Journal of Intelligent & Robotic Systems (2020) 100:863–877
https://doi.org/10.1007/s10846-020-01248-3

A Novel Strategy for Automatic Error Classification and Error Recovery for Robotic Assembly in Flexible Production

Ewa Kristiansen[1] · Emil Krabbe Nielsen[2] · Lasse Hansen[3] · David Bourne[4]

Received: 12 December 2019 / Accepted: 13 August 2020 / Published online: 18 September 2020

Abstract

In this article, we develop a novel strategy for automatic error classification and recovery in robotic assembly tasks. The strategy does not require error diagnosis. It allows for effective reduction of an undetermined number of error states to 4, without the need for further operator updates of error space. The strategy integrates existing methods for computer vision, active vision and active manipulation. Our solution is implemented in a generic software framework, which is independent from software and hardware for implementing error detection and allows for application in other assembly types and components. The value of our strategy was experimentally validated on a simple case, where we inserted a battery into a cell phone. The experiment was performed on 1500 assembly attempts and included 500 detected errors. The whole experiment ran for 42 hours, with no need for operator assistance or supervision. The resulting classification rate is 99.6% and the resulting recovery rate is 98.8%. The 6 unrecovered errors were successfully resolved in a successive assembly attempt.

Keywords Automatic error classification · Automatic error recovery · Robotic assembly · Flexible production · Semi-structured environment · Active vision

1 Introduction

Recently, companies must cope with short product lifecycles, which are caused by rapidly changing technologies and intense competition. When automating manufacturing processes companies can invest in dedicated, hard automation. Unfortunately, these production lines must be reconfigured for a new product variant, which is both time consuming and expensive. Alternatively, companies can use robots and programmable equipment to eliminate drastic changes to an existing manufacturing facility, with the changes being addressed by software adjustments. Increasing the flexibility of industrial robotic setups is an important goal for future production systems [1, 2] to make low volume production cost effective.

Flexible robotic assembly is difficult to automate because the variations between different batches and the uncertainties and compliance introduced by flexible fixtures and tools, make the conventional position-based motion planning unreliable. The existing force/torque based strategies offer the potential solution. A number of scientific articles addresses this issue in the context of flexible assembly. In [3–5], the authors develop the real-time force controller to guide the robotic assembly. In [6] contact forces are used for minimizing assembly time by empirical self-tuning of parameters by the robot and using a learning algorithm. In [7] the parameters are tuned with machine learning using success/failure matrices as an objective function. However, automated flexible assembly based on force based strategies requires complex models for part interactions and/or they need to be refined over many runs with machine learning before they are applied for controlling the assembly process.

In highly flexible assemblies, the uncertainties in the process decrease the success rate of the assembly. The

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s10846-020-01248-3) contains supplementary material, which is available to authorized users.

✉ Ewa Kristiansen
ewa@m-tech.aau.dk

[1] Department of Materials and Production, Aalborg University, Fibigerstræde 16, 9220, Aalborg Ø, Denmark

[2] Department of Electrical Engineering, Technical University of Denmark, Elektrovej 326, 2800, Kgs. Lyngby, Denmark

[3] Nel Hydrogen, Fueling and Solutions, Vejlevej 5, 7400 Herning, Denmark

[4] Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA

implementation of error prevention can be expensive and complex. This makes error detection and correction a central feature for increasing the reliability of the manufacturing process [8]. In [9] and [10] the authors present a general framework for an error handling process, which is consisting of; assembly monitoring, error diagnosis and error recovery. The flow chart in Fig. 1 is adapted from the framework. It consists of: error detection (returning a boolean value), error classification (returning an error type), error diagnosis (returning the cause of the specific error type) and error recovery (executing the recovery strategy, which is matching the specific error cause). This flow chart can be used for both discrete and continuous error handling and can be applied to any of the final stages of assembly: approach, contact, force imposition and completion.

Multiple research projects have documented subproblems found in error handling systems shown in Fig. 1. Continuous monitoring of an assembly process with force-torque sensors is studied in [11]. The measurements and piecewise affine model of a mating process are used for detecting early errors and applying different searching strategies to guide the robot motions and eventually recover from errors. In [12] the values of signals from force and visual servoing are related to soft and hard constraints. Soft constraints are favourable conditions for assembly: the data from the monitoring sensors are fed to the reactive controller, which is implemented to minimize deviation from the constraint boundary and recover from some early errors. Hard constraints are necessary conditions for successful assembly and refer to the completion phase: if the data from monitoring sensors violates hard constraints, the assembly terminates as unsuccessful. Hard constraints are not included in research [11] and [12].

Research presented in [8] and [7] is dealing with error detection after the completion phase by using learning algorithms and principal component analysis for force signatures to distinguish between failed and successful assemblies. Both methods are model free, but error classification, diagnosis and recovery are not handled. Other research [13] extends these methods with the classification of error states and [14] further contributes to the state of the art by relating the classified error states to different stages of assembly. However, both papers leave the error recovery to future research.

The most popular approaches for error diagnosis of a discrete event system are finite state automaton and petri nets, relating every fault to a unique data feature from the sensor signals. As the transitions between the states can be unobservable, the data features need to be classified through some classification algorithms. In [15] a Mamdani model is used as a rule implication in the fuzzy Petri nets and the authors report 93% accuracy in diagnosing five hardware induced failure root causes in a dual robot assembly process. In [16] the authors use a probability implication in the Bayesian Network where the diagnosis of various fixture failures is based on sensor measurement data during an assembly process. The work evaluates the optimal sensor placement in the fixture and the node probabilities of the Bayesian Network are obtained using Finite Element Analysis to perform faulty assembly scenarios. The work is evaluated on an automotive part showing it is efficient and feasible. An alternative method for representing system states is presented in [17]. The proposed framework is based on an interval analysis approach and in particular it is implementing max-plus algebra. The considered faults are associated with the discrepancy between the expected and monitored time used for transportation and for processing steps in assembly. The essence of this approach is also its disadvantage; namely depending on the type of error the unsuccessful assemblies can take exactly the same time to perform as the successful assemblies.

The complete solution for error handling in flexible assembly, as shown in Fig. 1 is presented in [9]. However, it requires an interaction with a human operator in initial training phase to create an extensive knowledge database of error types and associated recovery strategies. It is also a problem that the new error types in the execution of assembly need to be recovered manually and the knowledge

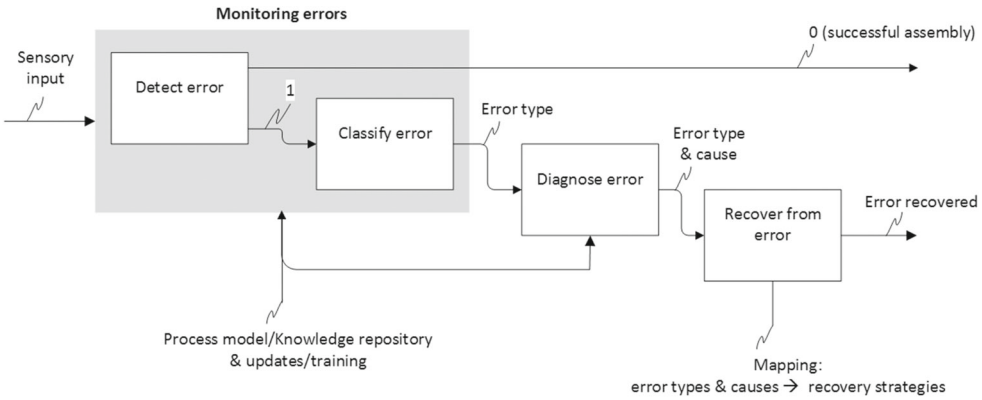

Fig. 1 Flow chart of an error handling process

database needs to be updated by a trained operator. In this way, the system history and the model of errors is continuously expanded, though contributing to a complex robot control program. According to [18], 90% of the system coding is devoted in automatic error recovery.

An interesting attempt to ease programming robots is proposed in [19]. The presented error recovery approach does not require error classification, error diagnosis or even explicitly written error handlers. Instead, it is using a reverse execution approach after an error is detected. A number of steps in the robot program are backtracked and then the assembly operation is repeated, allowing the probabilistic uncertainties to fix themselves. Backtracking is feasible only for some robot instructions and therefore this approach is also relying on the operator, here classifying the instructions as directly reversible, reversible with minor/major modifications or not reversible. Moreover, all the instructions need to be paired (for example 'open gripper' with 'close gripper') and for some operations the instructions cannot be reversed and their sequence needs to be overridden by the user. The reverse execution approach does not consider error propagation, giving tight requirements that the occurrence and detection of error happen during the execution of the same robot instruction. It was not quantified whether this approach is easier to program and integrate with error detection system compared to approach with error associated recovery strategy.

The literature review does not reveal a solution for error recovery in flexible assembly (especially for errors in the completion phase) that does not require complex process modelling and/or fault tolerant control and/or knowledge repository and/or human supervision for updates and handling new errors. The research presented in this paper is solving this research gap. Inspired by the definition of error recovery from [19] as bringing the assembly part to the known initial location, we are allowing the probabilistic uncertainties to fix themselves in the successive assembly trials. Our novel strategy for error classification and recovery is intended for the completion phase of flexible assembly and solves the problems introduced by an undetermined number of error states; it allows for effective reduction of number of error states to 4 with the corresponding 3 recovery strategies. With the fully defined model of errors, the robot control program is kept simple and without a need for further operator updates.

There are two main contributions of this paper. The first one is the development of a novel strategy for error recovery from unsuccessful assemblies (also referred to as errors in completion phase), which are imposed by semi-structured environment of flexible assembly. The second contribution is an implementation of this strategy in a generalized software framework for broad applicability to different assembly tasks in flexible production.

The errors handled in this paper originate from two sources: flexible design of fixtures with preloaded springs and compliance introduced to the robot tool when choosing a suction based gripper. The first error source causes position uncertainties of components prior and during the assembly process and results in an undesired pick-up location. Whereas the use of suction cups can result in no grasp, as well as object slips because of inertia, varying weight of objects or collision of parts. The higher the level of flexibility, the more unstructured environment and the bigger number of unsuccessful assemblies. We limit our work to handling errors that are due to hardware and system faults and execution failures, such as collisions and missing assembly components.

The value of our strategy was demonstrated by experimental validation of a robotic insertion of a battery into a Nextel Blackberry 7520 cell phone and without any human supervision. The conclusions are based on an extensive experimentation with 1500 assembly attempts and including 500 detected errors. The results show a high rate of classified errors and recovered errors. The developed framework also demonstrate an ability to self-recover from misclassified errors and unrecovered errors, which are due to the errors occurring during recovery activities themselves, such as battery slips and no grasp.

The rest of this paper is organised in the following way. Section 2 presents the developed software framework. Section 3 describes the strategy for error classification. Section 4 presents a method for image processing. Section 5 discusses error recovery. Section 6 describes the experimental setup and Section 7 presents the error detection system used for generating an input to the developed software framework. The experimental results and their analysis are presented in Section 8 and the key points are summarised in Section 9.

2 Software Framework for Error Classification and Recovery

The developed software framework relies on the redefined flow chart for automatic error handling, as shown in Fig. 2, which consists of only 3 subproblems, as opposed to 4 subproblems in the general framework shown in Fig. 1. The excluded subproblem is error diagnosis, as our system does not require any information about the cause of errors. The developed framework includes only: error detection, classifier of error states and error recovery.

The role of an error detection system is to distinguish between a successful and failed assembly. In case of a failure the error classification and error recovery systems are executed in the loop two times and the recovery is completed. The error classification system is applying

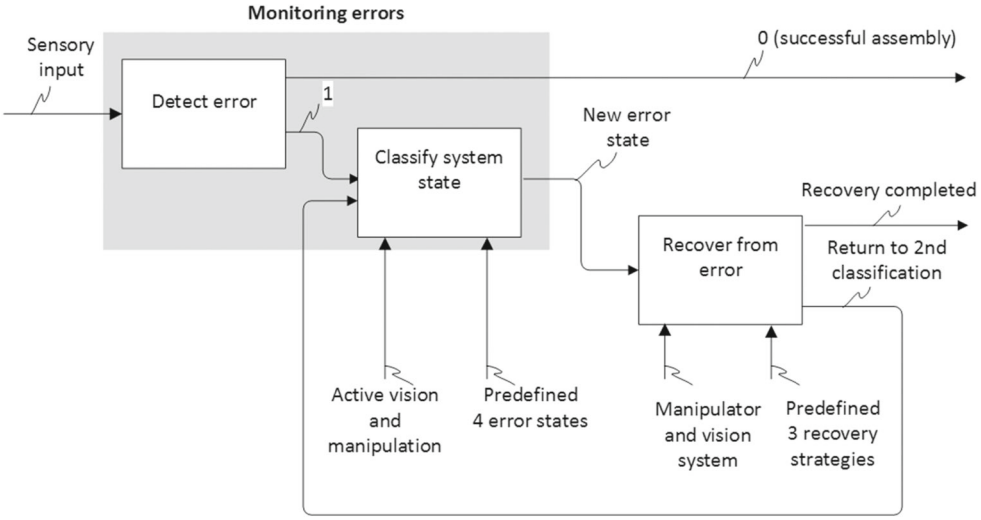

Fig. 2 Flow chart of a modified error handling process

active vision and manipulation to identify 1 out of 4 error states. The error state is then mapped to 1 out of 3 predefined recovery strategies.

The details of the developed software framework for error classification and error recovery, together with the interfaces between these two subsystems are shown in Fig. 3.

The two subsystems are tightly integrated with each other, in a way that error classification and error recovery are called two times. This software framework is the key for a very efficient reduction of number of possible error states and therefore also the number of recovery strategies. Detailed explanation can be found in the following three sections.

A coordination of hardware and software operation is provided by a high level controller implemented in the Robot Operating System (ROS). ROS is used as an operating system, and ROS packages are written to collect and process data from all sensors, communicate with the robot controller and to handle robot commands and interaction with a human operator.

3 Strategy for Error Classification

The system for classifying error states is activated by an error detection system, whenever the unsuccessful assembly is detected. The error detection system is described in Section 7.

The developed classification system determines the error state of an unsuccessful assembly. In this work we consider three sources of errors: position uncertainties, compliance of the tool and compliance of the fixture. Additionally the compliance of the tool can cause no grasp, misgrasp or undesired drop of the assembly object. These can bring the object to undetermined number of error states, which are representing here the object's undesired locations outside the ground body. In this work we suggest an approach how to convert this undetermined number of error states into only 4 error states and this way limit the search space of the classifier. The approach is generic and can be used for other assembly components and assembly types.

After an unsuccessful assembly is detected the object is relocated from the predefined pick up point to the predefined position in the pick up bin and also reoriented so that the reference frame of the assembly object, $\{B\}$, is aligned to the reference frame of the robot, $\{R\}$, as exemplified in Fig. 4. Such manipulation of the object makes the robot move out of the frame for the range camera and it makes error states not to be related to any primary position and orientation of the object when the unsuccessful assembly is detected, but instead it is linked only with a new orientation of the object in a known position.

The robot tool can unintentionally drop the assembly object on its way to the pick up bin. The assembly object can also fall down on the assembly platform or outside the assembly platform during the assembly process and therefore the assembly object can not be picked up from the pre-defined position, located in the center of the intended mating. Regardless of the result of *'Relocate and reorient object'* the succeeding function *'Localize approx. center of object'* from Fig. 3, can progress because the assembly object is either laying in the pick up bin or outside the ground body, which gives the proper background for image processing, described in Section 4.

Alignment of the frames brings the assembly object to one of four possible orientations, referred to as error states and explained on an example in Fig. 5; Not flipped - Not turned, *NF-NT*, Not flipped-turned, *NF-T*, Flipped-Not turned, *F-NT* and Flipped-Turned, *F-T*. The error states can relatively easily be distinguished by the presence of at least two unique features on the assembly object, facing in

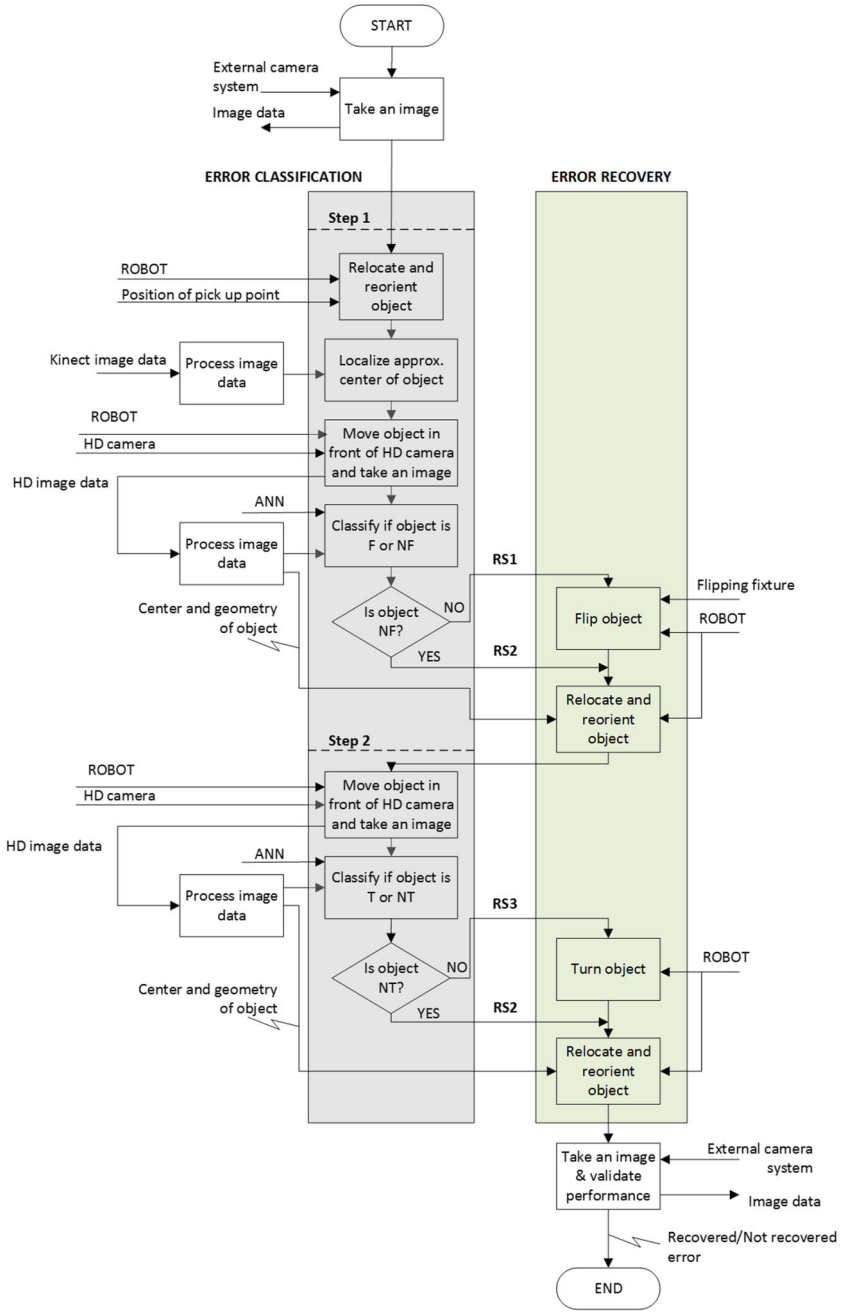

Fig. 3 The generalised software framework for automatic error classification and error recovery. RS1, RS2 and RS3 denote different recovery strategies

two predefined directions. System in state *NF-NT* has one feature, which is a sticker shown in Fig. 5, pointing upwards and the other feature, which is a connector in Fig. 5, on the left hand side.

These 4 error states define the whole search space for the classification system and represent any undesired location of the assembly object outside a ground body and within the work range of the range camera. Such a big reduction in number of error states is though at the expense of diagnosing the cause of the failure in the assembly. It is therefore not possible to conclude and learn from the unsuccessful assemblies, whether they are due to compliance of the tool, compliance of the fixture or positional accuracy of the assembly object in the bin.

Error states are implemented using the error state tree model, presented in Fig. 6. Error state tree model consists of two levels. On the first level it is distinguished whether the assembly object is flipped or not flipped and on the second level the additional information is added to the error state description, about the assembly object being turned or not turned. The two levels in the error state tree represent two steps in the classification procedure, as shown in Fig. 3.

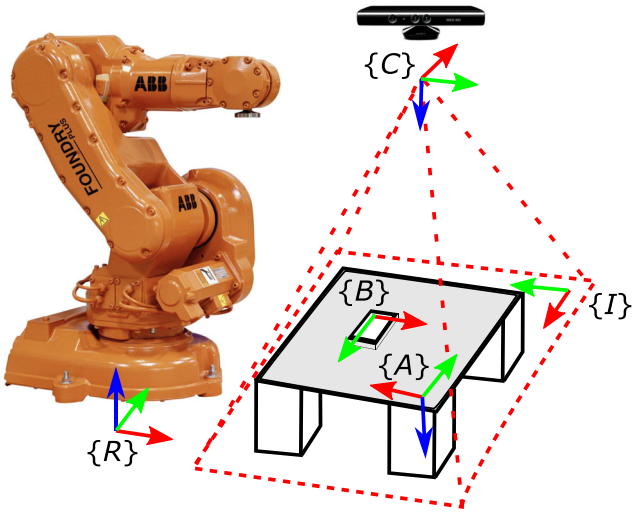

Fig. 4 Coordinate frames: robot {*R*}, range camera {*C*}, image {*I*}, platform {*A*} and assembly object {*B*}

Every step is proceeded with error recovery strategy for robotic manipulation of the assembly object, which includes relocation and reorientation.

Error classification uses processed image data, which is provided by two vision sensors; range camera and HD camera. Range camera is in charge of detecting the pick-up location of the assembly object. Data from the HD camera has a higher pixel resolution and is used for classification of error states, finding the center and geometry of the assembly object. Additionally in connection to the HD camera we use active vision, which is a robotic manipulation of the assembly object to the predefined vicinity of the camera, in order to avoid any issues related to the limited range of the camera as well as disruption of frames by other objects in the robotic cell. Active vision is represented in Fig. 3 by the function *'Move object in front of HD camera'*. In the first classification step the assembly object is moved in front of the HD camera such that a surfaces with a unique feature points towards the camera. In the second classification step another surface with a unique feature points towards the camera. Both orientations are exemplified in Fig. 7.

Since the outcome of the first classification step is not influenced by the specific location of the suction cup on

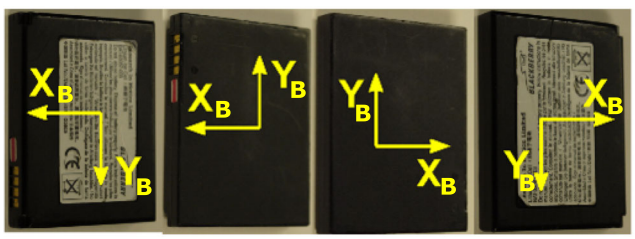

Fig. 5 The 4 error states exemplified on the orientation of the battery's coordinate frame {*B*}; NT-NT, F-NT, F-T, F-T

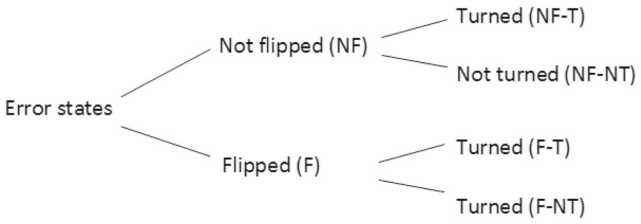

Fig. 6 State tree with 4 error states

the assembly object, the low pixel resolution data from the range camera is sufficient for localization of approximate center of the assembly object.

4 Image Processing

The image processing method presented in this section should be considered as a secondary contribution of the paper, which is though a necessary mean for implementation and testing of the developed system for error classification and error recovery.

The function *'Process image data'* from Fig. 3 implements a computer vision (CV) routine, which is inspired by [20] and [21]. The CV routine consists of the following steps; subtract the background image, add an adaptive threshold to the image, erode and dilate image, and perform blob analysis. Figure 8 shows images of CV routine, which are created from case data provided by the Kinect camera.

On the final image from this example it can be seen that when using the Kinect data it is difficult to specify the geometry of the assembly object, and in the case example from Fig. 8 to distinguish between the long and short side of the battery. This information is necessary for making reorientation in error recovery. For this purpose we process data from the HD camera, by fitting a rectangle to the battery's blob with the least square optimisation method. For the sake of simplicity of solution an Aritificial Neural Network (ANN) is used as a diagnostic tool, instead of Deep Neutral Network.

As shown on the right hand side of Fig. 7, the close-up image of the long side of the battery is taken by the HD camera. The raw image is prepocessed. First a region of interest containing connectors of the battery is chosen to achieve a higher classification accuracy, as only few training examples are used, considering that ANNs are commonly trained on thousands of samples. Next, a threshold is applied to convert the color image to a binary image in black and white.

An input layer to ANN is a vector with 2.760 binary nodes, corresponding to pixels of the binary image. The ANN is a multilayer perceptron (MLP) and has one hidden

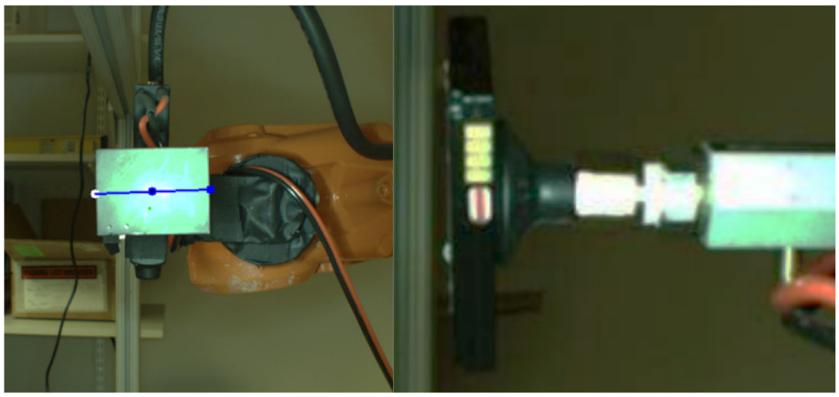

Fig. 7 Orientation of battery in front of HD camera in Step 1 and Step 2 of error classification

layer containing eight nodes and an output layer containing three nodes.

The three nodes in the output layer represent either of the following groups of error states:

– *NF-NT* - the unique feature is located at the top,
– *NF-T* - the unique feature is located at the bottom,
– *F-T* and *F-NT* - the unique feature is not detected.

The ANN is trained on 60 samples and then tested on 100 samples that are not part of the training data. Both the training set and the test set contain close to equal amount of error state groups. Thus for the training set, each error state group is represented by 20 images.

The ANN applied achieves an accuracy of 100% on the 100 test samples. The ANN is tested with a step size of 0.01, a stopping criteria of 10.000 steps or a maximum error of 0.0001. The prediction accuracy of the ANN saturates between 50 and 60 samples and training with more than 60 samples does not increase prediction accuracy of the ANN.

5 Strategy for Error Recovery

The goal of recovery system is to bring the assembly object back to the pick up bin, where both its position and orientation are fully defined. The assembly object is then ready for a new assembly attempt and the error is considered recovered. As shown in Fig. 3, the recovery system is executed two times and each call corresponds to 1 out of 3 predefined recovery strategies, $RS1$, $RS2$ and $RS3$, associated with a specific error state. The recovery strategies along with the software implementation of the error recovery system are generic and can be used for other assembly products and assembly types.

The recovery strategy, $RS1$, is executed when the error state is classified as Flipped, F. The recovery strategy rotates the battery $180°$ around Y_B by robotic manipulation and using a purposely designed flipping fixture to accommodate for the geometry of the battery. The flipping fixture is the only component dependent part of our solution. An example showing the flipping fixture and flipping procedure is shown in Fig. 9. The flipping procedure consists of the following steps; position the battery along one inner wall of the fixture, push it over to the other wall of the fixture and re-grasp the battery from the opposite side. After flipping the assembly object the recovery strategy, $RS2$, is executed.

The recovery strategy, $RS2$, is associated with the error states NF and NT. It relocates and reorients the assembly object, as described in Section 3.

The recovery strategy, $RS3$, is executed when the error state is classified as Turned, T. This recovery strategy rotates the assembly object $180°$ around Z_B. It is done by $180°$ rotation of the robot wrist joint and re-grasping. Lastly, the recovery strategy, $RS2$, is executed again and the error is considered fully recovered.

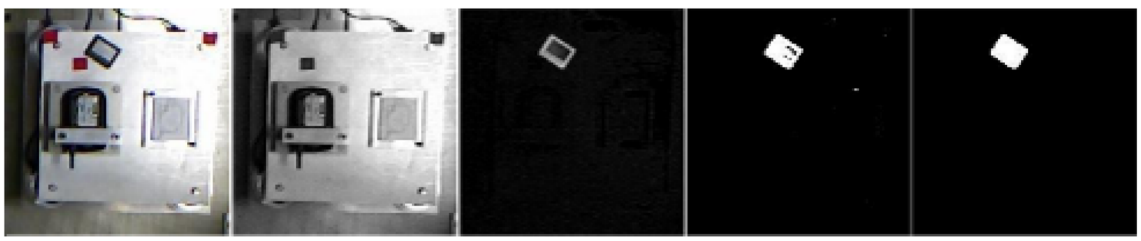

Fig. 8 CV routine: Input image, background image, adaptive threshold image, eroded and dilated image, and final image after blob analysis

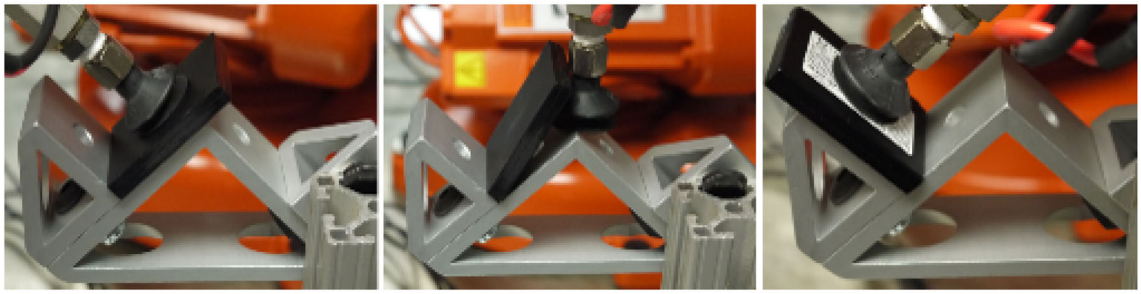

Fig. 9 An example of the fixture and flipping procedure for a battery

6 Experimental Setup

The added value of the developed strategy for automatic error classification and recovery is demonstrated by experimental validation, on the case example of insertion of a battery into a Nextel Blackberry 7520 cell phone. We chose assembly of these two components mainly because of two challenges this assembly implies; one related to the press fit and the other one related to the need of grasping from one side only.

6.1 Hardware Setup

The main hardware setup for physical experiments is shown in Figs. 10 and 11 and consists of:

1. ABB IRB-140 6DOF robot arm
2. Assembly platform with:
   (a) 4 one axis force sensors with a load capacity of 22.7 kg from Loadstar
   (b) 4 preloaded springs
   (c) fixture for a cell phone
   (d) cell phone
   (e) flexible fixture for a battery
   (f) battery
3. Suction cup
4. Point Grey HD camera
5. Microsoft Kinect camera

The assembly platform is constructed to provide mounting of; four force sensors, a fixture holding the cell phone and a flexible fixture for the battery, also referred to as the pick up bin.

The four springs are installed between the fixture for a cell phone and the assembly platform, in order to prevent harmful collisions and to allow for producing different values of force signals by varying the value of preloading.

The cell phone with its fixture and the pick up bin are fixed to the platform, but the battery is not fixed to the pick up bin. In order to anticipate process uncertainties and increase robustness to uncertainties the size of the pick up bin is relatively big to allow for adding a pseudo random noise to the initial, known location of the battery in the pick up bin. Detailed description of the noise used can be found in [7]. Introduction of a pseudo random noise makes it possible to control the nature of the errors, it also makes the experiments repeatable and comparable while emulating the flexible production environment, where the assembly object is delivered to the assembly area with random position and orientation and where the change to another type of assembly object does not necessarily require the change of fixture.

The Kinect camera is placed on top of the robot cell, facing downward. The low pixel resolution of the camera makes it difficult to accurately determine the orientation of the battery. The Kinect camera is thus used in this work for providing the location of the assembly platform, the location of the cell phone and the position of the

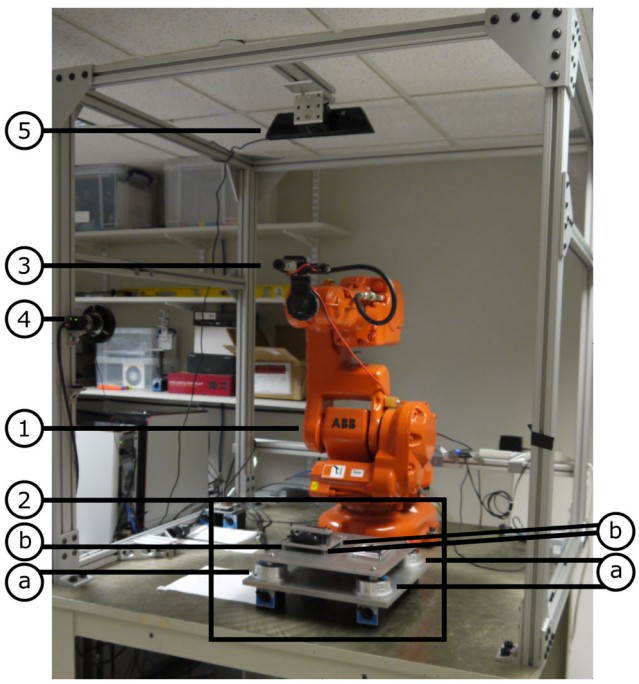

Fig. 10 Hardware setup

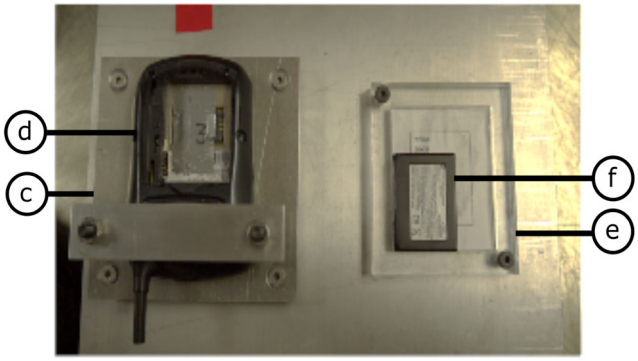

Fig. 11 Top view of the assembly platform

battery. The position of the battery is defined in 3D space, in Kinect's coordinate system $\{C\}$, shown in Fig. 4. The possible ${}^{C}x_B$, ${}^{C}y_B$ coordinates are constrained by the position and the range of the Kinect camera, defining the boundary of the work area for our developed system. In our particular case studies the ${}^{C}z_B$ coordinates are limited to only three possible battery's positions: on the top of the mobile, on the assembly platform or outside the assembly platform. Therefore in this study the ${}^{C}z_B$ coordinates for the battery's position are precoded, depending on its ${}^{C}x_B$, ${}^{C}y_B$ coordinates relative to the assembly platform. If the robot cell or the assembly components should have more complex geometry and should encounter for any ${}^{C}z_B$ coordinates for the battery's position, it might be necessary to upgrade hardware with stereo vision.

The HD camera is attached to the side of the robot cell. Its position is predefined and therefore giving the possibility to implement active vision. The HD camera is used whenever the higher resolution images are required.

The source of error in the presented hardware setup is introduced by:

- low position and orientation tolerances of battery in the pick up bin,
- compliance introduced to the robot tool, by choosing a suction cup gripper,
- compliance due to integration of the four preloaded springs to the design of assembly platform.

The assembly robot is programmed online for a specific pick-up location of the battery and the operator does not take into account the implications from semi-structured environment. The robot program has an influence on the number of successful assemblies. The authors acknowledge the importance of the motion plan for the total performance of the whole assembly operation. Though the success rate of the assembly is not the topic of this article, instead the focus is on the success rate of error classification and error recovery.

6.2 Software Setup

The developed software framework for automatic error classification and error recovery is a stand alone system. Though in order to conduct the validation experiments we implement it in the software framework for the entire flexible robotic assembly process, including autonomous parameter optimisation of fine motions and production, explained in Fig. 12.

First in function *'Calibrate frames'*, the reference frames are calibrated in order to avoid systematic errors. Calibration routine is described in details in the Supplementary Material. After calibration a classifier is trained in function *'Train classifier'* to distinguish between a successful and a failed assembly, based on data from the force signals. Then in function *'Optimize parameters'* the fine motion assembly parameters, such as angle of insertion, are optimised in order to achieve the highest probability for successful assembly with a pseudo random noise and compliance introduced to the fixture and the tool. The assembly is performed in function *'Assemble with pseudo random noise'* and the completed assembly is examined for success in function *'Detect unsuccessful assemblies'*. Detected failures activate the system *'Classify error and recover'*. With the optimised parameters the 'Parameter Optimisation' can terminate and the assembly can be executed in 'Production'. Due to existing chance for a failed assembly in 'Production' this process is also monitored by the system *'Detect unsuccessful assemblies'* and the system *'Classify error and recover'*. For further explanation of the method for classifier training, parameter optimisation and the noise refer to [7].

7 Error Detection

The proposed novel strategy for error classification and error recovery is initialized when the error detection system notifies about an unsuccessful assembly, Fig. 2. The only requirement for the interface is the boolean value. Therefore the strategy gives the flexibility to choose between different methods implementing the error detection. For testing the strategy we chose to use the error detection method presented in details in [7] and summerized in the following two sub-sections.

7.1 Preprocessing

The error detection method is utilizing a classifier. When training a classifier, it is possible to increase its accuracy by preprocessing the training data [22] and hence increasing distinction between the two classes; failed assembly and successful assembly. The raw signal is preprocessed in the following way:

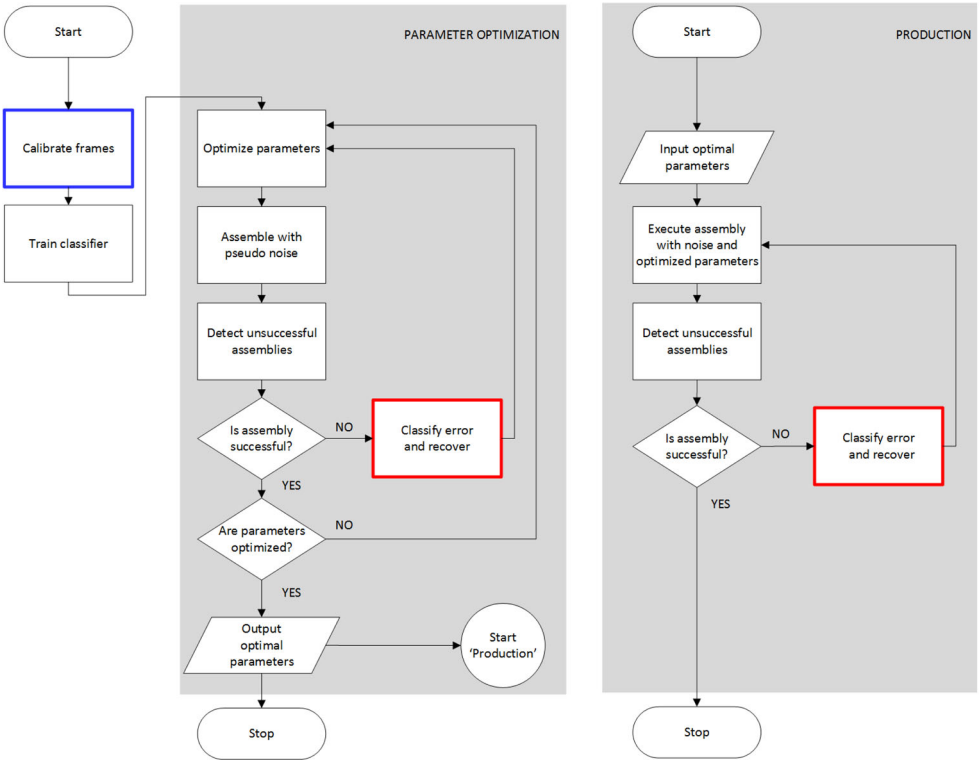

Fig. 12 Software framework for flexible robotic assembly. The function in a red box is a focus of this paper, the function in a blue box is described in the Supplementary Material and the functions in black boxes are presented in [7]

- Resampling: Linear Interpolation
- Feature combination: Root Mean Square
- Feature extraction: Principal Component Analysis
- Feature scaling: Range scaling

The classifier giving inputs to the optimization is a trained Support Vector Machine (SVM) [23]; this type of classifier performs well with little training data compared to many other classifiers [24]. A Radial Basis Function kernel with two parameters is used for the SVM. Grid search is used to find the optimal kernel parameters by using two-fold cross validation to evaluate them [25].

The force signal is resampled by linear interpolation because the sampling frequency of the force sensors is uneven. For each training example we are taking a number of observations, also referred to as features and indexed by $j \in [1; 81]$. For each feature we are measuring the force magnitude in millipounds, denoted as $x_{i,j}$, where i corresponds to the id of a force sensor. A requirement for the input signal to an SVM is that the features must be comparable with regard to occurrence in time across training examples [26].

For each feature, $j \in [1; 81]$, the force magnitudes from four force sensors, $[x_{1,j}, x_{2,j}, x_{3,j}, x_{4,j}]$ are combined into one signal, x_j. The results are shown in Fig. 13 for all 100 training examples and have been calculated using the following equation:

$$\forall j \in [1; 81]: \qquad x_j = \sqrt{x_{1,j}^2 + x_{2,j}^2 + x_{3,j}^2 + x_{4,j}^2} \qquad (1)$$

Principal Component Analysis (PCA) [27] can be used for transforming signals linearly to decrease a high correlation between features and decrease a large feature space dimensionality [8]. This is advantageous since some features can be regarded as noise if there is little or no difference between these features for both classes. The principal component represents the maximum variance in descending order of the transformed dataset. When calculating the principal components of x_j, the first principal component accounts for the maximum variance in all the training examples. All subsequent principal components will account for decreasing variance in decreasing order. The last principal component herby accounts for the feature with the least variance for all training examples.

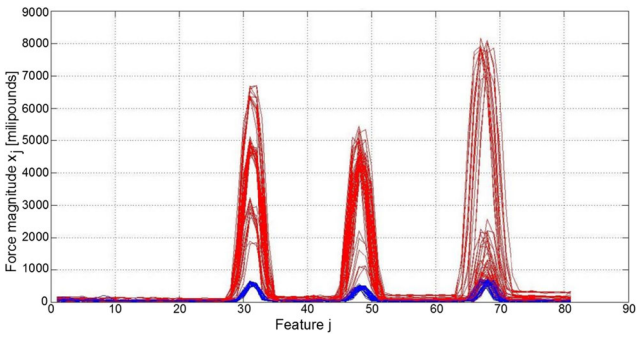

Fig. 13 Plot of 100 signals corresponding to 100 training examples classified by the operator from failed (red) and successful (blue) when tapping [7]

In order to improve the classification accuracy the last and final preprocessing step is to scale the value of the force magnitude, x_j, so that $x_j \in [0, 1]$ [26, 28]. One minimum and one maximum value of magnitude, denoted by min_j and max_j, is found for each j. The minimum and maximum value is found by searching through all training examples for each j, and is used for scaling by the following formula:

$$\forall j \in [1; 81]: \qquad X_{(i,j)} = \frac{x'_{(i,j)} - min_j}{max_j - min_j} \tag{2}$$

7.2 SVM Training

Data collection from four force sensors and involving tapping, as shown in Fig. 14, resulted in two distinct bands describing the successful or failed assemblies. The two bands are marked by different colors in Fig. 13. They are only distinct at three peaks corresponding to tree taps by the robot.

The classifier used for experimentation was trained on 100 training examples and 8 principal components, in order to represent all three taps on each corner of the battery slot, because the battery can be in a configuration where there will be no force exertion on one or two of the battery corners during tapping. This could result in misclassification when using only 1 principal component representing only the tap with force exertion.

8 Experimental Results

In order to validate the performance of developed strategy for error classification and error recovery, an external camera system is installed. The system makes a video of each assembly attempt and also captures two images; one prior and one after error classification and recovery, as shown in Fig. 3. The examples of images taken with the external camera system are shown in Fig. 15.

A total of 1500 assemblies are performed, including 500 assemblies with manually tuned parameters, 500 assemblies made during 'Parameter Optimization' and 500 assemblies made during 'Production, both procedures explained in Fig. 12. The performance of the whole experiment with 1500 assemblies took 42 hours. During this time the system was running automatically, without any need for operator assistance and supervision.

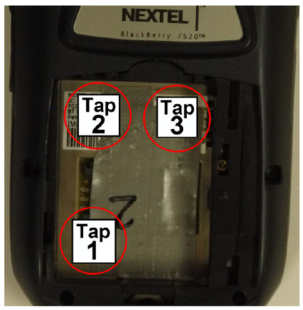

Fig. 14 Data collection strategy using three taps [7]

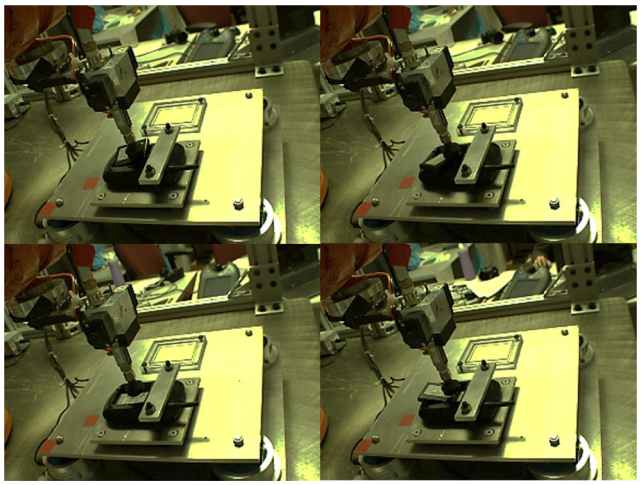

Fig. 15 Examples of unsuccessful assemblies, before execution of error classification and recovery

As the focus is not solely on optimisation of the motion plan, but primarily to test the performance of error classification and recovery, a high number of errors are required. For this reason a large amount of positional noise is added to the battery. As a result, out of 1500 assemblies, 500 assemblies were detected unsuccessful and used to test the performance of error classification and recovery system.

For each of 500 unsuccessful assemblies, two images are manually compared with the classified error states and recovered errors. The results of experiments are summarised in Tables 1 and 2. An example experiment is shown in the video (Online Resource 1).

8.1 Analysis Of Results From Error Classification

It is observed that 498 out of 500 unsuccessful assemblies are classified correctly and only 2 unsuccessful assemblies are misclassified. The total classification rate is therefore 99.6%.

All 496 correctly classified error states are *NF-NT*. The remaining 2 correctly classified error states are *NF-T* and *F-T*.

The 2 unsuccessful assemblies in 'Production' are misclassified as *NF-NT*. They are shown in Fig. 16 and the images are taken prior to classification. In the case shown on the right hand side of Fig. 16, the battery rests on the wall of the battery slot. In the case shown on the left hand side of Fig. 16, the battery is missing.

The incorrect classification as *NF-NT* happens when there is no battery attached to the suction cup while taking

Table 1 Experimental data from all assemblies

	Manually tuned parameters	Parameter optimization	In Production
Assemblies	500	500	500
Detected errors	192	189	119
Undetected errors	0	0	3
Detection rate [%]	100	100	99.4
Classified errors	192	189	117
Misclassified errors	0	0	2
Classification rate [%]	100	100	98.3
Recovered errors	190	187	117
Unrecovered errors	2	2	2
Recovery rate [%]	98.96	98.94	98.32

images with HD camera and it is due to the fact that ANN is not trained on the images with no battery present. The lack of battery on the HD image indicates that either the battery is dropped while being moved by the robot or the suction cup fails to pick up the battery.

The reason for misclassification is therefore the effect of introducing the compliance to the robot tool and the fixture, together with the lack of training samples for ANN representing the case with no battery. This type of misclassification can possibly be avoided by extending the error state tree model, presented in Fig. 6, with additional error state, Not available, *NA*. The new error state, *NA* should be implemented on both levels of error state tree model, in order to cover the whole time period when the battery is manipulated by the robot during error classification.

8.2 Analysis Of Results From Error Recovery

It is observed on the images that 494 out of 500 unsuccessful assemblies are recovered, giving the total recovery rate of 98.8%.

The assembly from the correctly classified error state *NF-T* is not recovered, due to unsuccessful pick up of the battery after execution of recovery strategy. There are also 5 unrecovered assemblies from the error state *NF-NT*. The 3 of those unrecovered assemblies are classified correctly, but were not placed back in the pick up bin because the suction cup dropped the battery during the robotic manipulation. The remaining 2 unrecovered assemblies are the ones being misclassified.

Therefore rather than measuring the quality of the developed system for error recovery, the results are proned by the misperformance of error classification system and disadvantage of introducing the compliance to the robotic tool and fixture. The recovery rate could possibly be improved by hardware implementation of sensors, which are giving feedback signal related to the presence of the battery on the suction cup.

Despite the unrecovered assemblies, the system is capable of running without an operator assistance. In the successive assembly attempt the function *Localize approx. center of object*, shown in Fig. 3, can find the battery unintentionally dropped by the suction sup in the previous assembly attempt, as long as it is within the work range of Kinect camera and outside the mobile's slot.

The achieved recovery rate of 98.8%, which is using discrete error detection and error recovery post assembly, exceeds the results in [19], reporting the recovery rate of 82% from 22 errors when using continuous detection and continuous recovery with reverse execution. Similar to the

Table 2 Detailed data from unsuccessful assembly experiments

	NF-NT	NF-T	F-T	Total	Rate [%]
Unsuccessful assemblies	498	1	1	500	
Classified errors	496	1	1	498	99.6
Misclassified errors	2	0	0	2	
Recovered errors	493	0	1	494	98.8
Unrecovered errors	5	1	0	6	

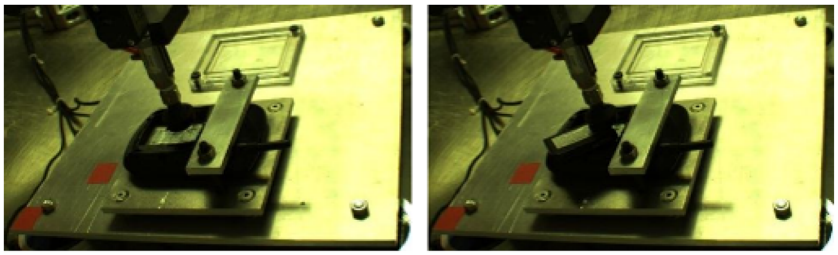

Fig. 16 Unsuccessful assemblies misclassified as *NF-NT*

study proposed in this paper, [11] introduces positional uncertainty to the assembly process. Recovery from the assembly errors is determined from the estimated contact forces and related to the pre-defined motion patterns. Four different strategies are evaluated, and a recovery rate of 94–98% is achieved.

A direct comparison of the results is, however, not possible as the shared database of errors does not exist, the types of errors differ and they occur under different conditions. All three recovery approaches are, however, flexible and require minor efforts in comparison to [9] and the methods reviewed in [10].

9 Conclusions

The novel strategy for automatic error classification and error recovery was developed and implemented to suit the challenges of robotic operation in the flexible assembly cell and arising due to low positional tolerances of parts and compliance in fixtures and robot tools. The designed software framework gave the possibility to reduce the number of error states to only 4, without the need for operator updates. It was only possible by introducing the unique relation of errors to the orientation of the assembly object after it has been manipulated by the robot when using active vision.

The proposed solution for error handling was tested on 1500 assembly attempts, whereas 500 were detected as unsuccessful and used as an input to error classification and error recovery system. The whole experiment was running for 42 hours, with no need for operator assistance or supervision. The resulting diagnosis rate is 99.6% and the resulting recovery rate is 98.8%. The 6 unrecovered errors were succesfully and automatically recovered from in the successive assembly attempt.

It is possible to further improve the performance rate of strategy for automatic error classification and error recovery, by software and hardware implementation, which allows for catching the exception when the battery is dropped by the robot tool and prevents proper functioning of the proposed error handling system. This can be done by extending the number of ANN training samples with the ones showing no battery to the HD camera, by introducing an extra error state, *NA*, in the error state tree model and/or by implementing a sensory feedback for detection of battery on the tool.

The future work can focus on optimisation of time for error classification and error recoveries, as current static image capture is inefficient time and space wise. The future research can investigate to which degree the presented strategy for error classification and recovery can be used for assembling other products and for handling other error types. It would also be interesting to study a hybrid system for error classification and recovery, which is addressing both the probabilistic uncertainties, using the strategy presented in this paper, and the exogenous errors, using knowledge database.

Acknowledgements We would like to thank Nishant Kelkar for his contributions to methodology and software. The authors would also like to thank Alberto Rodriguez and Robert Paolini for their help.

References

1. SPARC, euRobotics AISBL: Multi-annual roadmap - for robotics in europe (2016)
2. Robotics VO: A roadmap for u.s. robotics from internet to robotics (2016)
3. Vaaler, E.G., Seering, W.P.: A machine learning algorithm for automated assembly. In: Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on, pp. 2231–2237 (1991). https://doi.org/10.1109/ROBOT.1991.131962. IEEE
4. Newman, W.S., Zhao, Y., Pao, Y.-H.: Interpretation of force and moment signals for compliant peg-in-hole assembly. In: ICRA, pp. 571–576 (2001). https://doi.org/10.1109/ROBOT.2001.932611
5. Jörg, S., Langwald, J., Stelter, J., Hirzinger, G., Natale, C.: Flexible robot-assembly using a multi sensory approach. In: Robotics and Automation (ICRA), 2000 IEEE International Conference

on, pp. 3687–3694 (2000). https://doi.org/10.1109/ROBOT.2000.845306. IEEE
6. Marvel, J.A., Newman, W.S., Gravel, D.P., Zhang, G., Wang, J., Fuhlbrigge, T.: Automated learning for parameter optimization of robotic assembly tasks utilizing genetic algorithms. In: Robotics and Biomimetics, 2008. ROBIO 2008. IEEE International Conference on, pp. 179–184 (2009). https://doi.org/10.1109/ROBIO.2009.4913000. IEEE
7. Krabbe, E., Kristiansen, E., Hansen, L., Bourne, D.: Autonomous optimization of fine motions for robotic assembly. In: Robotics and Automation (ICRA), 2014 IEEE International Conference on, pp. 4168–4175 (2014). https://doi.org/10.1109/ICRA.2014.6907465. IEEE
8. Rodriguez, A., Bourne, D., Mason, M., Rossano, F.G., Wang, J.: Failure detection in assembly: Force signature analysis. In: Automation Science and Engineering, 2010 IEEE Conference on, pp. 210–215 (2010). https://doi.org/10.1109/COASE.2010.5584452. IEEE
9. Camarinha-Matos, L.M., Lopes, L.S., Barata, J.: Integration and learning in supervision of flexible assembly systems. IEEE Trans. Robot. Autom. **12**(2), 202–219 (1996). https://doi.org/10.1109/70.488941
10. Loborg, P.: Error recovery in automation - an overview. In: AAAI Technical Report SS-94-04, pp. 94–100 (2001)
11. Chen, F., Cannella, F., Huang, J., Sasaki, H., Fukuda, T.: A study on error recovery search strategies of electronic connector mating for robotic fault-tolerant assembly. J. Intell. Robot. Syst. pp. 257–271 (2015). https://doi.org/10.1007/s10846-015-0248-5
12. Hamner, B., Koterba, S., Shi, J., Simmons, R., Singh, S.: An autonomous mobile manipulator for assembly tasks. Auton. Robot. **28**(1), 131 (2010). https://doi.org/10.1007/s10514-009-9142-y
13. Hayami, Y., Shi, P., Ramirez-Alpizar, I.G., Harada, K.: Multi-dimensional error identification during robotic snap assembly. In: Advances in Mechanism and Machine Science, pp. 2189–2198 (2019). https://doi.org/10.1007/978–3–030–20131–9_217
14. Aronson, R.M., Bhatia, A., Jia, Z., Guillane-Bert, M., Bourne, D., Dubrawski, A., Mason, M.T.: Data-driven classification of screwdriving operations. In: Springer Proceedings in Advanced Robotics, pp. 244–253 (2016). https://doi.org/10.1007/978–3–319–50115–4_22
15. Wu, Z., Hsieh, S.-J.: A realtime fuzzy petri net diagnoser for detecting progressive faults in plc based discrete manufacturing system. Int. J. Adv. Manuf. Technol. **61**(1-4), 405–421 (2012). https://doi.org/10.1007/s00170-011-3689-4
16. Liu, Y., Jin, S., Lin, Z., Zheng, C., Yu, K.: Optimal sensor placement for fixture fault diagnosis using bayesian network. Assem. Autom. **31**(2), 176–181 (2011). https://doi.org/10.1108/01445151111117764
17. Majdzik, P., Akielaszek-Witczak, A., Seybold, L., Stetter, R., Mrugalska, B.: A fault-tolerant approach to the control of a battery assembly system. Control. Eng. Pract. **55**, 139–148 (2016). https://doi.org/10.1016/j.conengprac.2016.07.001
18. Hasegawa, M., Takata, M., Temmyo, T., Matsuka, H.: Modelling of exception handling in manufacturing cell control and its application to plc programming. In: Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on, pp. 514–519 (1990). https://doi.org/10.1109/ROBOT.1990.126031. IEEE
19. Laursen, J.S., Ellekilde, L.-P., Schultz, U.P.: Modelling reversible execution of robotic assembly. In: Robotica, pp. 625–654 (2018). https://doi.org/10.1017/S0263574717000613
20. El-Wardany, T.I., Gao, D., Elbestawi, M.A.: Tool condition monitoring in drilling using vibration signature analysis. Int. J. Mach. Tools Manuf. **36**(6), 687–711 (1996). https://doi.org/10.1016/0890-6955(95)00058-5
21. Hsueh, Y.-W., Yang, C.-Y.: Prediction of tool breakage in face milling using support vector machine. Int. J. Adv. Manuf. Technol. **37**(9-10), 872–880 (2008). https://doi.org/10.1007/s00170-007-1034-8
22. Batal, I., Hauskrecht, M.: A supervised time series feature extraction technique using dct and dwt, In: Machine Learning and Appications, Fourth International Conference on, pp. 735–739 (2009). https://doi.org/10.1109/ICMLA.2009.13. IEEE
23. Chang, C.-C., Lin, C.-J.: Libsvm: a library from support vector machines, software available at http://www.csie.ntu.edu.tw/cjlin/libsvm (2001)
24. Manning, C.D., Raghavan, P., Schötze, H.: An introduction to information retrieval. Cambridge University Press, Cambridge (2008). ISBN: 0521865719
25. Staelin, C.: Parameter selection for support vector machines. HPL-2002-354 (R.1) (2003)
26. Hsu, C.-W., Chang, C.-C., Lin, C.-J.: A practical guide to support vector classification (2016). https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf
27. Shlens, J.: A tutorial on principal component analysis. In: Systems Neurobiology Laboratory, Salk Institute for Biological Studies (2005). https://www.cc.gatech.edu/~lsong/teaching/CX4240spring16/pca_schlens.pdf
28. Edwards, C., Raskutti, B.: The effects of attribute scaling on the performance of support vector machines. In: AI 2004: Advances in Artificial Intelligence, pp. 500–512 (2004). https://doi.org/10.1007/978–3–540–30549–1_44. Springer

Ewa Kristiansen received her PhD degree from the Department of Production at Aalborg University, Denmark, in 2009, and continued 11 years with an academic career in the Department of Materials and Production at Aalborg University, Denmark. In 2007 she was a visiting graduate student at The Robotics Institute at Carnegie Mellon University, USA. Her research interests include automation of manufacturing processes, scheduling of industrial robots, and design and control of mechatronic systems. She is currently employed as a Senior Robot and Controls Engineer in Grundfos.

Emil Krabbe Nielsen received his MSc degree in the Department of Mechanical and Manufacturing Engineering at Aalborg University, Denmark, in 2013. In 2012 he was a visiting graduate student at The Robotics Institute at Carnegie Mellon University, USA. He is now working towards a PhD degree in the Department of Electrical Engineering in Technical University of Denmark. His research project is about functional modelling of water treatment systems and his research interests are related to fault diagnosis. He is currently employed as an Onboarding Specialist at Kairos Technology AS.

Lasse Hansen received his MSc degree in the Department of Mechanical and Manufacturing Engineering at Aalborg University, Denmark, in 2013. In 2012 he was a visiting graduate student at The Robotics Institute at Carnegie Mellon University, USA. He is currently employed as a team leader at Nel Hydrogen, Denmark, where he is leading the development of the next product generation of heavy duty hydrogen fueling station. He has previously worked as a Production Engineer in EagleBurgmann.

David Bourne is a Principle System Scientist in the Robotics Institute at Carnegie Mellon University, where he heads the Rapid Manufacturing Lab. Dr. Bourne's research focuses broadly on building intelligent systems for automated manufacturing. It is including the following research issues; representing behaviour in the design process, the semantics of tolerance and design for manufacturing and manufacturing feedback to design. He has published over 70 manufacturing publications and has had eight patents.