Proceedings of DETC'01
2001 ASME Design Engineering and Technical Conferences
and Computers and Information in Engineering Conference
September 9-12, 2001, Pittsburgh, Pennsylvania

DETC2001/ CIE-21261

PREDICTION AND DIAGNOSIS OF PROPAGATED ERRORS IN ASSEMBLY SYSTEMS USING VIRTUAL FACTORIES

Cem M. BAYDAR
(cbaydar@umich.edu)

Kazuhiro SAITOU[*]
(kazu@umich.edu)

Department of Mechanical Engineering
University of Michigan
Ann Arbor, MI 48109-2125, USA

station level automated assembly system [7] showed that, this approach is capable of identifying the possible failures and their likelihood as well as their 3D geometrical state. 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 to predict unexpected propagated errors before the actual assembly operation takes place thus providing efficient means for diagnosis and recovery for these types of failures.

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]. 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. WIFA aims to improve the current state of art of FMEA by knowledge-based support of the user. Each FMEA in WIFA should be performed in the following steps [9]:

1. Specify the system architecture,
2. Identify critical system elements for which the FMEA has to be conducted,
3. For each system element specify all its functions,
4. For each system element specify all its failures,
5. Link functions to functional structure,
6. Specify fault trees,
7. Fill out the remainder of the form, i.e. risk priority numbers (RPNs), actions and responsibilities, etc.

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. It is believed that a life-cycle evaluation tool requires a flexible set of data that contains pertinent information about the candidate design [11]. Ishii 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 enables to conduct service mode analysis (SMA) and life-cycle analysis since it expresses the relations of the objects in an assembly explicitly. Eubanks and Ishii [12] also used artificial intelligence methods to infer required repair labor actions from the design description and analyze life-cycle service costs.

Several systems were developed in the literature based on the anticipated error propagation scenarios. Chang and DiCesare [2] proposed and 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 is claimed that error propagation 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 logic of the proposed approach is summarized in Fig.1. 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. Unexpected errors, which may be *unforeseen* to human design experts before the operation of the line, can be predicted easily and outcomes of this prediction with providing necessary recovery logic will decrease the costly downtime for these types of systems. 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.

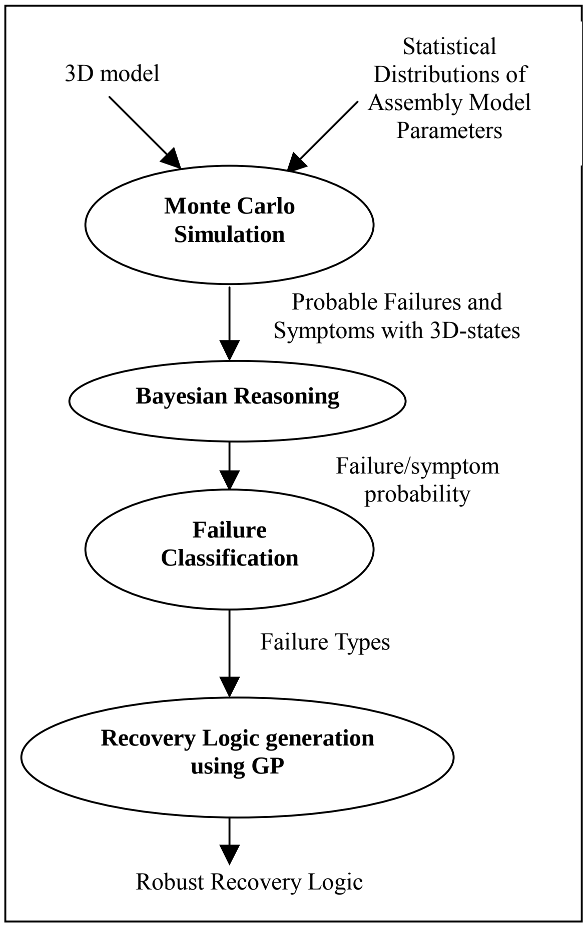

3D model
Statistical Distributions of Assembly Model Parameters
Monte Carlo Simulation
Probable Failures and Symptoms with 3D-states
Bayesian Reasoning
Failure/symptom probability
Failure Classification
Failure Types
Recovery Logic generation using GP
Robust Recovery Logic

Figure 1: Working mechanism of the off-line method.

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 from 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 Figure 2 and the detailed explanation of each module is as follows:

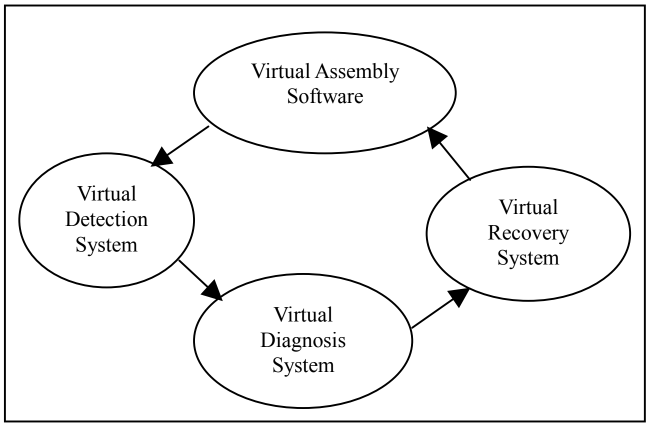


Figure 2: Layout of the Proposed Virtual Factory.

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 implemented 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.

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

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

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

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 system 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.

This module can also be embedded to the real line by using a computer system.

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.

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

Each strategy contains one or more recovery codes. The use of the appropriate code depends on the point where error has been detected. Another advantage of this module is that the generated codes can be downloaded to the assembly controller of the real line.

A multi-robotic system is modeled and the results are evaluated in the case studies section. Based on these results, the advantages of using this system are realized as follows:

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

CASE STUDIES

A multi-station assembly system, which is responsible from mounting and welding two workpieces together, is modeled using Workspace [16] and shown in Fig. 3.

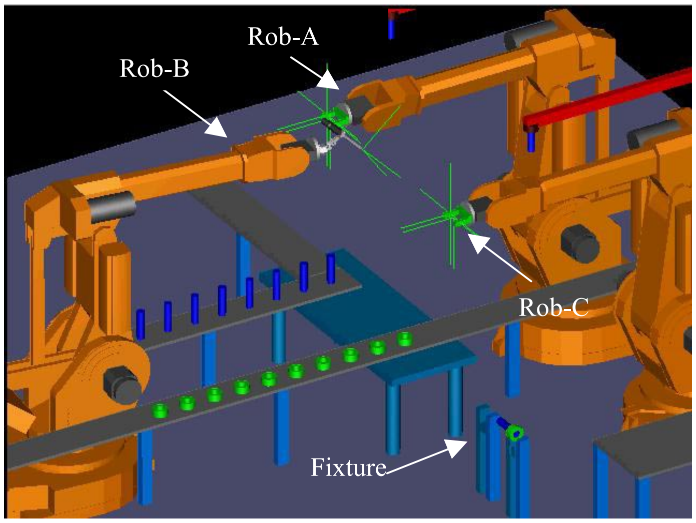


Figure 3: Modeled Assembly System.

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 to 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 process, two inspection cameras are used to verify the assembly process. These cameras are placed over the first station where Rob-A picks up the cylindrical piece and over the fixture respectively. The situation after the complete process is given in Fig. 4.

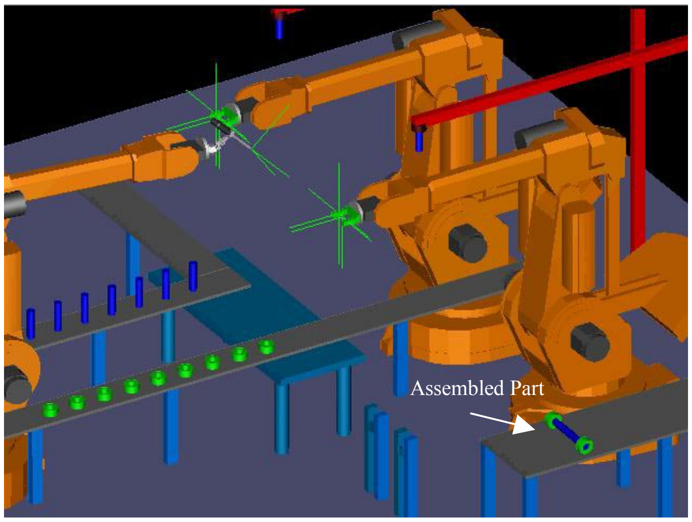


Figure 4: Completed Assembly.

The elements in the modeled symptom array and their signal codes are given in the following tables:

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

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:

Table 4: Camera Codes.

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

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.

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

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

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)

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. Two examples for these types of failures are discussed here.

a) 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 is 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, indicating that torque/force sensor of Rob-C detected collision. The state of the assembly system is shown in Fig.5.

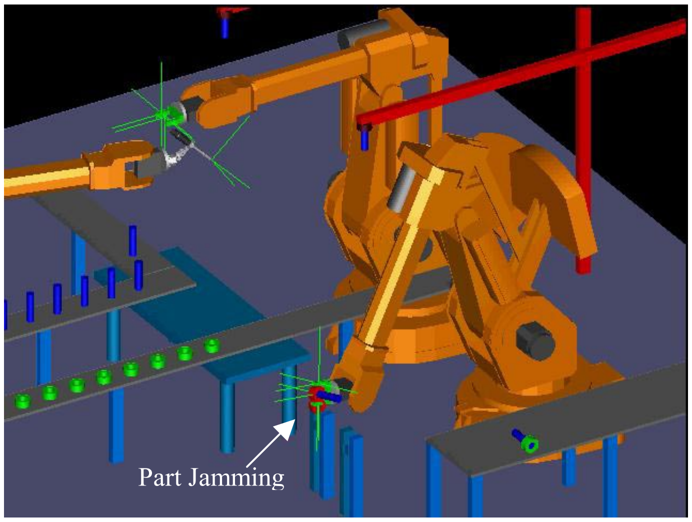


Figure 5: Part jamming at the fixture.

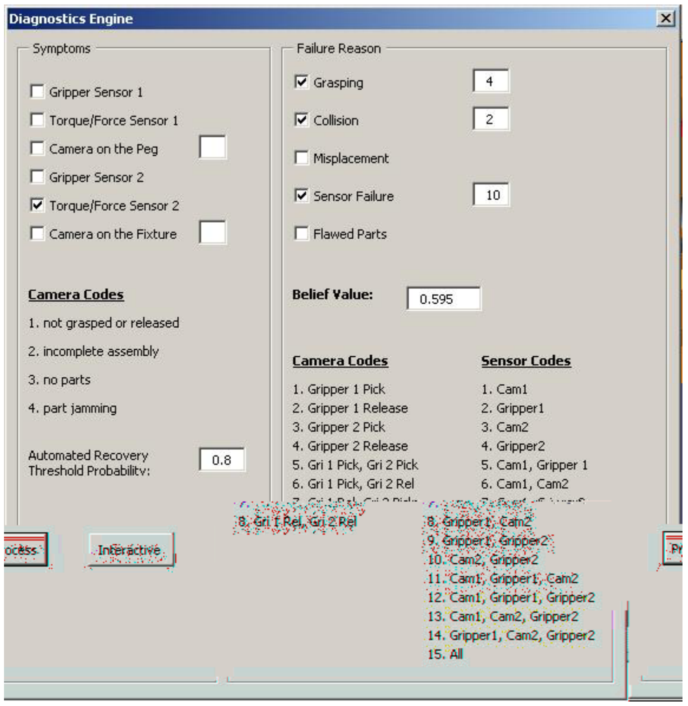
At this point, it is difficult to diagnose the situation for a human expert. The only input is from the torque/force sensor and there are other parameters to check thus there may be several different reasons for this problem.

Information from the sensor array is diagnosed by the Virtual Diagnosis module as shown in Fig.6. The diagnosed failure reason and its belief value are as follows:

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

As it can be seen from the table above, the calculated belief value is less than the threshold value. At this point 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 that component is working properly. Re-diagnosing based on this further data revealed another reason of failure as shown in Fig.7 and Table 8.



Diagnostics Engine

Symptoms
- [ ] Gripper Sensor 1
- [ ] Torque/Force Sensor 1
- [ ] Camera on the Peg
- [ ] Gripper Sensor 2
- [x] Torque/Force Sensor 2
- [ ] Camera on the Fixture

Camera Codes
1. not grasped or released
2. incomplete assembly
3. no parts
4. part jamming

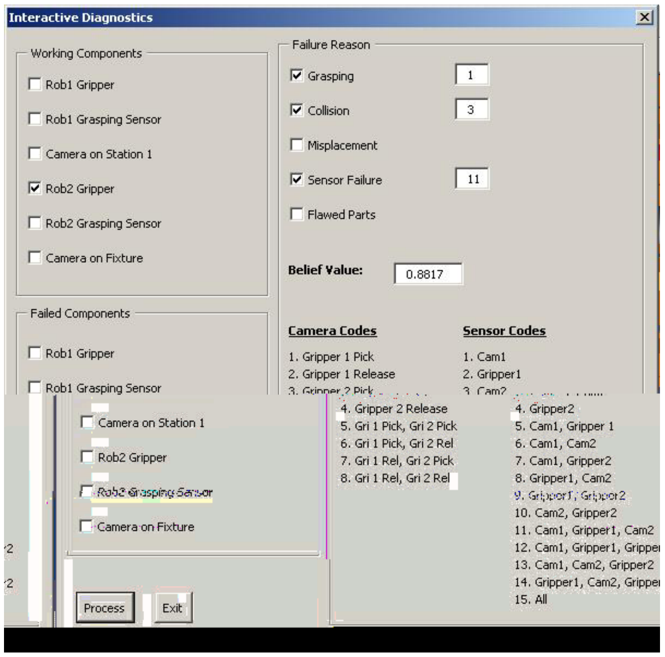
Automated Recovery Threshold Probability: 0.8

Failure Reason
- [x] Grasping 4
- [x] Collision 2
- [ ] Misplacement
- [x] Sensor Failure 10
- [ ] Flawed Parts

Belief Value: 0.595

Camera Codes	Sensor Codes
1. Gripper 1 Pick	1. Cam1
2. Gripper 1 Release	2. Gripper1
3. Gripper 2 Pick	3. Cam2
4. Gripper 2 Release	4. Gripper2
5. Gri 1 Pick, Gri 2 Pick	5. Cam1, Gripper 1
6. Gri 1 Pick, Gri 2 Rel	6. Cam1, Cam2
8. Gri 1 Rel, Gri 2 Rel	8. Gripper1, Cam2
	9. Gripper1, Gripper2
	10. Cam2, Gripper2
	11. Cam1, Gripper1, Cam2
	12. Cam1, Gripper1, Gripper2
	13. Cam1, Cam2, Gripper2
	14. Gripper1, Cam2, Gripper2
	15. All

Figure 6: Virtual Diagnosis Module.



Interactive Diagnostics

Working Components
- [ ] Rob1 Gripper
- [ ] Rob1 Grasping Sensor
- [ ] Camera on Station 1
- [x] Rob2 Gripper
- [ ] Rob2 Grasping Sensor
- [ ] Camera on Fixture

Failed Components
- [ ] Rob1 Gripper
- [ ] Rob1 Grasping Sensor
- [ ] Camera on Station 1
- [ ] Rob2 Gripper
- [ ] Rob2 Grasping Sensor
- [ ] Camera on Fixture

Failure Reason
- [x] Grasping 1
- [x] Collision 3
- [ ] Misplacement
- [x] Sensor Failure 11
- [ ] Flawed Parts

Belief Value: 0.8817

Camera Codes	Sensor Codes
1. Gripper 1 Pick	1. Cam1
2. Gripper 1 Release	2. Gripper1
3. Gripper 2 Pick	3. Cam2
4. Gripper 2 Release	4. Gripper2
5. Gri 1 Pick, Gri 2 Pick	5. Cam1, Gripper 1
6. Gri 1 Pick, Gri 2 Rel	6. Cam1, Cam2
7. Gri 1 Rel, Gri 2 Pick	7. Cam1, Gripper2
8. Gri 1 Rel, Gri 2 Rel	8. Gripper1, Cam2
	9. Gripper1, Gripper2
	10. Cam2, Gripper2
	11. Cam1, Gripper1, Cam2
	12. Cam1, Gripper1, Gripper
	13. Cam1, Cam2, Gripper2
	14. Gripper1, Cam2, Gripper
	15. All

Figure 7: Interactive Reinforced Diagnosis System

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

As it can be observed from the above table, 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.

b) Propagation resulted in part jamming at station 1:

In this case, a jamming error is reported at station-1 by the torque/force sensor of Rob-A. This time the reason for this failure is Rob-A did not release the piece in its gripper and camera-1 and Rob-A gripper sensor is malfunctioning to detect this initial error. Furthermore, this error is propagated with the camera-2 failure to detect the incomplete assembly. When the next cycle starts, the next workpiece collides with the previous part held still in Rob-A's gripper.

It is realized that although the malfunctioning components are all same with the previous case, the error has been detected at a different place only because there is a part releasing error rather than picking. The 3D state of the system is also different compared to the previous case. The failure situation is given in Fig.8.

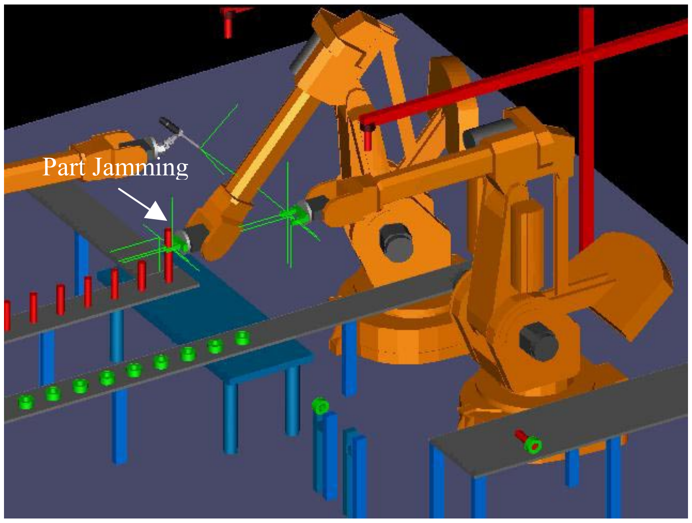
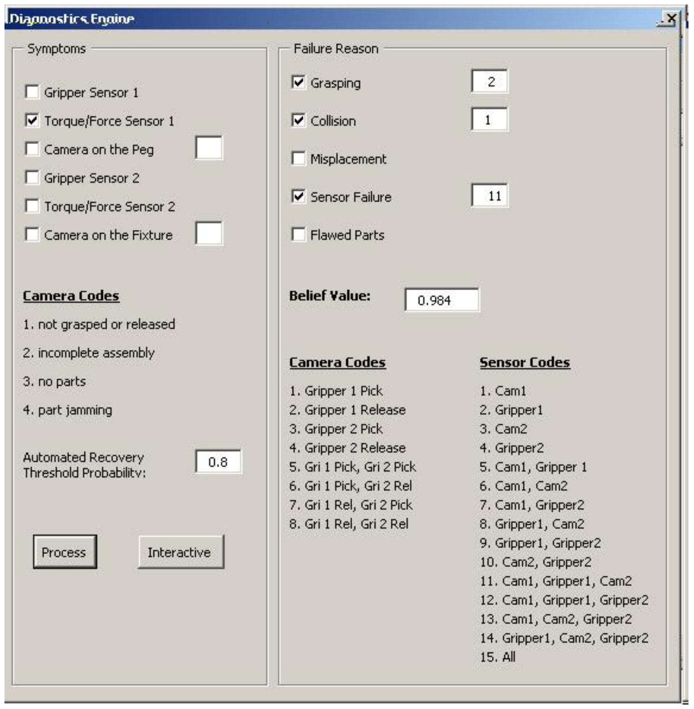


Figure 8: Part jamming at the 1st station.

Information from the sensor array is diagnosed by the Virtual Diagnosis module as shown in Fig.9. The diagnosed failure reason and its belief value are as follows:

Table 9: Output of Virtual Diagnosis Module

Diagnosed Failure:	
Belief Value	0.984
Grasping Failure	Rob-A Gripper release failure
Sensor Failures:	Cam.1, Rob-A Gripper, Cam.2



Diagnostics Engine

Symptoms
- ☐ Gripper Sensor 1
- ☑ Torque/Force Sensor 1
- ☐ Camera on the Peg
- ☐ Gripper Sensor 2
- ☐ Torque/Force Sensor 2
- ☐ Camera on the Fixture

Camera Codes
1. not grasped or released
2. incomplete assembly
3. no parts
4. part jamming

Automated Recovery Threshold Probability: 0.8

Process | Interactive

Failure Reason
- ☑ Grasping 2
- ☑ Collision 1
- ☐ Misplacement
- ☑ Sensor Failure 11
- ☐ Flawed Parts

Belief Value: 0.984

Camera Codes	Sensor Codes
1. Gripper 1 Pick	1. Cam1
2. Gripper 1 Release	2. Gripper1
3. Gripper 2 Pick	3. Cam2
4. Gripper 2 Release	4. Gripper2
5. Gri 1 Pick, Gri 2 Pick	5. Cam1, Gripper 1
6. Gri 1 Pick, Gri 2 Rel	6. Cam1, Cam2
7. Gri 1 Rel, Gri 2 Pick	7. Cam1, Gripper2
8. Gri 1 Rel, Gri 2 Rel	8. Gripper1, Cam2
	9. Gripper1, Gripper2
	10. Cam2, Gripper2
	11. Cam1, Gripper1, Cam2
	12. Cam1, Gripper1, Gripper2
	13. Cam1, Cam2, Gripper2
	14. Gripper1, Cam2, Gripper2
	15. All

Figure 9: Virtual Diagnosis Output

This time the proposed failure reason's belief value is greater than the threshold so the system proceeds with automated recovery. The suggested failure recovery strategy is calling the maintenance person to replace the malfunctioning sensors and components.

These case studies revealed the fact that although propagated errors occur in less likelihood, they cannot be avoided. Although the modeled system is composed of relatively less number of 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 on 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 undetected 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 and it is efficient to diagnose even the failure case is 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.
- Virtual Diagnosis module can be embedded to 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.

ACKNOWLEDGMENTS

The authors are grateful for the technical support of Flow Sofware Inc. for the Workspace^{TM} software. The authors also acknowledge Rackham Graduate School of Studies at The University of Michigan for supporting this work.

REFERENCES

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

[2] Chang S. J, DiCesare F., Goldbogen, G., (1991), *"Failure Propagation Trees for diagnosis in manufacturing systems"*, IEEE Transactions on System Man Cybernetics, v.21 n.4, pp. 767-776.

[3] Luxhoj J.T., Riis J. O., Thorsteinsson U. (1997), "Trends and Perspectives in Industrial Maintenance Management", Journal of Manufacturing Systems, Vol.16, No.6.

[4] Baydar C., 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., 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., 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 Engineering and System Safety 53, 247-264.

[9] Wirth R., Berthold B., Kramer A., Peter G. (1996), "Knowledge Based Support of System Analysis for the Analysis of Failure Modes and Effects", Engineering Applications in Artificial Intelligence 9(3), 219-229.

[10] Cai K. Y. (1996), "System Failure Engineering and Fuzzy Methodology: An Introductory Overview", Fuzzy Sets and Systems 83, 113-133.

[11] Ishii K. (1995), "Life Cycle Engineering Design", Transactions of the ASME 117, 42-47.

[12] Eubanks C.F., Ishii K. (1993), *"AI methods for Life-Cycle Serviceability Design of Mechanical Systems"*, Artificial Intelligence in Engineering 8, 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).