POLITECNICO DI TORINO
Repository ISTITUZIONALE

Combined Hidden-Markov Model and Siamese Network approach for assembly operation recognition and error detection

Original
Combined Hidden-Markov Model and Siamese Network approach for assembly operation recognition and error detection / Pelosi, M., Zanchettin, A.M., Rocco, P.. - In: INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH. - ISSN 1366-588X. - ELETTRONICO. - 64:17(2026), pp. 7457-7479. [10.1080/00207543.2025.2556484]

Availability:
This version is available at: 11583/3003491 since: 2025-09-30T09:18:16Z

Publisher:
Taylor and Francis

Published
DOI:10.1080/00207543.2025.2556484

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
Taylor and Francis postprint/Author's Accepted Manuscript

This is an Accepted Manuscript of an article published by Taylor & Francis in INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH on 2026, available at http://wwww.tandfonline.com/10.1080/00207543.2025.2556484

(Article begins on next page)

07 November 2026

RESEARCH ARTICLE

Combined Hidden-Markov Model and Siamese Network approach for assembly operation recognition and error detection

M. Pelosi[a b], A. M. Zanchettin[b], P. Rocco[b]

[a]Politecnico di Torino, Dipartimento di Automatica e Informatica (DAUIN), Corso Castelfidardo 34/d, 10138, Torino (Italy); [b]Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Piazza Leonardo da Vinci 32, 20133, Milano (Italy)

ARTICLE HISTORY
Compiled August 27, 2025

ABSTRACT
The growing demand for customized products is driving manufacturing towards more flexible production systems. In this context, manual assembly lines remain essential for complex or small-batch products, as human cognitive and advanced manipulation capabilities enable better adaptation to changes in production processes. However, the experience and working conditions of human operators significantly impact the quality of the final product. External factors such as stress, fatigue, and cognitive overload can lead to errors, especially in flexible environments that require rapid adaptation. Monitoring human activity to detect errors during the process is, therefore, crucial. This paper presents a Siamese network-based approach to recognize assembly steps and detect errors, easily adaptable to different assembly scenarios. By comparing images captured during the process with a reference dataset, the Siamese Neural Network (SNN) identifies the last assembly step performed and possible discrepancies. The SNN is integrated into a Hidden Markov Model framework to improve recognition accuracy and process reliability. Experimental validation demonstrated the model's adaptability to various assembly scenarios, including a real-world use case of industrial packaging, without the need to retrain the model. The method also proved to be robust to different working conditions and human variability.

KEYWORDS
Quality assurance; Hidden Markov Model; Siamese Neural Network; Assembly operation recognition; Human error detection;

1. Introduction

With the advance of Industry 4.0, manufacturing is moving from rigid and large-batch production to more flexible and highly customized systems (Y. Koren *et al.* 2010), (F. Zhou *et al.* 2013). This paradigm shift responds to the growing demand for personalized products and shorter life cycles, but also introduces significant challenges. Fully automated assembly lines, while widespread in industrial environments, struggle to meet the requirements imposed by frequent changes in production processes or product designs. These systems are optimized for consistent and repetitive tasks and excel in high-volume manufacturing. However, they lack the adaptability needed to

CONTACT M. Pelosi. Email: martina.pelosi@mail.polimi.it, martina.pelosi@polito.it

handle dynamic production scenarios and are neither practical nor cost-effective for small-batch-size production.
Human-Robot Collaboration (HRC) offers a promising alternative to address these challenges. Humans excel at decision-making, problem-solving, and adapting to new situations, while robots efficiently perform repetitive, precise, or physically demanding tasks. Together, they create a synergy that enables flexible and efficient production while reducing the physical and mental effort of human operators (A. Ajoudani *et al.* 2018), (A. Borboni *et al.* 2023).
However, for economic and technological reasons, complex products with small batch sizes are still preferably manufactured in manual assembly settings (A. Metzmacher *et al.* 2019). In these environments, product quality and assembly efficiency depend significantly on the skills, experience, and working conditions of the operator. In addition, factors such as stress, fatigue, and inconsistent working conditions can negatively affect performance, leading to errors during assembly (Y. Torres *et al.* 2021). These challenges are amplified in environments that require workers to quickly adapt to changing processes. Furthermore, flexible deployment of workers and tasks reduces the opportunity for adequate training, making it difficult to develop the experience necessary to handle complex tasks without mistakes. At the production process level, the high complexity and variability of products increase the cognitive load on workers, requiring them to process more information, make faster decisions, and adapt to new procedures (L. Merkel *et al.* 2018).
Within this scenario, Zero Defect Manufacturing (ZDM) emerges as a key paradigm to improve quality and reliability in flexible and human-centered production systems. ZDM aims to systematically remove defects by leveraging advanced digital technologies in four main fields: detection, prediction, prevention, and repair (F. Psarommatis *et al.* 2024). In the context of defect detection, monitoring human movements and actions in real-time, identifying errors as they occur, and providing immediate feedback are fundamental to address the challenges emerging both in HRC and fully manual settings. These Cyber-Physical monitoring Systems (CPS) can provide the required communication channels between humans and robots (H. Liu *et al.* 2018), but also be an effective tool for early defect detection in the assembly process, not only improving product quality, but also reducing rework, production downtime, and waste. This proactive approach ensures the mitigation of human limitations in flexible production environments, enabling more reliable and efficient manufacturing processes.
With these purposes, this paper introduces a vision-based comparative method to recognize assembly steps performed by a human worker, simultaneously evaluating their correctness and detecting errors during the process. The approach exploits a Siamese Neural Network (SNN) trained to highlight chromatic and geometric differences between two input images of the same product at different stages of assembly. More in detail, by comparing an image captured online with a reference dataset representing the correct assembly steps, the system identifies the most recently completed operation and flags any variations or errors in its execution. The SNN provides real-time observations of the current assembly state, which are then integrated into a Hidden Markov Model (HMM) framework. To enhance prediction accuracy, the assembly process is modeled as an HMM, leveraging the SNN's outputs as observation data for the current hidden assembly state.
The main contributions of this work compared to the state-of-the-art methods are listed below.

- **Adaptability to new products:** The model does not require retraining or

fine-tuned model variants for new products introduced in the production line, ensuring the required production flexibility and ready adaptability to new assembly processes. Instead, the pre-trained model is used as-is, in combination with a new set of application-specific reference images. The SNN compares the current assembly image with this reference set, identifying the most visually similar reference state, which is then flagged as the current assembly step. The training dataset includes a variety of assembly types, capturing the visual features commonly found in most industrial products, but does not contain any application-specific examples. This design makes the model scalable and applicable to a wide range of assemblies, including products with different complexity and configurations compared to the training data, provided that the visual differences between steps are adequately represented in both the training dataset and the reference images.

- **Integration of assembly dynamics:** By combining the predictions of the SNN with a-priori knowledge of assembly process dynamics embedded within an HMM, the framework filters out inconsistent network outputs. Aligning the predictions with the logical sequence of assembly steps increases the reliability of the SNN predictions.
- **Simultaneous error detection:** The model recognizes assembly operations, simultaneously detecting production faults and inconsistencies during the process.
- **Robustness to environmental variations:** The composition of the training dataset, including only images of real assembly parts and industrial items, and the use of an accurate segmentation model, make the network robust to external shadows, light reflections, and different light exposure.

2. State of the art

With the purpose of zero-defect manufacturing, detecting defects and production errors is a key aspect required to provide timely corrective interventions (F. Psarommatis *et al.* 2020). Several works in the scientific literature focus on analyzing manufacturing quality in the direction of zero-defect production, developing methods both to detect potential defects on the workpiece and to track human operations and execution errors (G. Genta *et al.* 2020), (Y. Le *et al.* 2012).
This section collects relevant works on quality assurance in industrial settings. Starting with state-of-the-art methods for detecting defects and production errors in Section 2.1, the discussion continues with the approaches for recognizing executed assembly steps in Section 2.2. Special attention is paid to the use of Siamese Networks and assembly dynamics for these purposes.

2.1. *Production error detection*

Traditional inspections, conducted at the end of the production line, result in delays in error detection and high reworking costs and waste. To address this, there is an increasing interest in leveraging Computer Vision algorithms during the assembly process to early detect non-conformities and meet the quality standards (C. Chen *et al.* 2020). For example, G. Di Leo *et al.* (2014) developed a camera-LED inspection system to measure electromechanical components' dimensions, correcting inconsistencies, while S. Bhardwaj (2016) applied image subtraction, template matching, and

histogram matching to highlight defects in Surface Mounted Devices in PCBs. Despite being effective in the validated contexts, these approaches are task-specific and lack adaptability to different products and applications.

AI algorithms offer promising solutions for quality assurance, supporting the production flexibility required by complex, dynamic manufacturing environments while reducing costs and resources, and improving productivity. (S. Thamm *et al.* 2021), (Y. Hui *et al.* 2022), (M. Getachew *et al.* 2024). For instance, M. Ojer *et al.* (2020) developed a PCB real-time guiding and monitoring system using classifiers trained with synthetic images demonstrating robustness for each mounted component in the tested systems. However, it requires retraining whenever a new part is introduced in the assembly, limiting applicability to frequently changing productions. Similarly, C. Chen *et al.* (2020) proposed a Fully Convolutional Network to recognize parts present in the assembled product, using CAD-rendered images as training data. Nevertheless, this approach only works if the product's CAD model is available, showing an additional limitation. Furthermore, A. Manimuthu *et al.* (2022) combined K-means and SVM to monitor vehicle assemblies, enhancing productivity in volatile manufacturing environments and reducing energy and material waste. While effective within the trained context, the method requires retraining for different assembly systems or new tasks, as it relies on system-specific training data.

Few-shot learning enhances Deep Learning adaptability to different scenarios, avoiding retraining or Transfer Learning. In particular, Siamese Neural Networks (SNNs) learn similarity metrics between two inputs and act as zero-shot methods by comparing the test instance with a reference (S. Jadon *et al.* 2020). While SNNs were initially applied to facial and handwriting recognition, their use has recently expanded to new areas, such as defect detection. For instance, J. Kurek *et al.* (2020) used an SNN to assess the drill wear level by comparing images of drilled holes. Moreover, H. Takimoto *et al.* (2022) proposed an SNN-based approach for anomaly detection in industrial parts, particularly useful when limited data representing defective parts are available.

Although training models to classify errors helps to focus attention on defect-related features, enhancing detection performance, this approach requires prior knowledge of all potential anomalies that might occur during the process. In dynamic industrial environments, anomalies could be multiple and varied, making it difficult to collect all defect-specific images. In this sense, recent studies on Large Language Models (LLM) and Large Vision Models (LVM) bring significant contributions to fault diagnosis, enabling better generalization across diverse fault types, locations, and scenarios (Z. Li *et al.* 2025). L. Peifeng *et al.* (2024) combined aviation knowledge graphs with LLMs through prefix-tuning for fault localization in complex assembly settings. Achieving 98.5% accuracy in real industrial tests and showing strong performance and explainability, the method enables efficient, domain-adaptive reasoning without high computational effort. However, its effectiveness depends on high-quality knowledge graphs and application-specific fine-tuning. Another example is provided by K. Qian *et al.* (2024), who employed large vision-language models and contrastive cross-modal training to enhance industrial anomaly detection and precise localization by aligning visual and textual features in a shared embedding space. While offering improved interpretability and performance in terms of detection accuracy and localization, the approach requires significant computational resources due to the complexity of LVMs-LLMs. Given the required fine-tuning for specific applications and high real-time computational effort, foundation models do not provide easily implementable plug-and-play solutions for various manufacturing environments.

Another solution to address the issue of partially available labeled data is provided

by unsupervised learning to detect unknown faults. For example, J. Ju *et al.* (2023) introduced a fault detection method using a dynamic convolutional GRU attention autoencoder, capable of effectively handling complex temporal dependencies in industrial processes in an unsupervised manner. Despite strongly performing in several processes, the approach is tailored to specific dynamic patterns, lacking generalizability to significantly different process dynamics. Another example is represented by the unsupervised fault detection method by Y. Byun *et al.* (2024), exploiting frequency-wise angular filtering to isolate normal samples from vibration data. It improves non-labeled fault detection but assumes normal patterns dominate the dataset, which may not hold in all industrial settings.
Similar to unsupervised methods, SNNs have also been explored for fault detection by identifying errors as deviations from reference manufacturing states, without classification. D. Lehman *et al.* (2024) proposed an error segmentation and localization method, exploiting synthetic image pairs from 3D CAD models. The system detects and localizes previously unseen errors by comparing test images with correct references without the need for manual labels. However, the model generalizes poorly to new states, requiring retraining for major configurations changes, and, also, was only tested on single-view 2D images, despite using 3D data during training.

2.2. *Assembly step recognition*

Detecting errors as deviations from expected assembly states requires recognizing all intermediate assembly steps composing the process. Classifying assembly correct steps instead of mistakes allows for exploiting the known process dynamics, in terms of feasible assembly sequences, to improve the accuracy of the data-driven detection, and to expand the range of identifiable errors, including sequencing-related faults. All studies in Section 2.1 use instantaneous detection without accounting for the whole assembly process, limiting the method's robustness and applicability.
In contrast, in our work, the process is modeled as an HMM that exploits the a-priori known system dynamics, inferring hidden assembly states by treating the SNN's predictions as observations of the state itself.
In industrial assemblies, an assembly state is the product's static configuration at a given step, while an assembly operation is the action enabling the transition from one state to the next. Knowing the previous and current states reveals the executed operation, allowing verification of correct steps through action monitoring rather than waiting for state completion. Several studies focused on operator movements rather than the appearance of the resulting assembled component, often using human-object interaction to infer ongoing operations. For example, C. Chen *et al.* (2020) trained a 3D CNN to classify operations from image sequences. Although the method achieved high accuracy, the training dataset is limited to nine common actions, requiring model retraining for new operations. Moreover, the dataset creation is complex, involving many participants and long recordings. Another example is the VR-based assistance system by L. Eversberg *et al.* (2021), classifying manual assembly steps in real time, using a multivariate time series convolutional neural network and 6DOF motion data of the user's head, hands, and tools. Although the model requires only a few assembly recordings for training, it does not generalize to unseen sequences of operations without task-specific training. Furthermore, this approach is not ideal for quality assessment, as it classifies operations without evaluating their correctness or focusing on the condition of the assembled part.

Table 1.: Methodology comparison with recent state-of-the-art approaches.

Reference	Method	Context	Input	Validation	Features		
					Scalability	Impl. effort	Comp. effort
G. Di Leo *et al.* (2014)	CV	Damage detection	Multi-vision	Online	None	Low	Low
S. Bhardwaj (2016)	CV	Fault detection	2D-vision	Online	None	Low	Low
M. Ojer *et al.* (2020)	ML	Assembly assistance Assembly monitoring	2D-vision	Online	Medium	Low	Medium
C. Chen *et al.* (2020)	FullyCN+3DCNN	Fault detection Operation recognition	3D-vision (tested only 2D)	Online	Medium	Low	Medium
A. Manimuthu *et al.* (2022)	SVM+Kmeans	Fault detection	2D-vision	Online	Medium	Low	Medium
J. Kurek *et al.* (2020)	SNN	Wear monitoring	2D-vision	Offline	Medium	Medium	Medium
H. Takimoto *et al.* (2022)	SNN	Anomaly detection	2D-vision	Offline	High	Medium	Medium
L. Peifeng *et al.* (2024)	KG+LLM	Fault detection Fault localization	Language data	Offline	High	Low	High
K. Qian *et al.* (2024)	LVM+LLM	Fault detection Fault localization	2D-vision	Offline	High	Low	High
J. Ju *et al.* (2023)	DCGRUA-AE	Fault detection	3D-vision	Online	Medium	Low	Medium
Y. Byun *et al.* (2024)	Unsupervised filtering	Fault detection	Vibration frequency data	Online	Medium	Low	Medium
D. Lehman *et al.* (2024)	SNN	Error detection Error localization	Multi-vision	Online (partially tested)	Medium	Low	Medium
L. Eversberg *et al.* (2021)	DNN	Assembly step recognition	2D-vision	VR	Low	Low	Low
A. Riedel *et al.* (2021)	YOLOv4+MQTT	Assembly guidance Fault detection	2D-vision	Online	Medium	Low	Medium
Our approach	SNN+HMM	Assembly step recognition Error detection	2D-vision (possible multi-view)	Online	High	Medium	Medium

Regarding the assembly evolution, most of the studies exploit the assembly dynamics for task allocation and scheduling to optimize manual, automated, or collaborative processes (Z. Jia *et al.* 2025), (M. Cai *et al.* 2025). In the quality assessment field, A. Riedel *et al.* (2021) introduced a modular assistance system using MQTT to deliver dynamic work instructions, preventing errors, and YOLOv4 for fault detection, highlighting present components. Validated in real industrial settings, it demonstrated efficient deployment with short data collection and annotation. However, it operates with 2D-vision data, limiting its spatial awareness, and it lacks flexibility, as it cannot generalize to new assemblies without fine-tuning the object detection model. In addition, while the framework accounts for the process dynamics, it is used only for guidance, not to enhance the YOLO recognition capabilities.
To the best of the authors' knowledge, our approach represents the first application of Hidden-Markov Models to enhance network predictions. Unlike previous methods, it eliminates the need to retrain the model for specific assembly tasks, involving only new reference images to adapt to different scenarios.
All relevant state-of-the-art methods have been compared with our approach in Table 1, highlighting key features like flexibility, use of multi-view vision, and implementation effort.

3. Siamese Neural Network for assembly step recognition and error detection

This Chapter presents the SNN specifically designed to recognize assembly steps and detect execution errors, as shown in Figure 1.
After each operation the human performs, an image of the assembled part is captured and compared with a set of reference images showing the correctly assembled part at each process step. The SNN outputs a dissimilarity value for each reference image, which is then compared to a predefined threshold properly customized for the specific assembly process. The assembly step executed by the operator is identified as the one corresponding to the lowest dissimilarity, provided this value is below the threshold.

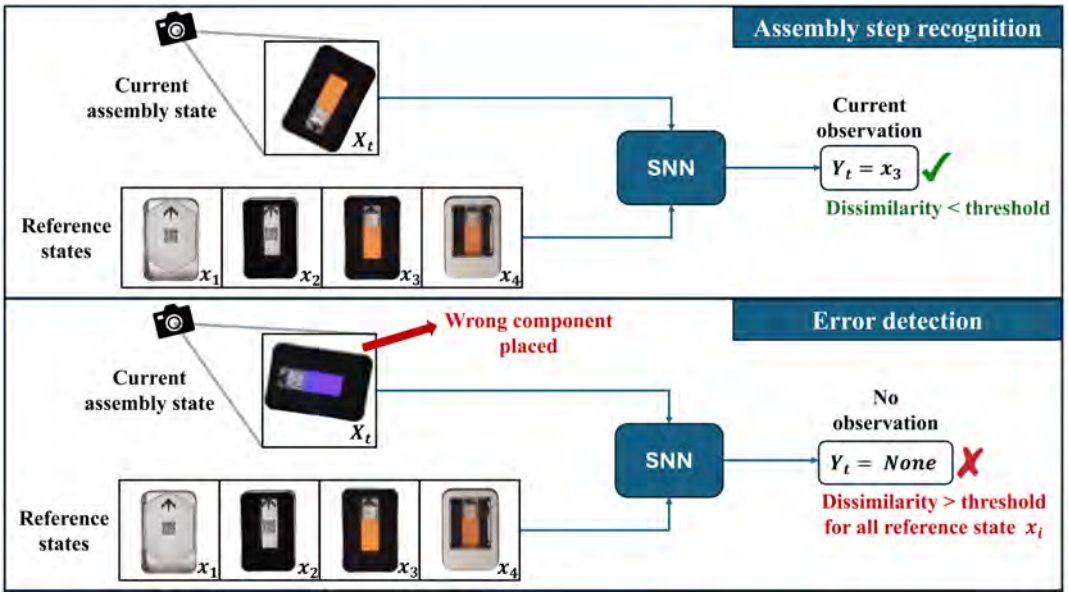


Figure 1.: SNN-based assembly step recognition and error detection mechanisms.

On the contrary, if no dissimilarity meets the threshold criteria, the system highlights an erroneous execution, indicating a possible mistake made by the operator.
This approach is based on the following assumptions, valid in a variety of real industrial assembly processes.

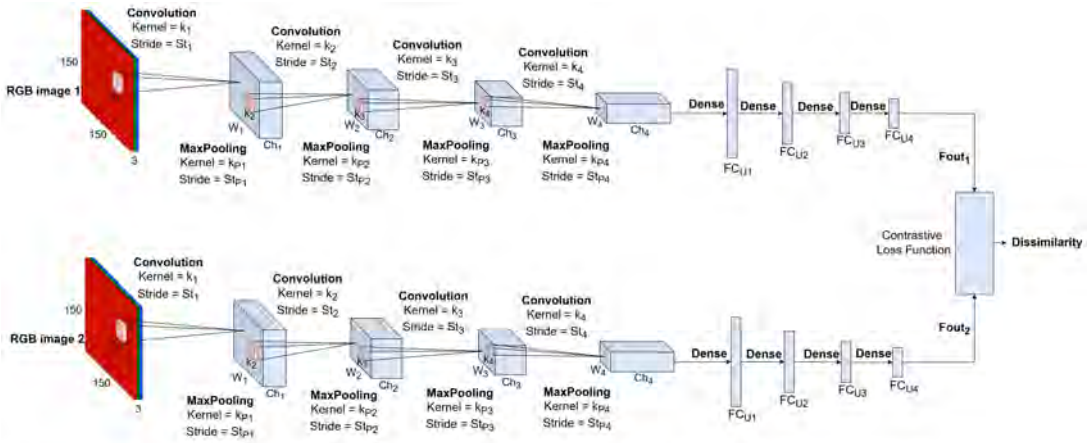
- **The assembly process can be monitored from a single perspective.** The assembly state is captured with a top-down camera view of the product, in accordance with the standard industrial monitoring systems, allowing easy installation, minimal intrusion into the workspace, and a wide field of view.
- **Assembly error detection is more critical than error classification.** This assumption reflects the industrial request for early fault detection, allowing a prompt human intervention to avoid error propagation and reduce rework costs. The method is designed to flag deviations from the expected feasible assembly state, without identifying the specific nature of the error. In real-world scenarios, errors can be varied or unknown in advance, making it impossible to collect sufficient faulty samples for each potential failure mode.

This Chapter first extends a previous SNN comparative approach for assembly step recognition developed by M. Pelosi *et al.* (2024) to address limitations encountered during early testing. Specifically, issues such as reflections on the assembled part, external shadows in the scene, or too subtle visual variations from one assembly step to the next were found to negatively impact the model's predictions. To handle these challenging conditions, the current approach includes an image pre-processing stage aimed at removing the background, thereby helping the network focus on relevant visual information. In addition, a chromatic data augmentation procedure was implemented to enhance the model's robustness to color and lighting variations. The training dataset was also significantly expanded to include a wider variety of objects and assembly processes, improving the network's generalization capability. Introducing more elements with very subtle distinguishing features, the dataset extension also aims to improve the model's ability to recognize fine-grained visual details. Furthermore, the SNN architecture was modified to improve feature learning and computational ef-

ficiency. The following sections detail the architectural improvements and the training dataset composition developed to overcome these limitations.

3.1. *Network's architecture*

The architecture of the SNN, as shown in Figure 2, consists of two identical Convolutional Neural Networks (CNNs) that process squared RGB images, transforming them into feature vectors to allow the SNN-based image comparison.
Compared to the original design, the input resolution has been reduced from 400x400



Hyperparameter	Learning rate	Batch size	Channel 1	Channel 2	Channel 3	Channel 4	Kernel 1	Kernel 2	Kernel 3	Kernel 4	Stride	MaxPool kernel	MaxPool stride	FC-units 1	FC-units 2	FC-units 3	FC-units 4
Symbol	λ	b	Ch_1	Ch_2	Ch_3	Ch_4	k_1	k_2	k_3	k_4	St_i	k_{P_i}	St_{P_i}	FC_{U_1}	FC_{U_2}	FC_{U_3}	FC_{U_4}
Value	0.0001	128	64	128	128	256	5	4	3	3	1	2	2	4096	512	512	32

Figure 2.: Siamese Network's architecture: two parallel Convolutional Neural Networks with four convolutional layers with kernel k_i, stride St_i and channel number Ch_i, four max-pooling layers with kernel k_{P_i} and stride St_{P_i} and four fully connected layers with FC-units FC_{U_i}

to 150x150 pixels, which is considered a good compromise to limit the computational time, significantly increased due to the expanded training dataset, while still well-visualizing object details. Additionally, including an image pre-processing stage enhances feature extraction, making this resolution reduction feasible without compromising performance. The images are processed by four convolutional layers (only three in the original network architecture), each followed by the Gaussian Error Linear Unit (GELU) activation function and a max pooling layer. The GELU replaced the more common Rectified Linear Unit (ReLU) activation function, as it was proven to be more effective and efficient, enhancing the extraction of valuable features and reducing the learning time (D. Hendrycks *et al.* 2016). The structure of the CNNs is completed by four fully connected layers, allowing the transformation of the input RGB image into the feature vector.
The comparative SNN architecture is enabled by the Contrastive Loss Function (CL), employed to learn the dissimilarity between feature vectors. It is defined as:

$$CL = (1-Y)\frac{1}{2}D^2 + Y\frac{1}{2}\max(0, m-D)^2, \text{ with } D = \|F_{out_1} - F_{out_2}\| \qquad (1)$$

Table 2.: Dataset composition: nine assembly groups and nine similar object groups

Group type	Group name	N. classes	Classes differences
Assembly	Angle grinder	15	Different assembly steps
	3D-printed planetary gearbox	11	Different assembly steps
	USB flash-drive case	5	Different assembly steps
	Torch	4	Different assembly steps
	Screws assembly	3	Different assembly steps
	Plastic statuette	4	Different assembly steps
	3D-printed gear wheel system	6	Different assembly steps
	Remote control (external)	4	Different assembly steps
	Remote control (internal)	5	Different assembly steps
Similar objects	Screwdriver	9	Different screwdrivers (or similar) in color or/and shape
	Pliers (or similar)	12	Different pliers (or similar) in color or/and shape
	Scissors	4	Open and closed configuration of scissors
	Colored cubes	4	Different colored cubes (same geometry)
	3D-printed base	2	Different bases in shape (same color)
	Paper clips	2	Different clips in shape (same color)
	Plastic statuette	10	Different colored statuettes (same shape)
	Plastic bars	3	Different bars in color or/and geometry
	Internal assembly configuration	2	Different component position

with F_{out_i} the output feature vector of the i-th CNN, $Y \in \{0, 1\}$ the label of the input image pairs (0 for identical images, 1 for different ones), and $m = 2$ the margin.

3.2. *Training dataset*

The goal is to train the model to detect subtle visual differences in the assembled part, characterizing distinct steps within the same assembly process. To achieve this, the dataset must contain image pairs that reflect these variations. Thus, including some pairs of images showing completely unrelated or structurally different assembled objects is not only unnecessary but also counterproductive. The differences between these pictures do not provide relevant information to learn for the desired task, introducing unnecessary variability, distracting the network from learning more critical visual details, and, thus, increasing the convergence time.
The following explains all the steps to collect and diversify the training dataset to meet these requirements.

3.2.1. Data collection and data augmentation

The dataset was composed from scratch, including 18 distinct groups of objects. The first nine groups involve various assemblies, each characterized by a different number of assembly steps (classes), ranging from simple to more complex processes. In addition, nine groups of objects, such as industrial tools, clips, and plastic parts, were added to further enhance the capability of the model to learn subtle visual variations: each group includes a different number of classes, representing objects of the same type slightly differing in shape, color, or configuration. These objects contain features common in real-world assembly components, such as holes, sharp edges, screws, and specific materials, enabling the network to learn discriminative visual features for various assembly contexts. As shown in Table 2, the 18 groups of items produce 106 classes of different assembly steps and similar objects in total. For each class, 20 images of the part were captured by randomly moving and rotating the object on a white work surface, to make the final model positional and rotational invariant, enabling the recognition of

the components even if rotated and translated within the assembly zone.
The collected data were digitally augmented to enlarge the dataset, resulting in 60 additional files for each class. Digital variations, such as small random horizontal and/or vertical translations up to 10% of the original image dimension, rotations up to 10°, and random contrast and brightness changes, made the model more robust to environmental differences.

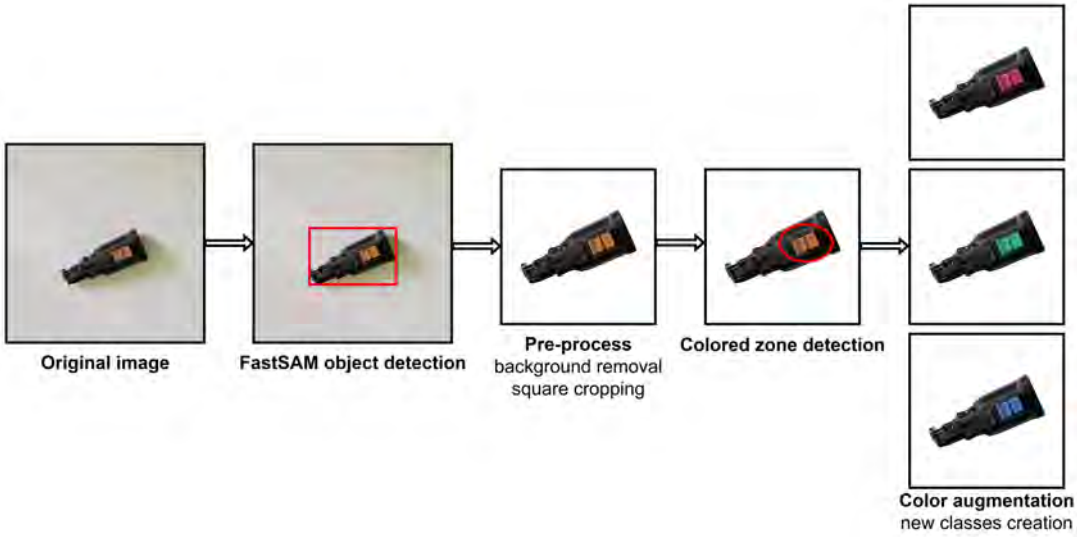
3.2.2. FastSAM-based object segmentation and background removal

After data collection and augmentation procedures, the entire dataset was pre-processed employing Fast-Segment Anything Model (FastSAM) (X. Zhao *et.al* 2023) to enable the network to focus on the object of interest. FastSAM, leveraging the YOLOv8-seg architecture for efficient instance segmentation, was used to segment the target subject. This process involved segmenting the object from its surroundings, removing the background and any potential external shadows in the workspace, and cropping the image around the object. These steps were designed to minimize irrelevant information in the image and reduce the number of non-essential features the network needs to process, allowing the final model to focus only on the relevant aspects for similarity detection. Furthermore, the use of FastSAM in the dataset creation improved the overall quality of the training data, making the images more uniform and drastically reducing noise caused by variations in lighting and shadows. To clarify, Figure 3a shows the pre-processing procedure applied to an example image of the training dataset.
FastSAM was employed not only during the training phase but also to test the model performance in the experimental validation. The use of this segmentation model ensured the network's adaptability across various surfaces, as background removal mitigated the influence of surface-specific features. For example, while the training dataset was prepared using a white surface, some experiments described in Section 5 were conducted on a green table with holes. Thanks to FastSAM's ability to remove backgrounds during training and testing, these surface variations did not significantly impact the network's performance.

3.2.3. Color augmentation

A systematic chromatic augmentation process was adopted to improve the model's ability to recognize chromatic variations, reducing the negative impact of external shadows and reflections in color recognition, thus improving the model's generalization across different assembly contexts. In fact, many real-world manufacturing systems produce components within the same industrial class of products, which often share similar shapes and geometries, but may differ in colors or textures. These chromatic differences can be critical for correctly identifying the current assembly state, especially when geometric cues are insufficient. Therefore, all those classes showing relevant chromatic features were digitally augmented. Specifically, for each class, three new classes were generated in the same object group by altering the color channels in the key chromatic regions, as shown in Figure 3a.
After adding the chromatically modified classes, 355 folders were obtained in total. With each class including 80 images, the final dataset contains 28400 files, resulting in 3.64 GB[1].

[1]The training dataset used to train the model can be found at the following link:
`https://osf.io/cvwqp/?view_only=bd2b68dd24a34bf3974701a388e38033`



(a) Dataset creation process: FastSAM-based pre-processing of the original image and color data augmentation.

(b) Examples of training image pairs with the corresponding label. 0: same assembly step/item with different position, orientation, and/or contrast and brightness; 1: different assembly steps and items with small visual variations.

Figure 3.: Dataset creation process and examples of training image pairs.

3.2.4. Image pairing and labeling

Image pairing is required to finally train the network, and a targeted approach was designed to improve the network's ability to highlight subtle differences between the input images. The first image is selected randomly from the entire dataset, while the second image is chosen from the same object group as the first. Moreover, for a balanced dataset, the second image is chosen from the same class and a different one with equal probability. The pairs of images are labeled with 1 when the data belong to different classes and 0 when they come from the same class. With this approach, the training pairs represent, with 50% probability, the exact same object or assembly step and, with 50% probability, similar objects or different assembly steps with small visual differences in color and/or shape. This maximizes the relevance of the training pairs to enhance the network's performance in discerning small chromatic and geometrical variations.
Some examples of training image pairs are shown in Figure 3b, together with the corresponding labels.

3.3. *Training configuration and results*

A grid search tuning process was conducted to optimize two critical hyperparameters significantly affecting model performance: learning rate (λ) and batch size (b).
The model was trained for 50 epochs for every possible combination of values from the following predefined sets:

$$\lambda = [0.001, 0.0001, 1 \cdot 10^{-5}, 0.0005, 5 \cdot 10^{-5}, 5 \cdot 10^{-6}]$$

$$b = [16, 32, 64, 128, 256]$$

After evaluating the loss function and accuracy trends for each combination, the optimal values were identified as:

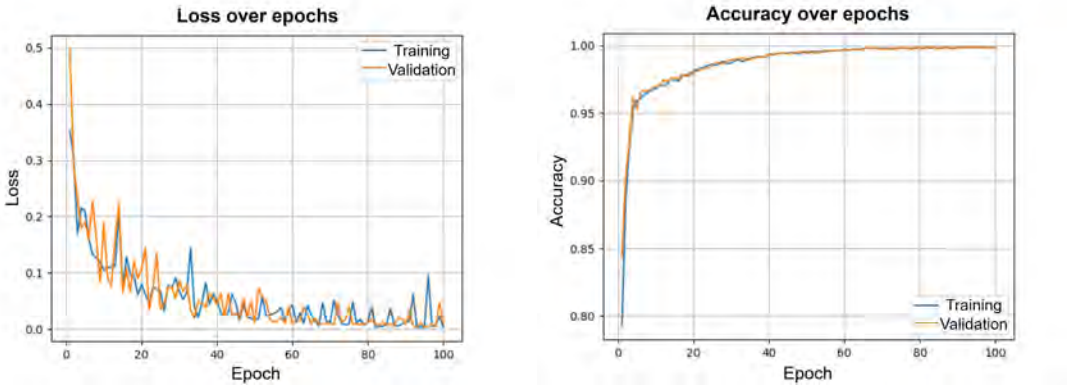
$$\lambda = 0.0001, b = 128$$

This combination represents a good balance between performance and computational efficiency. Higher learning rates caused unstable loss trends and reduced accuracy, probably due to the model overshooting minima in the optimization process. Smaller batch sizes resulted in more oscillatory loss functions with slower convergence and higher final loss values. On the other hand, lower learning rates and higher batch sizes improved stability but produced excessive computational costs, significantly increasing training times.
To further enhance the model, the most influencing architecture-related parameters were tuned using a random search approach. The adjusted key parameters include:

- Number of channels (Ch_i) and kernel size (k_i) for each convolutional layer
- Number of units (FC_{U_i}) for each Fully Connected layer

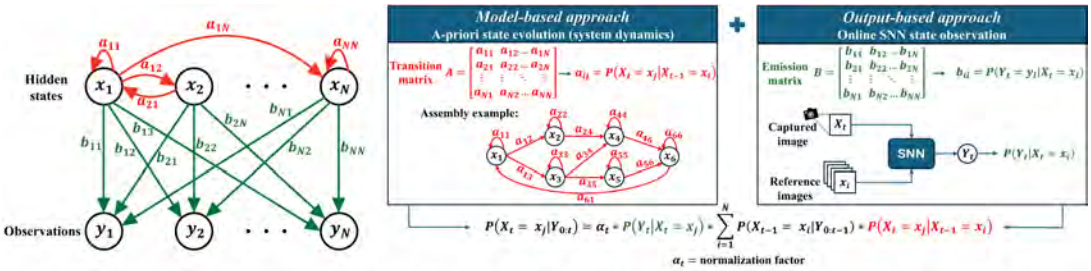
Final optimized values, derived from the overall tuning processes, are detailed in Figure 2 and were then employed for model training.
Stride values for both convolutional and max pooling layers (respectively St_i and St_{P_i}), as well as the kernel size for max pooling (k_{P_i}), were kept fixed and user-defined, as their impact was observed to be minimal compared to the other hyper-parameters.

After tuning, the network was trained for 100 epochs. The whole dataset was split into the training and validation sets, respectively, corresponding to 85% and 15% of the original. The resulting training and validation curves, depicted in Figure 4, show the trends over the epochs for loss and accuracy. A dissimilarity threshold (Th_{diss}) equal to 1 was used to compute the accuracy, classifying input image pairs as identical if their dissimilarity was below this threshold and different otherwise.
The resulting curves reveal a strong positive training trend, reflecting a significant improvement over epochs in the model's capability to classify instances correctly. The loss, initially oscillatory, stabilizes during later epochs, converging to approximately 0.01, and the accuracy steadily improves over epochs, ultimately reaching a near-perfect value of 0.99. Moreover, validation curves closely track training curves, indicating no overfitting and generalization capability to unseen data.



(a) Loss (b) Accuracy

Figure 4.: Training and validation curves for hyperparameters reported in Figure 2



(a) HMM structure and general parameters

(b) HMM data flow: transition probabilities from assembly dynamics and emission probability b_{li} from SNN observation Y_t

Figure 5.: Hidden Markov Model architecture and data flow

4. Integrating Hidden Markov Model with Siamese Network observations for enhanced assembly step recognition

As explained in Section 3, the method initially employs an SNN to recognize operations and detect errors. However, it relies only on the network's output, ignoring the dynamics of the process. Essentially, the SNN provides a snapshot of the current assembly state without considering how the process evolves over time. This approach does not use prior knowledge about the feasible sequences of operations to complete the assembly correctly. Instead, it relies only on the instantaneous predictions performed by the network.

To address this limitation, an HMM was introduced to model the assembly process. This new framework integrates the data-driven approach provided by the SNN with a model-based understanding of the process dynamics. The SNN delivers instantaneous observations related to the current hidden state. At the same time, the overall HMM framework refines the state predictions by incorporating knowledge about the expected sequences of operations, as shown in Figure 5a.

4.1. *HMM filtering*

Considering an assembly process made of M steps, an HMM-based filtering process is implemented to compute the probability distribution of the hidden state X_t at time

instant t, given the sequence of SNN observations $Y_{0:t}$ up to now:

$$P(X_t|Y_{0:t}) = [P(X_t = x_1|Y_{0:t}), P(X_t = x_2|Y_{0:t}), ..., P(X_t = X_N|Y_{0:t})]$$

with N the number of possible states describing the assembly process.
The forward recursion algorithm, going from step 0 to step $(M-1)$, is used to get the current assembly state.
At the first iteration, $t = 0$, the probability $P(X_0 = x_j|Y_0)$ is computed as:

$$P(X_0 = x_j|Y_0) = \alpha_0 \cdot P(Y_0|X_0 = x_j) \cdot P(X_0 = x_j) \tag{2}$$

where:

$$\alpha_0 = \frac{1}{\sum_{i=1}^{N} P(Y_0|X_0 = x_i) \cdot P(X_0 = x_i)} \tag{3}$$

$j = 1, ..., N$ and $P(X_0 = x_j)$ being the stationary probability of state x_j.
Then, for the next iterations, the filtering algorithm can be divided into two steps:

- *Prediction step*:

$$P(X_t = x_j|Y_{0:t-1}) = \sum_{i=1}^{N} P(X_{t-1} = x_i|Y_{0:t-1}) \cdot P(X_t = x_j|X_{t-1} = x_i) \tag{4}$$

- *Update step*:

$$P(X_t = x_j|Y_{0:t}) = \alpha_t \cdot P(Y_t|X_t = x_j) \cdot P(X_t = x_j|Y_{0:t-1}) \tag{5}$$

  where:

$$\alpha_t = \frac{1}{\sum_{i=1}^{N} P(Y_t|X_t = x_i) \cdot P(X_t = x_i|Y_{0:t-1})} \tag{6}$$

$j = 1, ..., N$,$P(X_t = x_j|X_{t-1} = x_i)$ being the transition probability from state x_i to x_j, and $P(Y_t|X_t = x_j)$ the emission probability of the SNN observation Y_t given the current state x_i at time instant t , as shown in Figure 5b.
To compute the probability distribution at each assembly step, allowing the final prediction of the hidden state as the one with the highest probability, the definition of the transition matrix, the emission matrix, and the stationary distribution is required.

4.1.1. State evolution: transition matrix and stationary distribution

To integrate the prior knowledge about the assembly state evolution, the transition probabilities from one state to the next are defined as:

$$P(X_t = x_j|X_{t-1} = x_i) = a_{ij}$$

$i, j = 1, ..., N$.
The general formulation of the transition matrix A is:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \dots & a_{NN} \end{bmatrix}$$

which is an $N \times N$ square matrix and is predefined and fixed.
Knowing the previous assembly state X_{t-1}, the probability of the current state is equally distributed among all possible immediate next states and the previous state itself. This probabilistic framework allows the operator the flexibility to either proceed with the assembly or leave the assembly state unchanged.
Furthermore, the first iteration of the forward algorithm requires the definition of the stationary distribution, i.e., the state probability distribution at time instant $t = 0$:

$$\Pi = P(X_0) = [P(X_0 = x_1), P(X_0 = x_2), ..., P(X_0 = x_N)] = [\pi_1, \pi_2, ..., \pi_N]$$

where π_j, with $j = 1, ..., N$, is predefined. In particular, π_1 is chosen close to 1 to privilege the first assembly state, as it represents the initial assembly condition required to start the process, while π_j is computed as:

$$\pi_j = \frac{1 - \pi_1}{N - 1} \tag{7}$$

$j = 2, ..., N$.
This formulation allows the model to set a strong prior bias toward the expected starting state x_1 while assigning a small non-zero probability to all other states. This choice ensures robustness to hand-tracking errors at the beginning of the process. For instance, if the first state is missed, due to tracking delays or missed detections, resulting in an initial observation corresponding to a later state, setting $\pi_j \neq 0$ for $j > 1$ prevents the model from rejecting potentially correct SNN predictions.

4.1.2. State observation: threshold customization and emission matrix

The second key element of the HMM formalism is the emission matrix, which encloses the probability distribution of the network observations given the state:

$$P(Y_t = x_l | X_t = x_i) = b_{il}$$

$i, l = 1, ..., N$.
The general formulation of the emission matrix B is:

$$B = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1N} \\ b_{21} & b_{22} & \dots & b_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ b_{N1} & b_{N2} & \dots & b_{NN} \end{bmatrix}$$

which is a $N \times N$ square matrix. Each row represents a probability distribution. Specifically, the element b_{il} is the probability of observing the feature embedding of state x_l when the true hidden state is x_i.
Note that since the SNN already provides an estimate of the present state, the emission probabilities are simply related to the accuracy obtained after the training process.
Matrix B is computed offline based on the reference dataset, containing N_{ref} images for each assembly state. The process involves the SNN to measure the dissimilarity between images from this dataset, which is used to estimate the probabilities of observing certain outputs given each specific hidden state.
A customized threshold value Th_{diss} is required to obtain network predictions. It is empirically computed for the specific assembly process based on the reference dataset as follows:

- Compute the average dissimilarity for:
  - Similar pairs (D_{equal}): images of the same assembly step.
  - Different pairs (D_{diff}): images of different assembly step.
- Calculate the customized threshold Th_{diss} as a weighted average:

$$Th_{diss} = D_{equal} + \beta \cdot (D_{diff} - D_{equal}) \tag{8}$$

  with $\beta = \frac{1}{7}$, empirically tuned.

The computation of the emission matrix can be summarized with the following steps.

- *Image pair definition and dissimilarity computation*
  An image in the reference dataset is chosen as the hidden state $X_t = x_i$. It is compared with N reference images, one from each reference state in the dataset. The SNN outputs the dissimilarity value as the Euclidean distance for each image pair (actual state, reference state).
- *Network prediction*
  For the actual state $X_t = x_i$, the network prediction Y_t is the reference state x_l corresponding to the lowest dissimilarity value, provided it is below the threshold Th_{diss}.
- *Iterative computation of prediction occurrences*
  The process is repeated for all combinations of reference images and every possible actual state x_i. For each actual state x_i, the number of predictions C_{il} of reference state x_l is recorded.
- *Emission probabilities computation*
  For each actual state x_i, the prediction occurrences are normalized to compute the required emission probabilities:

$$P(Y_t = x_l | X_t = x_i) = b_{il} = \frac{C_{il}}{\sum_{k=1}^{N} C_{ik}} \tag{9}$$

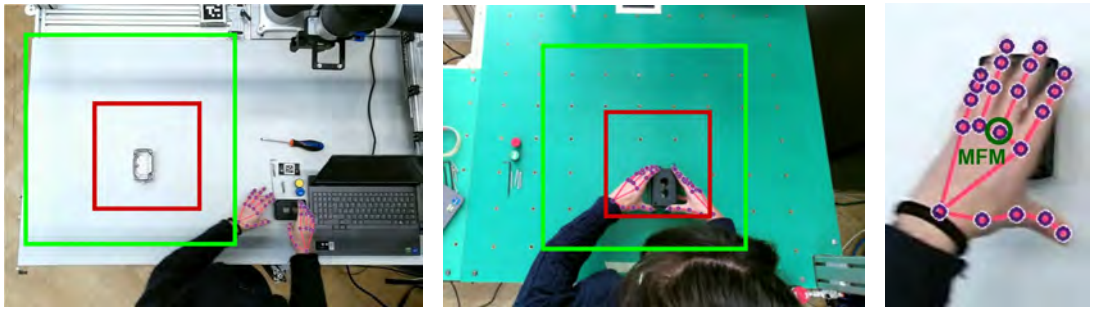
5. Experimental validation

This Chapter comprehensively summarizes the tests conducted to validate the proposed methodology. It includes an ablation study highlighting the performance improvements achieved by the new SNN model compared to the previous (M. Pelosi *et al.* 2024) and by introducing the HMM framework. In this Section, the experimental setup is described, as well as all the experiments performed to prove the robustness of

the method. Tests were conducted across different environmental conditions, human operators, and production faults.

5.1. *Experimental setup and hand-tracking strategy for detecting assembly step completion*

As SNN recognition requires an image of the assembled part at each assembly stage, tracking the operator's hands to highlight when each operation has been completed is necessary.
All the experiments exploit a Kinect V2 vision sensor. The camera was mounted above the working table, providing a suitable field of view to monitor the operator's hand movements during the assembly process while simultaneously capturing detailed visual features of the object.
The Mediapipe Hand Landmarker solution (F. Zhang *et al.* 2020) was used to track the operator's hands. Precisely, the position of the Middle Finger Metacarpophalangeal joint (MFM in Figure 6c) of both hands, corresponding approximately to the center of the hand, was monitored. When both MFMs leave the green reference working area, shown in Figure 6, to grab the next assembly component, the current step is considered completed. An image of the current assembly state, corresponding to the red assembly area, is taken and given as input to the SNN. Using a hand-tracking strategy avoids the need for explicit human communication or complex Human-Machine Interfaces (HMIs) to signal the operation's completion, enabling a seamless execution of the assembly. To ensure process continuity, avoiding production downtime caused by hand-tracking failures or occlusions, the following strategy was adopted: if the tracking system fails to detect one hand, due to either model limitations or occlusions caused by the other hand or external elements, the visible hand is still tracked and used to determine the completion of the assembly step. In cases where both hands are temporarily undetected, the system waits until no hands are present in the reference working area for a predefined number of consecutive frames before capturing the current assembly image, to reduce the risk of incorrect detections. This approach assumes that at the end of each assembly step, the operator removes the hands and any tools from the predefined working area. This assumption is based on the workspace layout, where components and tools are intentionally placed outside the green reference assembly zone. As a result, the operator naturally exits this area to retrieve the next item, enabling the system to detect the end of a step based on hand-tracking data. This design reflects common real-world industrial settings, where workspaces are organized to minimize clutter, maximize visibility, and support quality inspection or automation tasks in central assembly zones. Therefore, this assumption is not only valid in our experimental setup but also generalizable to many industrial contexts.
A Lenovo Legion 5 laptop, equipped with an Intel Core i7 processor and an NVIDIA GeForce RTX 3060 GPU, was used for both model training and experimental validation. Combined with a Windows 11 operating system and Python 3.9-based development environment, this hardware configuration ensures the computational efficiency required to run the algorithm online, maintaining consistent performance with minimal latency.
The experiments exploited two types of working tables, shown in Figures 6a and 6b. First, a uniform-white background was chosen to replicate the environmental conditions used during the composition of the training dataset. Then, a green table with threaded holes was also tested. This choice is motivated by the need to evaluate the



(a) Uniform-white background

(b) Holed-green background

(c) Middle Finger Metacarpophalangeal joint

Figure 6.: Experimental setup and hand-tracking strategy

method's robustness and adaptability to different industrial environments.

5.2. *Push-button panel assembly: tests and results*

To prove the method's performance, a push-button panel was chosen for the tests[2]. It includes all the challenges of the recognition problem to assess the model's capabilities. The assembly includes two buttons: a red button with a matte surface and a yellow-green striped button with a shiny surface. These buttons allow testing the ability to discern colors and robustness to possible reflections caused by the button with the shiny surface. In addition, four small screws complete the assembly, introducing a subtle visual detail that is challenging to detect. It is important to notice that the push-button panel assembly process was not part of the SNN training dataset, demonstrating the model's ability to generalize to unseen tasks.
The possible sequences to complete the assembly process are represented in the probabilistic automaton of Figure 7. The assembly involves five steps and can be completed in six valid sequences:

- *Assembly sequence 1*: $0 \rightarrow 1 \rightarrow 2\text{a} \rightarrow 3\text{a} \rightarrow 4$
- *Assembly sequence 2*: $0 \rightarrow 1 \rightarrow 2\text{a} \rightarrow 3\text{b} \rightarrow 4$
- *Assembly sequence 3*: $0 \rightarrow 1 \rightarrow 2\text{b} \rightarrow 3\text{a} \rightarrow 4$
- *Assembly sequence 4*: $0 \rightarrow 1 \rightarrow 2\text{b} \rightarrow 3\text{c} \rightarrow 4$
- *Assembly sequence 5*: $0 \rightarrow 1 \rightarrow 2\text{c} \rightarrow 3\text{b} \rightarrow 4$
- *Assembly sequence 6*: $0 \rightarrow 1 \rightarrow 2\text{c} \rightarrow 3\text{c} \rightarrow 4$

Overall, the assembly can take nine possible states during its evolution: 0, 1, $2a$, $2b$, $2c$, $3a$, $3b$, $3c$, and 4.
The transition matrix A and the stationary distribution Π, defined a-priori as detailed

[2]A demonstrative video of the push-button panel tests can be found here: `https://youtu.be/sFiczOTDWjk`

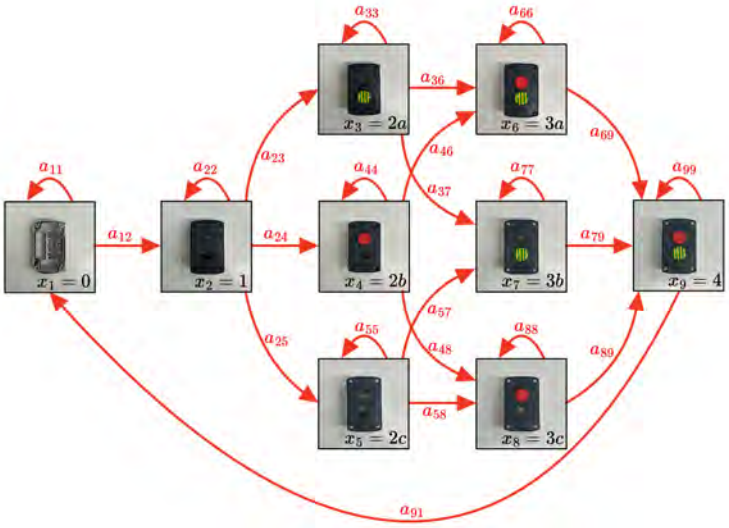


Figure 7.: Feasible transitions between hidden states of the push-button panel assembly

in Section 4.1.1, are given below:

$$A = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \end{bmatrix}, \Pi = \begin{bmatrix} 0.8 \\ 0.025 \\ 0.025 \\ 0.025 \\ 0.025 \\ 0.025 \\ 0.025 \\ 0.025 \\ 0.025 \end{bmatrix}$$

The experimental validation focused on two key objectives: recognizing assembly steps and detecting production faults. The performance of the SNN alone was compared to that of the complete HMM framework.

Regarding the first task, two sets of tests were conducted. The first test, discussed in Section 5.2.1, evaluated the method's robustness to background variations. In fact, by introducing FastSAM-based object segmentation during the image pre-processing stage, the method is designed to perform effectively on backgrounds different from those used during training without any degradation in performance. The second set of tests, shown in Section 5.2.2, involved multiple human operators to verify that the SNN model was robust enough to handle the variability introduced by the human factor.

Furthermore, to prove the method's capability to effectively integrate assembly state recognition with error detection, tests were conducted on two types of production errors introduced by the operator during the assembly process, as will be explained in

Section 5.2.3.

5.2.1. Assembly step recognition: robustness to different backgrounds

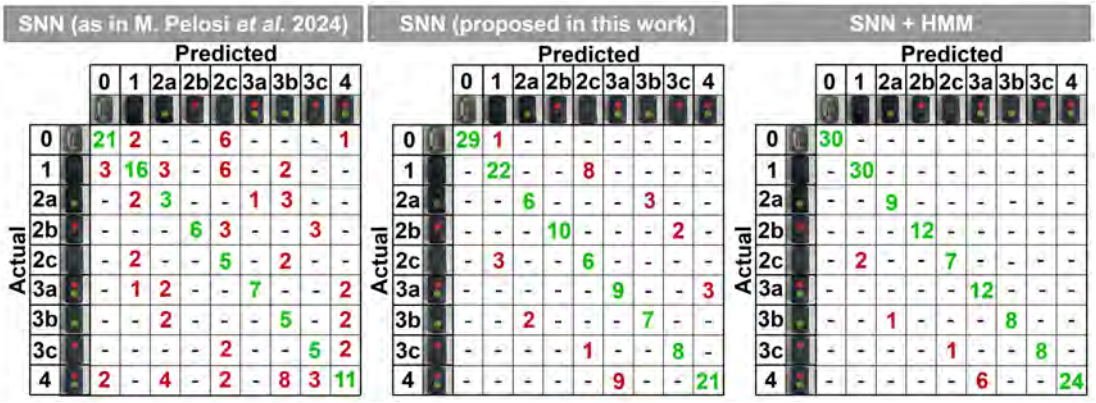
All six possible assembly sequences were tested five times each to validate the model, resulting in 30 complete assembly processes and 150 assembly steps to recognize. For each experiment, the position of the part in the assembly area was slightly changed to prove the rotational and translational invariance of the trained model.
These tests were performed using both the uniform-white and holed-green backgrounds. The reference dataset for each working table was collected offline by capturing 10 images for each assembly state.
Based on the reference datasets, the emission matrix B and the dissimilarity threshold Th_{diss} were computed for both backgrounds following the procedure detailed in Section 4.1.2. The resulting matrices and thresholds are shown below:

$$B_{white} = \begin{bmatrix} \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{0.73} & 0.01 & 0 & 0.25 & 0 & 0.01 & 0 & 0 \\ 0 & 0 & \mathbf{0.61} & 0 & 0 & 0 & 0.39 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{0.67} & 0 & 0 & 0 & 0.33 & 0 \\ 0 & 0.27 & 0.01 & 0 & \mathbf{0.71} & 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{0.59} & 0 & 0 & 0.41 \\ 0 & 0 & 0.42 & 0 & 0 & 0 & \mathbf{0.58} & 0 & 0 \\ 0 & 0 & 0 & 0.24 & 0 & 0.01 & 0 & \mathbf{0.75} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.35 & 0 & 0 & \mathbf{0.65} \end{bmatrix}$$

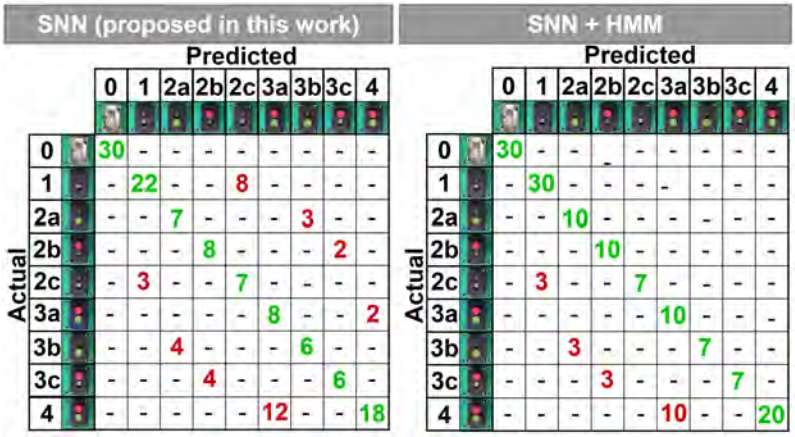
$$Th_{disswhite} = 0.61$$

$$B_{green} = \begin{bmatrix} \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{0.58} & 0 & 0 & 0.37 & 0 & 0.05 & 0 & 0 \\ 0.01 & 0.01 & \mathbf{0.65} & 0 & 0 & 0 & 0.31 & 0 & 0.02 \\ 0 & 0 & 0 & \mathbf{0.7} & 0 & 0 & 0.03 & 0.23 & 0.04 \\ 0 & 0.3 & 0 & 0 & \mathbf{0.67} & 0 & 0.03 & 0 & 0 \\ 0 & 0 & 0.01 & 0.02 & 0 & \mathbf{0.52} & 0 & 0.01 & 0.44 \\ 0.01 & 0.04 & 0.31 & 0 & 0.02 & 0 & \mathbf{0.61} & 0 & 0.01 \\ 0 & 0 & 0 & 0.38 & 0 & 0.02 & 0 & \mathbf{0.57} & 0.03 \\ 0 & 0 & 0 & 0.01 & 0 & 0.36 & 0 & 0.01 & \mathbf{0.62} \end{bmatrix}$$

$$Th_{dissgreen} = 0.87$$



(a) Uniform white background



(b) Holed green background

Figure 8.: Confusion matrices for step recognition tests on different backgrounds

The emission matrices show the highest values along the diagonal, indicating that the network can recognize most assembly states. However, only the assembly step 0 is always correctly identified by the SNN. For all the other states, off-diagonal entries with significant values reveal the network's tendency to confuse similar states. Looking at Figure 7, it is possible to notice that the mistakes are primarily related to the addition or removal of small screws that are difficult to see in the images. The pattern of the emission probabilities suggests the network has learned structural relationships in the data, as errors occur between visually similar states. Moreover, the matrices are similar for both backgrounds, revealing the SNN's robustness to background variations.
The results, in terms of confusion matrices, accuracy, macro-average precision, macro-average recall, and F1-score, are summarized in Figure 8 and Table 3a. These metrics compare the performance of the SNN alone and the complete HMM framework.
For the white background, the model previously proposed by M. Pelosi *et al.* (2024) was also tested to highlight the improvements of the updated approach. The confusion matrix of the previous model is characterized by several randomly positioned off-diagonal entries, which reveal unstructured recognition errors. On the contrary, the recognition mistakes of the new model are all related to the presence of the tiny screws, as anticipated by the pre-computed emission matrix. While the updated model improved performance, detecting such subtle details remains challenging. However, the

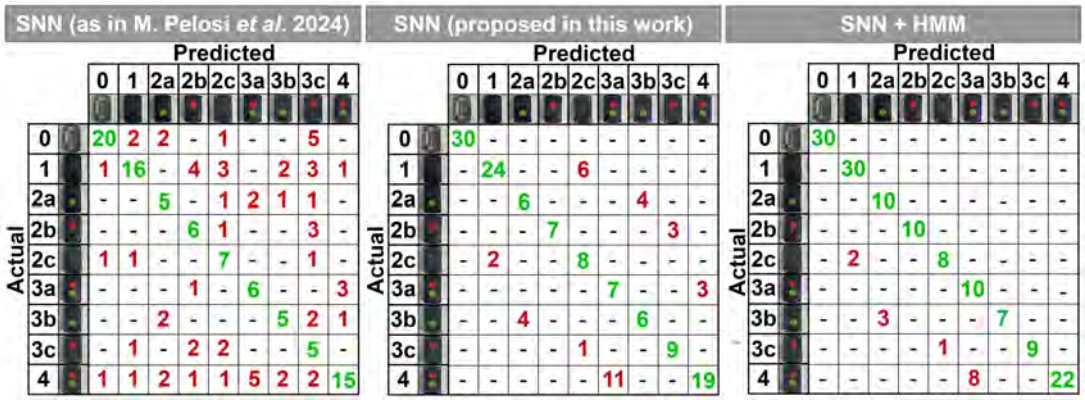


Figure 9.: Confusion matrices for step recognition tests with 10 different operators

HMM framework significantly improves the accuracy, solving the majority of the visual issues caused by the screws. This proves that by accounting for the assembly dynamics, the HMM framework can provide information about the evolution of the assembly state, greatly enhancing the accuracy in recognizing the overall assembly sequence. The remaining errors are related to SNN misrecognitions of feasible states in the transition tree that the HMM framework is not able to filter out.
Regarding the tests performed on the green table with threaded holes, a slight degradation in recognition performance was observed. This can be attributed to less accurate FastSAM-based object segmentation, as the holes occasionally interfere with background removal.

(a) Test results for assembly step recognition with different backgrounds and 10 different operators: ablation study with the previous SNN model (M. Pelosi *et al.* 2024), the SNN model proposed in this work, and the combined SNN-HMM model.

	DIFFERENT BACKGROUNDS					DIFFERENT OPERATORS		
	White-uniform			Green-holed				
	SNN (M. Pelosi *et al.* 2024)	SNN (this work)	SNN + HMM	SNN (this work)	SNN + HMM	SNN (M. Pelosi *et al.* 2024)	SNN (this work)	SNN + HMM
Accuracy	0.527	0.787	**0.933**	0.747	**0.873**	0.567	0.773	**0.907**
Precision	0.569	0.763	**0.931**	0.707	**0.883**	0.542	0.740	**0.906**
Recall	0.520	0.776	**0.928**	0.726	**0.863**	0.567	0.748	**0.904**
F1-score	0.543	0.769	**0.930**	0.716	**0.873**	0.554	0.744	**0.905**

(b) Test results for error detection at different assembly stages: comparison between the previous SNN model (M. Pelosi *et al.* 2024) and the SNN model proposed in this work.

	WRONG BUTTON PLACED				WRONG CASE PLACED			
	Blue button		Green button		Striped case		Damaged case	
	SNN (M. Pelosi *et al.* 2024)	SNN (this work)	SNN (M. Pelosi *et al.* 2024)	SNN (this work)	SNN (M. Pelosi *et al.* 2024)	SNN (this work)	SNN (M. Pelosi *et al.* 2024)	SNN (this work)
Step 1	-	-	-	-	9/15	11/15	10/15	12/15
Step 2a	3/5	4/5	2/5	4/5	-	-	-	-
Step 2b	1/5	3/5	2/5	4/5	-	-	-	-
Step 3a	4/10	7/10	7/10	8/10	-	-	-	-
Step 3b	4/5	5/5	3/5	4/5	-	-	-	-
Step 3c	2/5	2/5	2/5	3/5	-	-	-	-
Step 4	6/10	7/10	7/10	8/10	-	-	-	-
Accuracy	0.50	**0.70**	0.575	**0.775**	0.60	**0.733**	0.667	**0.80**

Table 3.: Test results for the push-button panel assembly: assembly state recognition and fault detection.

5.2.2. Assembly step recognition: robustness to human variability

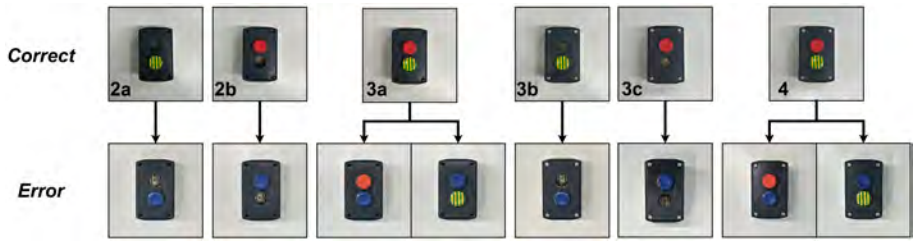
To evaluate the method's robustness to human variability, 10 operators were asked to assemble the push-button panel three times each, selecting the assembly sequence randomly for every attempt. In this experiment, the uniform white background was exclusively used to isolate the effect of human variability from the background's influence. These experiments are motivated by the need to assess the method's robustness in handling variations in part placement, which may arise from individual user experience or implicit assembly strategies. In addition, the people involved were not informed about the system's working principle to verify the possibility of deploying the application in real-world scenarios without requiring specialized training for the workers.
The total number of occurrences of each assembly sequence is reported below:

- *Assembly sequence 1*: 0 → 1 → 2a → 3a → 4, **5 times**
- *Assembly sequence 2*: 0 → 1 → 2a → 3b → 4, **4 times**
- *Assembly sequence 3*: 0 → 1 → 2b → 3a → 4, **7 times**
- *Assembly sequence 4*: 0 → 1 → 2b → 3c → 4, **5 times**
- *Assembly sequence 5*: 0 → 1 → 2c → 3b → 4, **5 times**
- *Assembly sequence 6*: 0 → 1 → 2c → 3c → 4, **4 times**

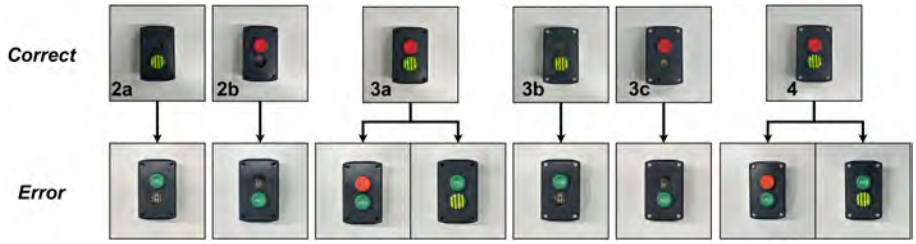
The results, in terms of confusion matrix, accuracy, macro-average precision, macro-average recall, and F1-score, are summarized in Figure 9 and Table 3a. Again, these metrics compare the performance of the old SNN model, the updated SNN model, and the complete HMM framework.
The results confirm that the variability introduced by changing human operators does not significantly impact performance, as similar KPI values were found in the previous experiments involving a single operator. The new SNN model demonstrates improved accuracy, precision, recall, and F1-score compared to the old model, while the HMM framework further enhances these metrics. As shown in Table **??**, the HMM framework achieves the highest accuracy and robustness in step recognition with different operators, highlighting its ability to effectively account for assembly dynamics. SNN misrecognitions that do not align with feasible assembly sequences are all filtered out.

5.2.3. Production fault detection

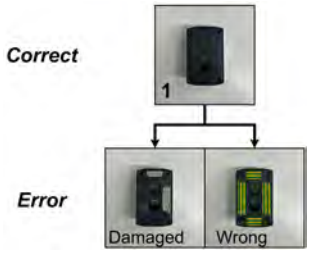
Two types of production faults were tested to evaluate the system's ability to detect errors during the assembly process: placement of wrong buttons and wrong external cases. First, two incorrect buttons, blue and green, were inserted in place of the red or striped buttons. These errors were tested at all feasible assembly stages, as shown in Figures 10a and 10b. The blue and green buttons replaced the red or striped buttons at the third, fourth, and final assembly steps. Each of these error scenarios was tested 5 times, resulting in 40 tests per button color. Second, the black external case was replaced with two incorrect cases, which can be seen in Figure 10c: one "damaged" and the other with a different design. To simulate the faded surface of the damaged case, some random white/grey zones were reproduced on the surface, while the wrong case presents some yellow and green stripes instead of being monochromatic. For these experiments, 15 tests per case type were conducted.
Only the performances of the original (M. Pelosi *et al.* 2024) and revised SNN models were compared for the error detection task. No comparison between SNN and HMM performances is provided, as the SNN logic manages error detection. Either involving the SNN alone or the overall HMM framework, an error message is instantaneously



(a) Wrong button placement (blue)



(b) Wrong button placement (green)



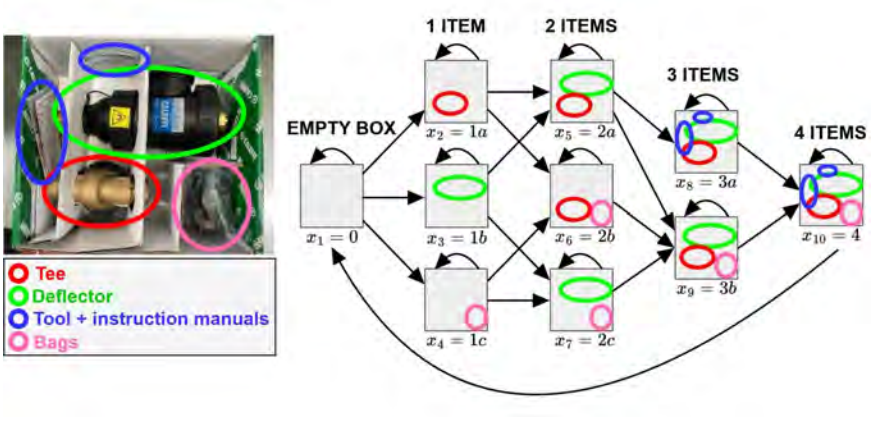
(c) Damaged and wrong case

Figure 10.: Tested assembly errors.

triggered when all dissimilarity values exceed the threshold, and the current assembly state cannot be recognized. All experiments were performed on the uniform-white background, applying the customized threshold computed for that environment.
The results, summarized in Table 3b, show a significant performance improvement with the new SNN model. Both models generally performed better in detecting errors with substantial visual differences. Detecting the placement of the wrong striped case is the most straightforward task, as it introduces significant chromatic variations compared to the reference state 1. On the other hand, detecting the blue button insertion was more challenging, as the dark blue color was less visible in the images.

6. Real industrial use-case

The push-button panel assembly, described in Section 5, serves as an exhaustive validation case to prove the method's robustness to different geometrical and chromatic features, environmental conditions, and human variability. It also provides significant examples of execution faults to demonstrate the network's ability to detect deviations from reference states.
However, this scenario involves a controlled manual assembly process that slightly differs from real-world industrial applications, which are often characterized by noisy and cluttered environments and more complex and varied components. To assess the



(a) Assembly process

(b) Assembly workstation captured by the 3D vision sensor Smart Robots

Figure 11.: Caleffi Hydronic Solutions use case: packaging of hydronic components.

method's applicability in real-world industrial contexts, it was applied to a practical use case involving the manual packaging of hydronic components on an active assembly line at Caleffi Hydronic Solutions in Novara (Italy). Unlike the push-button panel scenario, which was specifically designed for validation purposes, this industrial use case was not tailored to the method but represents a well-established and consolidated manual process provided by the company. Notably, the packaging assembly process was also not included in the training dataset, providing an additional demonstration of the model's generalizability to different tasks.

Figure 11a represents all feasible assembly states and transitions. The assembly process consists of five steps and can take ten possible states: 0, $1a$, $1b$, $1c$, $2a$, $2b$, $2c$, $3a$, $3b$, and 4.

Five reference images per assembly state were used to create the reference dataset for this use case. The transition matrix A_{pack}, the stationary distribution Π_{pack}, and the emission matrix B_{pack} are provided below.

$$A_{pack} = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} & \frac{1}{3} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \end{bmatrix}, \Pi_{pack} = \begin{bmatrix} 0.8 \\ 0.022 \\ 0.022 \\ 0.022 \\ 0.022 \\ 0.022 \\ 0.022 \\ 0.022 \\ 0.022 \\ 0.022 \end{bmatrix}$$

$$B_{pack} = \begin{bmatrix} \mathbf{0.8} & 0 & 0 & 0 & 0.04 & 0 & 0.16 & 0 & 0 & 0 \\ 0 & \mathbf{0.36} & 0 & 0.08 & 0 & 0.24 & 0 & 0 & 0.32 & 0 \\ 0 & 0 & \mathbf{0.56} & 0 & 0.32 & 0 & 0.12 & 0 & 0 & 0 \\ 0 & 0.08 & 0 & \mathbf{0.68} & 0 & 0.2 & 0 & 0.04 & 0 & 0 \\ 0 & 0 & 0.12 & 0 & \mathbf{0.44} & 0 & 0.04 & 0 & 0.4 & 0 \\ 0 & 0.36 & 0 & 0.04 & 0 & \mathbf{0.44} & 0.04 & 0 & 0.12 & 0 \\ 0 & 0 & 0.2 & 0 & 0.24 & 0.08 & \mathbf{0.44} & 0 & 0.04 & 0 \\ 0 & 0 & 0 & 0.08 & 0 & 0 & 0 & \mathbf{0.56} & 0 & 0.36 \\ 0 & 0 & 0 & 0 & 0.44 & 0 & 0.04 & 0 & \mathbf{0.52} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0 & \mathbf{0.8} \end{bmatrix}$$

$$Th_{disspack} = 1.06$$

Hydronic components were packaged five times by an operator working in the assembly line, recorded using the 3D vision sensor Smart Robots[3], as shown in Figure 11b. The operator followed the same sequence of operations each time: 0 → 1a → 2a → 3a → 4.

Table 4 summarizes the results for assembly state recognition, comparing the performance of the new SNN alone and the whole HMM framework. The resulting performance is slightly decreased compared to the value obtained with the push-button panel assembly due to the more complex and noisy industrial environment. However, the recognition accuracy of the HMM framework is still relatively high, demonstrating the model's generalizability across different and unseen use cases and the method's effectiveness when applied to real and consolidated industrial workflows.

[3]Smart Robots company website: `https://smartrobots.it/en/`

Table 4.: Test results for Caleffi Hydronic Solutions use-case: comparison between SNN alone and combined SNN-HMM framework.

	Step 0	**Step 1a**	**Step 2a**	**Step 3a**	**Step 4**	**Accuracy**
SNN proposed in this work	5/5	2/5	3/5	3/5	4/5	0.68
SNN + HMM	5/5	3/5	4/5	4/5	4/5	**0.80**

7. Managerial insights

The proposed approach offers significant advantages from an industrial management perspective, making it a practical, cost-effective, and scalable solution for real-world production quality assurance. The main advantages are listed here to better highlight the relevance of this solution:

- **Reduced rework and waste.** Step-by-step fault detection prevents error propagation along the production line, significantly reducing rework costs and production waste, major sources of annual operational losses in industrial settings.
- **Plug-and-play deployment.** The system requires no retraining. Only a few reference images are needed for new assemblies, enabling quick adaptation with minimal effort.
- **Improved Return Of Investment (ROI).** By reducing inefficiencies, supporting consistent product quality, and minimizing retraining costs, the system reduces the payback period, lowering deployment overheads and improving cost benefits.
- **No specialized personnel training required.** Designed to be easy to use, the method provides intuitive feedback to operators, who do not need technical knowledge of the algorithm's working principle. Experimental tests with users unfamiliar with the system confirmed its reliability and ease of use for the operators.
- **Reduced data collection time in manual processes.** Since the system operates online and is seamlessly integrated with ongoing human activities, it reduces the time and effort for data collection in manual industrial processes.

8. Conclusions

Positioned in the subfield of defect detection within the ZDM paradigm, this paper introduces a novel approach that integrates a Siamese Neural Network into a Hidden Markov Model to address the challenges of assembly operation recognition and error detection in flexible, human-operated manufacturing environments. Leveraging the SNN for visual comparison and the HMM framework to account for process dynamics, the proposed approach exhibits high adaptability to various assembly scenarios without the need for retraining in the case of new products. It only requires the offline collection of a set of reference images and the customization of the error detection threshold to suit the specific assembly case.
Experimental validation on both a lab-controlled test case and a real industrial use case, the latter not specifically tailored for the method, demonstrated the method's strong robustness to human variability, lighting changes, and background differences. The model achieved up to 93% accuracy in assembly step recognition and 80% in error detection, with consistent performance in the presence of different operators and

environments. The use of the HMM to filter SNN outputs based on logical assembly sequences, combined with FastSAM-based background removal, significantly improved prediction reliability. An ablation study confirmed that these additions improved classification accuracy up to 40%, compared to the previous version of the SNN model (M. Pelosi *et al.* 2024).
These results highlight the model's potential as a practical decision-support tool to improve product quality, reduce rework costs and production waste, and enhance process reliability in real-world production systems.

8.1. *Future perspectives*

Despite the promising results in real setups, the method currently relies on assumptions, such as top-view visibility of assembly processes and no need for assembly fault classification. Such assumptions, although valid in many industrial contexts, may limit its applicability in more complex and cluttered environments or in scenarios where the main purpose is error classification. For this reason, multi-view setups will be explored to handle occlusions and improve component visibility from different angles. In addition, fault classification will be investigated by incorporating labeled error data to enable not only detection but also identification of error types. Efforts will also be made to enhance the system's sensitivity to subtle variations by increasing the input images' resolution and by expanding the dataset, including more diverse components. Although the method's capabilities have been thoroughly tested using laboratory data, future work could further evaluate its fault detection performance in the context of real-world industrial assembly errors. Such validation was not conducted in the present study, as the data provided by the company included only correctly assembled cases. Additionally, extending the evaluation to encompass a wider range of faults, such as incorrect relative orientations, component misalignments, skipped steps, or sequencing errors, would provide further opportunities to demonstrate the versatility of the HMM framework in detecting deviations from the expected dynamics of the assembly process.

Acknowledgements

The authors gratefully acknowledge Caleffi Hydronic Solutions for supplying videos and images of their production stations, which were essential for validating the method in real-world industrial applications.

Data availability statement

The training dataset used to train the Siamese Network model is openly available at the following link:
`https://osf.io/cvwqp/?view_only=bd2b68dd24a34bf3974701a388e38033`

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This study was partially carried out within the MICS (Made in Italy – Circular and Sustainable) Extended Partnership and received funding from Next-Generation EU (Italian PNRR – M4 C2, Invest 1.3 – D.D. 1551.11-10 2022, PE00000004). CUP MICS D43C22003120001

Notes on contributor(s)

Martina Pelosi was born in Piacenza (Italy) in 1998. She received her Master's degree cum laude in Mechanical Engineering from Politecnico di Milano in 2022. From November 2022, she is a national PhD student in Artificial Intelligence at Politecnico di Torino. Her research focuses on improving the efficiency of human-robot collaborative production lines.

Andrea Maria Zanchettin was born in Cremona (Italy) in 1983. He received his PhD in Information Technology, with honour, from Politecnico di Milano in 2012. From January 2012 until February 2014 he has been a temporary research assistant at the Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB). From March 2014 until September 2019 he has been a fixed-term assistant professor at DEIB where he is now a tenured Associate Professor. His research interests are about mechatronic systems, automatic control, and intelligent human-robot interaction.
Andrea Zanchettin has been member of the IEEE Robotics and Automation Society since 2009. Since 2017, he has been co-founder and co-chair of the IEEE RAS Technical Committee on Collaborative Automation for Flexible Manufacturing (CAFM). Dr. Zanchettin has been chair of the Italian Chapter of the IEEE RAS (I-RAS) from 2019 until 2022, as well as vice president for industrial activities of the Italian Institute of Robotics and Intelligent Machines (I-RIM). He is also co-founder and member of the Board of Directors of Smart Robots, a spin-off company of Politecnico di Milano.

Paolo Rocco is a Full Professor in automatic control and robotics with the Politecnico di Milano, Milan, Italy. He is also a Co-Founder of Smart Robots, a spin-off company of Politecnico di Milano. He has authored about 200 papers in the areas of robotics, motion control, and mechatronics. His current research interests include industrial robotics, with a particular focus on safe and productive human-robot interaction.
Dr. Rocco has served in various positions in the editorial boards of journals and conferences. At present, he serves as a Senior Editor for the IEEE Transactions on Automation Science and Engineering and as an Associate Editor for other journals. He serves in the Board of Directors of euRobotics, the association of all stakeholders in robotics in Europe, in the Board of Directors of I-RIM, the Italian Institute of Robotics and Intelligent Machines, and in the Board of Governors of Politecnico di Milano. He is also head of the Systems and Control area at Politecnico di Milano and responsible for projects in MADE, the Industry 4.0 Competence Center in Milan. He has been in charge of several research projects with industrial partners and public bodies.

References

M. Pelosi, L. Repizzi, A. M. Zanchettin and P. Rocco (2024). *Siamese Network for assembly step recognition and quality assessment for Human-Robot Collaboration.* IEEE 2024 20th International Conference on Automation Science and Engineering (CASE). 3811-3817.

Y. Torres, S. Nadeau and K. Landau (2021). *Classification and quantification of human error in manufacturing: a case study in complex manual assembly.* Applied Sciences, MDPI.

A. Ajoudani, A. M. Zanchettin, S. Ivaldi, A. Albu-Schäffer, K. Kosuge and O. Khatib, (2018). *Progress and prospects of the human-robot collaboration.* Autonomous robots.

A. Borboni, K. V. V. Reddy, I. Elamvazuthi, M. S AL-Quraishi, E. Natarajan and S. S. Azhar Ali (2023). *The expanding role of artificial intelligence in collaborative robots for industrial applications: A systematic review of recent works.* Machines, MDPI.

Y. Koren (2010). *The Global Manufacturing Revolution: Product-Process-Business Integration and Reconfigurable Systems.* John Wiley &Sons, Inc.

A. Metzmacher, T. Hellebrandt, M. Ruessmann, I. Heine and R. H Schmitt (2019). *Advances in Production Research: Proceedings of the 8th Congress of the German Academic Association for Production Technology (WGP).* Springer. 715-729.

F. Zhou, Y. Ji and R. J. Jiao (2013). *Affective and cognitive design for mass personalization: status and prospect.* Springer.

L. Merkel, J. Atug, C. Berger, S. Braunreuther and G. Reinhart (2018). *Mass customization and paperless assembly in the learning factory for cyber-physical-production systems: Learning module 'from paperbased to paperless assembly.* IEEE 2018 18th International Conference on Advanced Learning Technologies (ICALT). 270-271.

F. Psarommatis, G. May, V. Azamfirei (2024). *Zero defect manufacturing in 2024: a holistic literature review for bridging the gaps and forward outlook.* International Journal of Production Research. 1–37.

H. Liu and L. Wang (2018). *Gesture recognition for human-robot collaboration: A review.* International Journal of Industrial Ergonomics.

Gianfranco Genta, Maurizio Galetto and Fiorenzo Franceschini (2020). *Inspection procedures in manufacturing processes: recent studies and research perspectives.* International Journal of Production Research.

C. Chen, C. Zhang, T. Wang, D. Li, Y. Guo, Z. Zhao and J. Hong (2020). *Monitoring of assembly process using deep learning technology.* Sensors, MDPI.

G. Di Leo, C. Liguori, A. Paolillo, A. Pietrosanto, E. Adiutori and F. Promutico (2014). *Online visual inspection of defects in the assembly of electromechanical parts.* IEEE 2014 International Instrumentation and Measurement Technology Conference (I2MTC).

S. C. Bhardwaj (2016). *Detection and verification of missing components in SMD using AOI techniques.* International Journal of Computer Graphics.

M. Ojer, I. Serrano, F. Saiz, I. Barandiaran, I. Gil, D. Aguinaga and D. Alejandro (2020). *Real-time automatic optical system to assist operators in the assembling of electronic components.* The International Journal of Advanced Manufacturing Technology. 2261-2275.

A. Manimuthu, VG Venkatesh, V. Raja Sreedharan, V. Mani, Venkatesh (2022). *Modelling and analysis of artificial intelligence for commercial vehicle assembly process in VUCA world: a case study.* International Journal of Production Research. 4529-4547.

Z. Li, X. Wu, H. Du, H. Nghiem, G. Shi (2025). *Benchmark evaluations, applications, and challenges of large vision language models: A survey.* arXiv:2501.02189.

K. Qian, T. Sun, W. Wang (2024). *Exploring Large Vision-Language Models for Robust and Efficient Industrial Anomaly Detection.* arXiv:2412.00890.

L. Peifeng, L. Qian, X. Zhao, B. Tao, Bo (2024). *Joint knowledge graph and large language model for fault diagnosis and its application in aviation assembly.* IEEE Transactions on Industrial Informatics.

J. Yu, S. Li, X. Liu, Y. Gao, S. Wang, C. Liu (2023). *Dynamic convolutional gated recurrent unit attention auto-encoder for feature learning and fault detection in dynamic industrial processes.* International Journal of Production Research. 7434-745.

Y. Byun, D. Maeng, J. Baek (2024). *Unsupervised fault detection using frequency-wise angular filtering in contaminated vibration signals*. International Journal of Production Research. 1-27.

L. Eversberg, P. Grosenick, M. Meusel and J. Lambrecht (2021). *An industrial assistance system with manual assembly step recognition in virtual reality*. IEEE 2021 International Conference on Applied Artificial Intelligence (ICAPAI). 1-6.

S. Jadon and A. Jadon (2020). *An overview of deep learning architectures in few-shot learning domain*. arXiv:2008.06365.

J. Kurek, I. Antoniuk, B. Świderski, A. Jegorowa and M. Bukowski (2020). *Application of siamese networks to the recognition of the drill wear state based on images of drilled holes*. Sensors, MDPI.

Foivos Psarommatis, Gökan May, Paul-Arthur Dreyfus and Dimitris Kiritsis (2020). *Zero defect manufacturing: state-of-the-art review, shortcomings and future directions in research*. International Journal of Production Research. 1-17.

Y. Hui, X. Mei, G. Jiang, F. Zhao, Z. Ma, Ziwei and T. Tao (2022). *Assembly quality evaluation for linear axis of machine tool using data-driven modeling approach*. Journal of Intelligent Manufacturing. 1-17.

Y. Le, S. Qiang, and S. Liangfa (2012).*A novel method of analyzing quality defects due to human errors in engine assembly line*. IEEE 2012 International conference on information management, innovation management and industrial engineering. 154-157.

S. Thamm, L. Huebser, T. Adam, T. Hellebrandt, I. Heine, S. Barbalho, S. K. Velho, M. Becker, V. S Bagnato and R. H Schmitt (2021). *Concept for an augmented intelligence-based quality assurance of assembly tasks in global value networks*. Procedia CIRP. 423-428.

Mehret Getachew, Birhanu Beshah, Ameha Mulugeta and Daniel Kitaw (2024). *Application of artificial intelligence to enhance manufacturing quality and zero-defect using CRISP-DM framework*. International Journal of Production Research. 1-25.

Hironori Takimoto, Junya Seki, Sulfayanti F. Situju and Akihiro Kanagawa (2022). *Anomaly Detection Using Siamese Network with Attention Mechanism for Few-Shot Learning*. Applied Artificial Intelligence

M. Cai, G. Wang, X. Luo, X. Xu (2025). *Task allocation of human-robot collaborative assembly line considering assembly complexity and workload balance*. International Journal of Production Research. 1-27.

Z. Jia, S. Xie, W. Zhang (2025). *Flexible task assignment and assembly scheduling for human-robot collaboration cell considering uncertainty*. International Journal of Production Research. 1-21.

A. Riedel, J. Gerlach, M. Dietsch, S. Herbst, F. Engelmann, N. Brehm and T. Pfeifroth (2021) *A deep learning-based worker assistance system for error prevention: Case study in a real-world manual assembly*. Advances in Production Engineering & Management

D. Lehman, T. J Schoonbeek, S. H. Hung, J. Kustra, P. HN de With and F. van der Sommen (2024) *Find the Assembly Mistakes: Error Segmentation for Industrial Applications*. arXiv:2408.12945

D. Hendrycks and K. Gimpel (2016). *Gaussian error linear units (gelus)*. arXiv:1606.08415

F. Zhang, v. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C. L. Chang and M. Grundmann (2020) *Mediapipe hands: On-device real-time hand tracking*. arXiv:2006.10214

X. Zhao, W. Ding, Y. An, Y. Du, T. Yu, M. Li, M. Tang and J. Wang (2023). *Fast segment anything*. arXiv:2306.12156