

Article

Determining Exception Context in Assembly Operations from Multimodal Data

Mihael Simonič ^{1,2,*} , Matevž Majcen Hrovat ¹ , Sašo Džeroski ^{3,4} , Aleš Ude ^{1,2}  and Bojan Nemec ^{1,4} 

- ¹ Department of Automatics, Biocybernetics and Robotics, Jožef Stefan Institute, Jamova Cesta 39, 1000 Ljubljana, Slovenia
- ² Faculty of Electrical Engineering, University of Ljubljana, Tržaška Cesta 25, 1000 Ljubljana, Slovenia
- ³ Department of Knowledge Technologies, Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia
- ⁴ Jožef Stefan International Postgraduate School, Jamova Cesta 39, 1000 Ljubljana, Slovenia
- * Correspondence: mihael.simonic@ijs.si

Abstract: Robot assembly tasks can fail due to unpredictable errors and can only continue with the manual intervention of a human operator. Recently, we proposed an exception strategy learning framework based on statistical learning and context determination, which can successfully resolve such situations. This paper deals with context determination from multimodal data, which is the key component of our framework. We propose a novel approach to generate unified low-dimensional context descriptions based on image and force-torque data. For this purpose, we combine a state-of-the-art neural network model for image segmentation and contact point estimation using force-torque measurements. An ensemble of decision trees is used to combine features from the two modalities. To validate the proposed approach, we have collected datasets of deliberately induced insertion failures both for the classic peg-in-hole insertion task and for an industrially relevant task of car starter assembly. We demonstrate that the proposed approach generates reliable low-dimensional descriptors, suitable as queries necessary in statistical learning.



Citation: Simonič, M.; Majcen Hrovat, M.; Džeroski, S.; Ude, A.; Nemec, B. Determining Exception Context in Assembly Operations from Multimodal Data. *Sensors* **2022**, *22*, 7962. <https://doi.org/10.3390/s22207962>

Academic Editors: Abdelaziz Benallegue and A. El Hadri

Received: 27 August 2022

Accepted: 14 October 2022

Published: 19 October 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: sensor fusion; predictive clustering trees; autonomous exception handling; autonomous assembly; peg-in-hole

1. Introduction

Assembly tasks, such as inserting parts into fixtures, are among the most common industrial applications [1]. Robot assembly typically requires a good understanding of the procedure and knowledge about part properties and geometry [2]. Therefore, most of the deployed robotic systems used today are carefully programmed [3]. As such, they are limited to performing a specific assembly task in structured environments without external disturbances. Nevertheless, they can fail due to various errors that cannot be foreseen in advance. Possible causes include deviations in the geometry of the workpiece, imprecise grasping, etc. In such cases, it is necessary for the operator to manually eliminate the cause of the error, reset the system, and restart the task [4]. Current robotic systems do not learn from such situations. If a similar situation repeats, human intervention is needed again. To ensure robust execution of robot assembly tasks, it is increasingly important to handle such exception scenarios autonomously, possibly by incorporating previous experience.

The first step toward building such an autonomous system is to determine the reason for the failure. For example, a robot may fail to assemble two parts, but it is unclear whether it has failed because the parts do not match or because of an ineffective manipulation strategy [5]. Understanding or at least classifying the reason for the failure is, therefore, crucial for the successful design of a corresponding exception policy.

Recently, we have proposed a framework for the learning of exception strategies [6], which is based on determining the context of the failure. The extracted context is associated with different robot policies needed to resolve the cause of the error. In the event of an error,

the system stops, and the robot switches to gravity compensation mode. Using incremental kinesthetic guidance [7], the operator performs a sequence of movements to allow the continuation of regular operation. First, the robot builds a database of corrective actions and associates them with the detected error contexts. Then, using statistical methods, it computes an appropriate action by generalizing the corrective actions associated with different error contexts. In this way, the robot becomes increasingly able to resolve errors on its own and eventually does not require human intervention to resolve assembly failures anymore (see Figure 1).

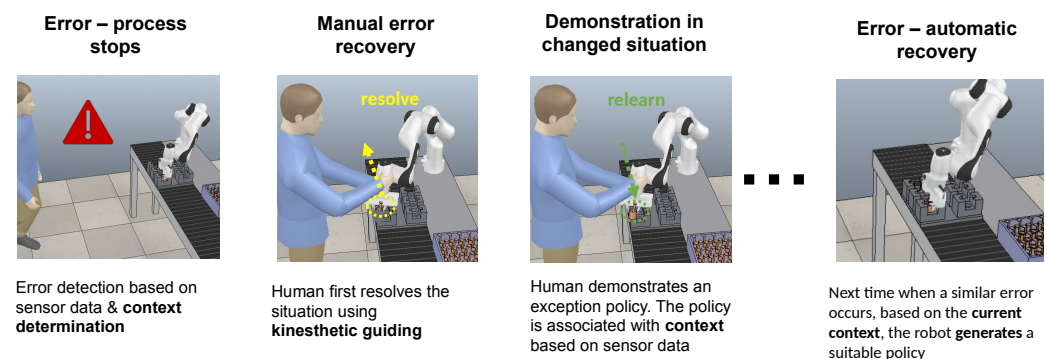


Figure 1. Simplified exception handling workflow [6]. This paper is about context determination, which is an essential requirement of the workflow.

Modern robotic systems are equipped with a wide variety of sensors that can be used to detect a possible failure. On the other hand, context determination can be seen as inferring the circumstances that have resulted in the given outcome. The first step in the exception strategy learning framework is, therefore, context determination, which can be seen as inferring a minimal representation of the circumstances that have resulted in the given outcome. As raw sensor data are usually high-dimensional, they cannot be directly used with statistical learning methods that allow us to relate the observed state (context) to the previous states and generate an appropriate robot action to resume regular operation. Moreover, for reliable context determination, it is often necessary to combine complementary information from different sensor modalities. This process is known as data fusion and can lead to improved accuracy of the model compared to a model based on any of the individual data sources alone [8]. Ensemble learning methods have proven to be appropriate for addressing multimodal classification and regression problems in many domains [9].

We propose to train models that generate low-dimensional context descriptions based on multimodal sensor data from vision systems and force-torque sensors. In this sense, context determination can be defined as determining of the type of circumstances from multimodal data. We use an intermediate-fusion approach, where we first extract modality-specific features, as shown in Figure 2. We rely on a state-of-the-art neural network model for image segmentation to extract features from images, whereas we use contact point estimation to extract data from the measured forces and torques. To generate a low-dimensional context description of the circumstances that resulted in the given outcome from the extracted features, we use ensembles of predictive clustering trees (PCTs) [10], which are well suited for handling hierarchical multi-label classification (HMLC) tasks. With the proposed hierarchical approach, the training of the context estimation model can be divided into multiple phases, allowing for an incremental approach. The time-consuming training of the image segmentation model only needs to be performed once, whereas fast training of the high-level ensemble model can be performed each time a new class needs to be added.

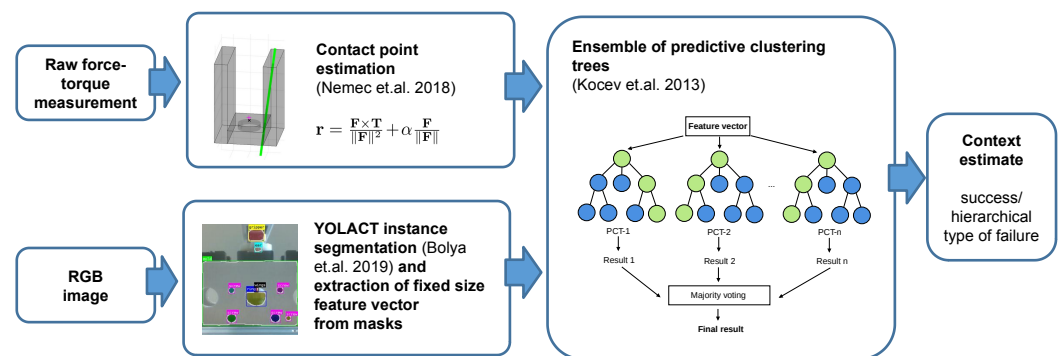


Figure 2. Context determination pipeline. Features are processed for each modality separately (Sections 3.4–3.6) and later merged by using ensembles of predictive clustering trees (Section 3.7). References: [10–12].

The evaluation of the proposed approach comprised two scenarios. Peg-in-hole assembly is chosen as the first use case because it reflects the typical complexity of industrial assembly tasks [13]. We show that it is possible to apply the method to other tasks by performing an evaluation of a car starter assembly task. The approach can also be applied to further situations. Apart from identifying error classes, the only process-specific step is selecting the image segments of interest and training the instance segmentation model accordingly.

The main highlights of our context determination approach are:

- with the use of multimodal data we get an improved predictive performance of ensembles;
- it is easy to add new classes (this is necessary as we discover new failure cases incrementally as they arise);
- the approach generalizes well to new cases (we can make useful predictions based on a model trained on a limited amount of data).

The paper is organized into six sections. Section 2 reviews current strategies to handle exceptions in robot assembly and the usage of tactile and vision sensor data in robotics. The details of our approach are presented in Section 3. In Section 4, the predictive performances of different variations of the developed model for context determination are evaluated. Section 5 discusses the results as well as future plans. We conclude with a brief summary of the paper in Section 6.

2. Related Work

Fault-tolerant robotic systems that are able to detect and autonomously deal with system failures have been the subject of research for many years [14]. While some researchers are concerned with fault tolerance in medical, space, nuclear, and other hazardous applications, our research focuses on industrial processes, where we can ensure operator presence, at least in the learning phase. In such environments, strategies based on various heuristic movement patterns (random search, spiral search, dithering, vibrating, etc.) are often used to deal with unexpected situations [15,16].

Laursen et al. [17] proposed a system that can automatically recover from certain types of errors by performing the task in reverse order until the system returns to a state from which the execution can resume. Error recovery can also be performed collaboratively so that the robot recognizes when it is unable to proceed and asks for human intervention to complete the task [18]. Recently, it was proposed to use ergodic exploration to increase the insertion task success rate based on information gathered from human demonstrations [19]. Another method exploits variability in human demonstrations to consider task uncertainties and does not rely on external sensors [2].

On the other hand, another line of research highlights the importance of sensorimotor interaction for future learning methods in robotics [20]. During the assembly task execution, monitoring the exerted forces and torques is necessary to prevent damaging the parts or

robot [6]. These data can be used to avoid large impact forces exploiting compliance and on-line adaptation [21], to speed up the process in the subsequent repetitions [22], to determine contact points and learn contact policies [11], and to predict [23] or classify [24] robot execution failures.

Various previous works have studied the idea to calculate contact points based on force-torque measurements [11,25–27]. In our previous work [6], we used force-torque measurements to calculate trajectory refinements that enable successful insertion despite the grasping error. Force-torque data carry enough information to generate an appropriate refinement, given that we already know the nature of the problem (orientation vs. position grasping error). However, in general, sufficient information cannot be obtained from force-torque data only. For example, the policies for correcting unsuccessful assembly attempts often depend on which part of the peg is in contact with the environment [28]. Thus additional sensors are required. Using a force-torque sensor only is also problematic due to sensor noise. Many applications in process automation, therefore, rely on vision systems to extract the necessary information. While vision can be quite sensitive to calibration errors and typically requires a well-designed workcell to ensure optimal lighting conditions and avoid occlusions, it is fast and can be used for the global detection of multiple features [29].

The advantage of combining visual and contact information has been investigated in multiple works in robotics over a longer period of time. This research includes dimension inspection [30], object recognition [31], and localization [29]. In the context of robot assembly, visual and tactile sensing has been used to continuously track assembly parts using multimodal fusion based on particle filters [32] and Bayesian state estimation [13]. We share the principal idea of combining data from visual and force-based sensing. We want to further develop these concepts towards structured representations of the task context in order to develop an integrated solution for the automatic handling of failures in assembly processes.

Multimodal fusion combines information from a set of different types of sensors. Detection and classification problems can be addressed more efficiently by exploiting complementary information from different sensors [8]. Different methods for data fusion from multimodal sources exist. Generally, we can distinguish three levels of data fusion: early fusion, where the raw data are combined ahead of feature extraction and the result is obtained directly; intermediate fusion, where modality-specific features are extracted and joined before obtaining the result; and late fusion, where the modality-wise results are combined [33,34]. It may seem that combining multimodal data at the raw data level should yield the best results, as there would be no loss of information. However, due to the unknown inter-dependencies in raw data, fusion at a higher level of abstraction may be a more helpful approach in practice [35].

Ensemble learning is a general approach in machine learning that seeks better predictive performance by combining the predictions from multiple models [9]. Ensemble learning methods have proven to be an appropriate tool to address multimodal fusion, achieving comparable results or even outperforming other state-of-the-art methods in many other domains [36,37]. The idea of ensemble learning is to employ multiple models and combine their predictions. This is often more accurate than having a complex individual model to decide about a given problem. Data from heterogeneous sources, such as different modalities [38], can easily be combined. In this paper, we consider an ensemble of predictive clustering trees (PCTs) [10] to perform hierarchical multi-label classification (HMLC). PCTs are a generalization of ordinary decision trees and have been successfully used for a number of modeling tasks in different domains, i.e., to predict several types of structured outputs, including nominal/real value tuples, class hierarchies, and short time series [39,40]. A detailed description of PCTs for HMLC is given by Vens et al. [41].

3. Materials and Methods

In this section, we first describe the robotic workcell used to collect the data described in Section 3.1. The assembly tasks to perform the evaluation are presented in

Sections 3.2 and 3.3. Next, we present our contact determination method in detail. The approach consists of three main parts. First, force-torque sensor measurements are processed using a method for contact point estimation described in Section 3.4. Data from the vision modality are passed through a neural-network model performing instance segmentation (Section 3.5), and features are extracted from the instance masks using standard computer vision methods (Section 3.6). Finally, the features from both modalities are combined using an ensemble of predictive clustering trees, as described in Section 3.7.

3.1. Experimental Environment

In our research, we focused on robotic assembly and considered two tasks—square peg insertion using the Cranfield benchmark [42] and the industrially relevant car starter assembly [43]. To perform both tasks and collect data for the context determination model, we rely on a modular workcell design that enables easy mounting of task-specific equipment, e.g., robots, sensors, and auxiliary devices [44] and a ROS-based software architecture that allows for easy integration of new components [45].

The workcell consists of two modules and a control workstation. The first module supports a seven-degree-of-freedom collaborative robot, Franka Emika Panda. The other module is equipped with sensors and cameras to support the specific assembly process, as shown in Figure 3. An Intel RealSense D435i RGB-D camera is used to supervise the insertion visually. To control the light conditions, we utilize an adaptive lighting setup based on Aputure Amaran F1 LED panels. Besides images, we can also capture forces and torques. To measure the forces exerted in the peg-in-hole task, we utilize an ATI Delta force-torque sensor mounted under the Cranfield benchmark plate. To measure the forces exerted in the copper ring insertion task, we utilize a wrist-mounted ATI Nano25 sensor. Additional peripheral devices, visible in Figure 3b, are used to assist different aspects of the human–robot collaboration, which is not the subject of this paper.

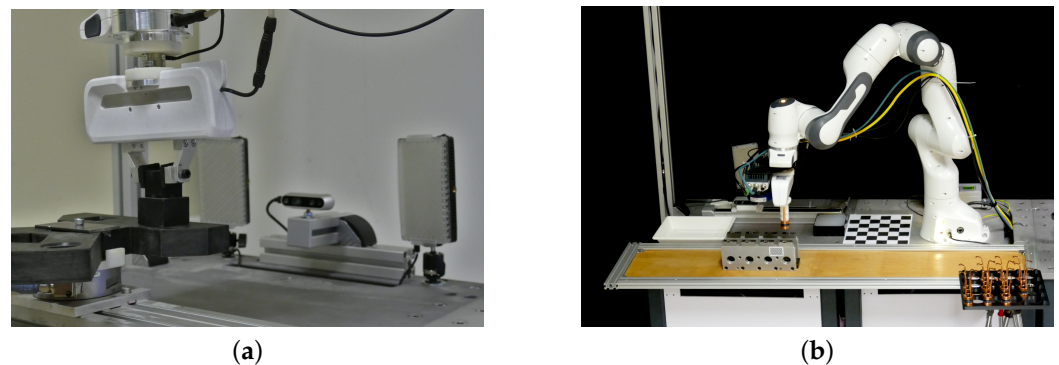


Figure 3. Experimental setup for testing exception strategy learning using multimodal data. (a) Setup for the *Cranfield benchmark*. (b) Setup for the copper ring insertion task.

To perform the assigned tasks, we applied a passivity-based impedance controller for manipulators with flexible joints [46]. We assume that the controller parameters were carefully tuned to ensure stable and compliant operation in unstructured environments, where we can expect deviations in task parameters.

3.2. Peg-in-Hole Insertion Task

The peg-in-hole (PiH) task is an abstraction of the most typical task in assembly processes, accounting for approximately 40% of the total assembly tasks [47]. Over time, many different approaches and control strategies to address this problem have emerged. Nowadays, the efficiency of the applications is enhanced by integrating machine vision and other sensor technology accompanied by artificial intelligence approaches. As such, it is a commonly accepted benchmark in assembly research.

To generate a dataset for comparing different methods for failure context determination, we repeatedly executed the task of square peg insertion using the Cranfield benchmark.

It requires the insertion of a square peg into the corresponding hole of the base plate. The main challenge is the transition of the peg from free space into a highly constrained target hole. Relatively tight tolerances combined with imprecise positioning can prevent the successful completion of the insertion process.

Different factors influence the outcome of the PiH task. For instance, both imprecise grasping and wrong target position can lead to insertion failure, as shown in Figure 4.

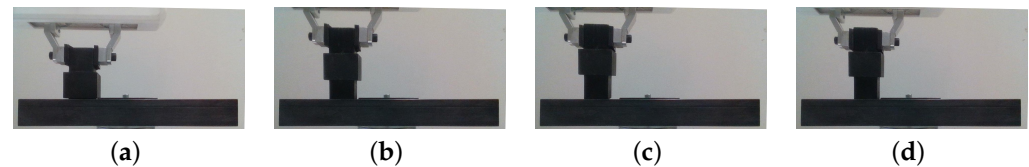


Figure 4. Different outcomes of the PiH task. (a) Successful insertion with correct parameters. (b) Insertion failure due to grasping error. (c) Insertion failure due to a positional error in the x -direction. (d) Insertion failure due to a positional error in the y -direction.

In order to collect a database of different insertion outcomes, we deliberately set different positional offsets in either the x or the y -direction from -10 to 10 mm in 1 mm steps. In this way, we generated 40 cases that resulted in insertion failure and 1 that led to successful insertion. Due to the offset, the robot fails to insert the peg and stops the execution when it exceeds a force threshold, set to 10 N in the z -direction. The insertion is successful when there is no positional offset in either direction.

In total 180 data entries were recorded. The robot attempted to insert the peg into the plate three times for each failure case. Additionally, 60 successful attempts were recorded. Robot pose, force and torque measurements, and RGB images of the outcome were captured when the insertion was complete or stopped (force threshold exceeded).

Note that the data can be organized hierarchically into three categories (no error, positional error in x direction, and positional error in y direction). The latter two categories can be further split in half depending on the direction of the error ($x/+$, $x/-$, $y/+$, $y/-$). Finally, we can split based on the magnitude of the error (e.g., $x/+2$, meaning that we have a 2 mm error in the $x+$ direction).

3.3. The Task of Inserting Copper Sliding Rings into Metal Pallets

The car starter assembly process includes inserting copper sliding rings into metal pallets, as shown in Figure 5. This can be categorized as a multiple peg-in-hole problem, as it is necessary to insert the bottom of the copper ring and both upper part lugs correctly. The insertion process is challenging due to the deformability of the sliding rings. This task has been taken from a real production process where it is performed manually. Previous automation attempts have failed due to the low success rate that was achieved. To ensure robust insertion, we proposed to use the exception strategy framework [43].

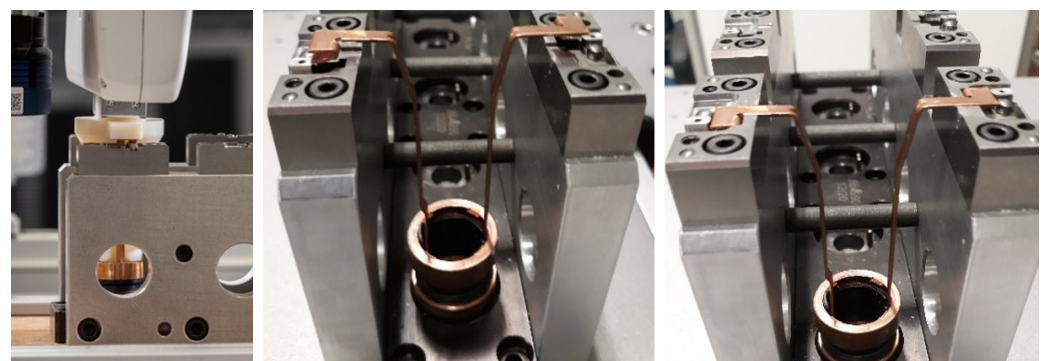


Figure 5. Left: Insertion of the ring into a modeling fixture. Center: Incorrect insertion. Right: Correct insertion.

We have collected a database of twelve copper ring insertions, which includes both successful insertions and deliberately induced insertion failures. The failures were caused by the displacement of the target position for insertion in the x and y directions:

- no displacement, leading to successful insertion;
- positional displacement in the x direction, with Δp_x between 1 and 3 mm in 1 mm steps;
- positional displacement in the y direction with Δp_y between 1 and 3 mm in 1 mm steps, both leading to unsuccessful insertion.

Additionally, we recorded insertion attempts with deformed parts, which also led to an unsuccessful insertion. For each of the cases, we recorded at least four insertion attempts. The process was repeated for all four slot positions in the molding cast. Each entry consists of a snapshot of the outcome of the insertion task (cropped RGB image) and the time series of force $\mathbf{F} = (F_x, F_y, F_z)$ and torque $\mathbf{T} = (T_x, T_y, T_z)$ measurements. Similarly to the PiH dataset, the gathered data can be organized hierarchically.

3.4. Force-Torque Data Extraction: Contact Vector Estimation

In our previous work [6], we have considered only errors due to the offset in the grasping angle and have shown that force-torque data can be used to determine a suitable context descriptor using principal component analysis (PCA), which correlated most strongly with the grasping error. Such a dimensionality reduction is beneficial because the generation of an appropriate refinement trajectory based on statistical learning is sensitive to the dimension of the feature space. Another possible approach to reduce the dimensionality of force-torque measurements is to determine contact points [48].

The point of contact between two parts can be estimated based on the relationship between force $\mathbf{F}$, torque $\mathbf{T}$, and lever $\mathbf{r}$ by using the following formulation [11]

$$\mathbf{r}(\alpha) = \frac{\mathbf{F} \times \mathbf{T}}{\|\mathbf{F}\|^2} + \alpha \frac{\mathbf{F}}{\|\mathbf{F}\|}, \quad (1)$$

where α is a suitably chosen constant so that the vector $\mathbf{r}$ touches the environment as illustrated in Figure 6a. The measured forces and torques must be expressed in the robot end-effector coordinate system.

However, the contact point estimation cannot always distinguish between the different types of errors, as illustrated in Figure 6b. Thus, forces and torques, as well as the positional data, cannot uniquely determine the context. In order to resolve this ambiguity, we introduce another modality, as discussed in the remainder of this paper.

Nevertheless, Equation (1) provides a suitable representation that can distinguish between different conditions of the same outcome type. A graphical example of contact vector estimation for both experiments is shown in Figure 7.

Our preliminary results have shown that the inclusion of raw force-torque data as features decreases the performance of the final model. Thus the feature vector for the FT modality was chosen to include only the contact point vector estimate. For each example $k \in \mathcal{E}$, the feature vector is calculated as:

$$\mathbf{f}_{\text{FT}}^k = (r_x, r_y, r_z), \quad (2)$$

where r_x, r_y, r_z are components of the vector $\mathbf{r}(\alpha)$.

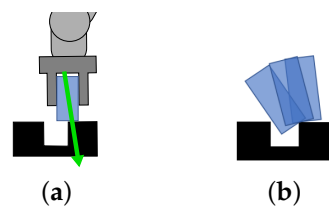


Figure 6. A scheme depicting contact point estimation (a) and another example where the grasped part comes into contact with the environment at the same point (b). The robot and the gripper are represented by the dark gray shape, whereas the grasped part and the environment are shown in blue and black, respectively. The contact vector estimate is shown with a green arrow.

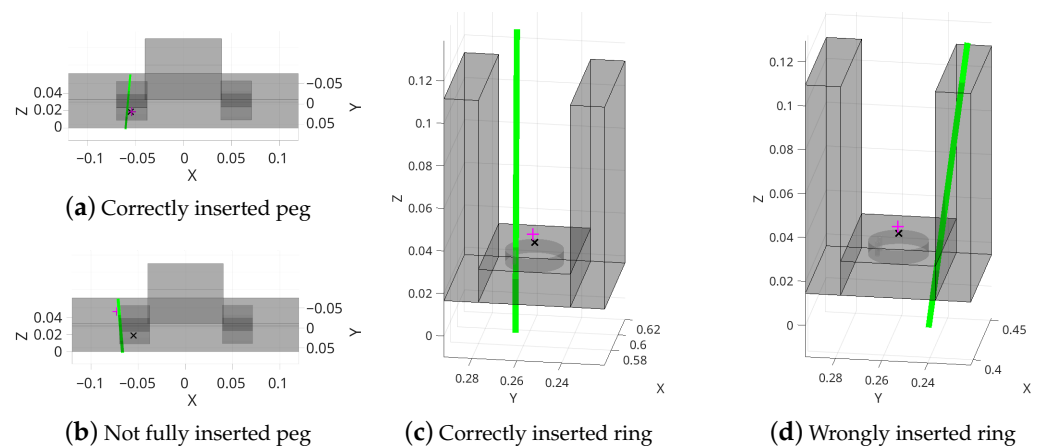


Figure 7. Contact vector estimation in four examples of the considered assembly tasks. A schematic model of the Cranfield base plate and the casting mold is shown in (a–d), respectively. The green line shows the contact vector, whereas the pink plus symbol shows the robot’s tool center point (TCP) at the time of contact and the black cross shows the target reference position.

3.5. Vision Data Extraction: Instance Segmentation with YOLACT

We applied deep neural networks (DNN) to perform feature extraction from image data. They provide good flexibility because pre-trained NN models and frameworks can be re-trained by using a custom dataset for a specific use case, in contrast to the classic computer vision (CV) algorithms, which tend to be more domain specific [49]. Compared to the traditional computer vision methods (e.g., edge detection), they often require less manual fine-tuning.

Various convolutional neural networks (CNNs) have proven to be suitable for analyzing image data. An essential issue with a custom network that directly extracts features is that retraining is needed when a new error class is added or the camera position is changed. For these reasons, we rely on models that are designed to be less prone to changes in object position in the picture. This has been extensively studied in object detection and instance segmentation models. Instance segmentation is an enhanced type of object detection that generates a segmentation map for each detected instance of an object in addition to the bounding boxes.

In order to meet the above-listed requirements, we used the state-of-the-art instance segmentation model YOLACT [12]. YOLACT builds upon the basic principles of RetinaNet [50] with the Feature Pyramid Network [51] and ResNet-101 [52] as a convolutional backbone architecture for feature extraction. It utilizes a fully convolutional network to directly predict a set of prototype masks on the entire image. Lastly, a fully connected layer assembles the final masks as linear combinations of the prototype masks, followed by bounding box cropping. Compared to most of the previous instance segmentation approaches, such as Mask R-CNN [53], which are inherently sequential (the first image is scanned for regions with object candidates, then each of them is processed separately), YOLACT is a one-stage algorithm that skips this intermediate localization step. This allows for nearly real-time performance. By using shallower computational backbones, such as ResNet-50, even faster performance can be achieved at a minimal accuracy cost when compared to ResNet-101 [12].

The (re)training of YOLACT requires labeled images and ground truth image masks. We have used an open-source graphical image annotation tool, Labelme, to annotate images in our datasets (<https://github.com/wkentaro/labelme>, accessed on 13 October 2022). For the PiH dataset, we manually annotated 30 images for each position using four different light settings. We manually annotated 10 images for each position using two different light settings for the copper ring insertion dataset. We split the annotated datasets into training and validation partitions, with 80% and 20% of the data, respectively. Finally, the annotations had to be transformed into a format compatible with the YOLACT

training script (COCO). For this purpose, we prepared an open-source tool—labelme2coco (<https://github.com/smihael/labelme2cocosplit>, accessed on 13 October 2022).

In the proposed pipeline, we configured YOLACT to use a computationally lighter ResNet-50 as the backbone. This enabled us to use original-resolution images while retaining high training and inference speed. We trained two models for each of the above-presented datasets. During training, the algorithm used a batch size of 8, weight decay of 0.0005, and image size of 1280×720 pixels (PiH dataset) or 221×381 pixels (copper ring insertion dataset). The initial learning rate was set to 0.001. The model was trained for 40,000 iterations, and the decay rate of 0.1 was applied once each 10,000 iterations. The training took 8 h on a GeForce GTX 1060 GPU for the PiH dataset and 6 h for the copper ring insertion dataset.

Once the models were trained, we deployed them to a workstation in the robotic workcell. The integration was done using a modified yolact_ros package (https://github.com/smihael/yolact_ros, accessed on 13 October 2022), which also allows using the learned model for inference without a GPU, thus lowering the computational requirements.

The results are shown in Figure 8. The PiH model is trained to distinguish between the peg and the base plate, whereas the copper ring insertion model is able to distinguish the following segments: gripper, mold, screws, ring base, wings, and ears (lugs).

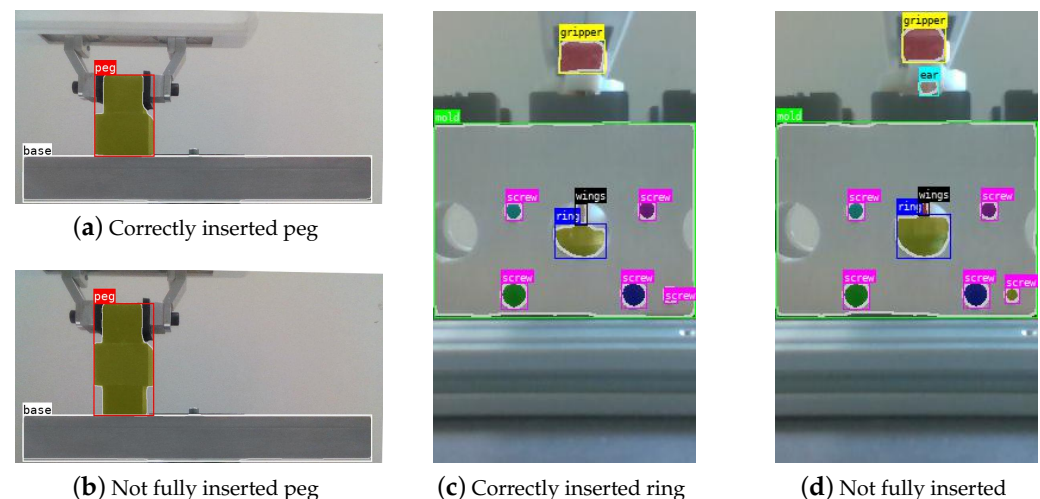


Figure 8. Bounding boxes and masks obtained by YOLACT using snapshots of the outcome of both tasks as input. For the PiH task (a,b), the base plate and peg are detected, regardless of the position/occlusion of the latter. In the copper ring insertion task (c,d), the gripper, mold, screws, ring base, wings, and ears are detected. Notice that ears are only detected when the part is not fully inserted.

Note that the PiH model can be equally used for any of the two insertion slots in the PiH task. Likewise, the copper rings model can be used for any of the four slots in the copper ring insertion task. Since the model is position invariant, meaning that the model is able to correctly mark the area of different image parts regardless of where in the image they appear, we can apply it for the analysis of new error cases.

3.6. Extracting a Fixed-Size Feature Vector from Instance Segmentation Results

Using the trained YOLACT segmentation models, we obtain bounding boxes and image masks for all images in the PiH and copper ring insertion datasets. The information obtained from instance segmentation needs to be further processed to be used in further steps of the pipeline, as shown in Figure 9. We extract fixed-size feature vectors, as ensembles of predictive clustering trees do not operate over image masks.

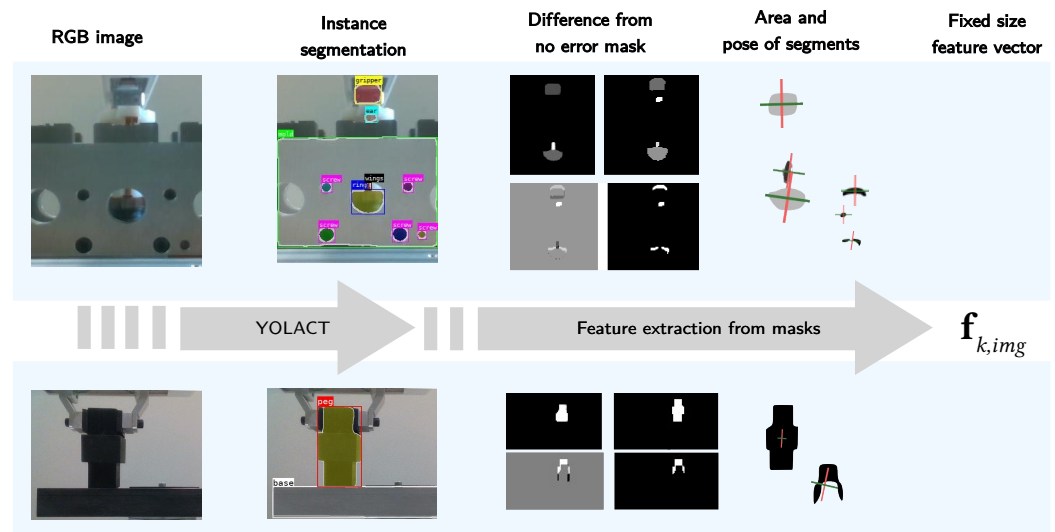


Figure 9. Image features extraction pipeline. The RGB snapshot of a situation is processed by a YOLOACT model to obtain masks of different parts of interest. The obtained masks are then processed to obtain a low-dimensional fixed-size feature vector.

An image is represented as a $w \times h \times 3$ matrix of pixels $\mathbf{I}(x, y, c) \in \{0, 1, \dots, 255\}$, representing the RGB color channels. The image can contain multiple instances of different objects. For each segmented object instance s , we obtain its type, the bounding box, and the mask. The bounding box $\mathbf{B}_s$ is given as a pair of pixel coordinates of two diagonal corners $\{(x_1, y_1), (x_2, y_2)\}$. The bounding box can be represented by the centroid $\mathbf{c}_s$, width w_s , and height h_s of the rectangle

$$\mathbf{c}_s = [c_{s,x}, c_{s,y}]^T = \left[\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2} \right]^T, \quad (3)$$

$$w_s = x_2 - x_1, \quad (4)$$

$$h_s = y_2 - y_1. \quad (5)$$

The pixels belonging to the specific object instance s are represented with masks. Each mask is a $w \times h$ binary matrix $\mathbf{M}_s \in \mathbb{B}^{w \times h}$, which tells whether a pixel is part of the mask or not. Using PCA, we determine the first principal component for each instance's mask $\mathbf{e}_{s,1} = (x, y)$. This result can be used to calculate the orientation of the part in the image plane (visualized in Figure 10)

$$\varphi_s = \arctan\left(\frac{-y}{x}\right) - \frac{\pi}{2}. \quad (6)$$

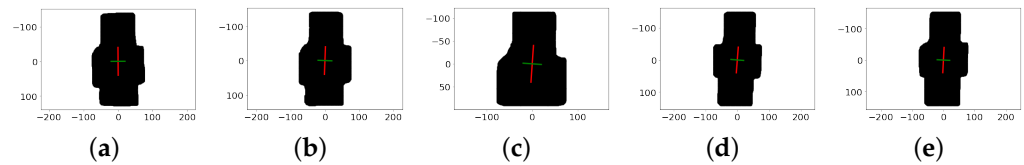


Figure 10. The extracted peg mask for different executions of the copper ring insertion task. Red and green lines show the principal directions and determine the mask's orientation. (a) $\Delta p_x = -10$ mm; (b) $\Delta p_x = -5$ mm; (c) $\Delta p_x = 0$ mm; (d) $\Delta p_x = 5$ mm; (e) $\Delta p_x = 10$ mm.

Additionally, we calculate the pixel area of each instance mask as a total number of all true elements in the instance matrix

$$a_s = \sum_{x=0}^w \sum_{y=0}^h m_s(x, y). \quad (7)$$

In both experiments, we define a set of object of interests $\mathcal{S}_{\text{int}}$ (see Figure 8). For the peg-in-hole insertion task, it consists of the peg only, while for the copper ring insertion task, $\mathcal{S}_{\text{int}}$ contains the gripper, ring, wings, and ear. Note that additional features, e.g., screws on the molding cast or the base of the Cranfield benchmark, can be used as calibration features. In our case, this was not needed as the datasets were recorded with a fixed camera position.

From the set of examples with no visible errors $\mathcal{E}_0$, we calculate the average segment mask $\bar{\mathbf{M}}_s$ for each object instance s of interest from $\mathcal{S}_{\text{int}}$. The average mask is calculated as the element-wise mean of the mask matrices:

$$\bar{\mathbf{M}}_s = \frac{1}{|\mathcal{E}_0|} \sum_{k \in \mathcal{E}_0} \mathbf{M}_s^k \in \mathbb{R}_+^{w \times h}. \quad (8)$$

For other examples, we compute the difference between their masks and the average mask of examples with no error $\tilde{\mathbf{M}}_{s,\text{diff}}^k = \mathbf{M}_s^k - \bar{\mathbf{M}}_s$, and take only its positive elements to define binary matrix $\mathbf{M}_{s,\text{diff}}^k$,

$$m_{s,\text{diff}}^k(x, y) = \begin{cases} 1, & \tilde{m}_{s,\text{diff}}^k(x, y) > 0 \\ 0, & \text{otherwise} \end{cases}. \quad (9)$$

An example is shown in Figure 11. For each of the obtained difference segments, we then calculate its center $\mathbf{c}_{s,\text{diff}} = [c_{s,\text{diff},x}, c_{s,\text{diff},y}]^\top$ and pixel area $A_{s,\text{diff}}$ using Equations (3) and (7), respectively.

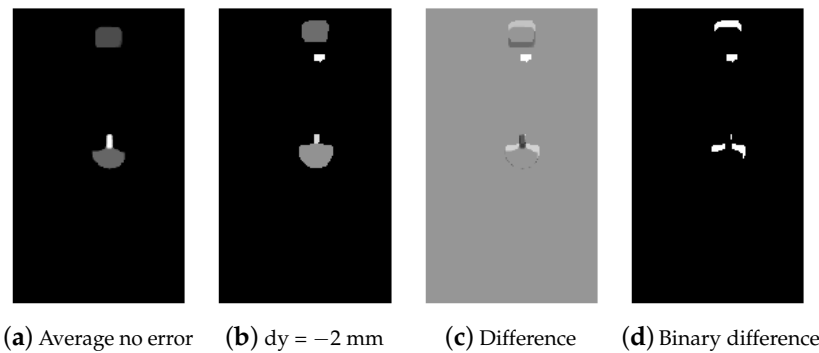


Figure 11. From left to right: (a) average segmentation mask for the successful copper ring insertion attempts, (b) segmentation mask for a failed insertion attempt (positional error in the y -direction), (c) difference of the segmentation masks, and (d) positive part of the difference.

In this way, we obtain an image feature vector for each example $k \in \mathcal{E}$ and object of interest $s \in \mathcal{S}_{\text{int}}$:

$$\mathbf{f}_s^k = [\mathbf{c}_s^{k\top}, w_s^k, h_s^k, \phi_s^k, a_s^k, \mathbf{c}_{s,\text{diff}}^{k\top}, a_{s,\text{diff}}^k]^\top. \quad (10)$$

3.7. Combining Image Features and Force-Torque Measurements Using Ensembles of Predictive Clustering Trees

We formulate the determination of the outcome of the insertion task as a hierarchical multi-label classification (HMLC) problem. Given the extracted image features and the estimate of the contact point, the type of outcome should be predicted. For the different types of outcomes, a hierarchy of class labels defines the direction and magnitude of the underlying error, as described below.

We applied ensembles of predictive clustering trees (PCTs) [10] for this task. PCTs are a generalization of ordinary decision trees [41]. Generally, in a decision tree, an input is entered at the top and as it traverses down the tree, the data gets bucketed into smaller and smaller sets until the final prediction can be determined. The PCT framework, however, views the decision tree as a hierarchy of clusters: the top node corresponds to the cluster containing all of the data, which is recursively partitioned into smaller clusters so that per-cluster variance is minimized [39]. In this way, cluster homogeneity is maximized, and consequently, the predictive performance of the tree is improved.

PCT ensembles consist of multiple trees. In an ensemble, the predictions of classifiers are combined to get the final prediction. For an ensemble to have better predictive performance than its individual members, the base predictive models must be accurate and diverse [9]. The diversity between trees in the PCT framework is obtained by using multiple replicas of the training set and by changing the feature set during learning, as in the random forest method [54].

In our setting, each example k from the set of examples $\mathcal{E}$ consists of all extracted features $\mathbf{f}_k$ and the corresponding label vector $\mathbf{l}_k$. The feature vectors are obtained by concatenating per-modality features:

$$\mathbf{f}_k = \left[\mathbf{f}_{\text{FT}}^k, \mathbf{f}_1^k, \mathbf{f}_2^k, \dots, \mathbf{f}_{|\mathcal{S}|}^k \right]^\top, \quad (11)$$

with $\mathbf{f}_{\text{FT}}^k$ and $\mathbf{f}_s^k$, $s = 1, \dots, |\mathcal{S}|$, defined as in Sections 3.4 and 3.6, respectively. To define the corresponding label vector, we first observe that in HLMC, each example can have multiple labels. Classes are organized in a hierarchical structure, i.e., an example belonging to a class also belongs to all of its superclasses. The resulting ordered set of classes is used to define a binary label vector $\mathbf{l}_k \in \mathbb{B}^L$. The components of $\mathbf{l}_k$ are equal to 1 if the example is labeled with the corresponding class and 0 otherwise. L denotes the number of all classes in the hierarchy.

For the PiH task, the set of labels at the first hierarchical level consists of “no error”, and “x” and “y” for the error in one of the two directions. At the second level, we have “x+” and “x−” as subclasses of “x”, and “y+” and “y−” as subclasses of “y”. Likewise, we have “x + 1”, “x − 1”, “y + 1”, “y − 1”, “x + 2”, ..., “y − 10” at the third hierarchical level. For the copper ring insertion task, the set of labels at the first hierarchical level consists of: “no error”, “bad part”, and “x” and “y” for the error in one of the two directions. Similarly, as in the PiH task, the sub-classes at the second and third levels are representing various magnitudes of error (ranging from −10 to 10 mm in 1 mm steps) in both considered directions (x and y).

In summary, to train the ensemble of predictive trees, we collect the dataset $\mathcal{E}$

$$\mathcal{E} = \{\mathbf{f}_k, \mathbf{l}_k\}_{k=1}^K. \quad (12)$$

After training we can use the resulting ensemble of predictive trees to predict the labels $\mathbf{l}$ given the extracted feature vector $\mathbf{f}$.

We trained multiple ensembles for two different tasks. See Section 4 for more details. We used PCT ensembles, i.e., random forests of PCTs, as implemented in the CLUS system (CLUS is available for download at <http://source.ijs.si/ktclus/clus-public>, accessed on 13 October 2022) for this purpose. Each ensemble consisted of 50 trees. As a heuristic to evaluate the splits in decision trees, we used the variance reduction [39]. The variance for the set of examples $\mathcal{E}$ is defined as the average squared distance between each example’s label vector $\mathbf{l}_k$ and the set’s mean label vector $\hat{\mathbf{l}}$, i.e.,

$$\text{Var}(\mathcal{E}) = \frac{1}{|\mathcal{E}|} \sum_{k \in \mathcal{E}} d(\mathbf{l}_k, \hat{\mathbf{l}})^2 \quad (13)$$

The distance measure used in the above formula is the weighted Euclidean distance:

$$d(\mathbf{l}_1, \mathbf{l}_2) = \sqrt{\sum_{i=1}^L w(c_i)(l_{1,i} - l_{2,i})^2}, \quad (14)$$

where the class's weight $w(c_i)$ depends on its depth within the hierarchy. The similarities at higher levels in the hierarchy are considered more important than the similarities at lower levels. Therefore, the class weights $w(c_i)$ decrease with the depth of the class in the hierarchy. $w(c_i)$ is typically set as w_0^d , where d is the depth of the label in the hierarchy: w_0 was set to 0.75 in our experiments. The number of randomly selected features at each node was set to $\lfloor \sqrt{L} \rfloor + 1$, where L is the total number of features.

To combine the predictions of all classifiers in the ensemble and obtain the final prediction, their average is taken.

4. Results

In this section, we evaluate the performance of our proposed approach along two dimensions: its generalizability to handle unseen data and the effect of including/excluding features from separate modalities. Finally, the setup was experimentally verified in the robotic workcell.

4.1. Generalizability of Classification

In order to verify how well the approach can generalize to unseen data, we train a model on a subset where we do not include any examples of a particular outcome case. Since the database for the copper ring insertion task is not sufficiently fine-grained, this aspect was evaluated only for the peg-in-hole task. We excluded all cases where the positional error in any direction equals 5 mm and observed if the model could correctly predict the direction of error for the excluded examples. The results are shown in Figure 12. The model correctly predicted the direction of error for all the excluded examples, both at the first and the second level of hierarchy. As the predictions at the third hierarchical level describe the magnitude of error, they can also be evaluated using root mean square error (RMSE). At the third hierarchical level, it assigned all the excluded examples to the closest lower error class that was presented in training for the x direction ($\text{RMSE}_x = 1$ mm), whereas for the y direction it did so for 4 of the 6 examples (resulting in $\text{RMSE}_y = 1.91$ mm).

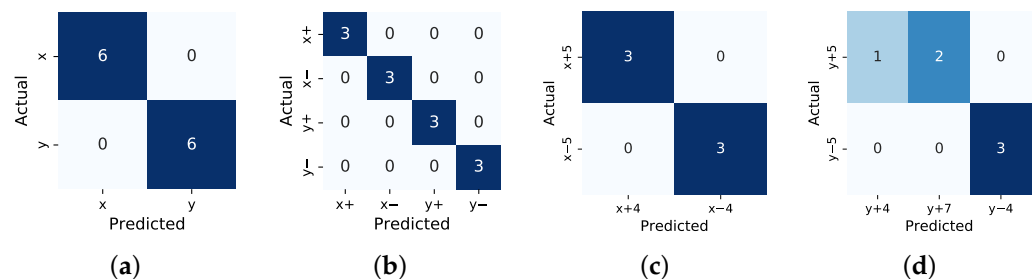


Figure 12. Confusion matrices for classification at different levels of the hierarchy. (a) First level of hierarchy (x or y displacement). (b) Second level (negative or positive displacement). (c) Third level (magnitude of x displacement); (d) Third level (magnitude of y displacement).

4.2. Single Modality versus Multimodal Models for Classification

We assessed the effectiveness of including/excluding features from the individual modalities by training multiple PCTs on different subsets of features for both tasks:

- only features based on the image data (see Section 3.6);
- only features based on the force-torque sensor data (see Section 3.4);
- features from both modalities.

Models were trained using 80% of the data and tested on the remaining 20% of the data.

The results for the PiH task are given in Figure 13. We found out that the model performed best when all features were included, indicating that the features from both modalities are complementary and improve the model's performance. The overall classification accuracy was calculated for the multi-class classification problem by taking the sum of the true positives and true negatives for each label, divided by the total number of predictions made. The accuracy was then averaged by support (the number of true instances for each label). For the model that uses all features, the overall classification accuracy at the first hierarchical level was 0.98. At the second level, the accuracy was 1.0. For the third level, the overall classification accuracy was 0.68. For error classes in the x and y directions, the classification accuracy was 0.55 and 0.5, respectively. For the model that only uses features from the vision modality, the overall classification accuracy at the first two hierarchical levels stayed the same, indicating that vision features can distinguish well between different types of outcomes. The overall classification accuracy at the third level was 0.68, and 0.5 and 0.55 for the x and y directions, respectively. When evaluating the model that only uses features from the FT modality, the overall classification accuracy at the first level dropped to 0.92, at the second to 0.95, and at the third to 0.61, whereas it was 0.5 and 0.35 for the x and y directions, respectively.

The results for the copper ring insertion task are given in Figure 14. Similar to before, we found that the model performed best when all features were included. For the model that uses all features, the overall classification accuracy at the first two hierarchical levels was 0.88. For the third level, the overall classification accuracy was 0.81. For error classes in x direction, the classification accuracy is 0.67, and 0.9 for y direction. For the model that only uses features from the vision modality, the overall classification accuracy at the first two hierarchical levels dropped slightly to 0.85. The overall classification accuracy at the third level was 0.62, and 0.5 and 0.6 for the x and y directions, respectively. The drop was even more pronounced when evaluating the model that only uses features from the FT modality. The overall classification accuracy at the first level was 0.81 and at the second and third it was 0.77, whereas it was 0.58 and 0.9 for the x and y directions, respectively. When comparing the results of the vision- and FT-features-only models, it is evident that while the earlier achieved a higher overall accuracy, the latter achieved higher accuracy when distinguishing among different magnitudes of error in the y direction.

4.3. Verification of Error Context Determination for the Generation of Exception Strategies

The proposed framework was experimentally verified on both the PiH and the copper ring insertion task. The initial PiH policy was carefully programmed and executed in the workcell with the same setup as described in Section 3.2. In order to cause an exception, the target position was displaced by 6 mm. The proposed approach correctly estimated the error context to " $x/-/6$ ". Since the exception strategy for this case has not yet been programmed, the robot stopped and prompted the operator. Using kinesthetic guidance, the operator guided the robot back along the policy to an appropriate point, where it is possible to resume the operation. The operator then demonstrated the correction, which resolved the problem. When we displaced the target position by 6 mm again, which resulted in a similar outcome, the robot again classified it as " $x/-/6$ ". As the exception strategy is now known, the robot could resolve the situation using the demonstrated exception strategy. In a similar manner, the operator demonstrated policies for the case where the target was displaced by 4 mm in a positive x direction. When we displaced the target position by 5 mm, the robot correctly classified the context to be " $x/-$ ", whereas the magnitude was not determined precisely (4 mm) as we used the model that did not include error contexts of this magnitude in the training set. Nevertheless, by combining the policies demonstrated for the other two cases in the " $x/-$ " category and using locally weighted regression, as proposed in [6,43], the robot was able to perform the insertion successfully.

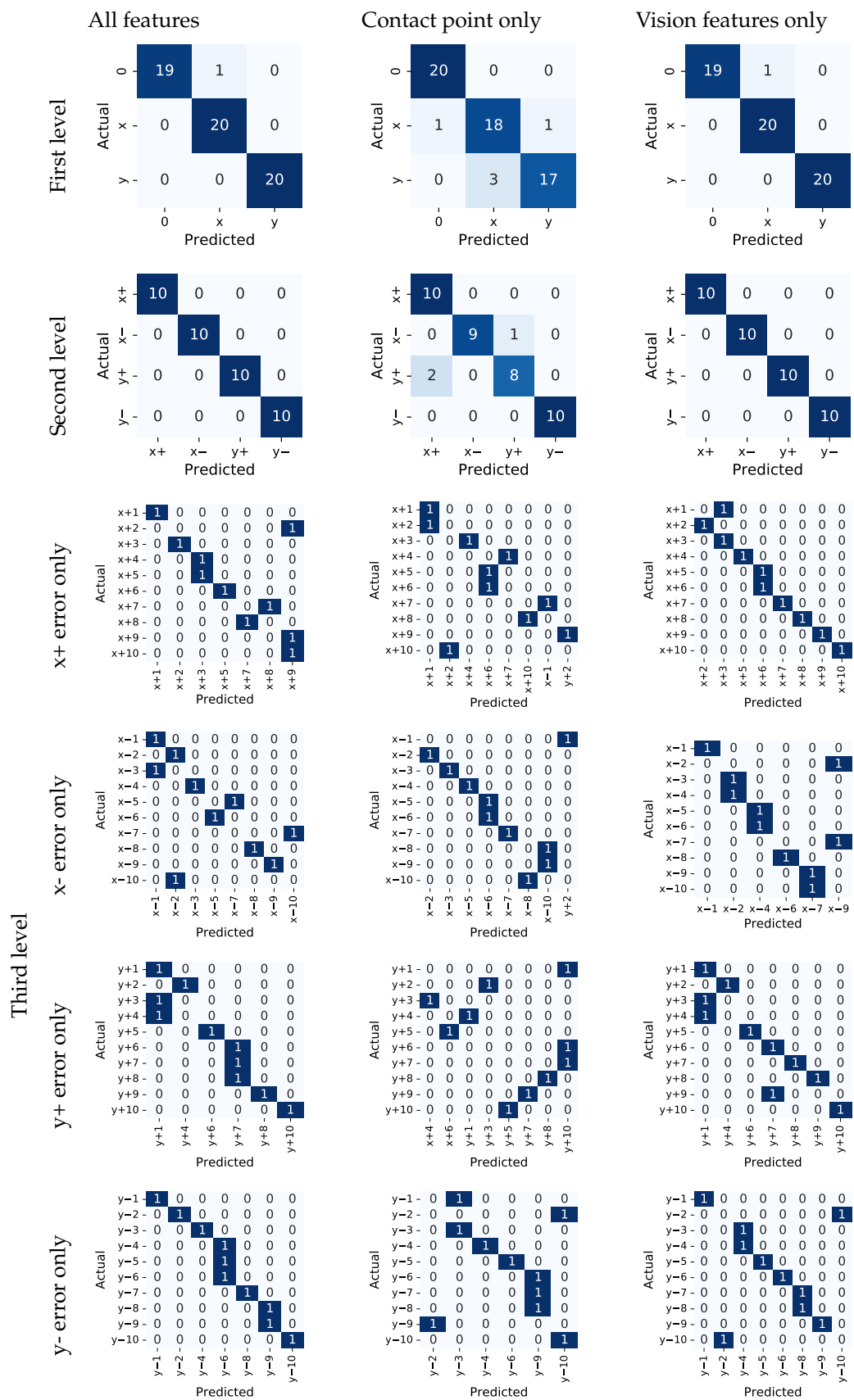


Figure 13. Confusion matrices for the PiH use case.

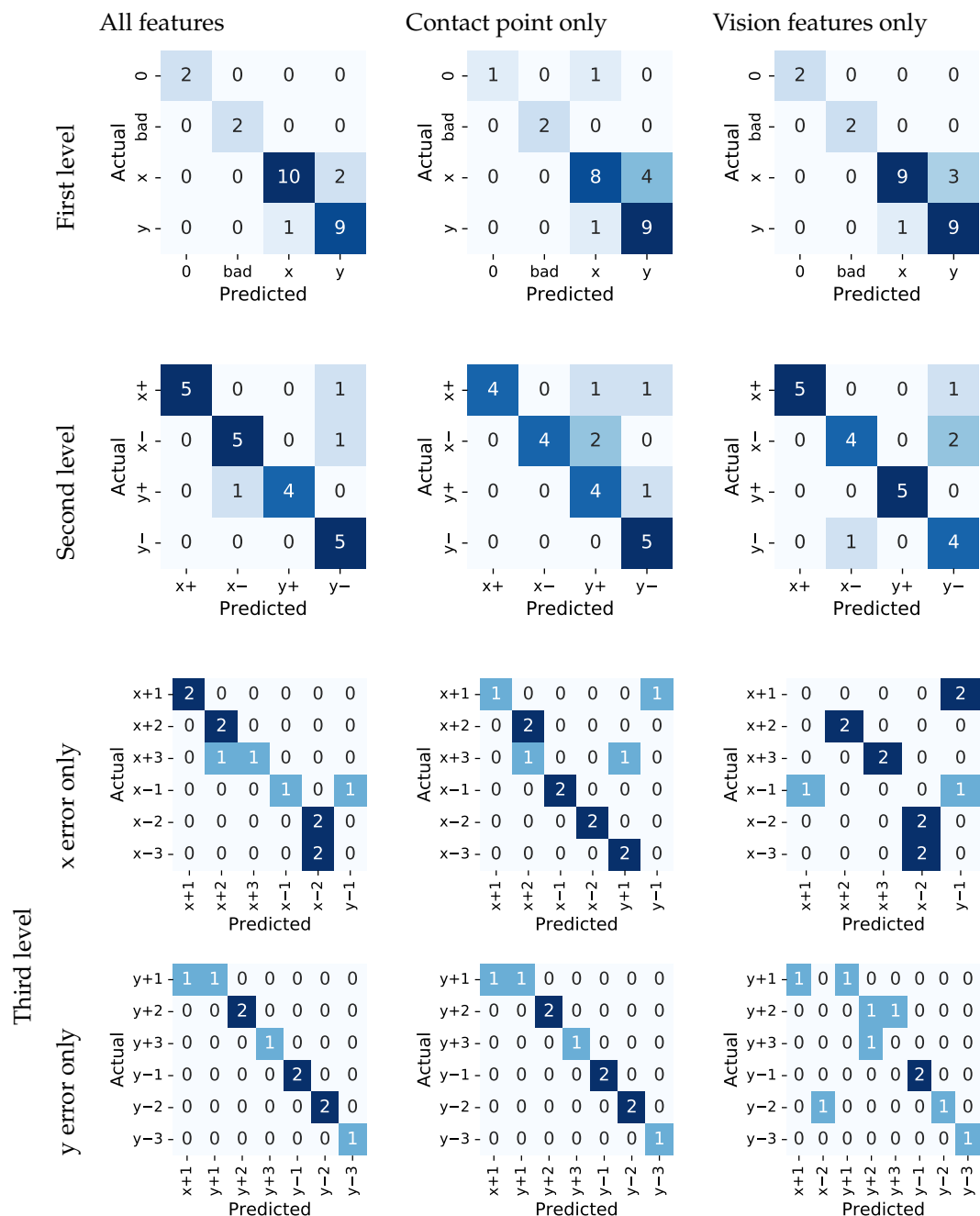


Figure 14. Confusion matrices for the copper ring insertion task.

When inserting a sliding ring into the casting mold, there are two major types of errors. The first type is when the base of the ring is not properly seated into the mold (see Figure 5 middle). This type of error mainly arises due to imprecise grasping or due to errors in the target position. The second type of error occurs when the sliding ring is deformed. Both types of error can be reliably determined by using the proposed approach. We first displaced the target position by 2 mm in the negative y direction so that the insertion failed. As the exception strategy for this case has not yet been programmed, the operator demonstrated how to resume the operation and resolve the issue using iterative kinesthetic guidance [7]. When the target was displaced by the same offset again, the robot was able to resolve the problem. We also started the insertion procedure with a deformed part. It was correctly determined, and the robot placed it into the bin for deformed parts. An example video of both experiments can be found in the Supplementary Materials.

Note that the application of the exception strategy learning framework does not affect the cycle time in successful attempts. Once the model is deployed, the time to obtain context estimation is negligible. In unsuccessful attempts, where an alternate policy needs to be demonstrated or executed, the cycle time is, however, prolonged. However, since these situations are less frequent, this has very little effect on the average cycle time of an automated line.

5. Discussion

The results of our study indicate that the application of multimodal features leads to an improved classification accuracy of the ensemble models employed for classification. This implies that the features are complementary and taken together provide greater discrimination power than the features stemming from a single modality. Prediction errors that arose when applying vision-only-based models, showed the limitations of two-dimensional image data, thus depth information should be considered in the future.

It is important to note that, to a large extent, the models were able to correctly assign error types to examples with a magnitude of error not included in the training data. This is a critical finding as it indicates that the computed models are robust and can be used in real-world applications, also in less-structured non-industrial environments, where error types can not be predicted in advance. To evaluate this aspect, we excluded all examples with a certain magnitude of error from the training set. The results show that the models still perform well, indicating that they are not overfitting the training data.

Based on the observation that features obtained from different modalities contributed to improving the classification results at different levels, a more explicit hierarchical pipeline could be considered in the future, exploiting the robot as an agent that can interact with the environment. Data from different modalities would contribute towards the final prediction at different stages of the process, consisting of, e.g.,

- (1) the type of error (due to positional displacement, part geometry, imprecise grasping), determined based on image data;
- (2) the magnitude of error, based on force-torque or depth data.

The context determination does not have to occur instantaneously but can include exploring the environment as part of the pipeline. We could first use the vision data to determine the direction in which the robot should move in order to reduce the error (left/right). The robot can then move in this direction until it detects a new state (one of the force-torque components changes or a compliant robot stops moving as it hits an obstacle—see [55]). In the newly found state, the robot again estimates the direction in which it needs to continue or stop.

In the future, we intend to expand the proposed approach by considering other possible error types (e.g., arising from erroneous orientation when grasping) and their combinations (displacements in multiple degrees of freedom at the same time), as well as properly handling continuous data (regression at the lower hierarchical level instead of classification). We believe that the presented framework is not only applicable to learning error context but could also be extended to cognitive systems that will be able to respond autonomously to changes in the environment. To achieve these goals, the improved versions of our approach should consider additional modalities and alternative features extraction methods.

6. Conclusions

In this work, we have proposed a novel method for context determination based on multimodal features that can be used for learning exception strategies in various assembly tasks.

Our approach was validated on two tasks, the classic peg-in-hole, and the copper ring insertion. To evaluate its effectiveness, we deliberately induced different types of errors, which led to failed task executions. Using the proposed approach, the error type

was correctly obtained in all cases, allowing for correction of the task execution parameters and finally leading to successful task performance.

The study results indicate that the features used in the ensemble models are complementary and that the multimodal setup achieves the highest classification accuracy. Moreover, the model can correctly assign error types to examples with an unknown error magnitude.

In the current implementation, the context was calculated based on the measurement of forces and torques and RGB sensor data. The introduction of further sensor modalities (such as depth data) could lead to a further increase in classification accuracy.

Supplementary Materials: The following are available at <https://www.mdpi.com/article/10.3390/s22207962/s1>.

Author Contributions: Conceptualization, M.S. and B.N.; methodology, M.S., B.N. and S.D.; software, M.S., M.M.H. and S.D.; validation, M.S.; formal analysis, M.S. and A.U.; investigation, M.S.; resources, B.N. and A.U.; data curation, M.S. and M.M.H.; writing—original draft preparation, M.S.; writing—review and editing, M.S., S.D., A.U. and B.N.; visualization, M.S. and M.M.H.; supervision, B.N., A.U. and S.D.; funding acquisition, B.N. and A.U. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the European Union’s Horizon 2020 through the ReconCycle project (contract no. 871352), the CoLLaboratE project (contract no. 820767), and by the Slovenian Research Agency (core research funding no. P2-0076).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data analyzed in this study are openly available in <https://doi.org/10.5281/zenodo.7221443> (accessed on 13 October 2022) and <https://doi.org/10.5281/zenodo.7221387> (accessed on 13 October 2022).

Acknowledgments: The authors would like to thank Rok Pahič for valuable discussions on extracting image features and Martin Breskvar for help with the CLUS setup in the early stages of this research.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. International Federation of Robotics. *World Robotics 2021 Industrial Robots*; VDMA Services GmbH: Frankfurt, Germany, 2021.
2. Roveda, L.; Magni, M.; Cantoni, M.; Piga, D.; Bucca, G. Human–robot collaboration in sensorless assembly task learning enhanced by uncertainties adaptation via Bayesian Optimization. *Robot. Auton. Syst.* **2021**, *136*, 103711. [\[CrossRef\]](#)
3. Gašpar, T.; Deniša, M.; Radanovič, P.; Ridge, B.; Savarimuthu, R.; Kramberger, A.; Priggemeyer, M.; Roßmann, J.; Wörgötter, F.; Ivanovska, T.; et al. Smart hardware integration with advanced robot programming technologies for efficient reconfiguration of robot workcells. *Robot. Comput.-Integr. Manuf.* **2020**, *66*, 101979. [\[CrossRef\]](#)
4. Zhu, Z.; Hu, H. Robot Learning from Demonstration in Robotic Assembly: A Survey. *Robotics* **2018**, *7*, 17. [\[CrossRef\]](#)
5. Zachares, P.; Lee, M.A.; Lian, W.; Bohg, J. Interpreting Contact Interactions to Overcome Failure in Robot Assembly Tasks. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Xi’an, China, 30 May–5 June 2021; pp. 3410–3417.
6. Nemec, B.; Simonič, M.; Ude, A. Learning of Exception Strategies in Assembly Tasks. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 6521–6527.
7. Simonič, M.; Petrič, T.; Ude, A.; Nemec, B. Analysis of Methods for Incremental Policy Refinement by Kinesthetic Guidance. *J. Intell. Robot. Syst.* **2021**, *102*, 5. [\[CrossRef\]](#)
8. Roheda, S.; Krim, H.; Riggan, B.S. Robust multi-modal sensor fusion: An adversarial approach. *IEEE Sens. J.* **2020**, *21*, 1885–1896. [\[CrossRef\]](#)
9. Polikar, R. Ensemble learning. In *Ensemble Machine Learning*; Springer: Berlin/Heidelberg, Germany, 2012.
10. Kocev, D.; Vens, C.; Struyf, J.; Džeroski, S. Tree ensembles for predicting structured outputs. *Pattern Recognit.* **2013**, *46*, 817–833. [\[CrossRef\]](#)
11. Nemec, B.; Yasuda, K.; Mullennix, N.; Likar, N.; Ude, A. Learning by demonstration and adaptation of finishing operations using virtual mechanism approach. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 7219–7225.

12. Bolya, D.; Zhou, C.; Xiao, F.; Lee, Y.J. YOLACT: Real-Time Instance Segmentation. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019; pp. 9156–9165.
13. Nottensteiner, K.; Sachtler, A.; Albu-Schäffer, A. Towards Autonomous Robotic Assembly: Using Combined Visual and Tactile Sensing for Adaptive Task Execution. *J. Intell. Robot. Syst.* **2021**, *101*, 49. [\[CrossRef\]](#)
14. Visinsky, M.; Cavallaro, J.; Walker, I. Robotic fault detection and fault tolerance: A survey. *Reliab. Eng. Syst. Saf.* **1994**, *46*, 139–158. [\[CrossRef\]](#)
15. Abu-Dakka, F.; Nemec, B.; Kramberger, A.; Buch, A.; Krüger, N.; Ude, A. Solving peg-in-hole tasks by human demonstration and exception strategies. *Ind. Robot. Int. J.* **2014**, *41*, 575–584. [\[CrossRef\]](#)
16. Marvel, J.A.; Bostelman, R.; Falco, J. Multi-Robot Assembly Strategies and Metrics. *ACM Comput. Surv.* **2018**, *51*, 1–32. [\[CrossRef\]](#) [\[PubMed\]](#)
17. Laursen, J.S.; Schultz, U.P.; Ellekilde, L.P. Automatic error recovery in robot assembly operations using reverse execution. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–3 October 2015; pp. 1785–1792.
18. Nicolescu, M.N.; Mataric, M.J. Learning and interacting in human-robot domains. *IEEE Trans. Syst. Man Cybern. Part A* **2001**, *31*, 419–430. [\[CrossRef\]](#)
19. Shetty, S.; Silverio, J.; Calinon, S. Ergodic Exploration Using Tensor Train: Applications in Insertion Tasks. *IEEE Trans. Robot.* **2021**, *38*, 906–921. [\[CrossRef\]](#)
20. Sigaud, O.; Droniou, A. Towards deep developmental learning. *IEEE Trans. Cogn. Dev. Syst.* **2016**, *8*, 99–114. [\[CrossRef\]](#)
21. Nemec, B.; Abu-Dakka, F.J.; Ridge, B.; Ude, A.; Jørgensen, J.A.; Savarimuthu, T.R.; Jouffroy, J.; Petersen, H.G.; Krüger, N. Transfer of assembly operations to new workpiece poses by adaptation to the desired force profile. In Proceedings of the 2013 16th International Conference on Advanced Robotics (ICAR), Montevideo, Uruguay, 25–29 November 2013.
22. Vuga, R.; Nemec, B.; Ude, A. Speed adaptation for self-improvement of skills learned from user demonstrations. *Robotica* **2016**, *34*, 2806–2822. [\[CrossRef\]](#)
23. Diryag, A.; Mitić, M.; Miljković, Z. Neural networks for prediction of robot failures. *Proc. Inst. Mech. Eng. Part J. Mech. Eng. Sci.* **2013**, *228*, 1444–1458. [\[CrossRef\]](#)
24. Alonso-Tovar, J.; Saha, B.N.; Romero-Hdz, J.; Ortega, D. Bayesian Network Classifier with Efficient Statistical Time-Series Features for the Classification of Robot Execution Failures. *Int. J. Comput. Sci. Eng.* **2016**, *3*, 80–89. [\[CrossRef\]](#)
25. Bicchi, A.; Salisbury, J.K.; Brock, D.L. Contact sensing from force measurements. *Int. J. Robot. Res.* **1993**, *12*, 249–262. [\[CrossRef\]](#)
26. Kitagaki, K.; Ogasawara, T.; Suehiro, T. Contact state detection by force sensing for assembly tasks. In Proceedings of the 1994 IEEE International Conference on MFI'94, Multisensor Fusion and Integration for Intelligent Systems, Las Vegas, NV, USA, 2–5 October 1994; pp. 366–370.
27. Liu, S.; Li, Y.F.; Xing, D. Sensing and control for simultaneous precision peg-in-hole assembly of multiple objects. *IEEE Trans. Autom. Sci. Eng.* **2019**, *17*, 310–324. [\[CrossRef\]](#)
28. JSI; IDIAP; CERTH. *D4.7—Assembly Policy Learning and Improvement*; Deliverable, CoLLaboratE Project; CoLLaboratE Project Consortium: Thessaloniki, Greece, 2020.
29. Sachtler, A.; Nottensteiner, K.; Kassecker, M.; Albu-Schäffer, A. Combined Visual and Touch-based Sensing for the Autonomous Registration of Objects with Circular Features. In Proceedings of the 19th International Conference on Advanced Robotics (ICAR), Belo Horizonte, Brazil, 2–6 December 2019; pp. 426–433.
30. Nashman, M.; Nashman, M.; Hong, T.H.; Herman, M. *An Integrated Vision Touch-Probe System for Dimensional Inspection Tasks*; US Department of Commerce, National Institute of Standards and Technology: Gaithersburg, MD, USA, 1995.
31. Allen, P.; Bajcsy, R. Robotic Object Recognition Using Vision and Touch. In Proceedings of the 9th International Joint Conference on Artificial Intelligence, Milan, Italy, 23–28 August 1987.
32. Thomas, U.; Molkenstruck, S.; Iser, R.; Wahl, F.M. Multi Sensor Fusion in Robot Assembly Using Particle Filters. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Roma, Italy, 10–14 April 2007; pp. 3837–3843. [\[CrossRef\]](#)
33. Bayoudh, K.; Knani, R.; Hamdaoui, F.; Mtibaa, A. A survey on deep multimodal learning for computer vision: Advances, trends, applications, and datasets. *Vis. Comput.* **2021**, *38*, 2939–2970. [\[CrossRef\]](#) [\[PubMed\]](#)
34. Boulahia, S.Y.; Amamra, A.; Madi, M.R.; Daikh, S. Early, intermediate and late fusion strategies for robust deep learning-based multimodal action recognition. *Mach. Vis. Appl.* **2021**, *32*, 121. [\[CrossRef\]](#)
35. Lahat, D.; Adali, T.; Jutten, C. Multimodal Data Fusion: An Overview of Methods, Challenges, and Prospects. *Proc. IEEE* **2015**, *103*, 1449–1477. [\[CrossRef\]](#)
36. Dong, X.; Yu, Z.; Cao, W.; Shi, Y.; Ma, Q. A survey on ensemble learning. *Front. Comput. Sci.* **2020**, *14*, 241–258. [\[CrossRef\]](#)
37. Ren, Y.; Zhang, L.; Suganthan, P. Ensemble Classification and Regression—Recent Developments, Applications and Future Directions. *IEEE Comput. Intell. Mag.* **2016**, *11*, 41–53. [\[CrossRef\]](#)
38. Džeroski, S.; Panov, P.; Ženko, B. Ensemble Methods in Machine Learning. In *Encyclopedia of Complexity and Systems Science*; Meyers, R.A., Ed.; Springer: New York, NY, USA, 2009; pp. 5317–5325. [\[CrossRef\]](#)
39. Levatić, J.; Kocev, D.; Debeljak, M.; Džeroski, S. Community structure models are improved by exploiting taxonomic rank with predictive clustering trees. *Ecol. Model.* **2015**, *306*, 294–304. [\[CrossRef\]](#)
40. Levatić, J.; Ceci, M.; Kocev, D.; Džeroski, S. Semi-supervised Predictive Clustering Trees for (Hierarchical) Multi-label Classification. *arXiv* **2022**, arXiv:2207.09237. [\[CrossRef\]](#)

41. Vens, C.; Struyf, J.; Schietgat, L.; Džeroski, S.; Blockeel, H. Decision trees for hierarchical multi-label classification. *Mach. Learn.* **2008**, *73*, 185–214. [[CrossRef](#)]
42. Collins, K.; Palmer, A.; Rathmill, K. The development of a European benchmark for the comparison of assembly robot programming systems. In *Robot Technology and Applications*; Springer: Berlin/Heidelberg, Germany, 1985; pp. 187–199.
43. Nemec, B.; Mavsar, M.; Simonič, M.; Hrovat, M.M.; Škrabar, J.; Ude, A. Integration of a reconfigurable robotic workcell for assembly operations in automotive industry. In Proceedings of the IEEE/SICE International Symposium on System Integration (SII), Virtual, 9–12 January 2022; pp. 778–783. [[CrossRef](#)]
44. Radanovič, P.; Jereb, J.; Kovač, I.; Ude, A. Design of a Modular Robotic Workcell Platform Enabled by Plug & Produce Connectors. In Proceedings of the 20th International Conference on Advanced Robotics (ICAR), Ljubljana, Slovenia, 6–10 December 2021.
45. Simonič, M.; Pahič, R.; Gašpar, T.; Abdolshah, S.; Haddadin, S.; Catalano, M.G.; Wörgötter, F.; Ude, A. Modular ROS-based software architecture for reconfigurable, Industry 4.0 compatible robotic workcells. In Proceedings of the 20th International Conference on Advanced Robotics (ICAR), Ljubljana, Slovenia, 6–10 December 2021.
46. Albu-Schaffer, A.; Ott, C.; Hirzinger, G. A passivity based Cartesian impedance controller for flexible joint robots—Part II: Full state feedback, impedance design and experiments. In Proceedings of the IEEE International Conference on Robotics and Automation, (ICRA), New Orleans, LA, USA, 26 April–1 May 2004; Volume 3, pp. 2666–2672.
47. Jiang, J.; Huang, Z.; Bi, Z.; Ma, X.; Yu, G. State-of-the-Art control strategies for robotic PiH assembly. *Robot. Comput.-Integr. Manuf.* **2020**, *65*, 101894. [[CrossRef](#)]
48. Nemec, B.; Yasuda, K.; Ude, A. A virtual mechanism approach for exploiting functional redundancy in finishing operations. *IEEE Trans. Autom. Sci. Eng.* **2020**, *18*, 2048–2060. [[CrossRef](#)]
49. O'Mahony, N.; Campbell, S.; Carvalho, A.; Harapanahalli, S.; Hernandez, G.V.; Krpalkova, L.; Riordan, D.; Walsh, J. *Deep Learning vs. Traditional Computer Vision*; Advances in Computer Vision; Arai, K., Kapoor, S., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 128–144.
50. Lin, T.Y.; Goyal, P.; Girshick, R.B.; He, K.; Dollár, P. Focal Loss for Dense Object Detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2999–3007.
51. Lin, T.Y.; Dollár, P.; Girshick, R.B.; He, K.; Hariharan, B.; Belongie, S.J. Feature Pyramid Networks for Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 936–944.
52. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
53. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R.B. Mask R-CNN. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2980–2988.
54. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [[CrossRef](#)]
55. Simonič, M.; Žlajpah, L.; Ude, A.; Nemec, B. Autonomous Learning of Assembly Tasks from the Corresponding Disassembly Tasks. In Proceedings of the IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids), Toronto, ON, Canada, 15–17 October 2019; pp. 230–236.