pattern Recognit Workshops 390–391. https://doi.org/10.1109/CVPRW50498.2020.00203
47. Liu S, Qi L, Qin H, Shi J, Jia J (2018) Path aggregation network for instance segmentation. Conf Comput Vis Pattern Recog 8759–8768. https://doi.org/10.1109/CVPR.2018.00913
48. Kuhn HW (1955) The Hungarian method for the assignment problem. Naval Res Logist Q 2(1–2):83–97. https://doi.org/10.1002/nav.3800020109
49. Luo H, Jiang W, Gu Y, Liu F, Liao X, Lai S, Gu J (2020) A strong baseline and batch normalization neck for deep person re-identification. IEEE Trans Multimedia 22(10):2597–2609. https://doi.org/10.1109/TMM.2019.2958756
50. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. Conf Comput Vis Pattern Recogn 770–778. https://doi.org/10.1109/CVPR.2016.90
51 Wang Z, Zheng L, Liu Y, Li Y, Wang S (2020) Towards real-time multiobject tracking. Eur Conf Comput Vis-Lect Notes Comput Sci 12356:107–122. https://doi.org/10.1007/978-3-030-58621-8_7
52. Evangelidis GD, Psarakis EZ (2008) Parametric image alignment using enhanced correlation coefficient maximization. IEEE Trans Pattern Anal Mach Intell 30(10):1858–1865. https://doi.org/10.1109/TPAMI.2008.113
53. Du Y, Wan J, Zhao Y, Zhang B, Tong Z, Dong J (2021) Giaotracker: a comprehensive framework for mcmot with global information and optimizing strategies in visdrone 2021. Int Conf Comput Vision Workshops 2809–2819. https://doi.org/10.1109/ICCVW54120.2021.00315
54. Lin T-Y, Maire M, Belongie S, Hays J, Perona P, Ramanan D et al (2014) Microsoft coco: common objects in context. Eur Conf Comput Vis-Lect Notes Comput Sci 8693:740–755. Springer. https://doi.org/10.1007/978-3-319-10602-1_48