Contents lists available at ScienceDirect

CIRP Journal of Manufacturing Science and Technology

journal homepage: www.elsevier.com/locate/cirpj

Full-length article

Monitoring manufacturing systems using AI: A method based on a digital factory twin to train CNNs on synthetic data

Marcello Urgo [a,*], Walter Terkaj [b], Gabriele Simonetti [c]

[a] *Politecnico di Milano, Department of Mechanical Engineering, Milan, Italy*
[b] *National Research Council, Institute of Intelligent Industrial Technologies and Systems for Advanced Manufacturing, Milan, Italy*
[c] *Business Integration Partners S.p.A., Italy*

ARTICLE INFO

Keywords:
Digital twin
Deep learning
Synthetic datasets
Production monitoring

ABSTRACT

Modern cyber–physical production systems provide advanced solutions to enhance factory throughput and efficiency. However, monitoring its behaviour and performance becomes challenging as the complexity of a manufacturing system increases. Artificial Intelligence (AI) provides techniques to manage not only decision-making tasks but also to support monitoring. The integration of AI into a factory can be facilitated by a reliable Digital Twin (DT) that enables knowledge-based and data-driven approaches. While computer vision and convolutional neural networks (CNNs) are crucial for monitoring production systems, the need for extensive training data hinders their adoption in real factories. The proposed methodology leverages the Digital Twin of a factory to generate labelled synthetic data for training CNN-based object detection models. Regarding their position and state, the focus is on monitoring entities in manufacturing systems, such as parts, components, fixtures, and tools. This approach reduces the need for large training datasets and enables training when the actual system is unavailable. The trained CNN model is evaluated in various scenarios, with a real case study involving an industrial pilot plant for repairing and recycling Printed Circuit Boards (PCBs).

1. Introduction

Monitoring plays a fundamental role in modern manufacturing systems, supporting process optimization, quality control, predictive maintenance, and inventory management [1–3]. Cyber–Physical Production Systems (CPSS) [4] enable different monitoring solutions that are increasingly enhanced by automation, flexibility, and digitization. Industry 4.0 and the large availability of affordable sensors have enabled the acquisition of real-time data to support automation and control [5], helping to ground decisions on updated and reliable data [6].

After the disruptive rise of Artificial Intelligence (AI) and its successful application in various fields, the manufacturing industry is also adopting AI to manage both complex decision-making tasks and monitoring by analysing data acquired from manufacturing systems [7]. The integration of AI in the industry is strictly linked to the availability of a Digital Twin (DT) [8] of the factory, i.e., a digital replica of the real factory that includes not only geometric characteristics but also products, processes, resources and their integrated behavioural features [9,10]. The availability of a Digital Factory Twin provides comprehensive and structured information that could easily support the implementation of knowledge-based and data-driven approaches, especially AI.

Digital Twin and AI techniques are effectively applied in manufacturing to support decision-making, design and control within factories, from single production modules to whole company networks. In particular, these techniques combined with image data are becoming more and more relevant to address tasks such as monitoring processes, tracking objects, visual inspections, and handling parts.

Computer vision approaches are already crucial for monitoring production systems and processes; among these, convolutional neural networks (CNN) are one of the most promising tools for image analysis. Nevertheless, the need for a massive amount of data to train this class of models constitutes a significant barrier towards their adoption in real factories.

This work proposes a methodology that employs the Digital Twin of a factory to generate synthetic data to support the training of CNN-based object detection models. The focus lies in monitoring entities (parts, components, fixtures, tools, etc.) in manufacturing systems in terms of their position and state, as these data serve various objectives, such as tracking products or general entities, supporting and optimize handling operations, monitoring the progress of production activities, enabling advanced safety and ergonomics analyses.

In addition to reducing the burden of large training datasets, the proposed approach also enables training CNN models when the actual

* Corresponding author.
E-mail address: marcello.urgo@polimi.it (M. Urgo).

https://doi.org/10.1016/j.cirpj.2024.03.005
Received 8 August 2023; Received in revised form 28 January 2024; Accepted 19 March 2024
Available online 27 March 2024

system is non-existent or inaccessible. The proposed approach involves an automated workflow that automates the generation of synthetic data and annotations leveraging the DT of a factory. The CNN model is trained on these data, and its performance is tested in various scenarios. The proposed approach is tested on a real case represented by a re-demanufacturing pilot plant [11] dedicated to handling, repairing and recycling Printed Circuit Boards (PCBs), thus entailing the need to locate pallets, PCBs, and transporters.

The article is structured as follows. Section 2 presents related works in the literature. Section 3 defines the goals and requirements of the proposed methodology to generate data sets and train a CNN for object recognition, described in Section 4. The use case presented in Section 5 was employed to test the methodology, in particular checking the generation of synthetic data (Section 6), evaluating the performance of a CNN model trained on synthetic data (Section 7), and assessing the application to monitoring transport and handling operations (Section 8). Finally, the conclusions are drawn in Section 9.

2. State of the art

Sensor-based monitoring is a widely-used approach in the industry, benefiting from the reliability and benefiting from of tags (e.g., based on RFID technology) for tracking and monitoring products, components, and tools. However, an RFID tag is required for each monitored object, leading to costs proportional to the number of objects to be tracked.

In this category of applications, machine vision offers several advantages compared to sensor-based tracking [12]. It is possible to avoid the labelling of a high number of objects, reducing the investment cost when tracking a large number of items. Furthermore, machine vision can also overcome tag limitations, e.g., in high-temperature environments.

The following subsections delve into the state of the art of technologies that constitute the pillars of the proposed solution, namely computer vision (Section 2.1), Convolutional Neural Networks and object detection (Section 2.2), Synthetic datasets (Section 2.3), and Digital Factory Twin (Section 2.4).

2.1. Machine vision

Machine vision employs cameras and computational capabilities to execute vision-related tasks, such as object detection, position identification, human operator pose recognition, identification of specific situations, etc. Thanks to the continuous and rapid advancement of enabling technologies, machine vision approaches are gaining relevance and are diffusely adopted in the industry [13,14].

Machine vision in industry is exploited for various purposes, including visual inspection of products and machinery [15], tracking parts and components in production stations [12], monitoring manufacturing processes [16], and supporting the grasping and handling of objects by robots [17].

Implementing machine vision technology has significantly enhanced productivity and overall quality management, leading to a considerable competitive advantage if properly exploited [18]. Furthermore, machine vision has proven successful in various applications, such as production control in the food industry, quality inspection in textiles, and defect identification and classification in PCB manufacturing [19].

2.2. Convolutional neural networks and object detection

Detecting specific objects in images is a fundamental aspect of industrial monitoring systems. Specifically, the capability to function effectively in demanding environments characterized by diverse scenarios and operating conditions (e.g., light, dust, etc.) is a challenge for numerous machine vision tasks. In recent years, Deep Learning (DL) approaches, particularly deep convolutional neural network architectures, have attained impressive results and are emerging as one of the breakthrough technologies with potential applications in the industry [20].

Convolutional Neural Networks (CNNs) are a specific class of neural networks engineered to process images and extract distinct features via convolutions. CNNs comprise multiple layers of different types: convolutional, non-linear, pooling, and fully connected layers [21]. CNNs take a two-dimensional image as input and produce an identification of one of the possible functionalities in object recognition, i.e., the association of the image to a possible class if a single object is present (*image classification*) or the identification of potential objects within the image using bounding box, associated labels, and a measure of the confidence for the inference (*object detection*). These approaches have been successfully exploited for a wide range of objects [22], and for human beings [23,24]

Numerous deep convolutional architectures have demonstrated very good performance, e.g., RetinaNet [25], Faster R-CNNs [26], Yolo [27], SSD [28,29], and R-FCN [30]. These models are undergoing rigorous testing, validation, and showcasing their efficacy in supporting a wide range of applications, e.g., faults diagnosis [15], process control [16], medical imaging analysis [31], etc.

Nevertheless, a shared characteristic among all the approaches mentioned above is the need to train models with millions of parameters, demanding vast quantities of training data. Developing these datasets involves collecting and labelling images, which relies on human input and becomes time-consuming, error-prone, and costly. This likely constitutes the primary constraint to their widespread adoption in real-world applications, particularly within the industry.

2.3. Training CNNs on synthetic images

Training data can be partially or entirely generated synthetically to mitigate the challenge of acquiring the necessary training data or to address situations where it might even be impossible, e.g., if the objects to be monitored or the operating do not exist. This involves using computer-based tools such as CAD software or virtual reality environments to generate images and their corresponding labels. Thus, the CNN training is carried out with synthetically generated images, which are later applied to the real world. Training models with synthetic data is a very appealing approach. It not only streamlines the collection of extensive data and the associated annotation, but it also provides the ability to generate practically limitless training data. Furthermore, it is possible to customize the synthetic data generation settings, thereby enhancing the diversity of training datasets. For instance, one can select the objects to be included in the images, their positioning, the surrounding environments, and other relevant factors.

Different strategies and approaches have been developed to implement and enhance the effectiveness of training neural networks using synthetic data. The characteristics of the dataset are fundamental, as the performance of the detection heavily relies on the size, quality and richness of the dataset itself.

A straightforward approach has also been proposed [29] involving generating images by cutting pictures of objects to be detected and pasting them onto random images, thereby providing a variable background. A similar approach has been used in [32–34], focusing on indoor detection of everyday objects, in particular in kitchen environments.

Transfer learning is a common approach to mitigate the need of extremely large datasets in training, even when using synthetically generated data. Starting from pre-trained object recognition models (e.g., Faster-RCNN, R-FCN, Mask-RCNN), the layers responsible for feature extraction are kept frozen, assuming that what is learned on real images can also be exploited on synthetic images. On the contrary, the last layers are replaced with new ones considering the new classes of objects to be identified and subsequently trained on synthetic datasets.

Thus, the final training phase focuses on the parameters associated with a small subset of the model, requiring a significantly smaller amount of data. This approach has consistently demonstrated superior performance compared to training on synthetic images alone [35, 36]. Nevertheless, transferring the capabilities of object detection approaches between different domains and, even more critically, between synthetic and real images remains a significant challenge that needs to be tackled [33,37]. Alternative methods have been proposed to address this challenge, such as domain adaptation [38] and domain randomization [37]. The latter approach has been further enhanced by combining real background images and photo-realistic rendering [39].

Recent advancements in this field also aim to explore the exclusive use of synthetic training data, thus reducing reliance on real images. The approach proposed in [40] focuses on generating synthetic images comprising two main components: 3D background and foreground objects. The foreground objects are targeted for detection, whereas background objects are densely and randomly arranged to construct a realistic background image. Randomized illumination, blur, and noise are also incorporated to increase the variety of the dataset further. The authors demonstrate that this approach makes trained models robust to environmental changes and can even outperform models trained on real data [40]. This methodology is the foundation for SynthDet [41], an open-source project that demonstrates an end-to-end object detection pipeline using synthetic image data, leveraging the Unity Perception Package [42].

2.4. Digital factory twin

A Digital Twin (DT) is intended as the coupling of a real system, its digital counterpart, a set of models and algorithms to support decisions, a continuous flow of data coming from the field and a control bus to actuate the decisions in the real system [43].

A digital twin is a relevant enabling technology for realizing the paradigm of smart manufacturing and Industry 4.0 [44], involving an entirely digitized, complex system that affects all units and classes in a factory. Indeed, the widespread use of digital technologies in factories generates and requires vast amounts of data. A review of DT for factories is presented in the article by Terkaj et al. [45], introducing the concept of Digital Factory Twin (DFT) and reviewing literature contributions that focus on factory, system, and process levels, while highlighting current challenges.

DT have several possible applications in manufacturing, such as product design and production engineering [46], monitoring and optimization of manufacturing processes [47–49], analysis of tools [50], design of production systems [10], maintenance [51], ergonomics [52], system control [53]. DT is also a fundamental enabler for AI integration in industrial applications [54], such as aerospace, autonomous driving, smart city transportation, and smart manufacturing [55]. Specifically, applications in smart manufacturing include fault diagnosis methods [56], lifecycle management of complex equipment [57], additive manufacturing [58], manufacturing system reconfiguration and optimization [59], task learning, movement prediction, risk reduction [60]. A detailed factory DT, which comprises realistic, textured 3D models of parts, components, equipment and buildings, enables users to interact with these models in a Virtual Reality (VR) environment. A VR representation of a factory proves to be a powerful enabler for generating synthetic datasets. The combination of VR technology and realistic 3D models contributes significantly to creating diverse and comprehensive synthetic datasets, enhancing the potential of AI models for industrial applications.

A solution for recognizing parts in a working environment for robotics applications has been presented in [17]. The authors realized the CNN training on images of parts rendered using a 3D CAD and placed on a background of randomly arranged and distorted images of an industrial environment. The results are promising, although based on limited experiments involving single ring-shaped parts with relatively simpler geometric features than the wide variety of industrial parts and components. A similar methodology was presented in [61] where CNNs were employed to segment and detect aluminium profiles after training on images generated using a CAD system. The outcome was a model exhibiting high accuracy and deemed potentially suitable for the specified industrial application. However, the robustness of the model has not been thoroughly tested in various realistic environmental scenarios, such as changes in lighting, background, and other factors that may occur in real-world settings.

A framework for developing DT-driven Machine Learning models was presented in [62], addressing a use case by training vision-based recognition of parts' orientation thanks to DT models.

Another potential industrial application of these approaches has been proposed in [63], addressing the detection of objects in egocentric vision. This concept assumes the point of view of a human operator to support manual operations, such as assembly tasks. The authors demonstrate how it is possible to improve the training of a CNN by using a large synthetic dataset rather than relying on a smaller dataset of real images.

Recently, this methodology has been applied to the civil engineering domain, specifically to aid in building scene understanding (BSU) [64]. In this context, a Building Information Model (BIM) was leveraged to generate photorealistic images from various perspectives of an indoor environment. The use of synthetic images resulted in promising outcomes for training deep-learning models.

3. Problem statement

The problem addressed in this work revolves around monitoring factory objects by identifying their locations within a manufacturing system. This is achieved by harnessing the availability of images or videos captured within the factory environment.

The goal is to develop an effective object detection and localization method to support monitoring and automation processes in industrial contexts. This method aims to fulfil the following functional requirements (FR):

FR1 Identification of specific objects in the factory environment, such as parts, components, tools, fixtures, transporters, etc..
FR2 Detection of the position of the tracked objects in the captured image/videos to facilitate the identification of their absolute or relative location in the working area.
FR3 Identification of specific macroscopic characteristics of the tracked objects, for example, determining whether they are empty or full, etc.

In addition, the following non-functional requirements must be satisfied:

NFR1 Real-time capability. The method must operate in real-time (or near real-time) to provide timely responses.
NFR2 Accuracy. The method should offer adequate accuracy to ensure reliable and precise object detection and localization, meeting the demands of relevant industrial applications.
NFR3 Flexibility. The method must be flexible enough to track multiple different objects.

Furthermore, machine vision and, specifically, convolutional neural networks are the technologies of choice for addressing object detection and localization, because the rapid development and increasing adoption of CNNs have made them highly suitable for industrial applications.

Machine vision based on neural networks asks for large amounts of training data. Thus, the following technical requirements must be addressed:

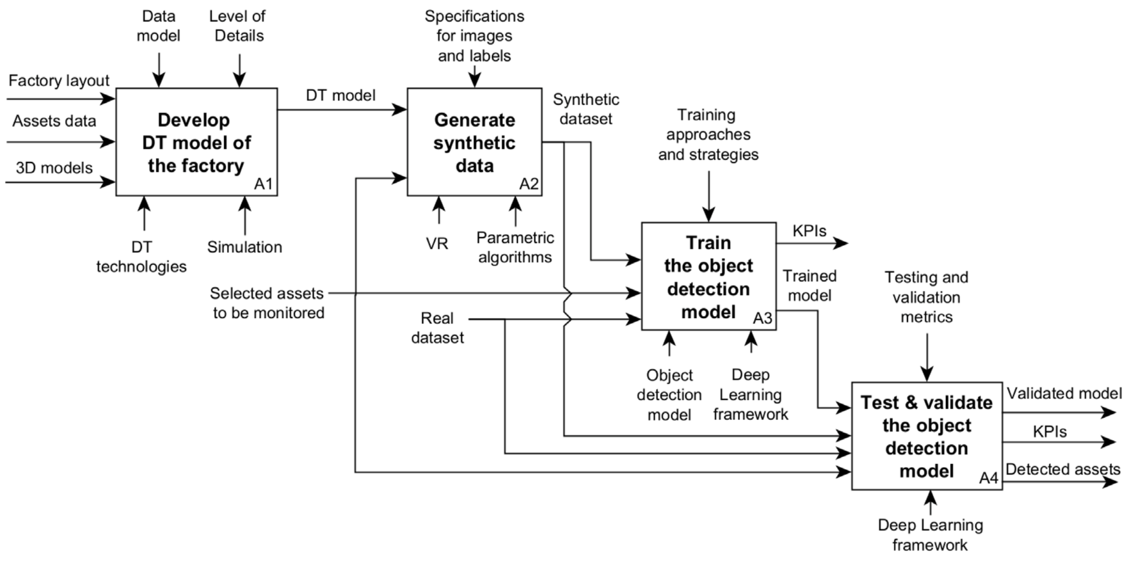

Data model
Level of Details
Factory layout
Assets data
3D models
Develop DT model of the factory A1
DT technologies
Simulation
DT model
Specifications for images and labels
Generate synthetic data A2
VR
Parametric algorithms
Synthetic dataset
Training approaches and strategies
Selected assets to be monitored
Real dataset
Train the object detection model A3
KPIs
Trained model
Object detection model
Deep Learning framework
Testing and validation metrics
Test & validate the object detection model A4
Validated model
KPIs
Detected assets
Deep Learning framework

Fig. 1. IDEF0 diagram representing the Solution Framework.

TR1 Data collection. The acquisition of large sets of data, specifically images, is essential for training the CNN models 2.2. These datasets need to be diverse and representative of the industrial environment.

TR2 Data annotation. The annotation of data (i.e., labelling the objects in the images) is needed to create ground-truth data for training the models. Human annotation is time-consuming and can be prone to errors.

TR3 Model training. Training deep learning models, especially large CNNs, demands specific hardware resources (e.g., powerful GPUs or specialized hardware accelerators), procedures, and time.

4. Solution framework

A comprehensive solution framework (see Fig. 1) has been developed to address the requirements outlined in the previous section while leveraging the use of synthetic images to train an object detection model.

At the core of the proposed approach is the Digital Twin (DT) model of the factory to be monitored. This model is developed through activity *A1*, incorporating data and knowledge related to the assets. The input data are formalized according to a specific data model and level of detail (Section 4.1).

The second phase of the approach (Activity *A2*) focuses on generating synthetic data leveraging the DT model (Section 4.2). The relevant factory assets to be monitored (e.g. objects flowing through the factory like parts, fixtures, pallets, or moving components of equipment) must be selected and identified in alignment with the DT model. The synthetic data are generated through the use of VR and algorithms configured based on the specifications for images and labels. This configuration enables the flexibility of employing various alternative frameworks and models for object detection.

Hence, an object detection model is selected and trained (Activity *A3*) by leveraging the functionalities of existing deep learning frameworks (Section 4.3). The object detection model receives synthetic data as input, along with a potential set of real images (i.e. *real dataset*) associated with the selected assets to be monitored.

Finally, the trained model is tested and validated (Activity *A4*) on both synthetic and real data, according to specific metrics designed for object detection approaches (Section 4.4). If the performance metrics (*KPIs*) meet the acceptable criteria, the results of the validated model will be used to detect the selected assets and support monitoring.

4.1. Development of the digital twin model of the factory

As anticipated in Section 2.4, the Digital Twin of a factory can support various business processes, including the generation of synthetic data to train AI models. In this case, the synthetic data consists of factory images, thus the DT must include the 3D representation of physical assets.

Several commercial and non-commercial methodologies and tools are available to develop the DT of a factory integrating the 3D visualization aspect. Nevertheless, most solutions generally involve the following common steps:

- The digital model of the factory is instantiated. In particular, the modelling of production resources, part types, and processes is crucial in this context.
- Generation of 3D models of factory assets. Typically, 3D CAD files are simplified and exported to exchange formats. The level of rendering quality can be adjusted based on the specific requirements for realism.
- Setup of a VR scene that is based on the digital model of the factory and the 3D models of the assets. In addition, cameras (i.e., point of view) and lights must be defined.
- Visualization of the factory in the VR environment.

In this work, a workflow based on free tools [65] is adopted to ensure data reuse and maximize accessibility to a wide range of potential users. The digital model of the factory is instantiated as an ontology model, leveraging existing meta-models designed for industrial applications [10]. The ontology instantiation can be streamlined by employing intermediate steps that involve commonly used data formats (e.g. spreadsheets, JSON files) and available data sources.

The 3D models are exported to .gLTF format, which has gained popularity as a widely adopted standard. This format enhances the realism of 3D models by incorporating visual details in the form of materials and textures.

The adopted VR environment is the prototype web application VEB.js (Virtual Environment based on Babylon.js).[1] VEB.js is reconfigurable and model-driven since the VR scene can be automatically generated from an ontology model or a JSON file. Its main functionalities encompass 3D navigation, reconfiguration of the layout, animation of assets, MQTT connection, (semi-)automatic generation of screenshots, etc.

[1] https://virtualfactory.gitbook.io/vlft/tools/vebjs

4.2. Generation of synthetic datasets

The CNN training for object detection relies on datasets containing labelled images. Each image in the dataset is coupled with information regarding the objects present in the scene, including the class of the objects and their corresponding bounding boxes.

As described in the previous subsection (Section 4.1), a DT of the factory, enhanced with a VR interface, serves as an ideal enabler for generating synthetic images. The camera of the VR environment determines the point of view within the virtual scene based on its position and orientation. This capability enables the capture of screenshots based on the specific perspective defined by the camera configuration. In addition, by using the underlying geometric model, it becomes feasible to automatically identify the visible assets in the scene and, more effectively, determine the coordinates of their bounding boxes projected on the viewport. This information can be directly extracted from the rendering engine, streamlining the process of generating synthetic images with accurate object annotations. Thus, for each relevant visible asset, the annotation includes the identifier of the asset, its class, and the size and position of the bounding box (cf. requirement *TR2*).

Different options are possible for the generation of the annotated (labelled) images:

- *Manual*. The user navigates through the VR scene and manually generates the labelled image for the selected perspective.
- *Semi-automatic*. While the user navigates through the VR scene, a timed capture is executed at regular intervals. This process ensures the generation of labelled images from various perspectives without requiring continuous manual intervention.
- *Automatic*. The VR camera is controlled by defining a series of sequential positions or by moving the camera on a sphere frontier around a target of interest (e.g., a selected factory asset) in its centre [13]. Thus, by specifying the parameters of the sphere (i.e., centre and radius) and the pattern to be followed for navigating (i.e., solid angle to be covered and horizontal and vertical step angles), the movement of the camera and the generation of the images can be entirely automated.

The semi-automatic and automatic options are highly recommended for generating large datasets, drastically reducing the time and cost associated with creating the training dataset (cf. requirement *TR1*). Furthermore this approach provides the capability of generating dataset with the necessary level of variety, which is essential for effectively training CNN object detection models.

The adopted VR environment VEB.js (Section 4.1) offers the necessary relevant functionalities for the automatic, semi-automatic, and manual generation of synthetic datasets in the form of images. These datasets can be exported either in JSON format (annotated version) or simple PNG format (non-annotated version).

4.3. Training the object detection model

The training of a CNN model for object detection conducted using synthetic data offers the advantage of being feasible even when real images are unavailable, e.g., in scenarios where the factory does not yet exist or is not readily accessible. However, models trained solely on synthetic data, while often beneficial and effective in various scenarios [40], may encounter challenges related to domain transfer [36]. To address this concern and improve performance on real images, a hybrid approach has been adopted, exploiting a second training stage where real images are incorporated into the training dataset. This aims to enhance the adaptability and robustness of the model when used in real scenarios.

In relation to the training approach and strategy, the initial training step on synthetic data has been operated according to a transfer learning scheme [35]. A pre-trained model is used and the weights associated with the first set of layers in the CNN are kept frozen during the training phase. By freezing these weights, their behaviour remains unchanged as they are responsible for *feature extraction* from the data. Hence, the training process only affects the last layers of the neural networks, which are responsible for the identification of specific objects.

To initiate the training process, it is essential to determine hyper-parameters defining the number of epochs, batch size, and learning rate have to be decided. These parameters significantly influence the effectiveness and duration of the training process. Their values are determined through a series of preliminary sets of experiments, aimed at minimizing the loss function used during the training [66].

After the initial training phase, when the loss function reaches a steady state, the first layers of the CNN can be unfrozen. Subsequently, a further training step is conducted on the same dataset to adjust the parameters of the already trained model.

Finally, a fine-tuning training step is also operated on real images. This iterative process of fine-tuning with synthetic and real data helps create a more reliable and accurate object detection model that can successfully operate in industrial applications.

The training of the CNN model should be conducted using an established deep learning framework providing a wide range of functionalities. Herein, the TensorFlow[2] framework provided by Google was employed, together with the Python module Keras.[3] Furthermore, the cloud platform Google Colaboratory[4] was used, providing the possibility to run Python code in the cloud using Python notebooks.

The last aspect to be addressed is the selection of the CNN model to be trained. The decision to operate on a cloud platform influences the decision on the CNN model for object detection and the resolution of the images to be processed, considering constraints related to computational load, memory, and running time. The landscape of AI-based object detection models is diverse and continuously evolving. Widely used frameworks like TensorFlow and PyTorch offer comprehensive libraries and tools for developing and deploying these models. Collections like COCO (Common Objects in Context) and ImageNet provide extensive datasets crucial for training and validating object detection algorithms. Additionally, cutting-edge object detection models are readily available. Presently, it is possible to select a specific object detection model and access repositories offering pre-trained versions ready for used. Examples are the TAO toolkit by NVIDIA,[5] the Tensorflow Object Detection API,[6] Model Zoo.[7] These models can detect multiple classes of objects in images, thus providing the required degree of flexibility (cfr. requirement *NFR3*).

As the focus of the current manuscript is on the definition of a general framework approach based on a DT model of a factory, the selection of the optimal object detection model is not a central aspect, particularly considering their availability through the mentioned libraries and repositories. To demonstrate the feasibility of the proposed framework, the YOLO-v3 model has been used. Although it may not be the most recent object detection model available, it provides good accuracy and a inference speed of less than 50ms [67], making it suitable for real-time object detection in videos at up to 20 fps (cfr. requirement *NFR1*). This choice takes into account the limitations introduced by the use of a cloud-based implementation. The selection of the YOLO-v3 model, together with the use of RGB images with a resolution of 416×416 pixels, is in line with the objective of achieving an optimal trade-off between inference speed and accuracy [67] (cfr. requirement *TR3*).

[2] https://www.tensorflow.org

[3] https://keras.io

[4] https://colab.research.google.com

[5] https://developer.nvidia.com/tao-toolkit

[6] https://github.com/tensorflow/models/tree/master/research/object_detection

[7] https://modelzoo.co

4.4. Testing and validation

Once the object detection model has been trained on synthetic or a combination of synthetic-real data, its performance is evaluated by operating the detection of objects on images. The evaluation is carried out using the COCO evaluation metrics [68], which align with standard practices in the field. The primary indicator of the model effectiveness in object identification is the *intersection-over-union* (IoU), which evaluates the agreement between a ground truth bounding box B_a and a predicted bounding box B_p. The IoU measures the overlap between the two bounding boxes, and it is widely used to assess the accuracy and quality of object detection models. The IoU is defined as:

$$IoU_{(B_p,B_a)} = \frac{B_p \cap B_a}{B_p \cup B_a} \in [0,1] \tag{1}$$

Upon the definition of a threshold T, the calculation of the IoU can define:

- $IoU > T$: the detection is valid (TP: True Positive);
- $IoU \leq T$: the detection is not valid (FP: False Positive);
- $IoU = 0$: it could be both a False Negative (FN) or a False Positive (FP).

Building upon these values, two additional metrics can be defined: precision (P), which assesses the ability of the model to accurately detect objects, and recall (R), which evaluates the capability of the model to correctly identify if an object is present in the image.

Furthermore, by plotting the P values as a function of the R values, obtained over the entire dataset, an average precision AP can be computed. By calculating the average precision AP for all classes of objects to be identified, the mean average precision mAP is derived. This comprehensive metric will be used to evaluate the performance of the trained models.

Multiple definitions exist for the mean average precision mAP. Specifically, the COCO challenge [68] proposes the definition in Eq. (2), i.e., the average of the values mAPs obtained by varying the threshold T for the IoU from 0.5 to 0.95 in steps of 0.05, yielding ten different values.

$$mAP_{COCO} = \frac{mAP_{(IoU=0.5:0.05:0.95)}}{10} \tag{2}$$

As the threshold T increases, the mAP is expected to decrease. Thus, the values of mAP_{COCO} will generally be lower than the ones calculated for $IoU = 0.5$. In this work, we will use both the metrics: mAP_{COCO} and $mAP_{IoU=0.5}$.

The evaluation of the approach will be carried out under two distinct operating conditions. Firstly, a test on synthetic images will be executed to verify the success of training on synthetic data. The trained model will be employed to identify objects in synthetic images sampled from the original dataset that has not been used for the training. Secondly, since the ultimate goal is to enable monitoring in a real factory setting, a separate test will be performed using real images. The models will be evaluated on real images to assess their capability to perform detections in a real-world environment.

Certainly, models trained using the mixed training scheme will not be tested on synthetic images. On the other hand, models that were exclusively trained on synthetic data will undergo evaluation in both scenarios. This approach aims to understand if a possible poor performance is primarily a result of the training on synthetic images or the subsequent domain transfer phase when dealing with real-world images.

5. Use case and digital twin

The proposed approach has been tested on a use case represented by a research laboratory, the Re- and De-manufacturing (RDM) Pilot Plant. This plant was designed to manage the end-of-life of mechatronics, in particular printed circuit boards (PCBs), implementing a circular manufacturing approach to support product disassembly, remanufacturing and recycling of materials [11]. The laboratory was installed at the CNR-STIIMA institute in Milan, Italy (Fig. 2(a)).

The plant consists of three stations connected by a modular transport system handling the pallets and routing them to the stations:

1. Hybrid disassembly of mechatronic components taking advantage of human–robot cooperation.
2. Testing and remanufacturing of printed circuit boards (PCBs).
3. Mechanical pre-treatment with low environmental impact (i.e. shredding and materials separation processes) for the recovery of high-value and critical raw materials from PCBs.

.

A key element of the pilot plant is the transport and handling of the pallets, taking advantage of a modular and reconfigurable system consisting of fifteen conveyor modules (Fig. 3(a)).

Pallets can host PCBs of different sizes through an adaptable fixture (Fig. 3(b)).

The conveyor modules are equipped with sensors, but they lack the capability to uniquely identify the position of the pallets.

The flexibility and reconfigurability of the transport system make a vision-based monitoring system a valuable tool for tracking the pallets and their contents. Therefore, the use case focuses on recognizing the positions of pallets and PCBs in the transportation system.

5.1. Dataset of real images

Photographs of the real system were captured to compose the datasets of real images. These images were taken using a Nikon D3100 camera paired with an 18–55 mm lens (Fig. 4).

When collecting real images, various scenarios were set up for the placement of pallets and PCBs:

- Black background: 414 images. Real objects on a black background (Figs. 5(a) and 5(b)).
- White background: 82 images. Real objects on a white background (Figs. 5(c) and 5(d)).
- Random objects: 32 images. Additional objects (not targeted for recognition) were placed against the black background to test the capability of the model to exclusively detect the objects of interest (Figs. 5(e) and 5(f)).
- Context: 195 images. These images feature the pallets and PCBs placed on the transport system, which represents the real operational context where monitoring is needed (Figs. 6(a) and 6(b)).

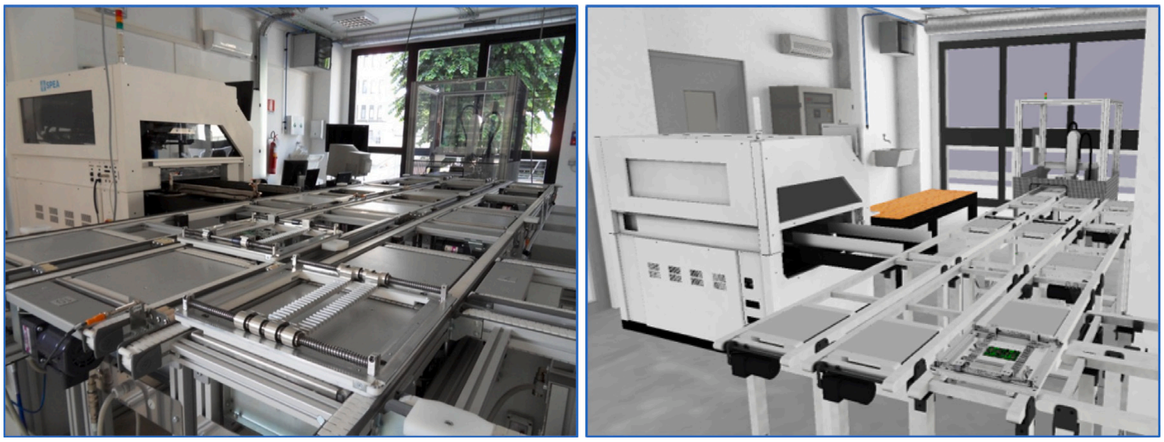
A summary of the real dataset and its contents is reported in Table 1. The dataset comprises a total of 722 images, encompassing 671 instances of pallets and 541 instances of PCBs. It is worth noting that transport modules are not included in this count, as capturing images of them in isolation was not feasible. The entire dataset has been partitioned into two sub-datasets, namely *DR0* and *DR1*, which serve specific phases of training and testing (see Section 7). All the images have undergone post-processed to enhance contrast and highlight object contrast and highlight object edges for better visibility. The real datasets and their labelling have been made available in a data repository [69].

5.2. Digital twin of the pilot plant

The digital twin of the RDM pilot plant was developed as described in the first phase of the approach (cf. Section 4.1) and relevant data are available online, including the ontology model,[8] the VR scene,[9] and the

[8] https://difactory.github.io/repository/ontoeng/VL/RdmPlant.ttl

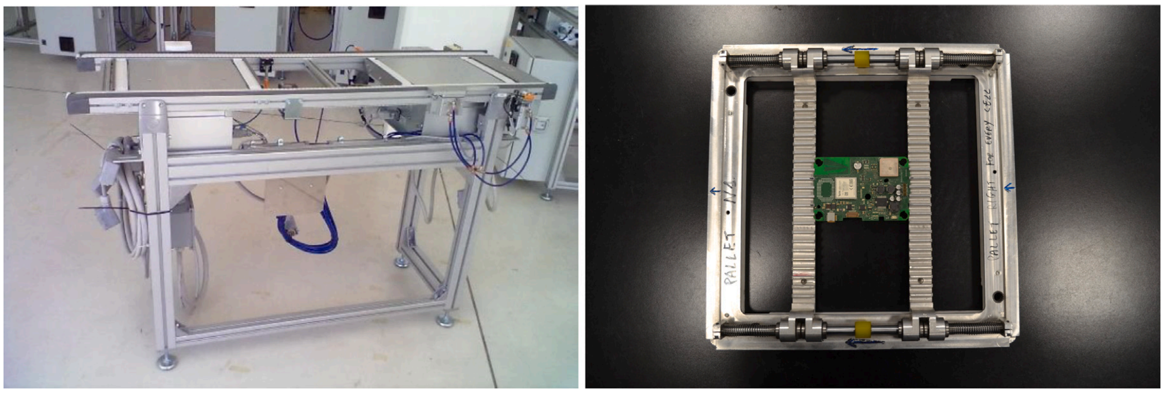
[9] https://difactory.github.io/repository/scenes/VL/RdmPlant.json



(a) Real RDM pilot plant.

(b) Digital Twin of the RDM pilot plant.

Fig. 2. Real and Digital versions of the RDM pilot plant.



(a) A conveyor module.

(b) A pallet loaded with a PCB.

Fig. 3. Conveyor module and pallet of the RDM pilot plant.

Table 1
Dataset acquired from the real plant.

Dataset	# images	# pallet instances	# PCB instances	# total instances
DR0	522	486	393	879
DR1	200	185	148	333
Total	**722**	**671**	**541**	**1212**

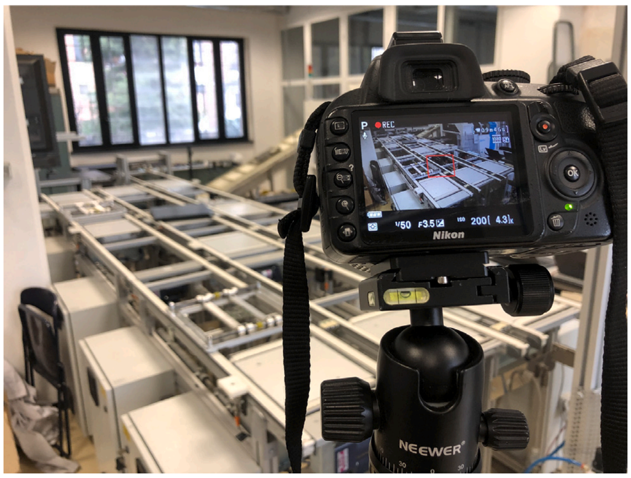


Fig. 4. Set-up for the collection of real images in the plant.

3D models.[10] The digital twin can be visualized[11] with the VEB.js web application (Fig. 2(b)) and supports the automatic generation of the synthetic datasets that are described in the next subsection.

6. Testing the generation of synthetic datasets

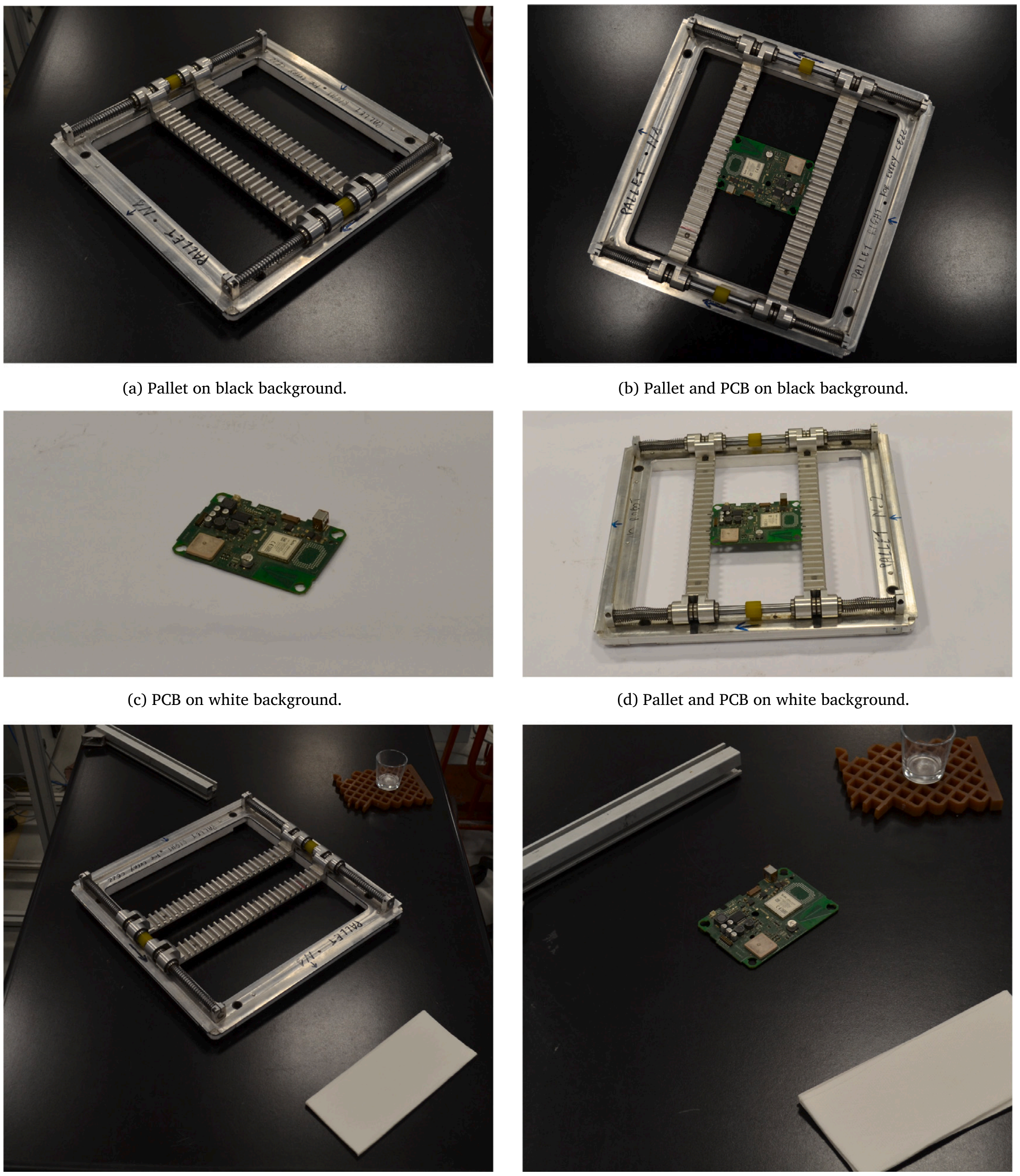
This section focuses on the generation of synthetic datasets (i.e. the second phase of the approach, cf. Section 4.2), providing details about the generation process tailored to the specific use case (Section 6.1) and testing the quality of synthetic data. Specifically, the outcomes of automatic labelling of the images (Section 6.2) and the outcomes of manual labelling (Section 6.3) are compared in Section 6.4.

6.1. *Synthetic dataset*

The generation of synthetic datasets (Section 4.2) takes advantage of the Digital Twin of the system and its VR representation in VEB.js

[10] https://github.com/difactory/repository/tree/main/models/VL/RDM

[11] https://difactory.github.io/DF/scenes/VL/RdmPlant.html



(a) Pallet on black background.

(b) Pallet and PCB on black background.

(c) PCB on white background.

(d) Pallet and PCB on white background.

(e) Pallet and random objects on black background.

(f) PCB and random objects on black background.

Fig. 5. Examples of images containing the objects to be identified (e.g., pallets, PCBs) on white and black backgrounds and in different scenarios (e.g, a single object, multiple objects, or together with other objects).

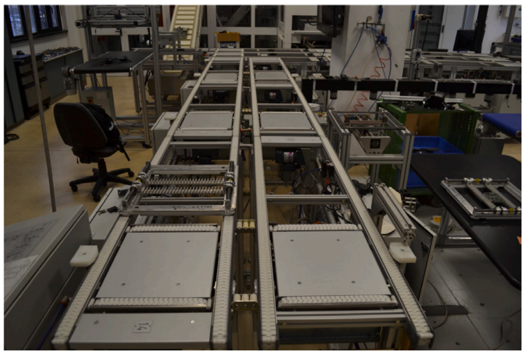
(Section 5.2). This VR environment facilitates the automatic generation and labelling of data.

Regarding the use case, the generation of synthetic data is limited to the transport modules and two specific classes of objects present in the system: pallets and PCBs. The process of generating synthetic data follows the following steps:

- The digital twin of the RDM pilot plant is updated, potentially including a subset of assets that are relevant to the training dataset. This specific subset is visualized within the corresponding VR scene using VEB.js.
- The virtual representations of assets are placed either on a neutral background (white, black or grey), or within the virtual representation of the building where the plant is actually located.
- Shadows are enabled to enhance the realism.
- PCBs are considered both in isolation and when loaded onto pallets.
- Lighting conditions, including tone and intensity, can be adjusted.
- A single asset is chosen at random from the set of relevant assets available in the VR scene.
- Screenshots are captured by moving the viewpoint along a spherical pattern, consistently oriented towards the selected asset.

Some examples of the synthetic images generated for the training dataset are shown in Figs. 7 and 8.

Seven datasets ($DS1, \dots, DS7$) have been generated by varying the number and type of objects (pallet, PCB, and conveyor) in the scene, considering different contexts for asset placement, and incorporating


(a) Pallet on the transport system.

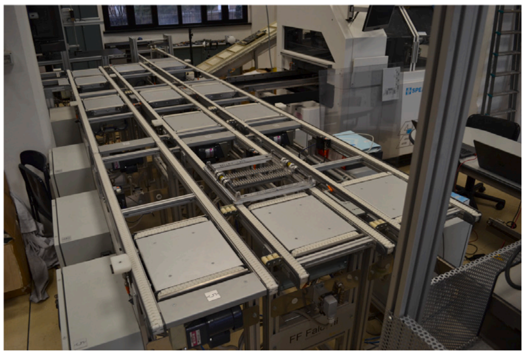

(b) Pallet on the transport system.

Fig. 6. Context.

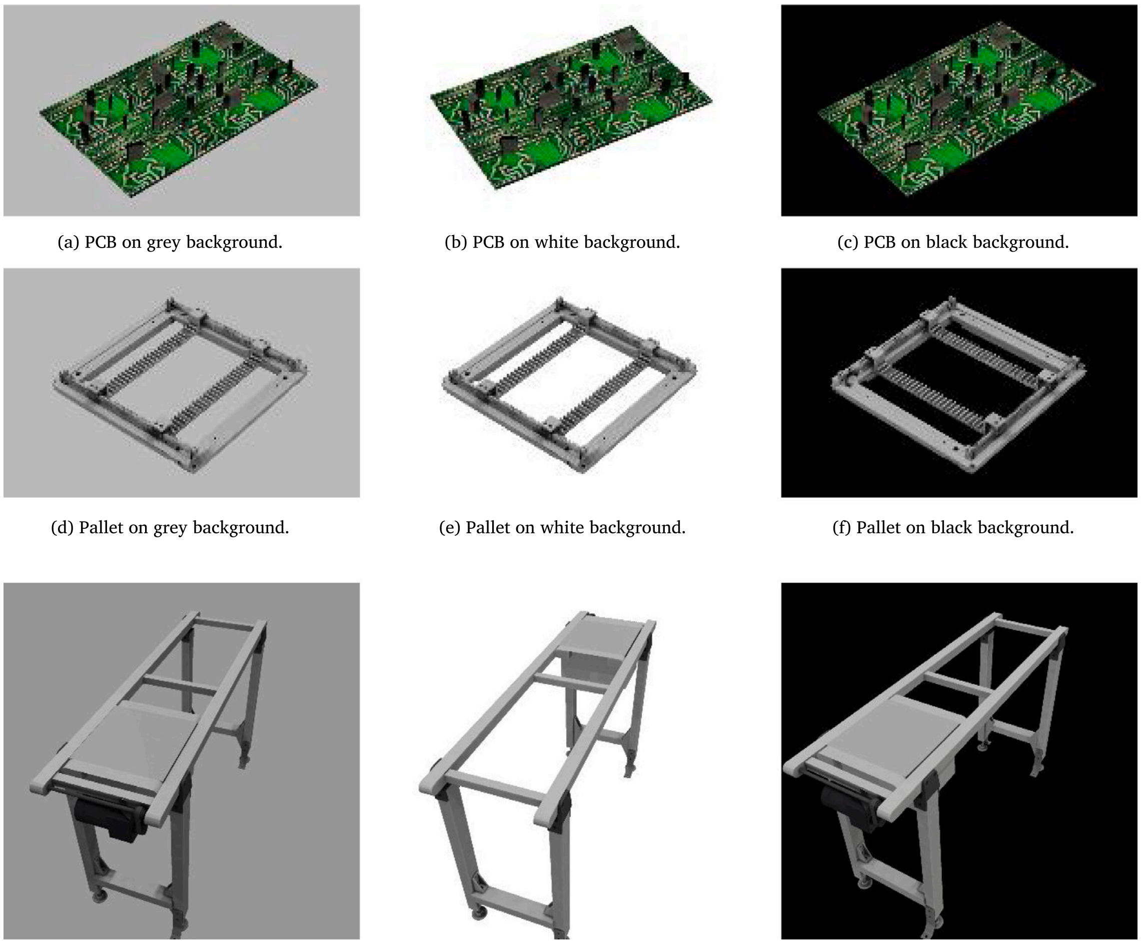

(a) PCB on grey background.
(b) PCB on white background.
(c) PCB on black background.
(d) Pallet on grey background.
(e) Pallet on white background.
(f) Pallet on black background.
(g) Conveyor on grey background.
(h) Conveyor on white background.
(i) Conveyor on black background.

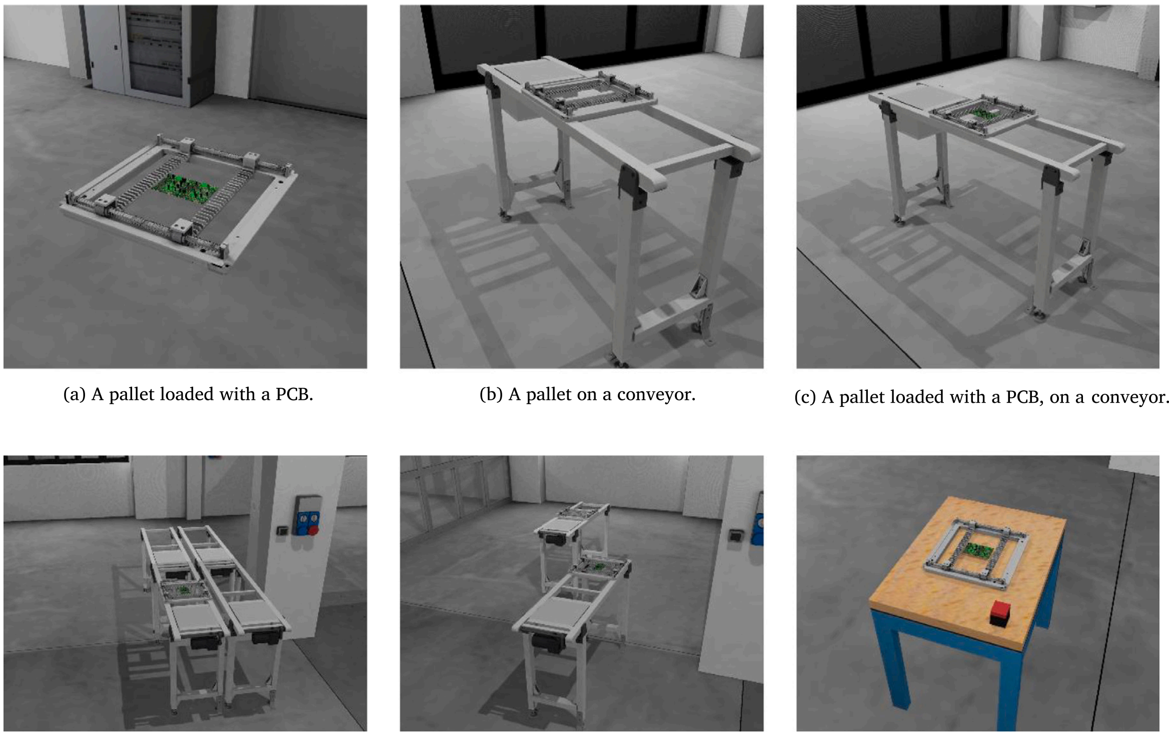
Fig. 7. Examples of synthetically generated images of PCBs, pallets and conveyors on neutral backgrounds.

randomization elements. Table 2 summarizes the characteristics of the datasets, while Table 3 reports the total number of generated images and instances of the different classes of objects.

The first dataset *DS1* was generated to assess the effectiveness of automatic labelling based on the information available in the digital twin and the VR scene. This was achieved by comparing the automatically generated labels with the manually annotated ones.

Additionally, three datasets (*DS2*, *DS3*, and *DS4*) were generated, each involving different combinations of object types within the scene. These combinations included a single conveyor, a single pallet, a single PCB, a conveyor with a pallet on it, a pallet with a PCB on it, and a conveyor with a pallet loaded with a PCB. The three datasets have a neutral light/dark grey, white or black background, for DS2, DS3 and DS4 respectively. For these datasets, the position and lighting have been randomized, except for the one with a black background, where different lighting has not been used, due to the minimal impact on the rendering.

The fifth dataset *DS5* contains images with the tree classes of objects placed in the realistic context, i.e., the virtual representation of the pilot plant. Dataset *DS6* was formulated to replicate the real configuration of


(a) A pallet loaded with a PCB. (b) A pallet on a conveyor. (c) A pallet loaded with a PCB, on a conveyor.

(d) A pallet and multiple conveyors. (e) Multiple pallets and conveyors. (f) A pallet loaded with a PCB on a table.

Fig. 8. Synthetically generated data within the realistic virtual representation of the building.

Table 2
Characteristics of the synthetically generated datasets.

Dataset	#pallets	#PCBs	#conveyors	Randomization	Context
DS1	0 – 1	0 – 1	0 – 6	Position	Real environment
DS2	0 – 1	0 – 1	0 – 1	Position, light	Grey background
DS3	0 – 1	0 – 1	0 – 1	Position, light	White background
DS4	0 – 1	0 – 1	0 – 1	Position	Black background
DS5	0 – 1	0 – 1	1	Position, light	Real environment
DS6	1 – 2	1 – 2	2 – 6	Position	Real environment
DS7	0 – 1	0 – 1	0	Position	Real environment

Table 3
Synthetically generated datasets.

Dataset	# images	# pallet instances	# PCB instances	# conveyor instances	# other instances	# total instances
DS1	4960	4640	2784	8960	0	16 384
DS2	10 176	8896	3840	2560	0	15 296
DS3	7616	6336	3840	2560	0	12 736
DS4	6400	5120	3840	2560	0	11 520
DS5	5304	4244	3764	2544	0	10 552
DS6	3448	3448	3448	18 136	0	25 032
DS7	2880	1920	1920	0	5760	9600

the pilot plant by including multiple instances of conveyors, pallets and PCBs.

Finally, dataset *DS7* was defined considering a single empty pallet, a single pallet loaded with a PCB and a single PCB on a table, together with other objects that do not need to be detected.

All the synthetically generated datasets, except *DS1*, and their labelling, have been made available in a data repository [69].

6.2. Automatic labelling

The phase of the approach described in Section 4.2 involves the automatic labelling of synthetically generated images. To assess the performance and quality of this automatic labelling, dataset *DS1* is partitioned into nine sub-datasets. Each sub-dataset contains images with varying numbers of objects to be labelled. The images are classified based on the degree of complexity of the rendered scene, categorized as low, medium and high complexity.

The time required for generating annotations is presented in Table 4. It takes about 19.15 s to label a single image while labelling a single instance within an image takes around 5.79 s on average. It is worth noting that the time needed to label images increases with the complexity of the scene. Indeed, the labelling process relies on defining bounding boxes for objects in the images, which is achieved through the rendering engine. Thus, scenes with more objects (higher complexity) take longer to render and label.

Furthermore, the time required for labelling the images is more closely related to the number of images rather than the number of

Table 4
Automatic labelling empirical data.

Dataset	Scene complexity	# images	# instances	Total time	Mean time [t/100 inst.]	Mean time [t/100 imgs]
DS1-1	Low	640	1920	1 m 54 s	5.93 s	17.81 s
DS1-2	Low	640	640	1 m 39 s	15.46 s	15.46 s
DS1-3	Low	640	1280	1 m 52 s	8.75 s	17.50 s
DS1-4	Medium	1 216	2 432	3 m 57 s	9.74 s	19.49 s
DS1-5	Medium	320	320	1 m 00 s	18.75 s	18.75 s
DS1-6	Medium	320	320	53 s	16.56 s	16.56 s
DS1-7	High	320	2 560	1 m 07 s	2.62 s	20.93 s
DS1-8	High	432	3 456	1 m 43 s	2.98 s	23.84 s
DS1-9	High	432	3 456	1 m 45 s	3.04 s	24.30 s
Overall		**4 960**	**16 384**	**15 m 50 s**	**5.79 s**	**19.15 s**

Table 5
Manual labelling empirical data.

Individual	# images	# instances to label	time [min]	Mean time [min/100 imgs]	Mean time [min/100 inst.]
P1	200	218	96	44.04	48.00
P2	235	590	312	52.88	132.76
P3	287	457	107	23.41	37.28
Overall	**722**	**1 265**	**515**	**40.11**	**72.68**

instances. For example, the average time needed to create 100 images for datasets *DS1-1* and *DS1-2* is similar, even though *DS1-1* has three times the number of instances compared to *DS1-2*.

6.3. Manual labelling

The manual labelling of images to support the training of object recognition models is a common practice. Nonetheless, due to its highly extremely time-consuming nature, it also poses a significant barrier to adopting these approaches. To address this challenge, various software applications have been developed to streamline this phase (e.g., LabelImg, VoTT, VGG Image Annotator).

Three individuals were tasked with the manual labelling process for two classes of objects, i.e. pallets and PCBs. Upon completing it, the time taken for labelling was documented, and participants were queried about any potential physical or psychological stress encountered during the task.

Similarly to the assessment of automatic labelling (Section 6.2), the time taken to identify (define the bounding box) and label 100 instances, along with the time needed to annotate 100 images, was used as an evaluation metric. The outcomes of this assessment are reported in Table 5, showing an average of roughly 40 min to label 100 object instances, while the average processing time is approximately 73 min for 100 images. Furthermore, the time to accomplish the task seems mostly dependent on the number of instances rather than the number of images.

Furthermore, participants reported the potential occurrence of physical and psychological stress, notably:

- Psychological stress. All participants noted that the task was highly repetitive while demanding a significant level of concentration. Consequently, they found it necessary to take frequent breaks to remain focused.
- Physical stress. Each participant mentioned experiencing eye strain due to the requirement of closely scrutinizing small elements on the screen. Some also reported experiencing tired eyes and mild headaches.

6.4. Manual vs automatic labelling

The comparison between automatic and manual labelling is conducted based on both time and quality performance.

In terms of time, the results show that manually locating and labelling 100 images takes an average of 40.11 min, whereas the automatic approach demands only 5.79 s on average.

Therefore, the automatic approach provides a substantial advantage in terms of time and cost savings. Additionally, it eliminates the potential for psychological or physical stresses, as well as the impact of human factors that could introduce errors during the labelling process.

Regarding the quality of labelling, the automatic generation of bounding boxes followed the procedure outlined in Section 4.2. As a result, these automatically generated bounding boxes were used as the ground truth after validating the automatic labelling process.

The quality of the labels is determined by how accurately the bounding box has been drawn around the object to be labelled. Ideally, a perfect bounding box should have its edges aligned with the edges of the object, without excluding any part of the target, while also minimizing the area enclosed by the bounding box.

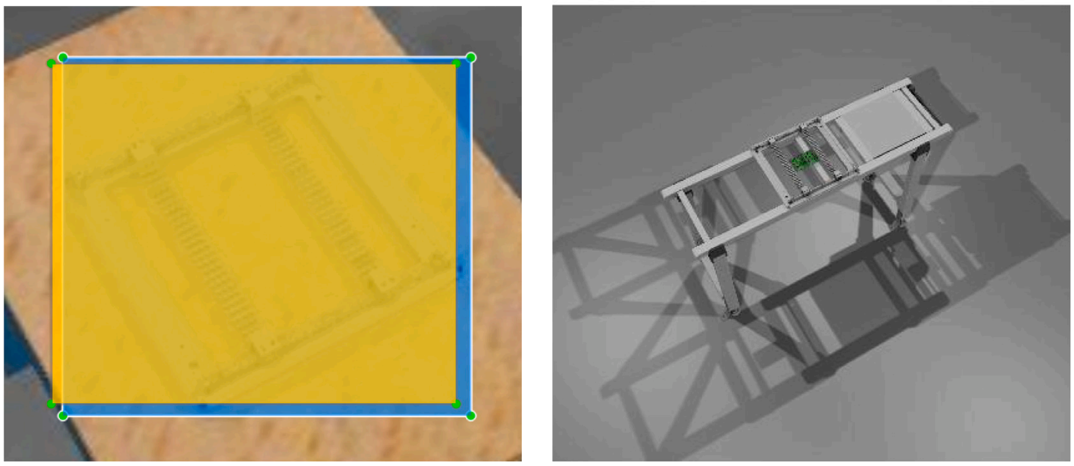
The comparison is made between the manually annotated dataset (see Section 6.3) and the same images annotated automatically (see Section 6.2). The accuracy is measured by calculating the IoU (Eq. (1)) between the two bounding boxes (manually and automatically generated) for the same object.

The results of the comparison are reported in Table 6, showing a high overall average IoU value (see Fig. 9a). This suggests that the manually labelled bounding boxes align closely with those obtained through automatic labelling. However, it is worth noting that despite the participants in the analysis being trained for image labelling, some variability in human-executed processes cannot be entirely eliminated.

Another notable observation is the lower average IoU achieved in the labelling of PCBs. This discrepancy can be attributed to the small size of PCBs (see Fig. 9b). The relatively low image resolution (416×416) chosen to match the input requirements of the YOLO-v3 model (see Section 4.3) makes it difficult to accurately place bounding boxes around these small objects. Although attempts to zoom in and locate the edges of small PCBs are made during the labelling process, inaccuracies may arise due to the limitations imposed by the resolution. A potential solution to this issue might involve increasing the resolution of images to enhance the rendering quality of the rendering of small objects.

7. Training a CNN model on synthetic dataset

The second set of tests involves training CNNs for object detection using synthetic data (i.e. the third phase of the approach, cf.



(a) Bounding boxes with IoU = 0.91

(b) Examples of an image with a PCB

Fig. 9. Dataset examples.

Table 6
Results.

Class	# instances	Max IoU	Min IoU	Average IoU
Pallet	351	1	0.067	0.94
PCB	273	1	0.63	0.84
Total	**819**	**1**	**0.90**	**0.91**

Section 4.3). It must be noted that a comprehensive and quantitative analysis to optimize parameters and hyperparameters is not within the scope of this study and will be addressed in future developments.

The training process will make use of both synthetic (Section 6.1) and real data (Section 5.1). As the conducted experiments are not exhaustive, a stepwise approach is adopted, assessing the viability of various alternative options and proceeding accordingly.

The training and testing of the CNN model will be conducted using various strategies. To support the experiments, the following alternative training and testing approaches have been defined:

- **Virtual Training**: training on synthetic data.
- **Mixed Training**: training on synthetic data plus a fine-tuning on real data.
- **Real Training**: training on real data.
- **Virtual Testing**: testing on synthetic data.
- **Real Testing**: testing on real data.

Table 7 outlines the alternative training and testing schemes. Based on these schemes, a set of experiments have been designed and executed. The specifics of these experiments are defined in Table 8. In the ensuing subsections, the outcomes of the experiments are detailed and analysed, aiming to demonstrate the efficacy of the proposed workflow.

All the experiments have been executed in the cloud on a Google Colaboratory[12] instance equipped with an NVIDIA Tesla V100-SXM2 with 16 GB RAM and NVIDIA CUDA[13] libraries version 11.0. These instances provide a Jupiter Notebook IDE in the cloud for rapid Python development.

A YOLO-v3 model has been implemented, leveraging the open source library Keras[14] version 2.2.3 and the TensorFlow[15] deep learning environment. The model developed has been based on an existing implementation of the YOLO-v3 model [70].

[12] https://colab.research.google.com
[13] https://developer.nvidia.com/cuda-toolkit
[14] https://keras.io
[15] https://www.tensorflow.org

7.1. S0: Training and testing on real data

A first set of experiments has been conducted to establish a baseline for training on synthetic data. In this context, the conventional training methodology has been employed. The dataset encompassing images acquired from the real plant (Section 5.1) has been used for training, validation, and testing (Table 8).

The training process for the model was divided into two steps. During the first step, which lasted 30 epochs, only the last layers of the CNNs were made trainable. Subsequently, the entire model was trained during the second step, which lasted for 45 epochs.

The train and validation loss graph is shown in Fig. 10(a). The graph exhibits a spike in both the train and validation loss around epoch 30. This is due to the transition into the fine-tuning phase, where all layers of the CNN are unfrozen. Thus, the loss computation involves parameters that have not been previously trained with the current dataset and require convergence. The total training time is 6 h and 25 min. At the end of the training, the training and validation losses are equal to 6.160 and 6.229, respectively (Table 9).

The trained model underwent testing using the images from the designated testing dataset (Table 8). The resulting performance values are reported in Fig. 10(b), displaying a COCO mean average precision of 34.66% and a mean average precision (mAP) of 71.77. It must be noted that the low values of the loss functions and mean average precision indicate that the current dataset might be insufficient to achieve good results. This emphasizes the importance of using synthetically generated data to enhance the training of these models.

The low mean average precision is reflected in the set of detections made by the model, as shown in Fig. 11: Fig. 11(a) shows a correct detection for both the pallet and the PCB, while the PCB detection is missing in Fig. 11(b). There are also two examples of false positives, where objects are detected (e.g., a PCB) even when they are not present in the image, as in Fig. 11(c) and (d).

7.2. SA: Virtual training and testing

In the second set of experiments, the object detection model is trained exclusively on synthetic data (Table 8). Similar to the previous experiments, the training of the model has been carried out in two steps lasting 15 and 8 epochs, respectively.

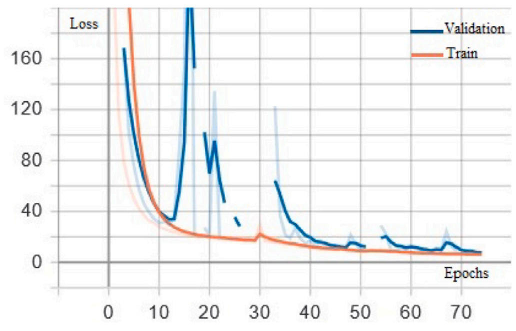
The graph for the train and validation loss is shown in Fig. 10(a). A discontinuity of the validation loss occurs as the fine-tuning phase starts after epoch 15. The train (orange) and validation (blue) loss curves converge around epoch 10. The total training time is 9 h and 42 min,

Table 7
Training and testing schemes.

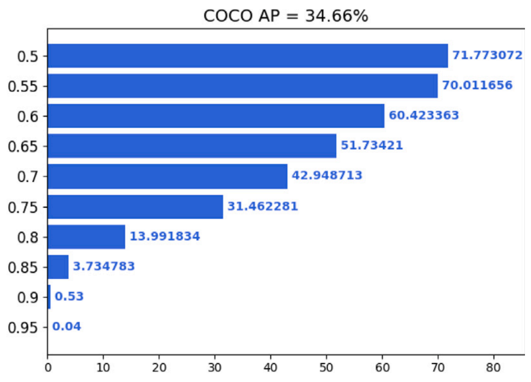
Id	Training	Testing	Description
S0	Real	Real	This is the classic training and testing scheme. It provides a benchmark for alternative strategies supported by synthetic data.
SA	Virtual	Virtual	This scheme completely relies on synthetic data. It serves to test the effectiveness of the training.
SB	Virtual	Real	This scheme entails a domain transfer of a model trained on synthetic data.
SC	Mixed	Real	This scheme assesses the benefits of an additional training step on real data to support the domain transfer.

Table 8
Experimental structure.

Scheme	Training	Training dataset	Validation dataset	Initial learning rate	Optimizer	Testing	Testing dataset
S0	Real	75% (All DR)	12.5% (All DR)	10^{-3}	Adam	Real	12.5% (All DR)
SA	Virtual	70% (All DS)	20% (All DS)	10^{-4}	Adam	Virtual	10% (All DS)
SB	Virtual	70% (All DS)	20% (All DS)	10^{-4}	Adam	Real	100% (DR1)
SC	Mixed	70% (All DS) + 90% (DR0) fine-tuning	20% (All DS) + 10% (DR0) fine-tuning	10^{-5}	Adam	Real	100% (DR1)



(a) Loss function



(b) mean Average Precision

Fig. 10. Results for training and testing on real data.

Table 9
Results for the training and testing in the different schemes.

	Epochs training	Time training	Epochs Fine-tuning	Time fine-tuning	Total time	Loss	Validation loss	mAP IoU=0.5	mAP COCO
S0	30+45	6 h 35 min	–	–	6 h 35 min	6.160	6.229	71.77	34.66
SA	15+8	9 h 42 min	–	–	9 h 42 min	1.279	1.269	99.05	73.52
SB	15+8	9 h 42 min	–	–	9 h 42 min	1.279	1.269	54.49	33.25
SC	15+8	9 h 42 min	35	2 h 16 min	11 h 58 min	3.241	4.470	90.00	48.52

with a final value for the train loss and validation loss equal to 1.279 and 1.269, respectively (Table 9).

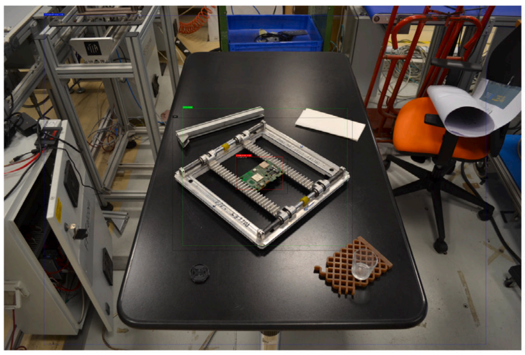
The trained model has been tested on the images of the testing dataset (Table 8). The obtained performance values are reported in Fig. 12(b) with a COCO mean average precision of 73.52% and a mean average precision (mAP) of 99.05. These values are extremely positive, denoting a very good capability of the model to identify the presence and position of the considered objects. Furthermore, these values align with the ones obtained by the YOLO-v3 model in the COCO challenge 2019 leaderboard for detecting common objects [22].

7.3. SB: Virtual training and testing on real data

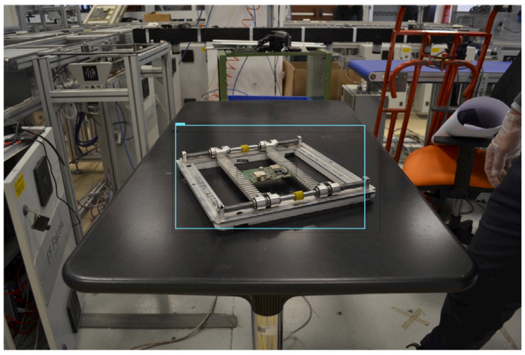
The third set of experiments is focused on the main objective of the proposed approach, i.e., training a CNN for object recognition using synthetic data and subsequently applying it to monitor a real system. The trained model is the same as the one trained in Section 7.2, but the testing phase is carried out using a dataset containing only real data, specifically *DR1* as indicated in Table 1.

The resulting performance values are reported in Table 9 with mAP_{COCO} equal to 33.25% and $mAP_{IoU=0.5}$ equal to 54.49%. These obtained results are unsatisfactory and highlight that training solely on synthetic data does not ensure sufficient detection performance on real images.

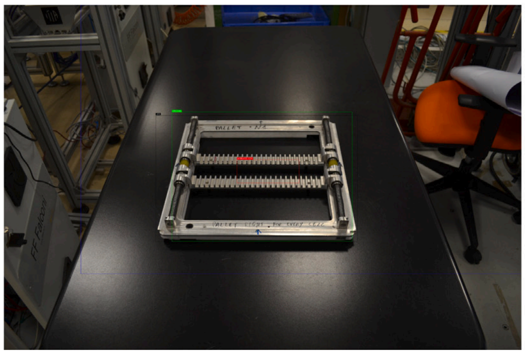
To understand the reasons for the poor performance, a deeper investigation was conducted by assessing the accuracy of the detections on images within the dataset. This qualitative analysis revealed that the model frequently experiences difficulties in accurately detecting objects in complex scenes with multiple objects and heterogeneous framing, resulting in missed and incorrect detections (Fig. 13). Conversely, the detection performance was higher for objects placed against a neutral background (black, white or grey) and positioned at the centre of the image.



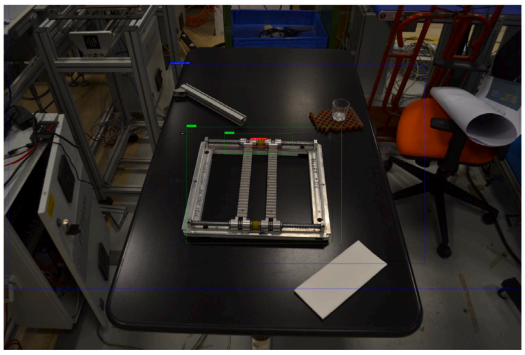
(a) Correct detection.



(b) Missing PCB detection.

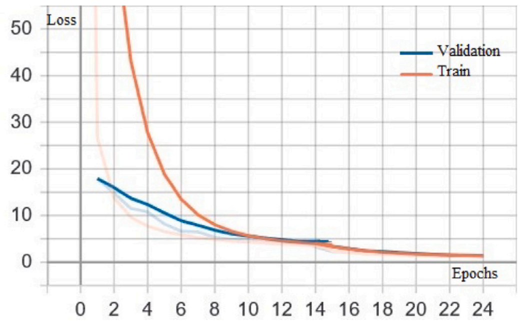


(c) False positive for a PCB.

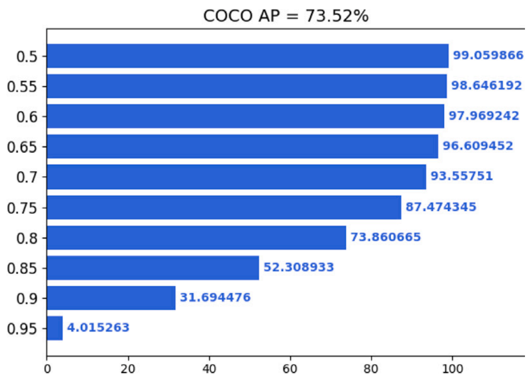


(d) False positive for a PCB and a pallet.

Fig. 11. Examples of detections operated on the test dataset.

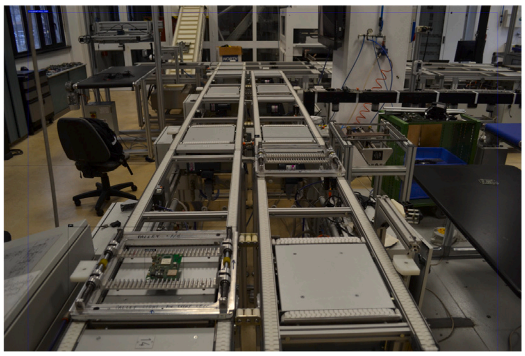


(a) Loss function

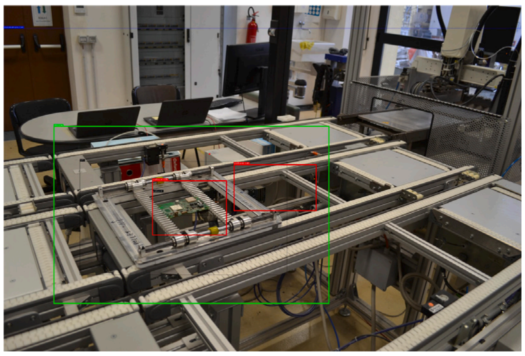


(b) mean Average Precision

Fig. 12. Results for virtual training and testing.

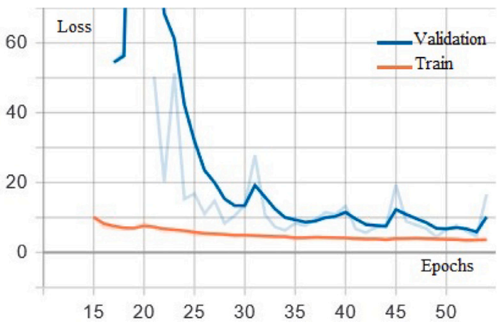


(a) Missing detections.

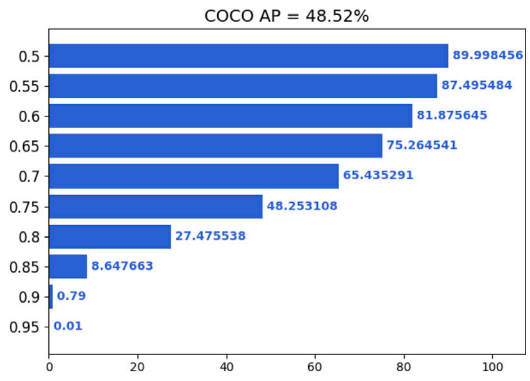


(b) False positives

Fig. 13. Examples of wrong detections in the testing dataset.

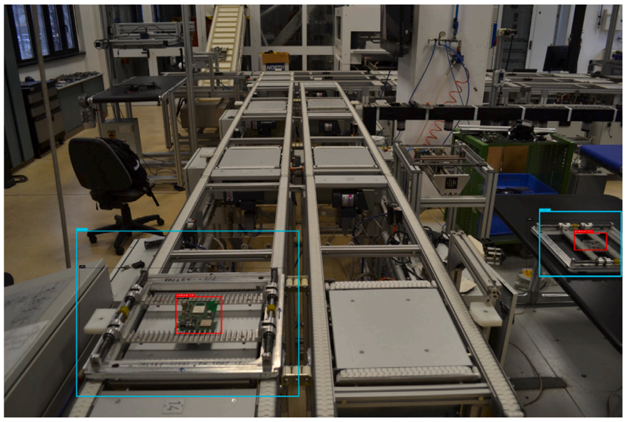


(a) Loss function

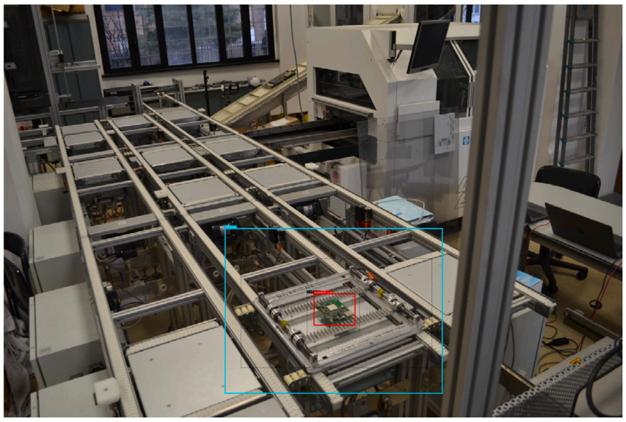


(b) COCO Average Precision

Fig. 14. SC: Training and testing.



(a) Detection in a real environment (confidence: PCB transport system 0.97, PCB table 0.76, Pallet transport system 0.99, Pallet table 0.88).



(b) Detection in a real environment (confidence: PCB 0.96, Pallet 1.00).

Fig. 15. Detection of objects in a real environment.

7.4. SC: Mixed training and testing on real data

To address the challenges experienced with the training scheme in Section 7.3, a different scheme has been adopted. Training is performed on a mixed dataset, containing both synthetic and real images. The CNN is initially trained on synthetic data, using the same dataset and methodology used in Section 7.3. Subsequently, a fine-tuning process is conducted on a dataset consisting of real images. The characteristics of the training dataset are outlined in Table 8.

The diagram of the loss function is plotted in Fig. 14(a). The graph illustrates the fine-tuning phase of the training, therefore the initial 15 epochs are not included since they relate to the first training. The fine-tuning process lasts 35 epochs, resulting in a combined total of 50 epochs for both training stages. The final value of the train and validation loss are 3.241 and 4.470, respectively.

The training time for the mixed training scheme is notably lower in comparison to the one presented in Section 7.1.

The testing of the model was conducted on the test dataset *DR1* (Section 7.3). The obtained performance values are reported in Fig. 14(b) and in Table 9 with a mAP_{COCO} equal to 48.52% and ($mAP_{IoU=0.5}$) equal to 90.00%. These results underscore the efficacy of the mixed training approach, leading to significant improvement in comparison to the performance obtained in Section 7.3.

7.5. Final comments and remarks

The experiments carried out using the proposed training and validation schemes shows that, pursuing scheme SC, it is possible to reach an mAP_{COCO} equal to 48.52% and ($mAP_{IoU=0.5}$) equal to 90.00%. This degree of accuracy is adequate for concrete applications of the object detection model to real application scenarios. This means that the model trained taking advantage of synthetic data is capable of correctly detecting the objects of interest in their real-world context, ensuring accurate localization of pallets and PCBs with respect to the transport system within the RDM plant (cf. requirements *FR1*, *NFR2*, and *NFR3*). To provide visual evidence of the experiment results, some examples are shown in Fig. 15, where blue bounding boxes indicate pallets, while red ones represent PCBs.

Some further remarks are worth to be provided to cover possible limitations, constraints and applicability of the proposed approach. The effectiveness and reliability of transfer learning approaches and training on synthetic images strongly depends on the variety and amplitude of the training dataset. Factors influencing this are the number of synthetic images, and their variety in terms of: framing, number and classes of objects, background objects, background environment (floor, walls, etc.), lighting conditions, etc. The proposed approach for the generation of synthetic data using a DT model of a factory is aimed at supporting the automatic generation of a very large number of images with higher variety. Nevertheless, the generated images are still realistic, i.e., they show the objects to be identified in realistic conditions as they are likely to happen in the factory. Additional approaches has been proposed generating unrealistic images that can push towards the variety of the dataset, contributing to obtain a higher effectiveness of the object detection models [41,71]. Pursuing these strategies is a mandatory option to improve the performance and effectiveness of the proposed class of approaches.

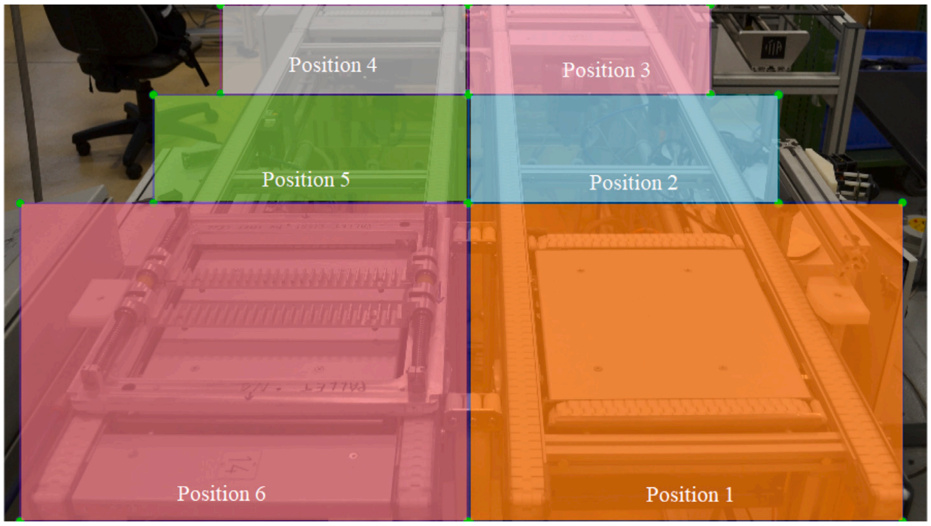


Fig. 16. Definition of positions within the transportation system.

Other factors, such as image rendering quality, resolution, and the specific characteristics of the objects to be detected, also influence the training effectiveness. For instance, PCBs, with their distinct textures and colours, are easier to detect, whereas metallic objects might require more careful consideration regarding rendering and resolution due to their differing appearance in virtual and real environments.

Regarding lighting conditions, both synthetic and real test images were subjected to varying levels of brightness. During the experiments, no significant impact of lighting variations on detection quality was observed.

However, the size and location of the objects in the image play a crucial role. Objects at the image's edge or smaller in size were more challenging to detect, particularly in dataset $DS6$. This limitation can be mitigated by using multiple cameras or high-resolution cameras in the factory, allowing multiple parallel detections on different portions of the same image.

8. Application to the monitoring of transportation and handling operations

The model trained and tested in Section 7.3 has been applied to demonstrate the feasibility of using a CNN for tracking parts as they move through a manufacturing system (fourth phase of the approach, cf. Section 4.4).

The proposed hypothesis involves the replacement of traditional position sensors with a network of cameras and trained CNN models. Although position sensors might be cost-effective, their integration into industrial monitoring systems can still incur significant expenses. Furthermore, relying on cameras and software-driven methods offers the monitoring system greater flexibility, particularly in scenarios where the system layout or transportation equipment might undergo changes.

The proof of concept is realized to monitor the movement of pallets along the conveyors within the transportation system of the RDM pilot plant (Section 5).

In addition, some hypotheses are defined:

1. Cameras are installed in fixed positions.
2. Objects that need to be monitored are completely visible within the field of view of the cameras.
3. The position of an object is discrete, indicating specific predefined locations (e.g., Position A or Position B), rather than continuous spatial coordinates.
4. The possible positions for objects are predetermined and known in advance.

For each camera, a bounding box (B_L) is defined for each of the selected possible positions of an object in the system. This can be done manually or leverage the DT of a factory, similarly to what has been done for the automatic labelling of synthetic images (Section 6.2). An example is provided in Fig. 16, showing the definition of the bounding boxes for six alternative pallet positions on the conveyors.

Thus, the trained CNN model is employed to detect the presence of pallets within the images captured by the cameras placed in fixed positions. Once a pallet is detected, the model predicts a bounding box (B_p) around it, indicating its location within the image. The evaluation of the possible presence of a pallet in a specific position is performed using the IoU metric (Eq. (1)), where the bounding box for the ground truth is the one associated with a position (B_L). As multiple positions are available, the prediction of the position of the pallet is executed by selecting the position with the highest value of the IoU:

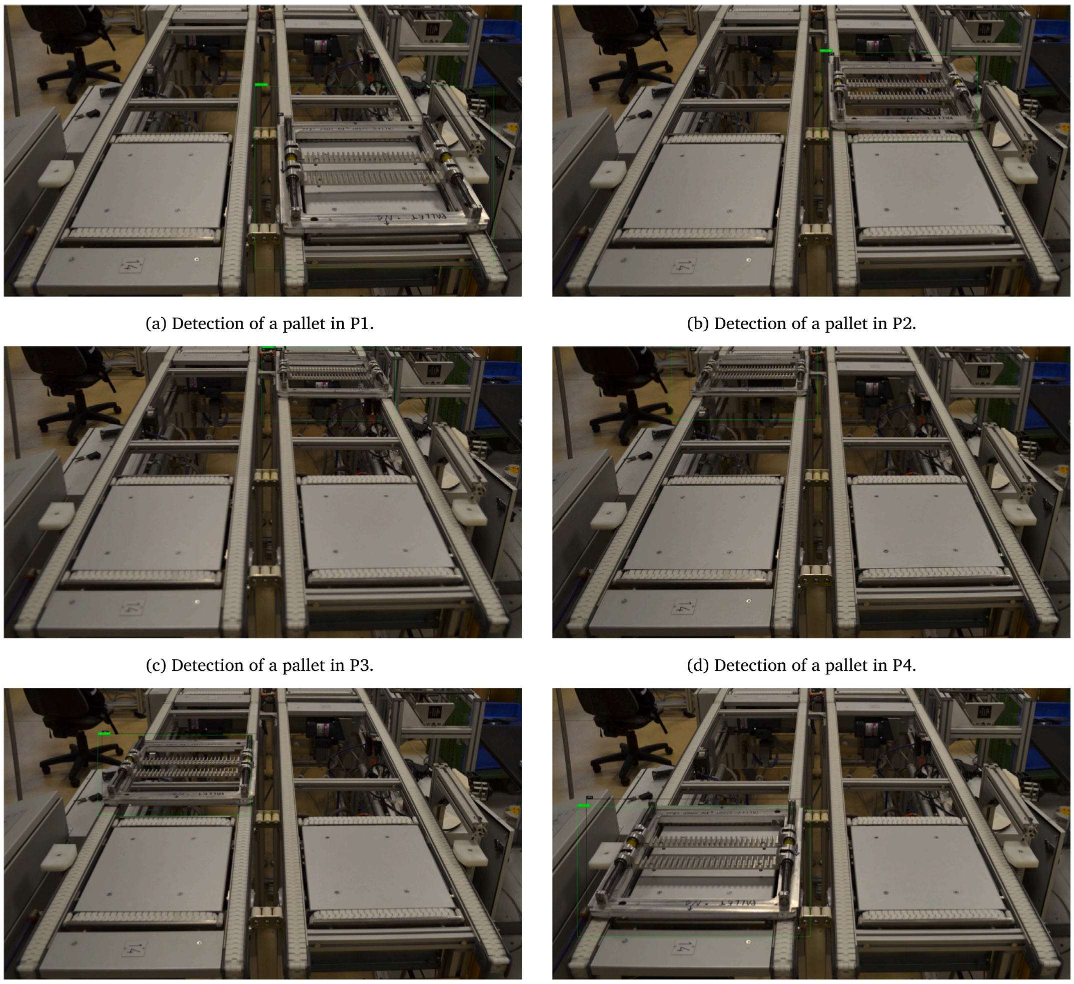
$$\hat{P} = \underset{h\in\{1,\dots,6\}}{\arg\max} \quad IoU_h \tag{3}$$

A test was carried out by moving a single pallet on the conveyor and collecting a set of images. Hence, the prediction of the position of the pallet is operated according to the steps described above. The results are reported in Table 10, showing that, at least for this simple scenario, the proposed approach can predict the correct position of the pallet (cf. requirement *FR2*). The detections associated with these images are also reported in Fig. 17.

A preliminary test has also been carried out with multiple pallets, both empty and loaded with a PCB. The results of the detections are reported in Fig. 18, demonstrating the capability of also identifying multiple pallets and their status (cf. requirement *FR3*).

The inference time for detecting the presence of the objects in the image was on the order of tens of milliseconds, consistent with the declared performance of YOLO-v3 models [67]. Considering the relatively low speed of the pallets moving within the plant, it is reasonable to sample a frame about every second. Thus, the proposed approach can operate a real-time monitoring of the selected objects in the system (cf. requirement *NFR1*).

Traditional and well-established monitoring methods typically rely on basic sensors like proximity and presence detectors, which are notably cost-effective. Implementing computer vision for monitoring introduces a more intricate workflow and demands greater computational resources, often resulting in less reliability. However, when considering the monitoring of an entire factory, the scenario changes. While individual sensors may be inexpensive, the cumulative cost of installing and integrating a vast number of them can become substantial, making the use of cameras and computer vision unexpectedly cost-effective.



(a) Detection of a pallet in P1.

(b) Detection of a pallet in P2.

(c) Detection of a pallet in P3.

(d) Detection of a pallet in P4.

(e) Detection of a pallet in P5.

(f) Detection of a pallet in P6.

Fig. 17. Identification of the position of a pallet on the conveyor.

Table 10
Results for the prediction of the position based on the IoU.

Actual position	IoU_{P1}	IoU_{P2}	IoU_{P3}	IoU_{P4}	IoU_{P5}	IoU_{P6}	Predicted position
P1	**0.652**	0.114	0.000	0.000	0.006	0.020	P1
P2	0.266	**0.646**	0.025	0.000	0.000	0.000	P2
P3	0.000	0.172	**0.631**	0.010	0.004	0.000	P3
P4	0.000	0.004	0.010	**0.656**	0.173	0.000	P4
P5	0.000	0.000	0.000	0.021	**0.677**	0.042	P5
P6	0.000	0.000	0.000	0.000	0.000	**0.671**	P6

Furthermore, the key advantage of camera-based systems lies in their flexibility and reconfigurability. In scenarios where the need arises to monitor new parts or track different routes, cameras can be easily adapted to these new requirements by simply retraining the object detection models to recognize these new elements. In contrast, updating traditional sensor-based monitoring systems could entail significant reconfiguration costs.

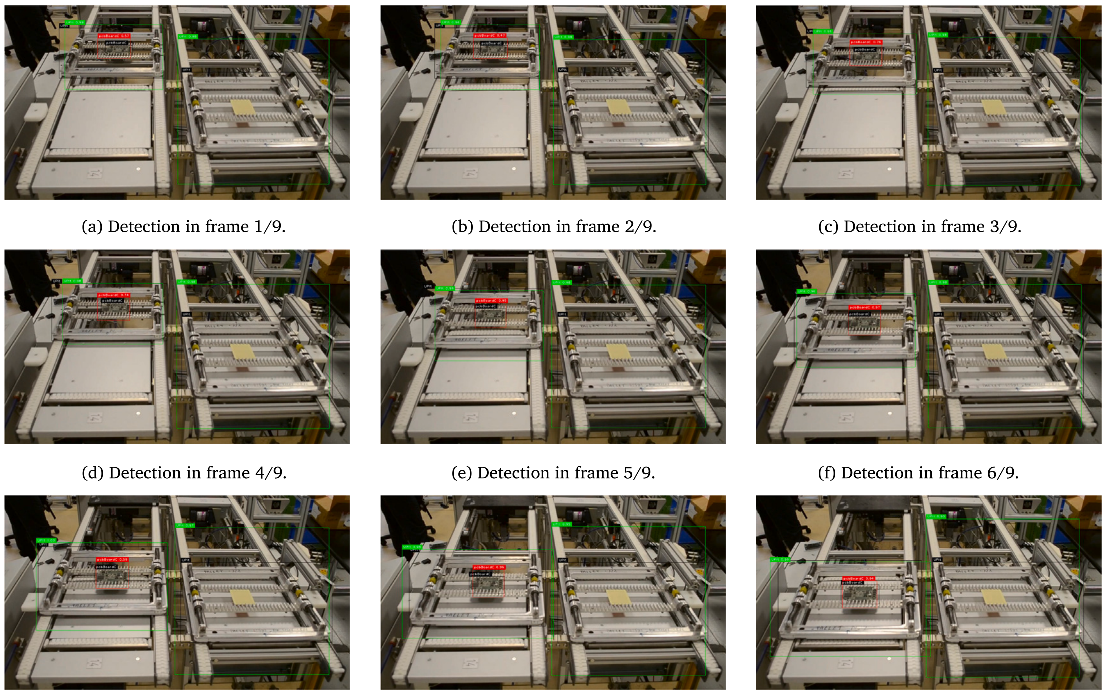
9. Conclusions

This study focuses on training CNNs using synthetically generated images to enhance object detection in a real-world setting. The motivation is to overcome two significant challenges in adopting these methods: the requirement for large datasets and the related annotations to train neural networks effectively.

The major challenge for using synthetic data is the definition of suitable scenarios to generate data that exhibits adequate diversity and volume, thus facilitating a successful domain transfer.

The proposed approach unequivocally showcases its capability to generate large amounts of labelled data with exceedingly minimal time and computational resources. This is a clear advantage with respect to the traditional image collection methods and manual labelling.

A YOLO-v3 model was trained on synthetic data and tested in different scenarios. It was then experimentally proven that models trained uniquely on synthetic images could achieve some correct detection on real images, even though they could not match the requirements in terms of accuracy. Integrating the synthetic dataset with a set of real images was considered a viable solution to achieve optimal results. The real images can be used for a second training phase (fine-tuning) of the models previously trained on synthetic datasets.



(a) Detection in frame 1/9. (b) Detection in frame 2/9. (c) Detection in frame 3/9.

(d) Detection in frame 4/9. (e) Detection in frame 5/9. (f) Detection in frame 6/9.

(g) Detection in frame 7/9. (h) Detection in frame 8/9. (i) Detection in frame 9/9.

Fig. 18. Identification of position and state of multiple pallets.

Furthermore, it has been demonstrated that conducting an initial training phase on a synthetic dataset significantly reduces training duration on real images. The experimental outcomes show that the average duration of the second training phase is less than 30% of the training time required for training exclusively on real images (cf. benchmark experiment in Section 7.1).

Finally, the effectiveness of the tracking operation leveraging a fine-tuned model was evaluated through a simplified use case relative to the transport system of the RDM pilot plant. The results indicated that the achieved degree of accuracy enables the pallet localization within the transport system and provides insights into its movement among predetermined positions.

Future work will address several developments related to the methodology and its application. First of all, although the performance of the proposed approach is quite good, the training of CNN models on synthetic data is likely to be influenced by multiple factors, e.g., the characteristics of the object, the resolution of the generated images, the randomization parameters used to generate synthetic images, the characteristics of the scene, etc.

Thus, the proposed approach will be tested and applied to different factory contexts, to assess its robustness in different application domains. Nevertheless, the availability of a digital representation of the factory offers the opportunity to fine-tune the factors described above and, thus, address the possible weakness feasibly, without the need to access and/or modify the real factory environment.

In addition, the output of the CNN model can be further elaborated and enhanced by a tighter integration with the ontology-based model, thus semantically representing the evolution of the monitored production system.

CRediT authorship contribution statement

Marcello Urgo: Conceptualization, Methodology, Validation, Formal analysis, Investigation, Resources, Writing – original draft, Writing – review & editing, Visualization, Supervision. **Walter Terkaj:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Visualization, Supervision. **Gabriele Simonetti:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Visualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] P. Zheng, H. wang, Z. Sang, R.Y. Zhong, Y. Liu, C. Liu, et al., Smart manufacturing systems for industry 4.0: Conceptual framework, scenarios, and future perspectives, Front Mech Eng 13 (2) (2018) 137–150, http://dx.doi.org/10.1007/s11465-018-0499-5.

[2] F. Tao, Q. Qi, A. Liu, A. Kusiak, Data-driven smart manufacturing, J Manuf Syst 48 (2018) 157–169, http://dx.doi.org/10.1016/j.jmsy.2018.01.006, URL https://www.sciencedirect.com/science/article/pii/S0278612518300062. Special Issue on Smart Manufacturing.

[3] K. Samir, A. Maffei, M.A. Onori, Real-time asset tracking; A starting point for digital twin implementation in manufacturing, Procedia CIRP 81 (2019) 719–723, http://dx.doi.org/10.1016/j.procir.2019.03.182.

[4] L. Monostori, B. Kádár, T. Bauernhansl, S. Kondoh, S. Kumara, G. Reinhart, et al., Cyber-physical systems in manufacturing, CIRP Ann 65 (2) (2016) 621–641, http://dx.doi.org/10.1016/j.cirp.2016.06.005, URL https://www.sciencedirect.com/science/article/pii/S0007850616301974.

[5] A. Schütze, N. Helwig, T. Schneider, Sensors 4.0 – smart sensors and measurement technology enable industry 4.0, J Sensors Sensor Syst 7 (1) (2018) 359–371, http://dx.doi.org/10.5194/jsss-7-359-2018.

[6] A. Frankó, G. Vida, P. Varga, Reliable identification schemes for asset and production tracking in industry 4.0, Sensors 20 (13) (2020) 3709, http://dx.doi.org/10.3390/s20133709.

[7] A. Correia, I. Reyes, et al., AI research and innovation: Europe paving its own way, 2020, http://dx.doi.org/10.2777/264689, European Commission. Directorate-General for Research and Innovation.

[8] R. Stark, C. Fresemann, K. Lindow, Development and operation of digital twins for technical systems and services, CIRP Ann 68 (1) (2019) 129–132, http://dx.doi.org/10.1016/j.cirp.2019.04.024.

[9] W. Terkaj, T. Tolio, M. Urgo, A virtual factory approach for in situ simulation to support production and maintenance planning, CIRP Ann - Manuf Technol 64 (1) (2015) 451–454, http://dx.doi.org/10.1016/j.cirp.2015.04.121, URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-84933679202&amp;doi=10.1016%2fj.cirp.2015.04.121&amp;partnerID=40&amp;md5=4b6f5d64e0cffc6e085d1ca6882a588d.

[10] W. Terkaj, P. Gaboardi, C. Trevisan, T. Tolio, M. Urgo, A digital factory platform for the design of roll shop plants, CIRP J Manuf Sci Technol 26 (2019) 88–93.

[11] T. Tolio, G. Copani, W. Terkaj, Key research priorities for factories of the future—Part II: Pilot plants and funding mechanisms, in: T. Tolio, G. Copani, W. Terkaj (Eds.), Factories of the future: the Italian flagship initiative, Springer International Publishing, Cham, 2019, pp. 475–494, http://dx.doi.org/10.1007/978-3-319-94358-9_21.

[12] T.H.J. Uhlemann, C. Lehmann, R. Steinhilper, The digital twin: Realizing the cyber-physical production system for industry 4.0, Procedia CIRP 61 (2017) 335–340, http://dx.doi.org/10.1016/j.procir.2016.11.152.

[13] E.N. Malamas, E.G.M. Petrakis, M. Zervakis, L. Petit, J.D. Legat, A survey on industrial vision systems, applications and tools, Image Vis Comput 21 (2) (2003) 171–188, http://dx.doi.org/10.1016/S0262-8856(02)00152-X.

[14] S. Thiede, P. Ghafoorpoor, B.P. Sullivan, S. Bienia, M. Demes, K. Dröder, Potentials and technical implications of tag based and AI enabled optical real-time location systems (RTLS) for manufacturing use cases, CIRP Ann 71 (1) (2022) 401–404, http://dx.doi.org/10.1016/j.cirp.2022.04.023, URL https://www.sciencedirect.com/science/article/pii/S0007850622000695.

[15] K.B. Lee, S. Cheon, C.O. Kim, A convolutional neural network for fault classification and diagnosis in semiconductor manufacturing processes, IEEE Trans Semicond Manuf 30 (2) (2017) 135–142, http://dx.doi.org/10.1109/TSM.2017.2676245.

[16] B. Zhang, S. Liu, Y.C. Shin, In-process monitoring of porosity during laser additive manufacturing process, Addit Manuf 28 (2019) 497–505, http://dx.doi.org/10.1016/j.addma.2019.05.030.

[17] M.V. Andulkar, J. Hodapp, T. Reichling, M. Reichenbach, U. Berger, Training CNNs from synthetic data for part handling in industrial environments, in: 2018 IEEE 14th international conference on automation science and engineering, 2018, pp. 624–629.

[18] V. Nandini, R.D. Vishal, C.A. Prakash, S. Aishwarya, A review on applications of machine vision systems in industries, Indian J Sci Technol 9 (48) (2016) 1–5, http://dx.doi.org/10.17485/ijst/2016/v9i48/108433.

[19] C. Pramerdorfer, M. Kampel, A dataset for computer-vision-based PCB analysis, in: 2015 14th IAPR international conference on machine vision applications, 2015, pp. 378–381, http://dx.doi.org/10.1109/MVA.2015.7153209.

[20] R. Chauhan, K.K. Ghanshala, R. Joshi, Convolutional neural network (CNN) for image detection and recognition, in: 2018 first international conference on secure cyber computing and communication, 2018, pp. 278–282, http://dx.doi.org/10.1109/ICSCCC.2018.8703316.

[21] S. Albawi, T.A. Mohammed, S. Al-Zawi, Understanding of a convolutional neural network, in: 2017 international conference on engineering and technology, 2017, pp. 1–6, http://dx.doi.org/10.1109/ICEngTechnol.2017.8308186.

[22] COCO - Common Objects in Context, COCO challenge leaderboard, 2019, https://cocodataset.org/#detection-leaderboard.

[23] M. Urgo, M. Tarabini, T. Tolio, A human modelling and monitoring approach to support the execution of manufacturing operations, CIRP Ann 68 (1) (2019) 5–8, http://dx.doi.org/10.1016/j.cirp.2019.04.052.

[24] M. Urgo, F. Berardinucci, P. Zheng, L. Wang, AI-based pose estimation of human operators in manufacturing environments, in: T. Tolio (Ed.), CIRP Novel Topics in Production Engineering: Volume 1, Springer International Publishing, Cham, 2024, pp. 3–38, http://dx.doi.org/10.1007/978-3-031-54034-9_1.

[25] T. Lin, P. Goyal, R.B. Girshick, K. He, P. Dollár, Focal loss for dense object detection, 2017, CoRR abs/1708.02002. arXiv:1708.02002. URL http://arxiv.org/abs/1708.02002.

[26] S. Ren, K. He, R.B. Girshick, J. Sun, Faster R-CNN: Towards real-time object detection with region proposal networks, 2015, CoRR abs/1506.01497. arXiv:1506.01497. URL http://arxiv.org/abs/1506.01497.

[27] J. Redmon, S.K. Divvala, R.B. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, 2015, CoRR abs/1506.02640. arXiv:1506.02640. URL http://arxiv.org/abs/1506.02640.

[28] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S.E. Reed, C. Fu, et al., SSD: Single shot MultiBox detector, 2015, CoRR abs/1512.02325. arXiv:1512.02325. URL http://arxiv.org/abs/1512.02325.

[29] D. Dwibedi, I. Misra, M. Hebert, Cut, paste and learn: Surprisingly easy synthesis for instance detection, 2017, CoRR abs/1708.01642. arXiv:1708.01642. URL http://arxiv.org/abs/1708.01642.

[30] J. Dai, Y. Li, K. He, J. Sun, R-FCN: Object detection via region-based fully convolutional networks, 2016, CoRR abs/1605.06409. arXiv:1605.06409. URL http://arxiv.org/abs/1605.06409.

[31] K. Suzuki, Overview of deep learning in medical imaging, Radiol Phys Technol 10 (3) (2017) 257–273, http://dx.doi.org/10.1007/s12194-017-0406-5.

[32] G. Georgakis, A. Mousavian, A.C. Berg, J. Kosecka, Synthesizing training data for object detection in indoor scenes, 2017, CoRR abs/1702.07836. arXiv:1702.07836. URL http://arxiv.org/abs/1702.07836.

[33] W.H. Yun, T. Kim, J. Lee, J. Kim, J. Kim, Cut-and-paste dataset generation for balancing domain gaps in object instance detection, IEEE Access 9 (2021) 14319–14329.

[34] K. Sarkar, K. Varanasi, D. Stricker, Trained 3D models for CNN based object recognition, in: International conference on computer vision theory and applications, 2022, pp. 130–137.

[35] S. Hinterstoisser, V. Lepetit, P. Wohlhart, K. Konolige, On pre-trained image features and synthetic images for deep learning, 2017, CoRR abs/1710.10710. arXiv:1710.10710. URL http://arxiv.org/abs/1710.10710.

[36] P.S. Rajpura, R.S. Hegde, H. Bojinov, Object detection using deep CNNs trained on synthetic images, 2017, CoRR abs/1706.06782. arXiv:1706.06782. URL http://arxiv.org/abs/1706.06782.

[37] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, P. Abbeel, Domain randomization for transferring deep neural networks from simulation to the real world, 2017, CoRR abs/1703.06907. arXiv:1703.06907. URL http://arxiv.org/abs/1703.06907.

[38] A. Rozantsev, M. Salzmann, P. Fua, Beyond sharing weights for deep domain adaptation, 2016, CoRR abs/1603.06432. arXiv:1603.06432. URL http://arxiv.org/abs/1603.06432.

[39] G. Varol, J. Romero, X. Martin, N. Mahmood, M.J. Black, I. Laptev, et al., Learning from synthetic humans, 2017, CoRR abs/1701.01370. arXiv:1701.01370. URL http://arxiv.org/abs/1701.01370.

[40] S. Hinterstoisser, O. Pauly, H. Heibel, M. Marek, M. Bokeloh, An annotation saved is an annotation earned: Using fully synthetic training for object instance detection, 2019, CoRR abs/1902.09967. arXiv:1902.09967. URL http://arxiv.org/abs/1902.09967.

[41] Y.C. Jhang, A. Palmar, B. Li, S. Dhakad, S.K. Vishwakarma, J. Hogins, et al., Training a performant object detection ML model on synthetic data using unity perception tools, 2020, https://blogs.unity3d.com/2020/09/17/training-a-performant-object-detection-ml-model-on-synthetic-data-using-unity-computer-vision-tools/.

[42] Unity Technologies, Unity perception package, 2020, https://github.com/Unity-Technologies/com.unity.perception.

[43] J. Stark, PLM and the digital twin, in: Product lifecycle management (volume 1), Springer, 2022, pp. 369–401.

[44] F. Tao, H. Zhang, A. Liu, A.Y.C. Nee, Digital twin in industry: State-of-the-art, IEEE Trans Ind Inf 15 (4) (2019) 2405–2415, http://dx.doi.org/10.1109/TII.2018.2873186.

[45] W. Terkaj, M. Annoni, B.O. Martinez, E. Pessot, M. Sortino, M. Urgo, Digital twin for factories: Challenges and industrial applications, in: L. Carrino, L.M. Galantucci, L. Settineri (Eds.), Selected topics in manufacturing: Emerging trends from the perspective of aITeM's Young researchers, Springer Nature Switzerland, Cham, 2024, pp. 255–274, http://dx.doi.org/10.1007/978-3-031-41163-2_13.

[46] B. Schleich, N. Anwer, L. Mathieu, S. Wartzack, Shaping the digital twin for design and production engineering, CIRP Ann 66 (1) (2017) 141–144, http://dx.doi.org/10.1016/j.cirp.2017.04.040, URL https://www.sciencedirect.com/science/article/pii/S0007850617300409.

[47] Z. Zhu, C. Liu, X. Xu, Visualisation of the digital twin data in manufacturing by using augmented reality, Procedia CIRP 81 (2019) 898–903, http://dx.doi.org/10.1016/j.procir.2019.03.223, URL https://www.sciencedirect.com/science/article/pii/S2212827119305281. 52nd CIRP Conference on Manufacturing Systems (CMS), Ljubljana, Slovenia, June 12-14, 2019.

[48] J. Bao, D. Guo, J. Li, J. Zhang, The modelling and operations for the digital twin in the context of manufacturing, Enterprise Inf Syst 13 (4) (2019) 534–556.

[49] D. Adeniji, J. Schoop, In-situ calibrated digital process twin models for resource efficient manufacturing, Trans ASME, J Manuf Sci Eng 144 (4) (2022).

[50] D. Botkina, M. Hedlind, B. Olsson, J. Henser, T. Lundholm, Digital twin of a cutting tool, Procedia CIRP 72 (2018) 215–218, 51st CIRP Conference on Manufacturing Systems.

[51] I. Errandonea, S. Beltrán, S. Arrizabalaga, Digital twin for maintenance: A literature review, Comput Ind 123 (2020) 103316, http://dx.doi.org/10.1016/j.compind.2020.103316, URL https://www.sciencedirect.com/science/article/pii/S0166361520305509.

[52] A. Greco, M. Caterino, M. Fera, S. Gerbino, Digital twin for monitoring ergonomics during manufacturing production, Appl Sci 10 (21) (2020) URL https://www.mdpi.com/2076-3417/10/21/7758.

[53] C. Zhuang, J. Liu, H. Xiong, Digital twin-based smart production management and control framework for the complex product assembly shop-floor, Int J Adv Manuf Technol 96 (2018) 1149–1163.

[54] Z. Lv, S. Xie, Artificial intelligence in the digital twins: State of the art, challenges, and future research topics, Digit Twin 1 (2022) 12, http://dx.doi.org/10.12688/digitaltwin.17524.2.

[55] A. Malik, P. Rajaguru, R. Azzawi, Smart manufacturing with artificial intelligence and digital twin: A brief review, in: 2022 8th international conference on information technology trends, ITT, 2022, pp. 177–182, http://dx.doi.org/10.1109/ITT56123.2022.9863938.

[56] Y. Xu, Y. Sun, X. Liu, Y. Zheng, A digital-twin-assisted fault diagnosis using deep transfer learning, Ieee Access 7 (2019) 19990–19999.

[57] Z. Ren, J. Wan, P. Deng, Machine-learning-driven digital twin for lifecycle management of complex equipment, IEEE Trans Emerg Top Comput 10 (1) (2022) 9–22.

[58] K. Bartsch, A. Pettke, A. Hübert, J. Lakämper, F. Lange, On the digital twin application and the role of artificial intelligence in additive manufacturing: A systematic review, J Phys: Mater 4 (3) (2021) 032005.

[59] F. Mo, H.U. Rehman, F.M. Monetti, J.C. Chaplin, D. Sanderson, A. Popov, et al., A framework for manufacturing system reconfiguration and optimisation utilising digital twins and modular artificial intelligence, Robot Comput-Integr Manuf 82 (2023) 102524, http://dx.doi.org/10.1016/j.rcim.2022.102524, URL https://www.sciencedirect.com/science/article/pii/S073658452200206X.

[60] M. Groshev, C. Guimarães, J. Martín-Pérez, A. de la Oliva, Toward intelligent cyber-physical systems: Digital twin meets artificial intelligence, IEEE Commun Mag 59 (8) (2021) 14–20, http://dx.doi.org/10.1109/MCOM.001.2001237.

[61] P.L. Mazzeo, A. Argentieri, F.D. Luca, P. Spagnolo, C. Distante, M. Leo, et al., Convolutional neural networks for recognition and segmentation of aluminum profiles, in: Proc.SPIE, vol. 11059, 2019, pp. 219–229, http://dx.doi.org/10.1117/12.2525687.

[62] K. Alexopoulos, N. Nikolakis, G. Chryssolouris, Digital twin-driven supervised machine learning for the development of artificial intelligence applications in manufacturing, Int J Comput Integr Manuf 33 (5) (2020) 429–439, http://dx.doi.org/10.1080/0951192X.2020.1747642.

[63] J. Cohen, J. Cohen, C. Crispim-Junior, C. Grange-Faivre, L. Tougne, CAD-based learning for egocentric object detection in industrial context, in: 15th international conference on computer vision theory and applications, 2022, pp. 644–651.

[64] H. Ying, R. Sacks, A. Degani, Synthetic image data generation using BIM and computer graphics for building scene understanding, Autom Constr 154 (2023) 105016, http://dx.doi.org/10.1016/j.autcon.2023.105016, URL https://www.sciencedirect.com/science/article/pii/S0926580523002765.

[65] F. Berardinucci, G. Colombo, M. Lorusso, M. Manzini, W. Terkaj, M. Urgo, A learning workflow based on an integrated digital toolkit to support education in manufacturing system engineering, J Manuf Syst 63 (2022) 411–423, http://dx.doi.org/10.1016/j.jmsy.2022.04.003, URL https://www.sciencedirect.com/science/article/pii/S027861252200053X.

[66] J. Hui, Real-time object detection with YOLO, YOLOv2 and now YOLOv3, 2019, https://jonathan-hui.medium.com/real-time-object-detection-with-yolo-yolov2-28b1b93e2088.

[67] J. Redmon, A. Farhadi, YOLOv3: An incremental improvement, 2018, arXiv:1804.02767 [cs]. arXiv:1804.02767.

[68] COCO - Common Objects in Context, COCO evaluation metrics for object detection, 2022, https://cocodataset.org/#detection-eval. [Accessed: 2022-12-02].

[69] M. Urgo, W. Terkaj, G. Simonetti, RDM dataset, 2023. http://dx.doi.org/10.5281/zenodo.8225189.

[70] david8862, TF keras YOLOv3 modelset, 2020, https://github.com/david8862/keras-YOLOv3-model-set.

[71] N. Morrical, J. Tremblay, Y. Lin, S. Tyree, S. Birchfield, V. Pascucci, et al., NViSII: A scriptable tool for photorealistic image generation, 2021, arXiv:2105.13962.