Prediction and Diagnosis of Propagated Errors in Assembly Systems Using Virtual Factories

Cem M. Baydar[1]

e-mail: cem-baydar@accentuate.com

Accenture Technology Labs, 3773 Willow Rd., Northbrook, IL 60062-6212

Kazuhiro Saitou

e-mail: kazu@umich.edu

Assist. Prof. Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI 48109-2125

Large-scale automated assembly systems are widely used in automotive, aerospace and consumer electronics industries to obtain high quality products in less time. However, one disadvantage of these automated systems is that they are composed of too many working parameters. Since it is not possible to monitor all these parameters during the assembly process, an undetected error may propagate and result in a more critical detected error. In this paper, a unique way of detecting and diagnosing these types of failures by using Virtual Factories is discussed. A Virtual Factory was developed by building and linking several software modules to predict and diagnose propagated errors. A multi-station assembly system was modeled and a previously discussed "off-line prediction and recovery" method was applied. The obtained results showed that this method is capable of predicting propagated errors, which are too complex to solve for a human expert.
[DOI: 10.1115/1.1411966]

Introduction

Large-scale robotic assembly systems, which are composed of several stations, are widely used in automotive, aerospace and consumer electronics industries. The main advantage of these systems is their performance of manufacturing high-quality products in less time than the manual or semi-automated assembly lines. However, since these large-scale systems are composed of many automated components, there are too many working parameters during the operation phase and it is not possible to monitor all of these parameters. This makes these systems vulnerable to any perturbations during the operation, which may result in unexpected failures.

The problem of having unexpected failures arises from the fact that these failures cannot be predicted easily before the actual assembly process. These failures generally result from the propagation of undetected errors during the assembly process. According to Abu-Hamdan and El-Gizawy [1], error-propagation is defined as carrying an undetected error from a previous task and coupling it with another error during a proceeding task. It is also stated in [2] that, when a failure is detected during an operation,

the operation that failed may not be the source of failure. The source of failure may have been propagated from earlier operations.

On-line diagnosis of these types of failures by human experts causes costly downtime and maintenance. It was reported that these unplanned maintenances cost around 200 billion dollars in the United States in 1990 [3]. The aim of this paper is to discuss an efficient way of predicting and diagnosing propagated errors in robotic assembly systems using *Virtual Factories* using "off-line programming." In *off-line programming*, any robotic system can be modeled virtually in 3D and the performance of the system can be evaluated accurately from the simulations. Previous discussions on the off-line error prediction, diagnosis and recovery were made in [4–7] and a method was proposed and evaluated using this approach. This study involves the integration of this method by using a commercial assembly simulation software with several developed modules to build a *Virtual Factory*. This structure will enable us to predict unexpected propagated errors before the actual assembly operation takes place thus providing efficient means of diagnosis and recovery for these types of failures when they happen. In this study, five different types of failures are under consideration and these are: grasping failures, sensor failures, collision errors, flawed parts and misplacement errors.

Previous Work

Prevention of the propagated errors has been a great interest in the automated recovery of robotic assembly systems. The established techniques of Failure Mode and Effect Analysis (FMEA), Fault Tree Analysis (FTA) and Event Tree Analysis (ETA) are used widely [8,9]. FMEA is used to examine all possible component failure modes and to identify their first order and final effects on the system. It is an important method to assure quality. FTA and ETA may be applied at various levels for examining the errors and failures in a system. FTA is a top-down technique for assessing the way in which several failures can cause a single outcome or a system failure. ETA is a 'forward' technique, which may be used to examine the propagation of an initiating event (or failure) with the presence of a number of other events, failures, faults or conditions. These methods are used during the design stage of the assembly system in order to predict possible propagated failure situations.

Probability theory is also used to analyze failure uncertainty. Fuzzy methodology has been applied in fault diagnosis, safety and risk engineering and structural reliability. A qualitative approach to the analysis of a failure (assuming the failure is a fuzzy element) is presented in [10] by Cai. Rather than defining success and failure in binary states, a probability of failure is introduced for each event.

Ishii concentrated on examining the life-cycle engineering design [11,12]. He proposed an effective representation scheme for assembly tasks by using a semantic network structure. The network is composed of components and subassemblies (nodes) and the relationships between the nodes (link). This representation supports to conduct service mode analysis (SMA) and life-cycle analysis since it expresses the relations of the objects in an assembly explicitly.

Several systems were developed in the literature based on the anticipated error propagation scenarios. Chang and DiCesare [2] proposed an algorithm for constructing and pruning failure propagation trees in manufacturing systems. The purpose of building a failure propagation tree is to identify possible failure causes and failure sources to allow for a planning process to recover from the error. A computer-aided monitoring system for assembly was developed by Abu-Hamdan and El-Gizawy [1]. The implemented expert system for detection is composed of task precondition and task execution monitoring parts and no assembly task is allowed to proceed unless the preceding task has been successfully completed. It was claimed that error propagation was totally eliminated with this approach. However, this system is inefficient when there is a failure among the detection system components.

The systems and methods discussed above depend on the prediction of failures by human experts so they do not cover most of the propagation situations.

An off-line error prediction, diagnosis and recovery technique was discussed previously in [4–7]. This method uses a commercial robotic assembly simulation software and Monte-Carlo simulation [13] to predict possible failures during the assembly operation. It also uses Bayesian Reasoning [14] to infer on the possible type of failure(s) and provide recovery logic based on the detected symptoms. The method is composed of five steps:

1 3D modeling of the assembly line using a commercial software package.
2 Prediction of the error cases by using Monte-Carlo simulation based on the statistical model of sensors, robots and products.
3 Off-line error diagnosis using Bayesian Reasoning.
4 Off-line generation of robust error recovery logic using Genetic Programming [15].
5 Downloading the generated codes to the system controller to prepare the system for automated recovery.

The main advantage of this approach is that it provides sufficient means of gathering information about the probable error situations during an assembly process and uses this information correctly to develop robust recovery plans. Since propagated errors cannot be predicted easily before the actual assembly process, this method can be used to predict and diagnose possible *propagated errors* in large-scale assembly systems.

Proposed Approach

The proposed system uses the previously discussed off-line error prediction, diagnosis and recovery method by building several software modules and linking them to establish a Virtual Factory. First, the assembly system is modeled in 3D. This model and the process parameters are inputted to the commercial software package. After that, possible error situations are identified by performing Monte Carlo simulation. These obtained situations are stored in a file in order to be used in the diagnosis stage. The next stage is generating recovery codes. In this step, Genetic Programming [15] is used to generate controller codes [4–6] in robot's language. Final stage is developing modules for the Virtual Factory.

The main module is the Virtual Assembly software, which is responsible for simulating the complete assembly process. The second module is the Virtual Detection module, which is used for detecting the component failures (gripper failures, sensor failures, etc.). Third module is the Virtual Diagnosis module for diagnosing errors. Fourth module is the Virtual Recovery module for applying the generated recovery codes. The factory structure is given in Fig. 1 and the detailed explanation of each module is as follows:

Structure of Virtual Factory. (a) *Virtual assembly software module*. This module is the commercial robotic simulation package and it includes the 3D model of the assembly system and assembly process codes. It also contains the realistic models of assembly robots, fixtures and products to simulate the process accurately. It is also possible to detect collision errors during the assembly process since this is an implemented feature in the package.

(b) *Virtual detection module*. Virtual Detection module is used for detecting the errors occurred during the assembly process. In assembly systems, there are two different monitoring types. The first type is called continuous monitoring, which a parameter is monitored continuously throughout the complete assembly process. As an example, torque/force sensors are checked continuously for collision detection. The second type is called discrete monitoring, which a parameter is monitored at certain steps of the assembly process. For example grasping sensors are checked during the part picking or releasing steps to ensure the process is completed successfully. Both types of monitoring are imple-
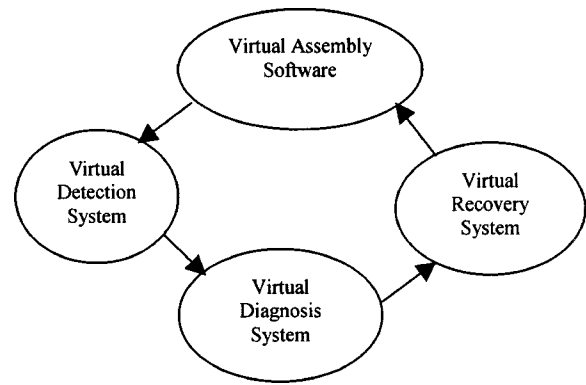

Fig. 1   Layout of the proposed virtual factory

mented in this module. A sensor array is defined and it contains information about each sensor's state. When an error is detected, current condition of the sensor array is passed to the diagnosis module to analyze the detected error.

(c) *Virtual diagnosis module*. This module uses the state of the sensor array to infer the possible failure reason. Based on the given symptom from the sensor array, the following Bayesian Reasoning formula is used to calculate a belief value for each failure type.

$$Bel(F_k) = \frac{P(Y_0 \backslash F_k) * P(F_k)}{\sum_{\forall l} P(Y_0 \backslash F_l) * P(F_l)} \tag{1}$$

In the above formula, the probability of having the specified symptom for each failure is calculated. Y_0 indicates the given symptom from the sensor array and F_k is the type of failure from the following failure array given in Table 1.

The failure, which has the highest belief value, is suggested as the reason after this calculation. However, in order to prevent incorrect automated recovery a threshold level for belief value is defined. If the diagnosed situation's belief value is greater than this threshold, system proceeds with automated recovery. If it is less than the threshold, system asks for user maintenance and the most possible failures are written into a log file.

When the system asks for user maintenance, an interactive reinforced diagnosis system is initialized. This subsystem is intended to be used on-line, after the system is prepared for the recovery process "off-line". It enables the user to input further data based on the manual diagnosis by entering the identified working and non-working components during the manual inspection process. Based on this additional data, the situation can be re-diagnosed. The usage of this sub-system is demonstrated in case studies.

(d) *Virtual recovery module*. This module is used for applying the recovery logic for the diagnosed failure. The outputs of the virtual diagnosis module are passed to this module and based on the failure type a strategy is followed for the recovery as shown in Table 2. Each strategy contains one or more recovery codes. The use of the appropriate code depends on the point where the error has been detected. Some failure types are dominant when multiple errors occurred (i.e., when a grasping failure occurred due to a flawed part and grasping, the system uses the recovery code to dispose the flawed part). Several rules are implemented to the system to recover from this type of possible scenarios.

Case Studies

A multi-station assembly system, which is responsible for mounting and welding workpieces together, is modeled using Workspace [16] and shown in Fig. 2.

Table 1   Failure array parameters

Failure Array = {d, e, f, g, h}
d= Grasping Error
e= Collision Error
f= Sensor Failure
g= Misplacement Error
h= Flawed Parts

Table 2   Failure recovery strategies

Failure Type	Strategy
Grasping Error	Try to grasp or release again
Collision Error	Use robust collision recovery
Sensor Failure	Call maintenance
Misplacement Error	Pickup the part and replace
Flawed Parts	Dispose the part and proceed

Fig. 2   Modeled assembly system

The system is composed of three IRB 6400-type industrial robots. The assembly process is as follows: At first, Rob-A picks up the cylindrical piece from the conveyor and inserts it into the hole of the second piece on the second station. Then, the welding robot (Rob-B) approaches the assembled two pieces and welds those two pieces together. After that, Rob-C picks up the welded piece and places it on the fixture and all of the parts are welded together by Rob-B. Finally the complete assembly is transferred on the conveyor by Rob-C. During the assembly operation, two inspection cameras are used for the verification process. These cameras are placed over the first station where Rob-A picks up the cylindrical piece and over the fixture respectively.

The elements in the modeled symptom array and their signal codes are given in the following Tables 3 and 4. Each symptom parameter, except the cameras, gets 0 or 1 depending on the signal feedback from the sensors. If there is a signal indicating an abnormal situation, then the associated parameter gets 1. For the inspection cameras the following codes are implemented as shown in Table 4.

The failure array is designed to provide information on the possible failure types as discussed before in Table 2. As shown in Table 5, all of the possible failure types of grasping and sensor failures are implemented to the model. In grasping, failure code 6 is different from 7 in the sequence of failures. Since sequence of failures is also important for the appropriate recovery these two codes are different from each other.

Several parameters are sampled from the assembly process. These parameters include robot repeatability for each robot, gripper reliability, gripper sensors reliability, sensor reliability for the inspection cameras and dimensional tolerances of each piece. Each parameter and its distribution type are given in Table 6. The values in the parenthesis indicate the mean and the standard values of the associated parameter:

Complete assembly process was simulated off-line 50000 times. The belief value threshold level for automated recovery is taken as 0.8. During this simulation process several types of error-propagation were observed.

Propagation Resulted in Part Jamming at the Fixture.   The actual reason for this problem is Rob-A did not pick the cylindrical part and this was not detected either by camera-1 over the first station or the Rob-A grasping sensor, since they are both malfunctioning. Besides, second camera over the fixture was also dead so it could not detect the incompletely assembled workpiece and because of this, that part could not be transferred to the conveyor. This type of error propagation caused part jamming at the fixture *during the next cycle*.

In this case, a collision error is detected at the fixture. The sensor array passes the information to the diagnosis module, indi-
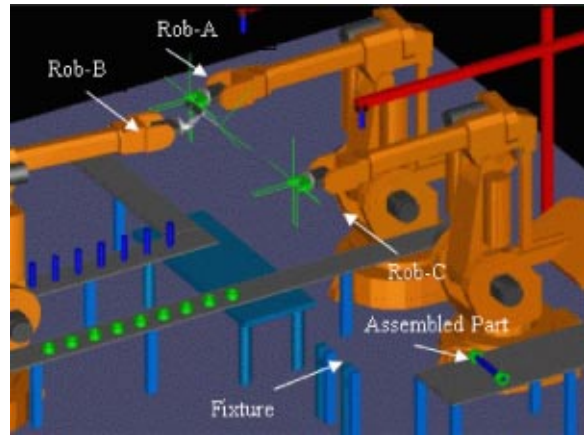
Table 3   Symptom array parameters

Symptom Array = {A, B, C, D, E,F}
A= Gripper Sensor of Rob-A
B= Torque/Force Sensor of Rob-A
C= Camera over the 1st Station
D= Gripper Sensor of Rob-B
E= Torque/Force Sensor of Rob-B
F= Camera over the fixture

Table 4   Camera codes

1= Workpiece not grasped or released
2= Incomplete Assembly
3= No parts
4= Part Jamming

Table 5   Failure codes

Grasping Failure Codes	Sensor Failure Codes
1: Rob-A Gripper Picking	1: Camera 1
2: Rob-A Gripper Releasing	2: Gripper A
3: Rob-C Gripper Picking	3: Camera 2
4: Rob-C Gripper Releasing	4: Gripper B
5: (Rob-A + Rob-C) Picking	5: Cam.1, Grip.A
6: Rob-A Pick + Rob-C Release	6: Cam.1, Cam.2
7: Rob-C Release + Rob-A Pick	7: Cam.1, Grip.B
8: (Rob-C + Rob-A) Release	8: Grip.A, Cam.2
	9: Grip.A, Grip.B
	10: Cam.2, Grip.B
	11: Cam.1, Cam.2, Grip.A
	12: Cam.1, Grip.A, Grip.B
	13: Cam.1, Cam.2, Grip.B
	14: Grip.A, Grip.B, Cam.2
	15: All of the sensors

Collision Failure Codes
1: Collision at Station 1
2: Collision at Station 2
3: Collision at Fixture

Table 6 Sampled parameters and values

Parameter:	Nominal Value / Distribution:
Robot Repeatability	0.02 mm/ Normal (0, 0.067 mm)
Grasping Ability	Uniform (0.9)
Gripper Sensor	Uniform (0.99)
Inspection Camera	Uniform (0.99)
Peg	49.92 mm / Normal (0, 0.0106)
Hole	50.3 mm / Normal (0, 0169)

Table 7 Output of virtual diagnosis module

Diagnosed Failure:	
Belief Value	0.595
Grasping Failure	Rob-C Gripper release failure
Sensor Failures:	Rob-C Gripper + Camera 2

Table 8 Output of interactive reinforced diagnosis

Diagnosed Failure:	
Belief Value	0.8817
Grasping Failure	Rob-A Gripper picking failure
Sensor Failures:	Cam.1, Gripper 1, Cam.2

cating that torque/force sensor of Rob-C detected collision. The state of the assembly system is shown in Fig. 3. At this point, it is difficult for a human expert to diagnose the situation. The only input is from the torque/force sensor and there are other parameters to check thus there may be several different reasons for this problem.

Information from the sensor array is diagnosed by the Virtual Diagnosis module. The diagnosed failure reason and its belief value are given in Table 7. As it can be seen from the table, the calculated belief value is less than the threshold value. If this case arises during the actual assembly operation, the system stops and asks for user maintenance and interactive reinforced diagnosis system is initiated. At this step, further obtained data can be inputted to the system. In order to simulate the situation, it is assumed that Rob-C gripper has been checked according to the initially proposed diagnose failure and it was found out that component is working properly. Re-diagnosing based on this further data revealed another reason of failure as shown in Table 8.

As it can be observed from Table 8, this time the correct reason for the failure has been found. The belief value is 0.8817. Although it is less than the threshold, the system is already in manual recovery mode so it can be recovered.

This case study revealed the fact that although propagated errors occur in less likelihood, they cannot be avoided. Although the modeled system is composed of relatively fewer components when compared to the large-scale auto-body assembly or consumer electronics lines, it is still difficult to analyze the system. Therefore, the developed Virtual Factory aids in prediction, diagnosis and recovery of the complex errors, which may be propagated during the assembly process.

Conclusions

Large-scale robotic assembly systems are used in industry extensively. However, these systems are composed of many components and it is not possible to monitor all the parameters of these components during the assembly process. For example, an unde-
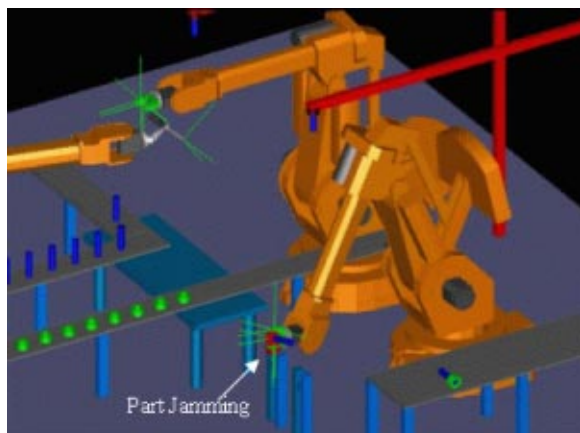
tected error in upstream of the line can cause a detectable error in further downstream of a line which is called the "*error-propagation.*"

The diagnosis and recovery of propagated errors are complex since they cannot be predicted easily prior to the operation of the assembly line. Several methods are used in the literature to predict the possible propagation of undetected errors using failure propagation trees, failure mode and effect analysis or using fuzzy logic. However, these methods are not adequate since they do not incorporate the 3D model of the system and they do not cover all of the possible error scenarios.

The advancements in both hardware and software technology revealed the concept of using *Virtual Factories* before building the real-lines. These systems can simulate the process repetitively in less time to predict the unpredictable, propagated errors before they occur. They can also be used for developing and verifying diagnosis and recovery logic.

A *Virtual Factory* was developed to predict and diagnose this type of failures before they happen. Several modules are developed as parts of this factory. The system is capable of detecting, diagnosing and recovering possible failures, which may occur during the real process.

In order to demonstrate the validity and the effectiveness of the proposed system, a multi-station assembly system is modeled and a previously discussed "off-line prediction and recovery" method was applied. The obtained results showed that the method is capable of predicting propagated errors, which are too complex to solve for a human expert. Following advantages of using a Virtual Factory are identified after conducting case studies:

- The developed Virtual Factory is capable of predicting and diagnosing propagated errors, which may cause problems although their likelihood of occurrence may be less.
- The Virtual Diagnosis module can be embedded into the real line and used for automated diagnosis.
- The developed interactive reinforced diagnosis system can help reducing downtime of the assembly system when manual diagnosis is required.

It is believed that this approach will decrease the costly downtime for failure inspection and recovery. The future work will involve examining the complex failure scenarios, which require the coordination of multiple agents for recovery and automated generation of recovery logic for this type of failures.

Fig. 3 Part jamming at the fixture

References

[1] Abu-Hamdan, M. G., and El-Gizawy, A. S., 1997, "Computer Aided Monitoring System for Flexible Assembly Operations," Computers in Industry, **34**, pp. 1–10.

[2] Chang, S. J., DiCesare, F., and Goldbogen, G., 1991, "Failure Propagation Trees for Diagnosis in Manufacturing Systems," IEEE Trans. Syst. Man Cybern., **21**, No. 4, pp. 767–776.

[3] Luxhoj, J. T., Rlis, J. O., and Thorsteinsson, U., 1997, "Trends and Perspectives in Industrial Maintenance Management," J. Manuf. Syst., **16**, No. 6

[4] Baydar, C., and Saitou, K., 2000, "Off-line Error Recovery Logic Synthesis in Automated Assembly Lines by Using Genetic Programming," *Proceedings of the 2000 Japan-USA Symposium on Flexible Automation*, Ann Arbor, MI.

[5] Baydar. C., and Saitou, K., 2000, "A Genetic Programming Framework for Error Recovery in Robotic Assembly Systems (extended abstract)," *Proceedings of the 2000 Genetic and Evolutionary Computation Conference* (GECCO-2000), Las Vegas, NV.

[6] Baydar, C., and Saitou, K., 2000, "Generation of Robust Recovery Logic in Assembly Systems Using Multi-Level Optimization and Genetic Programming," *Proceedings of the ASME-DETC2000/CIE Conference*, Baltimore, MD.

[7] Baydar, C., and Saitou, K., 2001, "Off-Line Error Prediction, Diagnosis and Recovery Using Virtual Assembly Systems," *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, Seoul, Korea.

[8] Khodabandehloo, K., 1996, "Analyses of Robot Systems Using Fault and Event Trees: Case Studies," Reliability Eng. Sys. Safety, **53**, pp. 247–264.

[9] Wirth, R., Berthold, B., Kramer, A., and Peter, G., 1996, "Knowledge Based Support of System Analysis for the Analysis of Failure Modes and Effects," Eng. Applic. Artif. Intell., **9**, No. 3, pp. 219–229.

[10] Cai, K. Y., 1996, "System Failure Engineering and Fuzzy Methodology: An Introductory Overview," Fuzzy Sets Syst., **83**, pp. 113–133.

[11] Ishii, K., 1995, "Life Cycle Engineering Design," Transactions of the ASME, J. Mech. Des. **117**, pp. 42–47.

[12] Eubanks, C. F., Ishii, K., 1993, "A1 methods for Life-Cycle Serviceability Design of Mechanical Systems," Artif. Intell. Eng., **8**, pp. 127–140.

[13] Kalos, M. H., Whitlock, P. A., 1986, *Monte Carlo Methods, Volume I: Basics*, John Wiley & Sons.

[14] Jensen, F., 1996, *An Introduction to Bayesian Networks*, Springer-Verlag.

[15] Koza, J., 1992, *Genetic Programming: On the Programming of Computers by Natural Selection*, MIT Press, Cambridge, MA.

[16] Workspace v.5 Educational User Guide Manual, 2000.