2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Autonomous optimization of fine motions for robotic assembly*

Emil Krabbe[1], Ewa Kristiansen[1], Lasse Hansen[1] and David Bourne[2]

Abstract— **In the past, robotic assembly has required rigid fixturing and special purpose robotic tools for every assembly component. Unfortunately, rigid fixtures and special purpose robotic tools often have to be customized for varying geometries. Alternatively, it is possible to operate in a semi-structured environment, defined by the use of softer fixtures (e.g. pick-up bin) and softer robotic tools (e.g. suction cups or compliant pads) that can be used for many assembly applications without modification, but they demand specific motion plans that can tolerate greater positional uncertainty.**

We have developed a system that supports autonomous generation of parameterized fine motion plans for assembly that are robust under positional uncertainty and compliance introduced by the use of a suction cup instead of a gripper. To accomplish this a classifier is trained, implemented and tested for performance in the semi-structured environment for distinguishing between a failed or successful assembly. The trained classifier is then integrated with the entire system and many robot-attended experiments are performed that vary the fine motion parameters, and optimize them for successful outcomes using an Interval Estimation optimization algorithm. An approach to machine learning based on Support Vector Machines and Principal Component Analysis is used to make the optimization autonomous.

We achieved a 99.7% classification accuracy with the trained classifier and by running repeated robot-attended experiments with artificial positional uncertainty and optimizing fine motion parameters, we were able to achieve a 38% improvement compared to fine motion plans with initial best guess parameters.

I. INTRODUCTION

The efficiency of innovation, research, development and implementation in consumer electronics has rapidly increased during the past many years, leading to shorter product life cycles due to more frequent introductions of new products to the market [1]. This has led to more frequent production changeovers.

When facing a production changeover a motion plan including an assembly strategy is programmed when using robots. A significant amount of time is often spent on manually tuning the robotic fine motions of the assembly strategy, which are those motions that involve either contact or the risk of contact and hence they have tight positional and dimensional tolerances for the parts to be assembled. In order to avoid the manual tuning of fine motions a framework is developed, which allows a set of a Cartesian coordinates and rotations, crucial to the outcome of the assembly, to be identified and optimized autonomously. Due to a decrease in the length of product life cycles, an autonomous assembly must be flexible and capable of accommodating a variety of parts in order to be feasible [2].

An optimization algorithm will be used to autonomously tune Cartesian coordinates of the fine motions. This is done by empirically searching a space where certain parameter values of the assembly strategy give a high chance for a successful assembly. The algorithm finds a set of parameter values that maximizes the probability for success despite purposely changing positional tolerances. The algorithm requires feedback information on whether an assembly has failed or succeeded. Human interaction is only required to manually train a classifier. The classifier then gives feedback so that the optimization can run autonomously.

We have intentionally added noise into the system to anticipate process uncertainties and increase robustness to uncertainties. Others have attempted to study assembly under noise through simulation of a flexible assembly cell [3]. The positional uncertainty is purposely added to the position of a battery, in the studied assembly task of the insertion of a battery into a Nextel Blackberry 7520 cell phone. The positional tolerances are defined as pseudo random noise, which is repeatable and makes different experiments comparable. The autonomous tuning system must accommodate for different variants of products in a semi-structured environment, making it applicable for a flexible assembly cell. A generic method for collecting data and training a classifier, which can aid parameter optimization, is hereby proposed. The chosen assembly task exemplifies the procedures and performance of the solutions presented in this paper.

II. PREVIOUS WORK

Earlier on the common trend in research within the field of autonomous robotic assembly has been on fault detection, fault diagnosis and error recovery. The focus was on learning how to take corrections during production to ensure a successful assembly. This has been done by employing various sensing techniques and learning algorithms, that would return information about why the assembly failed and how to proceed to assemble successfully [4].

In [5] the authors proposed a method for fault detection and diagnosis, which was based on the mathematical model of the frictional force signals during different phases of the assembly process and for different assembly outcomes. The outcome was classified either as successful or as one of four specified faulty cases. The frictional force model was established from different sensors mounted on the robot and by using a parameter estimation method, which was an

This work was supported by ABB Corporate Research and Institute of Product Development (IPU)

[1]Department of Mechanical and Manufacturing Engineering, Aalborg University, 9000 Aalborg, Denmark. ekn@ipu.dk, ewa@m-tech.aau.dk, lassehansen2@gmail.com

[3]Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA. db@cs.cmu.edu

adapted regular least-square estimation method. Contrary to this work, our paper treats the system as a black-box model, where the fault detection and optimization of assembly parameters are based on the successful and faulty experiments made prior to the actual production. Fault diagnosis and error recovery is not a topic of this paper.

In [3] a generic and autonomous robotic assembly architecture has been proposed to introduce flexibility. The system was built to detect, classify and explain causes and consequences of errors in order to continuously increase robustness. The error processing was used to tie different errors with recovery functions and sensor signals for a variety of products. It uses a force signal for the sensors mounted on the robot wrist. Machine learning is applied for distinguishing between different types of failures and successes and for reacting to different failure scenarios during the assembly, in order to achieve a successful assembly. The system in [3] develops gradually, whereas the framework presented in this paper does not change over time, because the assembly strategy with the highest probability of achieving a successful assembly has been chosen prior to production.

In [6] a genetic algorithm is used to optimize parameters of assembly strategies in order to reduce the assembly time. The assembly quality is evaluated by using machine learning to make the optimization autonomous. Instead of assembly time this paper aims to optimize the success rate of assemblies.

In [7] assemblies have been successfully classified as either a success or a failure on the basis of a force signal generated in the tool wrist. It was proven that by merely using 20 training examples with a classifier and a one axis force sensor built into the end effector, the failed assemblies could be classified. Principal Component Analysis was used as feature selection for reducing the amount of training examples in [7]. This paper is an extension of Rodriguez's framework, by optimizing the assembly success rate and automating the optimization. We are using four force sensors mounted to the fixture and not to the robot's end effector. The main contribution areas of the described papers are summarized in table I.

TABLE I

THE MAIN CONTRIBUTION AREAS OF THE PAPERS. AUTONOMOUS ABBREVIATED BY (A).

Paper	[3]	[4]	[5]	[6]	[7]	Our
Fault detection	✓	✓	✓	✓	✓	✓
Correction in production	✓	✓				
Optimization			✓	A		A
Continuous improvement	✓	✓				

III. SYSTEM SETUP

A. Hardware

The hardware setup is shown in figure 1 and includes:

(1) 6 axis ABB robot with:
(2) Suction cup
(3) Fixture with:
(4) Four force sensors
(5) Cell phone
(6) Battery
(7) HD camera
(8) Kinect camera
(9) Recovery fixture (see [8])

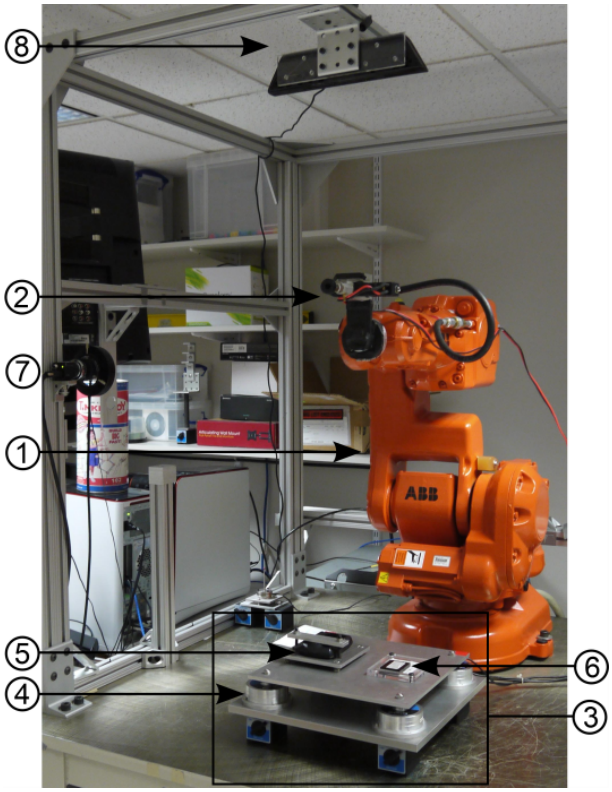

Fig. 1. Hardware setup.

What complicates this assembly of a battery and a cell phone are the limitations on the choice of a tool, since the battery can only be grasped from one side. The suction cup is not the best option for a tool in an assembly, where requirements for positional tolerances can be significant, due to the compliance. The compliance introduces flexibility which can however also be advantageous in a semi-structured environment where errors are more likely to occur.

The idea behind the fixture was to make it simple, and also modular to enable adaption for different parts. It is composed of four cheap one axis force sensors, a bin for the battery and a bin for the cell phone. The purpose of using four force sensors on the fixture instead of only one on the robot wrist, was to obtain information about the directions of the motions. A large force magnitude on only one force sensor indicates that the assembly motion is either in the direction of the force sensor, or that it takes place closest to the force sensor. This information might be necessary when the application of our solution should be extended to other assemblies than the battery and cell phone. Compliance has been introduced by using four preloaded springs to attach the cell phone bin to the fixture. The springs prevent harmful collisions and allow for producing different values of force signals by varying the value of preloading.

We are using a Kinect and a HD camera for error detection and recovery, as well as a recovery fixture for turning the battery upside down [8].

B. Software

Robot Operating System (ROS) has been used to increase the functionality of the ABB robot and interface with external sensors. ROS is installed on a computer with a Linux distribution and is used for handling all communication with optical sensors, force sensors and the ABB robot through nodes. A ROS package was written for execution of the assembly and data collection strategy as well as preprocessing, machine learning, error detection, error recovery and optimization. The interfaces between the different functionalities, and the hardware and software for parameter optimization are shown in figure 2. The optimization shown in the first box in figure 2 is implemented as a MATLAB script in the package and inputs a new Cartesian coordinate, B, and rotation, A, to the assembly. The package utilizes nodes for executing robot movements, for data collection with force sensors and for classification with the C++ library LIBSVM [9]. It also includes a package explained in [8] for the error detection and recovery, which utilizes nodes for the optical sensors and the library OpenCV for the computer vision processing and analysis.

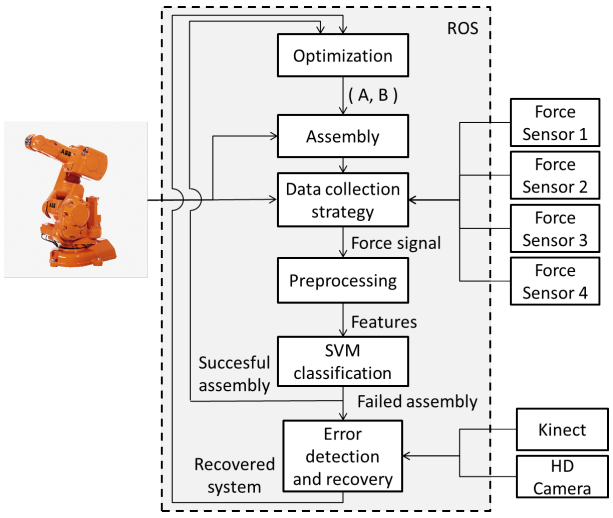

Fig. 2. Software and hardware architecture for parameter optimization, showing relations between functions.

IV. PROCEDURE

A classifier must first be trained for distinguishing between a force signal for a successful and a failed assembly. Afterwards a set of Cartesian coordinates can be optimized, in order to assemble with a high probability of succeeding in production.

A classifier is trained using the method presented in [7]. All force signals, called training examples, are manually labelled into one of the two classes: failure or success. Manual labelling refers to the operator's classification of the assembly as either a success or a failure, which is based on a visual inspection of the physical assembly. In order to collect this force signal a data collection strategy must be used. The force signal will be preprocessed before being used to train the classifier. Afterwards the classifier will be able to predict whether an assembly failed or succeeded based on the force signal. All steps for the training, optimization and production involved in this work are shown in figure 3. The dashed lines indicate that a step is not mandatory.

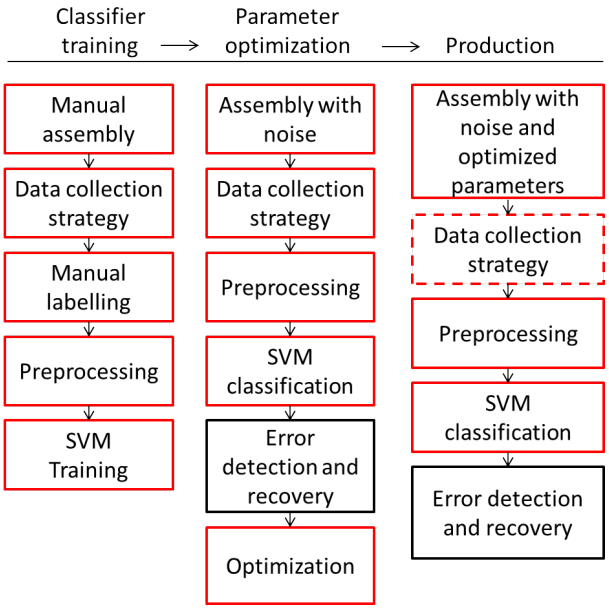

Fig. 3. Procedure for training the classifier, optimizing fine motion parameters and assembling in production. The functions shown in red boxes were developed in this work. The functions shown in black boxes were developed in [8].

The optimized parameter values are found by executing an assembly strategy and then a data collection strategy to obtain a force signal, which is then classified. This is repeated until the optimized parameter values are found. In case the assembly task fails, the system recovers by using the error detection and recovery system [8]. The optimization algorithm is an Interval Estimation Maximization Method (IEMAX) [10], which uses the upper limit of the Bayesian confidence interval. The values of the optimization parameters are constrained to an interval for each parameter, and the algorithm starts by searching these intervals, when performing the assembly with different parameter values. These sets of parameter values are classified as either a successful assembly (1) or a failure (0). In the beginning IEMAX searches stochastically and then focuses on the most promising area. IEMAX then maximizes the probability for a success by choosing the set of parameters with the highest upper limit for the confidence interval for this probability. A new set of parameters are found based on the feedback of the outcome of the previous assembly. The new set of parameters has the highest upper limit for the Bayesian confidence interval. The parameter optimization continues until a search criterion is fulfilled.

In production the optimized parameters are used for every

assembly. The classifier is used to label assemblies in order to recover from failed assemblies. The data collection strategy can be omitted and a classifier trained on the force signal during assembly can be used to save time. In this work the data collection strategy has not been omitted.

V. STRATEGIES

A. Strategy for assembly

The strategy for manual assembly during classifier training is to manually place the battery by hand into the cell phone, so that it results in either a succes or failure. The strategy for assembly when optimizing fine motions is composed of five steps shown in figure 4, where the robot (1) first lowers the battery into the slot at an angle A [°] and a position B=(0,Y_B) in the coordinate frame in the center of the battery slot in figure 4. The battery is then aligned (2) with the edges of the slot, (3) moved towards the connector pins and (4) aligned with the bottom of the slot. At last the suction cup releases the battery (5) and pushes down on the battery edge farthest away from the pins. After executing the assembly strategy the data collection strategy shown on figure 5 is used to produce a force signal as shown on figure 6. The parameters to be optimized should be chosen from a step which has the most influence on the outcome of the assembly. There is a vast amount of ways to tune the assembly, which could be coordinates, rotations or speeds of any fine motions involved in the assembly strategy. In this work we chose to optimize the values of parameters A and B.

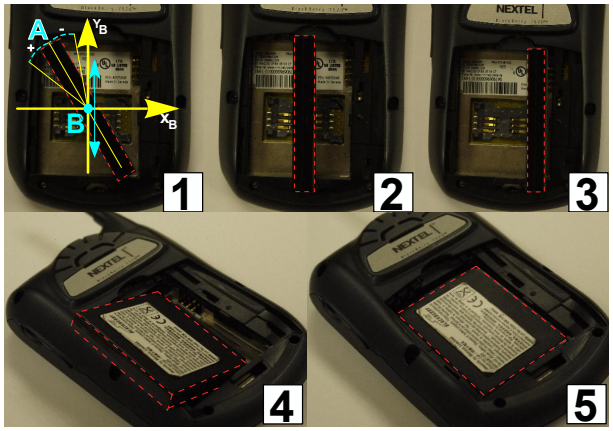

Fig. 4. 5 steps of the assembly strategy and the tuning parameters.

B. Strategy for data collection

In order for the classifier to classify an assembly correctly the data collection strategy must produce a force signal for a failed assembly, which is different from a successful assembly. The data collection strategy is shown in figure 5, and it consists of tapping at three corners of the battery slot with the suction cup. The robot is applying a force when tapping with a constant speed of 150 mm/s on the battery. The outcome of the assembly, as either successful or failed, depends on the values of the measured force exerted on the force sensors. The data collection strategy is kept simple in order to easily transfer the strategy to similar parts or similar assembly types. The data collection strategy can be omitted in production by training a new classifier on a signal from the assembly strategy, since executing this strategy introduces excess time in production.

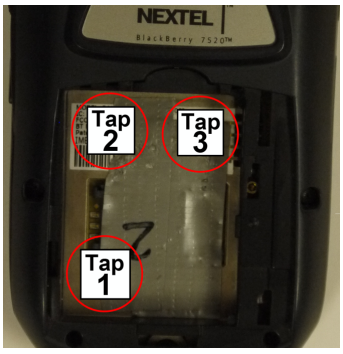

Fig. 5. Data collection strategy using three taps. The taps are recorded using the four force sensors positioned as shown in figure 1.

VI. EXPERIMENTAL RESULTS OF CLASSIFIER TRAINING

A. Preprocessing

When training a classifier, it is possible to increase the classification accuracy by preprocessing the training data [11]: and hence increasing distinction between the two classes; failed assembly and successful assembly. The raw signal is preprocessed in the following way:

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

The classifier, giving inputs to the parameter optimization, is a trained Support Vector Machine (SVM) [9]; this type of classifier performs well with little training data compared to many other classifiers [12]. 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 [13].

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 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 regards to occurrence in time across training examples [14].

For each feature, $j \in [1;81]$, the force magnitudes from the 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 figure 6 for all 100 training examples and have been calculated by using the following equation:

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

Principal Component Analysis (PCA) [15] can be used for transforming signals linearly to decrease a high correlation between features and decrease a large feature space dimensionality [7]. 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 components represent 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 feature with the maximum variance in the magnitude of the force signal for all the training examples. All subsequent principal components will account for a decreasing variance in decreasing order. The last principal component hereby accounts for the feature with the least variance for all training examples.

In order to improve classification accuracy the last and final preprocessing step is to scale the value of the force magnitude, x_j, so that $x_j \in [0,1]$ [14] [16]. One minimum and one maximum value of the force magnitude, denoted 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:

$$\forall j \in [1;81]:$$
$$x'_{(j)} = \frac{x_{(j)} - min_j}{max_j - min_j} \quad (2)$$

B. SVM Training

Two different strategies for SVM training are tested, and their performance is compared with each other. One training strategy is based on the force signal produced during assembly and the other one during the tapping strategy. By training a classifier on the force signals generated during assembly without any subsequent tapping, it was not possible to obtain distinct force signals for failures and successes resulting in a desired classification accuracy of at least 99%. The resulting classification accuracy was at most 92.9 % when using 100 training examples. The accuracy varied between 92 and 92.9 % when varying the number of principal components between 1-12. Increasing the number of principal components further, led to a reduction in accuracy.

Using the data collection strategy involving tapping, resulted in two distinct bands describing the successful or failed assemblies. The two bands are marked by different colors in figure 6. They are only distinct at three peaks corresponding to each of the three taps by the robot.

The tapping strategy performed well with as few as 4 training examples and 1 principal component, getting a 100 % testing accuracy independent of the number of principal components. When training the classifier 80% of the training examples are used to train the classifier on, and 20% to test it on. These classification accuracies are obtained by testing the

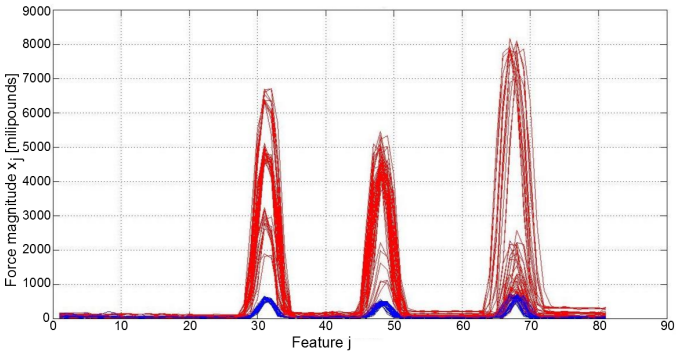

Fig. 6. Plot of 100 signals corresponding to 100 training examples classified by the operator from failed (red) and successful assemblies (blue) when tapping.

classifier on force signals, which the classifier was not trained on. The classifier used for experimentation was trained on 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 the tapping. This could result in misclassification when using only 1 principal component representing only the tap with force exertion.

The tapping strategy performs well with fewer training examples than the assembly strategy. The performance comes at expense of extra time consumption. The entire assembly including picking up the battery, taking it out of the cell phone and putting it back into the bin takes 13.5 seconds. The tapping strategy takes 5.5 seconds. When tapping the cycle time is prolonged with 41%.

In this case the tradeoff is between time and accuracy. Two possible reasons for good performance when using the tapping strategy compared to the assembly strategy are that the variance of the force magnitude between the successes is very low, and that force magnitudes representing successes are very distinct from the force magnitudes representing failures. During the assembly strategy the battery may be positioned farther up the edge of the cell phone slot, so that the battery exerts a large force onto the side of the cell phone slot during step No. 5 of figure 4, even though the assembly is successful. This results in force signals with high force magnitudes for successful assemblies that are similar to force signals of failed assemblies. The change in values of the optimization parameters, A and B, also has an influence on the force signals during the assembly strategy thus making any classification more difficult. However the change in values of the optimization parameters, A and B, do not influence the signal during tapping. The accuracy of the classifier was evaluated, as the production procedure shown in figure 3, by 1500 assemblies with introduced positional noise which will be explained in section VII.

When using a classifier trained on 100 training examples, 3 out of 1000 assemblies based on the tapping strategy were misclassified, resulting in a 99.7 % accuracy. Using a classifier trained on 4 training examples, 13 out of 500 assemblies based on the tapping strategy were misclassified, thus giving

an accuracy of 97.4 %. Choosing between the number of training examples is a matter of how well the classifier generalizes new force signals by avoiding overtraining and the interaction time of the operator. Using only 4 training examples will give the least operator interaction time and the best ability to generalize well, unless the two bands of failures and successes lie close together. The classifier trained on 100 training examples is used for further experimentation, because of the better classification accuracy.

VII. Experimental results of parameter optimization

The procedure for optimizing parameter values is as follows:

While (optimization != converged)

1 Drag the battery to the corner of the bin to know the absolute battery position, see figure 7.1 and 7.2.
2 Induce pseudo random positional noise by:
   – Rotating battery by an angle R, see figure 7.3.
   – Translating Battery in X-Y direction, see figure 7.3.
3 Pick up battery at the supposed center coordinate.
4 Move through steps 1-5 in figure 4 using a specific value of the optimization parameters A and B.
6 Start logging force signal from sensors 1-4.
7 Execute data collection strategy from figure 5.
8 Stop logging force signal from sensors 1-4.
9 Preprocess and classify signal.
10a If failed assembly
   – Run error detection and recovery routine [8].
10b Else
   – Pick the battery out of the slot and place it in the bin.
11 Input classification 0 or 1 to optimization algorithm and find new parameter values for A and B.

The optimized fine motion parameters have to be robust to the positional noise - therefore pseudo random noise has been added to the position and orientation of the battery before being picked up for assembly. The pseudo random noise is added as a rotation $R \in [-4^\circ; +4^\circ]$ and a translation in both the x- and y-direction by [-3 mm; +3 mm]. The value of the added noise always allows the suction cup to pick up the battery. The advantage of using pseudo random noise is that it is physically repeatable, which makes comparison on different assembly strategies possibly. A convergence criterion of 1000 assemblies was used.

The applied optimization algorithm IEMAX [10] may produce several local maxima, each of which may have different performances against the positional noise. The optimal parameter values however has the highest probability for assembling correctly independent of only one type of noise and the magnitude of this noise. To begin with the values of the optimization parameters A and B are chosen stochastically. Afterwards A and B are chosen statistically. The set of parameter values of A and B with the highest upper bound are chosen. The upper bound is calculated by using the following equation [10]:

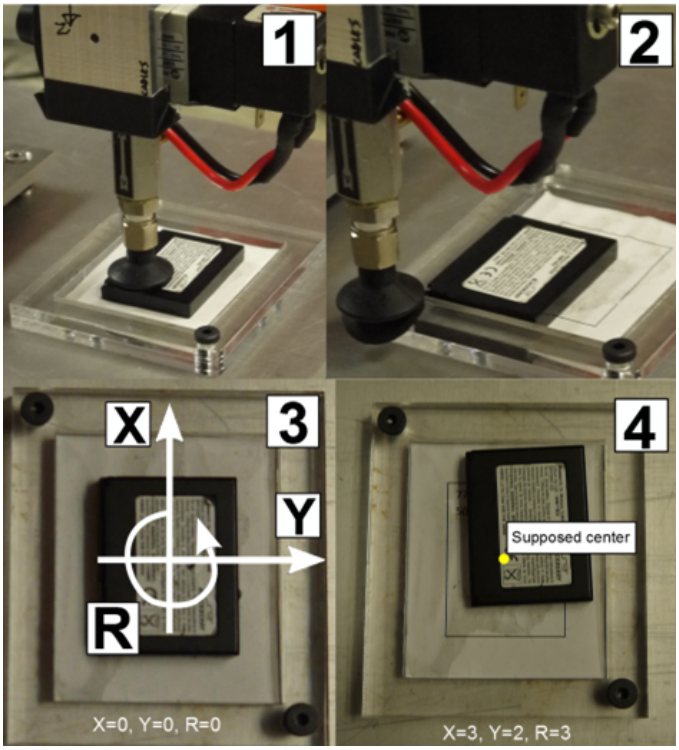

Fig. 7. Procedure for adding noise. From left step (1) to step (4) on right

$$ub(x,n) = \frac{\frac{x}{n} + \frac{z^2_{\alpha/2}}{2n} + \frac{z_{\alpha/2}}{\sqrt{n}}\sqrt{\frac{x}{n}\left(1-\frac{x}{n}\right) + \frac{z^2_{\alpha/2}}{4n}}}{1 + \frac{z^2_{\alpha/2}}{n}} \quad (3)$$

In this equation n denotes the number of assemblies with a given parameter value, and x is the number of successful assemblies for that value. For this system a confidence level of 95% has been chosen, for the assembly to be successful, hence $\alpha = 0.05$. The z-score $z_{\alpha/2}$ can then be found from the percentile $\alpha/2 = 0.025$. The upper bound $ub(x,n)$ is calculated for both the A and B parameter value for each assembly, and if another set of parameter values has a larger sum of these two upper bounds it is chosen as the next set of parameter values. If there are no other parameter values that have larger or equal upper bounds, the same set of parameter values is used again. When the optimization starts every parameter value of the optimization parameters A and B has a 100% probability of assembling successfully. In this way the probability of a local maximum is decreased by assembling with parameter values at the maximum, until a new local maximum has higher probability. This continues until a maximum does not decrease.

VIII. Experimental results of production

The assembly in production has been repeated 500 times with two sets of fine motion parameters: the optimized fine motion parameters A$= 4.0^\circ$ and B$= 3.0$mm and the original fine motion parameters A$= 0.0^\circ$ and B$= 0.0$mm. The two set of parameters were both tested on the exact same set of

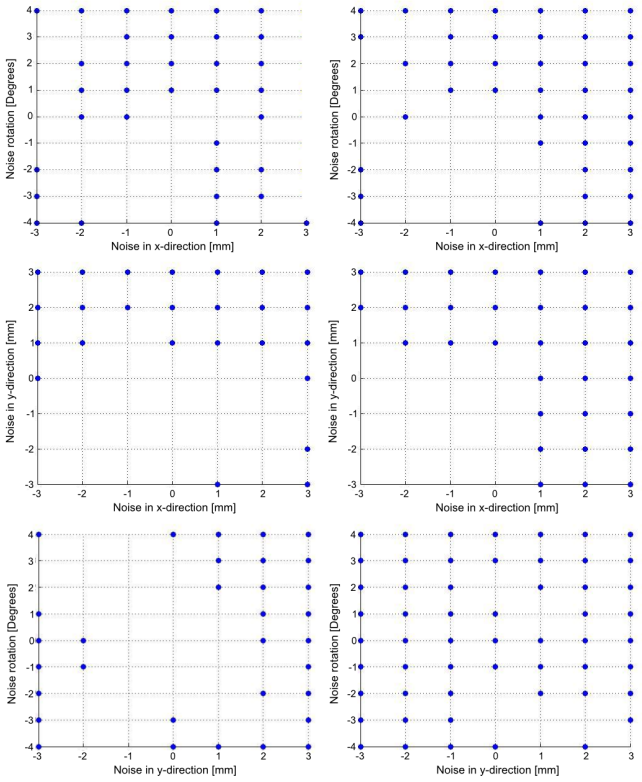

Fig. 8. 2D plot of failed assemblies with optimized fine motion parameters (3 left) and without (3 right).

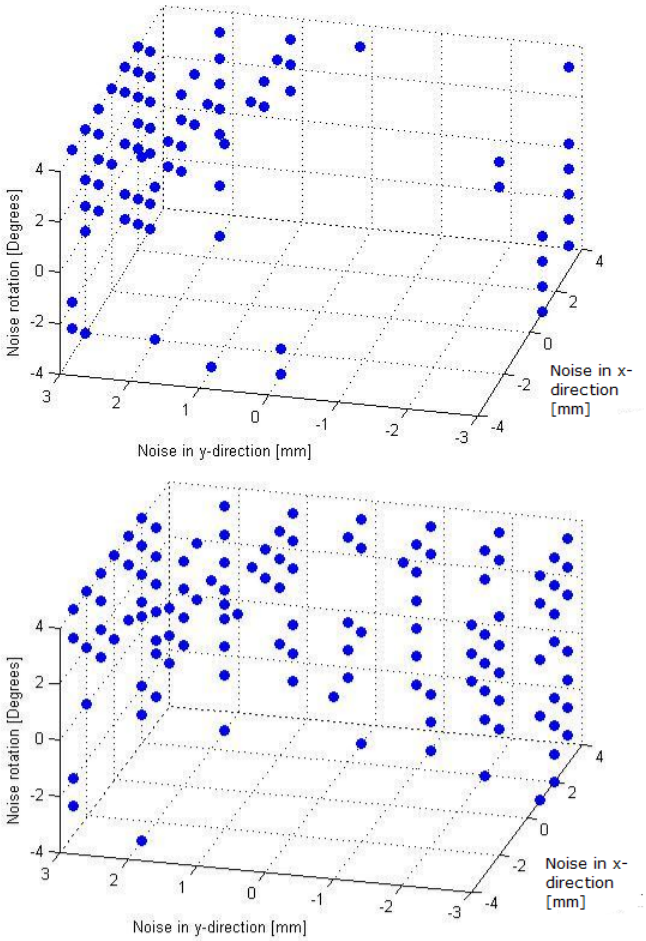

Fig. 9. 3D plot of failed assemblies for optimized (upper) and original (lower) parameter sets.

pseudo random noise. Since the added noise is discrete there are $7 \cdot 7 \cdot 9 = 441$ possible outcomes.

Assemblies with the original and the optimized set of fine motion parameters, which cannot tolerate specific positional noise, and results in failures, are marked by points in figure 8 and 9.

The optimized fine motion parameters tolerate positional noise better, as there are fewer failed assemblies when adding noise in the x-direction and negative y-direction, as well as when adding rotational noise and noise in the y-direction. The performance can be seen in table II for the optimized and original set of parameters against positional noise during production. The optimized parameter set resulted in 73 fewer failed assemblies, which is a decrease of 38% in the failure rate. However it is assessed that this result can be improved further by experimenting with different assembly strategies and random noise.

TABLE II

PERFORMANCE OF OPTIMIZED AND ORIGINAL SET OF FINE MOTION PARAMETERS AGAINST PSEUDO RANDOM NOISE.

Parameter set	No. of assemblies	No. of failures	Success rate
Optimized	500	119	76.2 %
Original	500	192	61.6 %

Neither of the parameter sets perform well against positive noise for the rotation and y-direction. The reason is that a positive translation in the y-direction makes the assembly take place above the slot, whereas a negative exploits the compliance and aligns the battery despite a rotation.

The influence of positional noise on the insertion is shown in figure 10. The crucial difference in performance before and after optimization is that the optimized parameters perform well against noise in the positive x-direction, which the original does not. This is because the optimized angle A in figure 4, makes the insertion less likely to hit the side of the cell phone circled in figure 10, when the battery is lowered into the slot.

It is noticeable how all the failed assemblies are primarily clustered in the positive direction of the noise, this is a result of the compliance in the suction cup, which in this case is only able to tolerate and compensate for negative noise. This means that in the instances with this particular assembly the compliance in the suction is one sided as it compensates for noise in only one direction.

IX. FUTURE WORK

Assuming this framework to have any commercial relevance, it must be capable of detecting failures for assemblies

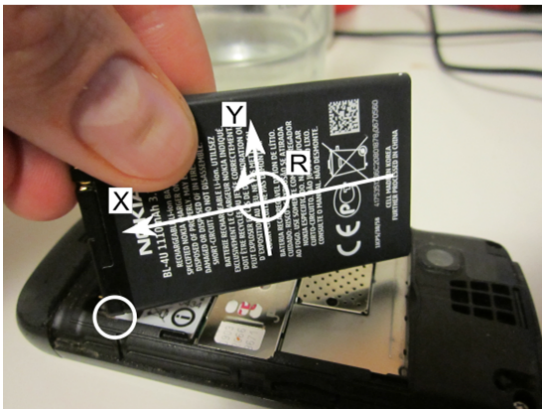

Fig. 10. Noise influence during fine motions in x-, and y-direction and the rotation.

of a wide range of different parts. This could be done by finding generic methods for producing distinguishable force signals dependent on e.g. the assembly type, the shapes or the constraints. Another approach would be to come up with a different way to preprocess the signal. If this can be done efficiently, it may even be possible to avoid using a data collection strategy such as tapping. If tapping can be avoided, it will not be necessary to first train a classifier on the force signal from tapping for optimization, and then a classifier on the assembly signal for production, in order to save time from executing the extra tapping motions during production.

It is more reasonable to focus on finding a method for making an assembly strategy, which gives a high probability of success rather than to only focus on fine motion optimization. If the strategy is not suitable for the assembly, optimizing the fine motion parameters will most likely not increase the ratio of which the assembly succeeds. When trying to come up with assembly strategies for the battery, some strategies were very difficult to assemble the cell phone and battery with, and seemed more prone to failure with positional noise. The future focus should be on finding an assembly strategy with a high probability of success. Another interesting thing to examine would be to benchmark the applied noise with the tolerances in a structured environment.

The presented framework allows for comparisons on the vitality and efficiency of different assembly motions for specific parts. The same goes for the optimization parameters, in order to identify correlations between fine motions or parameters of these motions and certain types of noise. Any two assembly strategies can now be interchanged, tweaked, tuned and compared on the same pseudo random noise.

X. CONCLUSIONS

By making a framework capable of detecting failed assemblies and recovering from them, it was possible to optimize a set of fine motion parameters to be robust against positional noise of parts to be assembled. The optimization resulted in 38 % fewer failed assemblies compared to the original parameter values. It is however necessary to identify the noise that must be optimized against, and find a parameter of the fine motions which is influenced by this specific noise. By physically finding the absolute position of the battery and afterwards adding pseudo random positional and rotational noise to the battery, it was possible to make comparisons based on the results of the physical experiments instead of simulation results from a model of the experiments.

Flexibility can be introduced in robotic assembly by using softer robotic tools and fixtures. To cope with the introduced compliance a set of parameters of the motion plan that are crucial to the outcome of the assembly, can be optimized in order to reduce the number of failed assemblies with potentially 38%.

XI. ACKNOWLEDGMENTS

The authors would like to acknowledge and thank especially Nishant Kelkar for his contributions to the project, as well as, Alberto Rodriguez, Robbie Paolini, Mabaran Rajaraman and ABB Corporate Research for their inspiration and help.

REFERENCES

[1] X. Xianhao, and S. Qizhi, Forecasting for products with short life cycle based on improved BASS model, 19th International Conference on Production Research, 2007.
[2] C. Gaimon and V. Singhal, Flexibility and the choice of manufacturing facilities under short product life cycles, European Journal of Operational Research, vol. 60, No. 2, July 1992, pp. 211-223.
[3] L. S. Lopes and L. M. Camarinha-Matos, Learning to Diagnose Failures of Assembly Tasks, Annual Review in Automatic Programming, vol 19, pp. 97-103, Oct. 1994.
[4] M. Santochi and G. Dini, Sensor Technology in Assembly Systems, CIRP Annals - Manufacturing Technology, vol. 47, No. 2, pp. 503-524, 1998.
[5] J. Huang, U. Nagoya, T. Fukuda, T. Matsuno, Model-Based Intelligent Fault Detection and Diagnosis for Mating Electric Connectors in Robotic Wiring Harness Assembly Systems, IEEE/ASME Transactions on Mechatronics, vol. 13, No. 1, pp. 86-94, 2008.
[6] J. A. Marvel, W. S. Newman, D. P. Gravel, G. Zhang, J. Wang and T. Fuhlbrigge, Automated learning for parameter optimization of robotic assembly tasks utilizing genetic algorithms, IEEE International Conference on Robotics and Biomimeticst, vol. 11, pp. 179-184, 2009.
[7] A. Rodriguez, D. Bourne, M. Mason, G. F. Rossano and J. Wang, Failure Detection in Assembly: Force Signature Analysis, IEEE Conference on Automation Science and Engineering, Aug. 2010.
[8] L. Hansen, E. Krabbe, E. Kristiansen and D. Bourne, Error handling in robotic assembly optimization process, To be submitted for publication.
[9] C.-C. Chang and C.-J. Lin, LIBSVM: a library for support vector machines, 2001, software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.
[10] L. P. Kaelbling, Learning in Embedded Systems, MIT Press, 1990.
[11] I. Batal and M. Hauskrecht, A Supervised Time Series Feature Extraction Technique Using DCT and DWT, Machine Learning and Applications, Fourth International Conference on, vol. 0, pp. 735-739, 2009.
[12] C. D. Manning, P. Raghavan and H. Schütze, Introduction to Information Retrieval, Cambridge University Press, ISBN: 0521865719, 2008.
[13] C. Staelin, Parameter selection for support vector machines, HPL-2002-354 (R.1), 2003.
[14] C.-W. Hsu, C.-C. Chang and C.-J. Lin, A Practical Guide to Support Vector Classification, 2010.
[15] J. Shlens, A tutorial on Principal Component Analysis, Systems Neurobiology Laboratory, Salk Institute for Biological Studies, 2005.
[16] C. Edwards and B. Raskutti, The effect of attribute scaling on the performance of support vector machines, Proceedings of the 17th Australian joint conference on Advances in Artificial Intelligence, pp. 500-512, 2004.