Siamese Network for assembly step recognition and quality assessment for Human-Robot Collaboration

Siamese Network for assembly step recognition and quality assessment for Human-Robot Collaboration

Martina Pelosi[1,2], Letizia Repizzi[1], Andrea Maria Zanchettin[1], Paolo Rocco[1]

Abstract— **Interest in Human-Robot Collaboration is continuously growing in the industrial field, leveraging human cognitive capabilities with the consistent repeatability and reliability of robots to enable flexible production while enhancing working conditions. Within this context, the goal of this paper is to recognize assembly steps by presenting a Siamese Network-based approach adaptable to different assembly scenarios. The Siamese Network compares images captured during the process with a previously acquired reference dataset to identify the last executed assembly step and evaluate its correctness. The model's effectiveness was proved offline by comparing its performance with pre-existing OpenCV feature-matching tools. Additional online experiments were performed on three assembly cases exploiting a hand-tracking algorithm to communicate the operation's completion. The results highlight the versatility and reliability of the proposed methodology, which offers a straightforward solution for recognizing assembly steps without the need for extensive training or complicated setups.**

I. Introduction

With the advance of Industry 4.0, manufacturing is moving towards flexible and customized production. Interest in Human-Robot Collaboration is steadily growing due to its promising advantages: unlike traditional automated setups, collaborative systems can leverage exceptional human cognitive abilities alongside the high robot repeatability, precision, and fatigue resistance. This enables the required flexibility while also relieving humans from physically or mentally demanding tasks [1],[2]. Nevertheless, several challenges must be addressed to maximize the technology's potential, fostering effective and efficient collaboration. In collaborative assembly scenarios, human actions play a crucial role. Thus, the primary goal is to enable robots to quickly perceive and comprehend human behavior, facilitating efficient intervention and accurate action execution [3]. Consequently, developing intuitive communication systems between the two, without the need for complex interfaces, is essential [4]. Furthermore, human involvement is a potential source of errors, leading to imperfections in the final assembly [5]. Therefore, human error detection methods employed during assembly execution are fundamental to mitigate error propagation.

This paper employs a vision-based comparative approach to recognize the assembly step carried out by the human operator, simultaneously evaluating its correctness and highlighting possible execution errors. A Siamese Neural Network (SNN) was properly trained to highlight chromatic and geometric differences between two images of the same product at different assembly steps. Specifically, an online acquired image is compared with a reference set for the assembly steps to detect the last executed operation and possible human errors. The main contributions of the method are listed below.

- No model re-training is needed for new products entering the production line. Capturing new reference images is only required, as the assembly steps are identified by comparing the current instance with a reference dataset representing specific assembly actions, and the training dataset includes various assembly types, collecting all visual features typical of most industrial products.
- The model recognizes assembly operations, simultaneously detecting human errors and inconsistencies during the process.
- The training dataset composition allows the network to be robust to the shadow variations of the assembled part and different light exposure.

In Section II, the previous works about human error detection and assembly operation recognition are collected. Section III describes the Siamese Network structure, the composition of the training dataset, and the training results. In Section IV, the SNN performances are tested offline and compared to pre-existing OpenCV feature detectors. Section V contains the experimental validation of the method. In Section VI, some concluding remarks are exposed.

II. Related works

Several works in scientific literature study human error detection and activity recognition, often treating these tasks separately [6],[7]. Traditional automated inspection occurs at the end of the assembly line, resulting in delayed error detection and significant reworking costs. Consequently, there is a growing interest in Computer Vision algorithms to identify non-conformities during the assembly process, aiming to mitigate errors and ensure the required quality level [8]. For instance, Automated Optical Inspection (AOI) is crucial for complex defect detection in the Printed Circuit Boards (PCBs) field. Bhardwaj *et al.* [9] employed classic image processing techniques, such as image subtraction, template matching, and histogram matching, to detect and locate defects and missing components in Surface Mounted Devices used in PCBs. Unlike our approach, this method requires uniform light intensity to remove component shadows interfering with detection. Using Machine Learning or Deep Learning classifiers could solve this problem. For

[1] The authors are with Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Piazza Leonardo da Vinci 32, 20133, Milano (Italy). e-mail: martina.pelosi@polimi.it

[2] The author is with Politecnico di Torino, Dipartimento di Automatica e Informatica (DAUIN), Corso Castelfidardo 34/d, 10138, Torino (Italy)

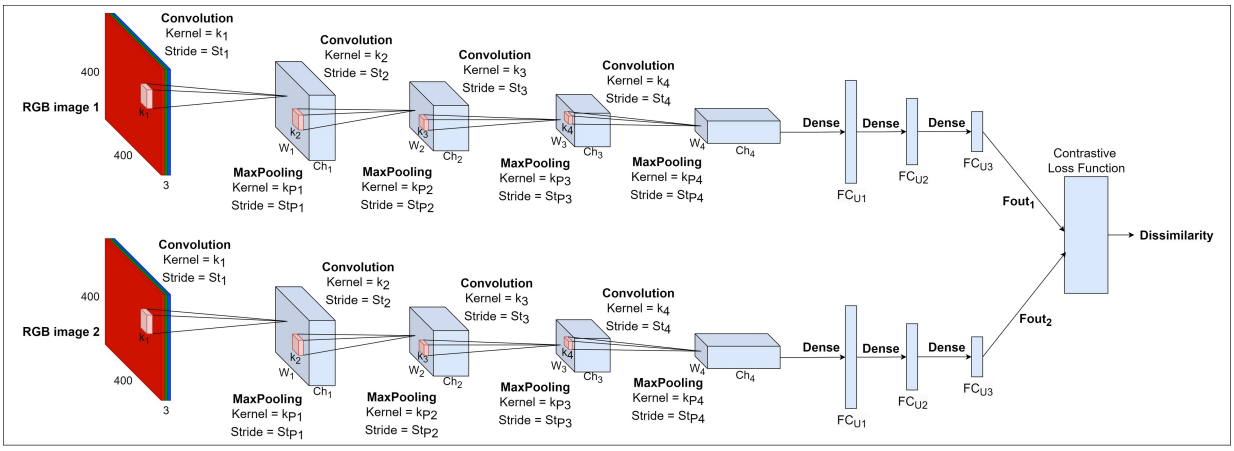


Fig. 1: Siamese Neural Network architecture: two parallel Convolutional Neural Networks with four convolutional layers with kernel k_i, stride St_i and channel number Ch_i, four max-pooling layers with kernel k_{P_i} and stride St_{P_i} and 3 fully connected layers with FC-units FC_{U_i}

example, Ojer *et al.* [10] developed a real-time monitoring system based on classifiers trained with synthetic images representing the board with and without the part mounted to separately verify the presence of each PCB component. Nevertheless, new training is necessary whenever a new assembly component is introduced. Chen *et al.* [8] explored solutions for assembly action recognition and missing components detection. They trained a 3D CNN with different image sequences to recognize ongoing operations. Despite achieving high accuracy, the training dataset is limited to only 9 common assembly actions, requiring extension and model re-training for new operations. In addition, they used a Fully Convolutional Network to segment and recognize the presence of parts in the assembled product, using images rendered from the assembly CAD model as training data. However, the method relies on the CAD model of the analyzed product, showing a possible limitation.

Regarding human operation recognition, several studies exploit human-object interaction to infer ongoing actions. For instance, Eversberg *et al.* [11] trained a Deep Neural Network to classify assembly steps based on the poses of the worker's hands and head and relevant tools and objects for operations. Although only a few assembly recordings are enough to train the model, this approach requires retraining for each new assembly process and was validated only on a non-industrial use case. Lucci *et al.* [12] overcame the training-related issue with neural networks by using a library of predicates to detect objects in hand and occupied working volumes, enabling both action recognition and human error monitoring. While offering seamless human-robot collaboration, the approach requires reflective stripes on the object surfaces to locate them, not allowing for actual object recognition.

In addition, few-shot learning techniques offer good prospectives in making Deep Learning algorithms more easily adaptable to different applications. In particular, Siamese Neural Networks (SNN) learn similarity metrics between inputs and work as zero-shot learning methods by comparing the current instance with a reference set [13]. Several SNN applications can be found in facial and handwriting recognition, but their use has extended to defect detection and human action recognition. For example, Kurek *et al.* [14] applied an SNN to verify drill wear states by comparing images of drilled holes. Concerning the action recognition problem, vision-based SNNs have been employed in sports to classify table tennis strokes [15] or to assign the score comparing the athlete's performance with reference sequences [16]. In addition, SNN models were explored to classify generic human actions using human movement data, exploiting, for example, accelerometer data [17] or 3D human skeletal joint sequences [18]. Nevertheless, complex dataset collections and wearable devices are typically required in these applications. To the best of the authors' knowledge, our approach represents the first application of SNN for assembly operation recognition, requiring just one online image acquisition per assembly step instead of time sequence data and exploiting a hand-tracking algorithm for easy communication.

III. The Siamese Neural Network

A. Network's architecture

The Siamese Neural Network (SNN) is the key component of the method, as it allows the recognition of assembly steps without the need for repetitive retraining to learn new operations. Only new reference images are required, as it detects the current assembly step by comparing an image taken during the process with a small reference dataset. It comprises two Convolution Neural Networks (CNNs), sharing identical weights and structural parameters, as depicted in Figure 1. The input of each CNN is a squared RGB image with a resolution of 400x400 pixels. This value was chosen as a trade-off between computational cost and the possibility of discerning small details. The decision to use RGB images aims to enable the network to learn both geometrical and chromatic features of objects. The input image is processed through four convolutional layers, interspersed by the Rectified Linear Unit (ReLU) activation function and

Fig. 2: Training data examples. Above, 0-labeled pairs of identical assembly stages or industrial tools; below, 1-labeled pairs of various assembly stages of the same product, objects with varied colors, and completely different objects

a max-pooling layer to enhance computational efficiency by reducing convolutional output dimensions. Additionally, three Fully Connected layers complete the architecture of both CNNs, allowing the final transformation of the input images into feature vectors. To learn dissimilarities between feature vectors, the SNN architecture employs an Euclidean Distance (D) - based Contrastive Loss Function (CL):

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

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

B. Training dataset

The training dataset was composed to enable the network to detect small variations between images, facilitating the distinction between different steps in the same assembly process. With this purpose, it comprises images of assembled parts captured at different stages of various kinds of assembly. Some industrial tools and objects were also included to expand and diversify the dataset allowing the model to learn specific features of industrial parts, like the presence of holes, screws, or sharp edges. Additionally, different colored objects were introduced to enable the network to recognize chromatic features. The data collection process produced 120 distinct classes, each containing 20 images of the same item by color and geometry. These images were captured by rotating the objects manually on a white surface while keeping a constant distance between the camera and the table. This setup enhances the object-background contrast, helping the network to focus on the item and reducing the misclassification risk. Moreover, maintaining a fixed camera-table distance allows the network to learn to recognize objects of different sizes. Furthermore, assuming a fixed light source, manual rotation of the object causes its shadow always to change shape while remaining on the same side of the object. Under these conditions, including images of a specific object captured from different angles within the same class allows training a rotational-invariant model, ensuring that changes in the shadow's appearance do not affect the network's predictions.

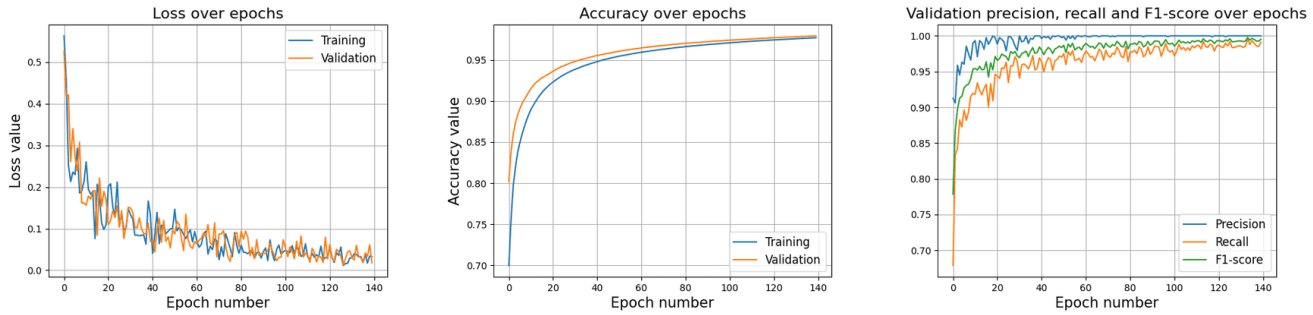
To expand the dataset, original data were digitally augmented, generating additional 60 images per class. First, 20 images were created through horizontal and vertical translations of the originals, with a maximum shift of 10% of the size. Subsequently, the resulting 40 images were further augmented by introducing small random variations in contrast and brightness to enhance the network's robustness to different levels of contrast and light exposure. This process led to a final dataset of 9600 images, divided into training and validation sets: 80% of the images were used to train the model and the remaining 20% to validate the results. Equal and different images were randomly paired with a 50% probability, labeling with 1 pairs of images belonging to different classes, while pairs from the same class were labeled with 0. Some examples of image pairs included in the dataset are shown in Figure 2, together with their labels.

C. Hyper-parameters tuning and training results

A grid search tuning process was applied to key hyper-parameters affecting performance, specifically learning rate (lr) and batch size (bs). The model was trained for 50 epochs for every possible combination of values from the following predefined sets: $lr = [0.001, 0.0001, 1 \cdot 10^{-5}, 0.0005, 5 \cdot 10^{-5}, 5 \cdot 10^{-6}]$, $bs = [16, 32, 64, 132]$. After analyzing the loss function and accuracy trends, the optimal values were determined to be $lr = 0.0001$ and $bs = 132$ as a trade-off between performance and computational efficiency. Higher lr caused unstable loss trends and decreased accuracy, while smaller bs led to more oscillatory loss functions with higher convergence values. On the other hand, lower lr and higher bs produced excessively high computational costs, significantly increasing training times. In addition, the most influencing architectural-related parameters were tuned using a random search approach. These parameters include the number of channels Ch_i and kernel size k_i for each convolutional layer, as well as the number of units FC_{U_i} for each Fully Connected layer. Table I outlines the final values derived from the overall tuning processes, which were then employed for model training. The stride values for both convolutional and max pooling layers (St_i and St_{P_i}), as well

lr	bs	Ch_1	Ch_2	Ch_3	Ch_4	k_1	k_2	k_3	k_4	St_1	St_2	St_3	St_4	k_Pi	St_Pi	FC_{U_1}	FC_{U_2}	FC_{U_3}
0.0001	128	64	128	128	256	8	7	4	3	3	1	1	1	2	2	4096	256	2

TABLE I: Network hyper-parameters values after tuning



(a) Loss (b) Accuracy (c) Validation precision, recall, F1-score

Fig. 3: Training and validation curves: descending loss function trend converging to 0.02, increasing accuracy converging to 0.98, and ascending validation precision, recall, and F1-score converging around 0.99

as the kernel size for max pooling (k_{P_i}), were user-defined, as they were observed to have less impact compared to the other parameters.

After tuning, the network was finally trained for 140 epochs. The resulting training and validation curves are depicted in Figure 3, showing the trends over epochs of loss, accuracy, as well as precision, recall, and F1-score for the validation. Computation of the accuracy involved defining a dissimilarity threshold Th_{diss}, set equal to 1 in this phase. Pairs of input images leading to dissimilarity below Th_{diss} are classified as equal images, while higher dissimilarity indicates differing images. The resulting curves show a positive model training trend, reflecting a significant improvement over epochs in the model's capability to classify instances correctly. There is no sign of overfitting, as validation curves closely track training ones.

IV. Offline Performance Validation and Comparison with OpenCV Feature Detectors

An offline testing campaign was performed to verify the model's ability to discriminate between different objects. The SNN's performance was compared with two OpenCV feature detectors: Scale-Invariant Feature Transform (SIFT) and Oriented FAST and rotated BRIEF (ORB). These tools are specifically designed to identify and match similar features across two input images, even when rotations or translations are involved. A testing dataset was created following the same procedure outlined in Section III for composing the training dataset but choosing assembled parts and tools not used for training. The final testing dataset includes 25 classes, each containing 80 images. To thoroughly analyze the trained model's capability to distinguish between similar and dissimilar objects, three types of experiments, involving 5 tests each, were conducted. First, pairs from the same and different classes were randomly selected with a 50% probability. The second experiment was performed by pairing images only from the same class. Lastly, only pairs of different images were involved. The network's performance was assessed against the OpenCV feature detectors for each experiment by computing four KPIs: accuracy, precision, recall, and F1-score. Table II presents the average results for each test type. The outcomes show that the Siamese Neural Network consistently outperforms the pre-existing feature matching tools, demonstrating robust performance in all experiments. Although SIFT and ORB appear a bit more accurate than the SNN in recognizing different images, their effectiveness collapses dramatically in matching features in similar images. The different appearance of shadows in the images, produced by the manual object rotation, probably interferes with the prediction of these models since, unlike the SNN, they have not been trained to avoid it.

		Accuracy	Precision	Recall	F1-score
Random selection	**SNN**	0.867	0.817	0.938	0.873
	ORB	0.553	0.532	0.995	0.733
	SIFT	0.561	0.536	0.993	0.696
Equal pairs	**SNN**	0.789	0	-	-
	ORB	0.103	0	-	-
	SIFT	0.117	0	-	-
Different pairs	**SNN**	0.932	1	0.932	0.971
	ORB	0.995	1	0.995	0.997
	SIFT	0.993	1	0.993	0.996

TABLE II: Offline testing results: performance comparison between SNN and OpenCV feature detectors, SIFT and ORB

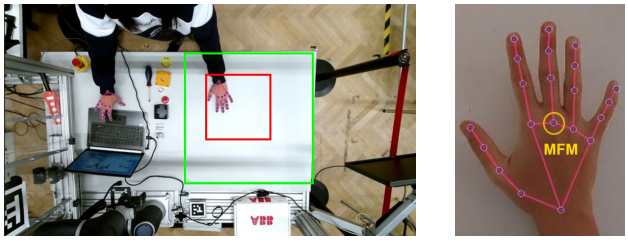
V. Experimental Validation and Use Case

The SNN's ability to recognize assembly operations and errors was validated online by testing three different use cases. This Section explains the experimental results, providing an overview of the setup, the hand-tracking algorithm for communicating operation completion, and the process for selecting the dissimilarity threshold to identify human errors.

A. Experimental setup and validation algorithm

The experiments were conducted by exploiting a KinectV2 vision sensor to capture the scene. It was placed 1.25 meters above the working table to ensure optimal data capture,

obtaining detailed visual information while maintaining an adequate field of view. Human hands are tracked using the Mediapipe Hand Landmarker solution [19], providing real-time information on the location of 21 hand keypoints. When the Middle Finger Metacarpophalangeal joints (MFM in Figure 4b) of both hands leave the green reference working area (Figure 4a) to grab the next assembly component, the current assembly stage is considered complete. The current image of the assembled part, corresponding to the red assembly area, is captured and input to the SNN. The hand-tracking algorithm enables natural interaction between the robot and the operator, eliminating the need for explicit human communication or Human-Machine Interfaces (HMIs) to signal the step completion.



(a) KinectV2 sensor field of view (b) MFM joint

Fig. 4: Hand-tracking algorithm: when the ongoing operation is completed, i.e. both MFM joints exit the green reference area, an image of the assembled part is captured, cropping the red assembly area

B. Dissimilarity threshold computation

For operation correctness evaluation, dissimilarity values output by the Siamese Network are compared with the threshold Th_{diss}. During training and offline validation, this value was manually set to 1, intentionally oversized, since the testing dataset contains images of completely different objects characterized by highly distinct features in color and geometry. However, during online testing, images are exclusively chosen from the same assembly process. Therefore, the threshold could not remain the same as offline validation but was adjusted to suit the assembly case under analysis. For each assembly case study, 10 reference images were taken for each step, showing the correctly assembled part in various positions and orientations. Each image was individually compared to all others within the same class and those in different classes, and the resulting dissimilarities from the network were recorded. An average dissimilarity was computed for both pairs of equal images (D_{equal}) and pairs of different images (D_{diff}) across all assembly steps. Then, the dissimilarity threshold for online validation was determined as follows:

$$Th_{diss} = D_{equal} + \frac{D_{diff} - D_{equal}}{8} \tag{2}$$

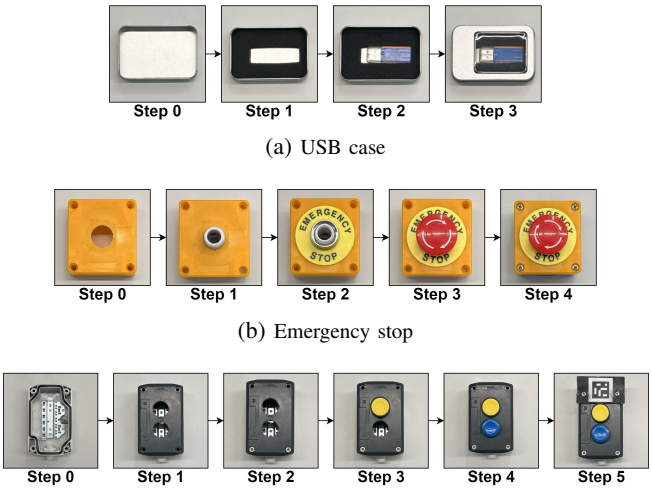
This formula allows to mitigate the potential risk of misinterpreting incorrect operations as correct, setting the value of Th_{diss} closer to D_{equal} than to D_{diff} in a conservative approach. Table III summarizes the resulting values for D_{equal}, D_{diff}, and the final computed threshold Th_{diss} for the assembly cases analyzed.

Dissimilarity threshold	USB case	Emergency stop	Push-button panel
D_{equal}	0.520	0.208	0.635
D_{diff}	3.854	1.997	2.457
Selected value Th_{diss}	**0.937**	**0.431**	**0.863**

TABLE III: Customized threshold values for tested assembly scenarios

C. Use cases and experimental results

Three use cases were tested to assess the model performance: the assembly of a USB flash drive case, an emergency stop, and a push-button panel [1]. Figure 5 shows the correct assembly sequences for the three scenarios. For each use case, 15 tests were performed, successfully completing the entire assembly sequence without error. Based on the hand-tracking strategy explained in Section V-A, an image is captured for each completed assembly step and compared with the 10 reference images collected for each class. The average dissimilarity score is computed for each reference step to minimize the influence of individual values, enhancing the overall accuracy of the result. Based on the dissimilarity referred to each assembly step, the network can make the prediction: the class with the lowest value is considered the assembly operation actually completed by the human, and its correctness is evaluated by comparing this value with Th_{diss} associated with the current assembly. The resulting network's predictions for the three case studies are shown in Table IV, respectively.



(a) USB case

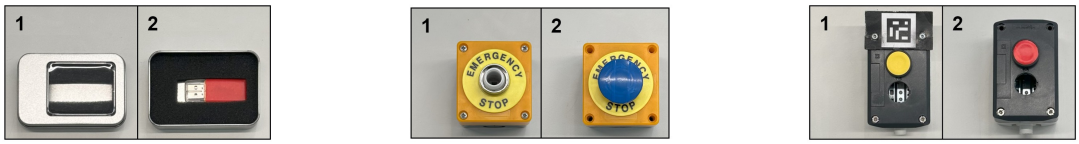
(b) Emergency stop

(c) Push-button panel

Fig. 5: Correct assembly step sequence for the three assembly cases

Overall, a 100% success rate in recognizing the actual assembly step was achieved for the USB flash drive case, 85,3% for the emergency stop, and 91,1% for the push-button panel. This is coherent since all components of the USB flash drive introduce significant visual differences between

[1]The video of the experimental validation can be seen here: `https://youtu.be/5SN3H_zYYZY`



(a) USB case (b) Emergency stop (c) Push-button panel

Fig. 6: Tested error cases: (a) missing USB stick (1) and wrong USB stick (2), (b) missing emergency button (1) and wrong emergency button (2), (c) missing lower button (1) and wrong upper button (2) in the panel

one step and the next. On the other hand, in emergency stop and push-button panel assemblies, some operations involve minimal visual changes from one step to the next, making recognition more challenging. Specifically, errors in the emergency stop assembly typically occur when the central white body is introduced (between steps 0 and 1) and when four small screws are added (between steps 3 and 4). As for the push-button panel, main errors arise when two small screws are introduced (between steps 1 and 2) or due to unwanted reflections on the surface of the blue button (between steps 3 and 4).

Actual Step \ Predicted Step	Step 0	Step 1	Step 2	Step 3
Step 0	15	0	0	0
Step 1	0	15	0	0
Step 2	0	0	15	0
Step 3	0	0	0	15

(a) USB case

Actual Step \ Predicted Step	Step 0	Step 1	Step 2	Step 3	Step 4
Step 0	11	4	0	0	0
Step 1	1	14	0	0	0
Step 2	0	0	15	0	0
Step 3	0	0	0	11	4
Step 4	0	0	0	2	13

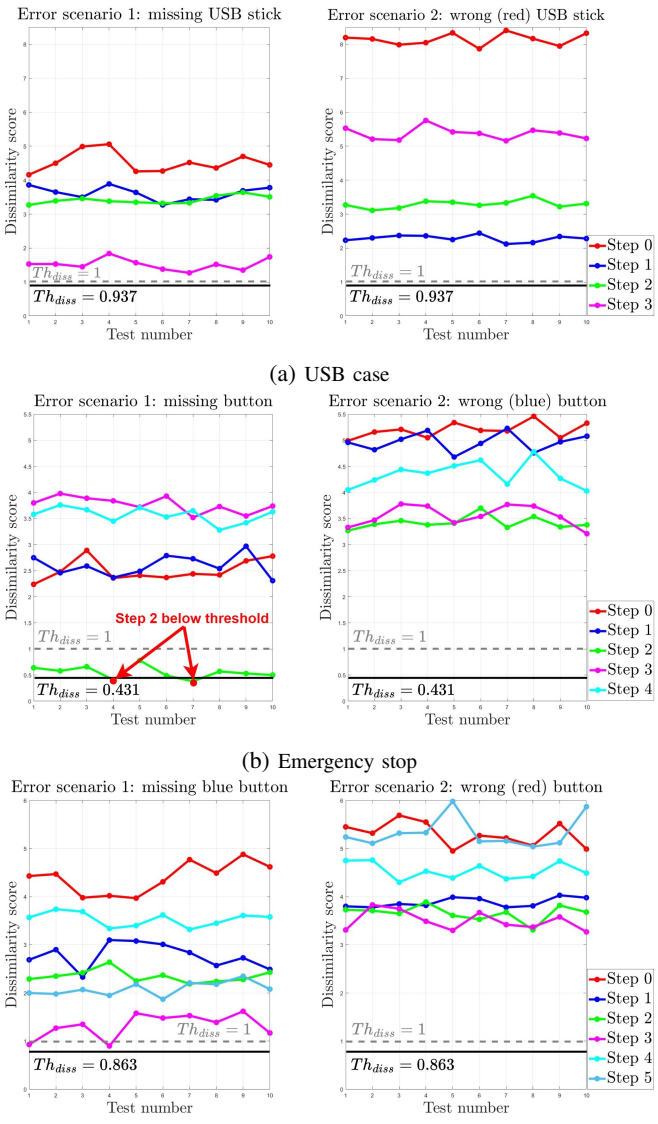
(b) Emergency stop

Actual Step \ Predicted Step	Step 0	Step 1	Step 2	Step 3	Step 4	Step 5
Step 0	15	0	0	0	0	0
Step 1	0	12	3	0	0	0
Step 2	0	2	13	0	0	0
Step 3	0	0	0	14	1	0
Step 4	0	0	0	2	13	0
Step 5	0	0	0	0	0	15

(c) Push-button panel

TABLE IV: Test results in correct assembly step recognition for the three assembly cases

Additionally, as shown in Figure 6, two error scenarios for each assembly were tested 10 times to assess the network's capability to reveal human errors during the process. Again, the online image is compared with the reference dataset. If the network returns a dissimilarity value above or equal to the threshold for all the reference assembly steps, an error in executing the last operation is recognized, and a warning is shown on the screen. The plots in Figure 7 show the dissimilarity values obtained for all tests for each error scenario. The network made only two errors in cases where the emergency stop button was missing, as indicated by the dissimilarity value for step 2 falling below the threshold Th_{diss}. However, the difference between this assembly step and the error scenario (the addition of four small screws at the corners) is a visually challenging detail to detect. As can be noticed from the plot, the customized threshold for the tested assembly improves error detection, enabling the correct identification of errors that would go unnoticed with the original threshold value.



(a) USB case

(b) Emergency stop

(c) Push-button panel

Fig. 7: Dissimilarity score over error scenarios tests

VI. Conclusion

The Siamese Neural Network-based approach eliminates the need for recurrent model training when introducing new parts or operations. It combines assembly step recognition and human error detection thanks to a customizable threshold, making it cost-effective for industrial use. Furthermore, hand-tracking allows operators to execute tasks without complex robot interactions. Offline validation shows promising SNN performance in recognizing both the same and different objects, even in the case of small details, unlike OpenCV's SIFT and ORB, which are only robust in detecting different instances. Online validation reveals high success rates in identifying assembly steps and human errors across different use cases, with issues occurring only with very small details or unwanted reflections.

Future work will involve introducing a robot to assist the operator during assembly to prove the method's effectiveness in a collaborative context. Furthermore, future development could focus on detecting ongoing operations to allow faster robot reactions. For instance, training the SNN to identify the tool grabbed by the operator could help anticipate the next assembly step, enhancing efficiency.

Acknowledgment

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

References

[1] A. Ajoudani, A. M. Zanchettin, S. Ivaldi, A. Albu-Schäffer, K. Kosuge, and O. Khatib. "Progress and prospects of the human–robot collaboration". In: *Autonomous Robots* 42 (2018), pp. 957–975.

[2] A. Borboni, K. V. V. Reddy, I. Elamvazuthi, M. S. AL-Quraishi, E. Natarajan, and S. S. Azhar Ali. "The Expanding Role of Artificial Intelligence in Collaborative Robots for Industrial Applications: A Systematic Review of Recent Works". In: *Machines* 11.1 (2023), p. 111.

[3] H. Oliff, Y. Liu, M. Kumar, and M. Williams. "Improving human–robot interaction utilizing learning and intelligence: A human factors-based approach". In: *IEEE Transactions on Automation Science and Engineering* 17.3 (2020), pp. 1597–1610.

[4] H. Liu and L. Wang. "Gesture recognition for human-robot collaboration: A review". In: *International Journal of Industrial Ergonomics* 68 (2018), pp. 355–367.

[5] Y. Torres, S. Nadeau, and K. Landau. "Classification and quantification of human error in manufacturing: a case study in complex manual assembly". In: *Applied Sciences* 11.2 (2021), p. 749.

[6] V. Di Pasquale, S. Miranda, W. P. Neumann, and A. Setayesh. "Human reliability in manual assembly systems: a Systematic Literature Review." In: *Ifac-Papersonline* 51.11 (2018), pp. 675–680.

[7] L. M. Dang, K. Min, H. Wang, M. J. Piran, C. H. Lee, and H. Moon. "Sensor-based and vision-based human activity recognition: A comprehensive survey". In: *Pattern Recognition* 108 (2020), p. 107561.

[8] C. Chen, C. Zhang, T. Wang, D. Li, Y. Guo, Z. Zhao, and J. Hong. "Monitoring of assembly process using deep learning technology". In: *Sensors* 20.15 (2020), p. 4208.

[9] S. C. Bhardwaj. "Detection and verification of missing components in SMD using AOI techniques". In: *International Journal of Computer Graphics* 7.2 (2016), pp. 13–22.

[10] M. Ojer, I. Serrano, F. Saiz, I. Barandiaran, I. Gil, D. Aguinaga, and D. Alejandro. "Real-time automatic optical system to assist operators in the assembling of electronic components". In: *The International Journal of Advanced Manufacturing Technology* 107.5-6 (2020), pp. 2261–2275.

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

[12] N. Lucci, A. Monguzzi, A. M. Zanchettin, and P. Rocco. "Workflow modelling for human–robot collaborative assembly operations". In: *Robotics and Computer-Integrated Manufacturing* 78 (2022), p. 102384.

[13] S. Jadon and A. Jadon. "An overview of deep learning architectures in few-shot learning domain". In: *arXiv preprint arXiv:2008.06365* (2020).

[14] J. Kurek, I. Antoniuk, B. Świderski, A. Jegorowa, and M. Bukowski. "Application of Siamese Networks to the Recognition of the Drill Wear State Based on Images of Drilled Holes". In: *Sensors* 20.23 (2020), p. 6978.

[15] P.-E. Martin, J. Benois-Pineau, R. Péteri, and J. Morlier. "Fine grained sport action recognition with Twin spatio-temporal convolutional neural networks: Application to table tennis". In: *Multimedia Tools and Applications* 79 (2020), pp. 20429–20447.

[16] H. Jain, G. Harit, and A. Sharma. "Action quality assessment using siamese network-based deep metric learning". In: *IEEE Transactions on Circuits and Systems for Video Technology* 31.6 (2020), pp. 2260–2273.

[17] S. Berlemont, G. Lefebvre, S. Duffner, and C. Garcia. "Class-balanced siamese neural networks". In: *Neurocomputing* 273 (2018), pp. 47–56.

[18] S. Yucer and Y. S. Akgul. "3D human action recognition with siamese-LSTM based deep metric learning". In: *arXiv preprint arXiv:1807.02131* (2018).

[19] Google. *MediaPipe Solutions: Hand Landmarker detection guide*. 2022. URL: `https://developers.google.com/mediapipe/solutions/vision/hand_landmarker`.