This paper focuses on the requirements for achieving automated diagnosis and error recovery in a robotic assembly cell. Untended robust operation is the goal which is discussed, and the use of a knowledge-base oriented diagnostic/control system to support on-line error recovery is advocated. It is proposed that error recovery attempts should be made, not only based on information gathered by system sensors, but also by using additional cell-specific knowledge pertaining to recovery strategies. Recovery attempts are to be made even in the absence of complete diagnosis, in order to gather additional information. A control structure developed to implement this approach is presented.

1. Introduction

Robot-based manufacturing systems present a domain which is inherently complex and is characterized by a high degree of interdependence among the devices which make up a manufacturing system. The control and coordination of these devices is a task which demands strict attention to detail, as well as an ability to understand the broader goals being pursued by a system. Traditional algorithm-based control of manufacturing systems has shown a good capability for handling the many details involved, however, deficiencies are evident when it comes to incorporating a 'higher-level' understanding of what is desired of a system. Algorithm-based control systems have no problem dealing with well-ordered, highly structured environments. However, robotic workcells tend to be very dynamic, and it requires great efforts and sometimes severe restrictions to ensure that a robotic workcell is predictable enough to function reliably while being controlled by a control algorithm.

Researchers are addressing the limitations of algorithm-based control through the application of various artificial intelligence (AI) techniques. These techniques present improvements over algorithmic methods, and are said to be directed toward creating machine intelligence. Many AI techniques are being explored, but the technique which is arguably the most mature involves the use of so-called 'expert systems' or 'knowledge-based' systems. Knowledge-based systems provide a powerful tool for manipulating information and knowledge through the use of inferencing techniques. Research into the application of these techniques has been widespread; however, much remains to be done.

One of the criticisms of robotic workcells is that they tend to be fairly sensitive to change, and thus are often of questionable reliability. Part of the cause of this problem is that the control algorithms which drive these systems are very rigid and do not respond well to 'errors' which occur during production. This rigidity is not as necessary in knowledge-based control systems, and thus increased robustness is expected of robotic workcells which incorporate knowledge-based control methods.
Current robotic workcells are highly automated systems which, during normal operation, require little or no human assistance. (This excludes, of course, those systems specifically designed as combined robot/human cells.) These highly automated systems race merrily along until, at some point during production, an error occurs. Many modern systems have some limited ability to function while impaired, or make some adjustment in operation for well known failures. However, when faced with an unexpected event or error condition, most simply shut down and start setting off alarms and lights. The system then is attended to by an 'operator' who attempts to diagnose what has failed and take whatever corrective action is thought to be appropriate. Researchers have begun exploring the use of knowledge-based systems to incorporate automated diagnosis into today's systems and, in some cases, even recommend the corrective actions to be taken by the operator.

There is a fundamental problem though. As systems become more complex and more capable, operators have less experience dealing with errors and trouble-shooting. They become less qualified to take corrective actions. At the extreme point of this situation, operators become little more than people assigned to read the right manual when an error light comes on and then push the right button. It is at this point, or actually before this point, when the system itself should be given the responsibility (and authority) to take corrective action on its own. To obtain this ability, systems must be provided with on-line automated diagnostics and error recovery capability.

Research concerning these issues tends to divide into two fairly distinct fields, the artificial intelligence field of automated diagnosis, and the robotics field of automated error recovery. There is overlap, and it will be discussed, however, the dominant work in each field is fairly independent of the other domain. Thus, this discussion will first focus on the relationship of the proposed research to each of the fields individually and will then discuss unifying factors.

2. Diagnostics

Research into automated diagnosis has historically been dominated by two schools of thought. One school has advocated so-called 'deep reasoning' as a tool for diagnostic reasoning, the other has stressed 'shallow reasoning'. In the field of deep reasoning, the recognized leader is Randall Davis (Davis et al. 1988, Davis 1982, 1983, 1984) of MIT. Davis advocates diagnostics through the use of models of the system being considered. These models are used to describe the structure, function, and behaviour of the system, and rely on 'reasoning from first principles'. Many of the models come from the various abstraction techniques used by engineers to represent their designs. For instance, computer designers use circuit diagrams to represent the connections between the physical components they assemble to create a computer. Civil engineers use 'ideal truss' diagrams to model bridges and other support structures.

These and other engineering modelling tools are used to capture vital technical knowledge while discarding large amounts of 'irrelevant' information. This model-based reasoning approach is very popular and is considered very powerful. Davis's work has concentrated on electronic trouble-shooting, and has been demonstrated to be very proficient in this area. The real disadvantage that this approach has is that deep systems tend to be very slow, and thus not suitable for on-line decision-making.

The second approach, shallow reasoning, has been more commercially successful, and is the basis for virtually all successful 'expert system' projects in industry. The
feature that defines this approach is that the knowledge used to conduct diagnosis is usually very specific to the system being considered. Many of these systems take the form of a rule-based knowledge base which uses forward or backward chaining to analyze the state of the system. These rules normally are of an ‘IF-THEN’ structure and are based on historical information about the system or heuristically determined relationships provided by a human expert. This type of reasoning is criticized by some as being too limited since the relationships must be explicitly stated. However, as Chandrasekaran and Mittal (1983) point out, deficiencies in a compiled system often result from inadequacies in the underlying knowledge, not inadequacies in the implementation technique. The real advantages of this approach involve the speed at which today’s computers can deal with this type of system.

As would probably be expected, most research being conducted today is focusing on the combination of deep and shallow reasoning into an approach called simply the ‘hybrid reasoning’ approach. Research has shown that hybrid reasoning can perform well in the diagnosis of complicated systems (Bublin and Kashyap 1989, Yang and Chang 1989, Graham and Alexander 1990, Borchelt 1991). Most researchers in the field of diagnostics are content to obtain a diagnosis and then stop. The field itself is primarily oriented toward that goal. Once the diagnosis is made, then the rest is left up to the user of diagnostic system. This occurs because most diagnostic systems are viewed as tools to be used ‘off-line’ to help fix a system or product. Even those researchers who are concerned with on-line diagnosis tend to rely on system operators to recover from the errors (Bublin and Kashyap 1989, Yang and Chang 1989, Graham and Alexander 1990).

3. Error recovery

The field of error recovery is another area, and it has its own viewpoint. Normally, error recovery is something which depends heavily on the environment in which the error occurred and on the specific capabilities of the system under study. As a result, this part of the discussion will focus on error recovery in robotic work-cells, and specifically on errors which are related to ‘process’ or ‘operational’ errors.

This type of error normally results from equipment problems, alignment failures, missing parts etc. Some of these errors can be eliminated by the use of sophisticated sensors (vision systems, tactile pads, etc.), however, the expense of these sensors cannot always be justified.

In order to make automated systems truly productive, it is necessary that they be capable of recovering from process errors unaided. This fact has been recognized by the field of ‘error recovery’, but authors who write in this area have tended to simply address error recovery techniques, and have not utilized any but the most basic diagnostic approaches. There are many descriptions of various robotic workcells and automated manufacturing systems which incorporate some type of error recovery technique. However, most of these authors focus primarily on the specific characteristics of their system rather than the broader ‘techniques’ used. In addition, error recovery in most implementations is very limited and included only for the sake of completeness. Two authors who have written in some detail about issues related to robotic error recovery are Maria Gini of the University of Minnesota, and M. H. Lee of University College of Wales, Aberystwyth. These authors describe the techniques that they have used to implement error recovery and focus primarily on the requirements and structures used for that purpose.
Gini has conceived of an approach which concentrates primarily on developing what are referred to as 'augmented programs' for the robot workcell (Gini and Gini 1983, 1986, Gini 1987). This approach takes the robot's 'manipulator-level' program, developed for a given assembly task, and modifies it to add in checkpoints, interlocks, and other features which help to accomplish error recognition and recovery. When an error occurs, the augmented program turns operational control over to a 'recoverer' which attempts to back up to where the error is believed to have occurred and create a 'patch' program which modifies the augmented program and attempts to go on with the assembly. This seems to be a powerful technique and has been implemented in a simple robotic workcell conducting a pick and place task (Smith and Gini 1992).

One of the deficiencies of Gini's work is that it fails to specify how diagnostics is being achieved. What can be gathered from the descriptions given is that a system similar to what is called a 'fault dictionary' is being used. Errors are assumed to be fairly predictable in nature and thus the sensors in the system are used to detect the error and match it with a defined error in memory. Once this is done (i.e. a diagnosis has been made) the error recovery portion takes over. This work mentions 'general strategies' and seems to identify 'types' of errors, but the 'recoverer' uses an ordered set of recovery methods which are based on the anticipated types of errors. It first tries clearing away and discarding the current part, then asks an operator to clear any obstructions (if it thinks there is one), picks up (or requests) a replacement part, and tries again. While this fixed response undoubtedly works well in the testbed for which it was designed, such a 'canned' response is probably not flexible enough for a more complex environment.

Lee et al. have also focused interest on a robotic workcell (Lee et al. 1983–1985, 1990, Hardy et al. 1988). Much of this work addresses the issues of sensory monitoring and software structure. A 'framework' named AFFIRM (Aberystwyth Framework For Industrial Robot Monitoring) was created, based on what is referred to as '... the artificial intelligence "frames" formalism'. Essentially, AFFIRM sets forth an action and expects a specific consequence. When it doesn't get that consequence, it declares an error.

Their diagnostic approach is not defined in any significant detail. They discuss the 'considerable potential knowledge' available for diagnosis and stress the importance of having confidence in the diagnosis before moving on to a specific error recovery attempt. However, rather than discussing their diagnostics, they simply say that 'This has many similarities with that branch of Artificial Intelligence know as Expert system research'. Although little detail is given about the actual error recovery structure, earlier papers (Lee et al. 1983, 1984) point out some intriguing ideas. In those papers, they discuss the concept of defining a 'spectrum of recovery processes'. Unfortunately, further mention of this idea is not found in later writings. Despite the fact that their software (AFFIRM) is not commercially available, it is based on frames and thus a similar structure can be developed using commercially available packages.

This tendency to rely on a diagnostic system to come up with a specific diagnosis and then begin an error recovery attempt is simply common sense. If the problem can be specified, it makes it much easier to pick a solution which is likely to work. The problem is that, due to the nature of the robotic environment, it is often impossible to obtain a specific fault diagnosis. Even in systems which are heavily 'sensed' and well defined, errors occur which are too complicated or compounded to allow for a specific failure to be recognized. The most common response in this situation is to simply not try anything.
4. Unifying diagnostics and error recovery

AI-based diagnostics can provide powerful insights into the workings of a system, and should be used to drive the error recovery process. Rather than insisting on a specific complete diagnosis, error recovery attempts should be made even in the absence of complete diagnosis. Often, a human operator (or repair technician) will not know exactly what is wrong with a system when they first consider it. Rather, they have suspicions and they begin 'trouble-shooting' to try to gain more information. They try easy things first, and move on to more complex tests as easier ones fail. They try fixing what is inexpensive to fix, and see if that helps. This gradual probing and testing is what automated systems should be allowed to do.

Past research into error recovery has emphasized the necessity of having a high degree of confidence in the diagnostic conclusion before beginning any error recovery procedures (Lee et al. 1985). The authors propose that it is not always possible to know exactly what has gone wrong, and that, even in the absence of complete knowledge, the system must be capable of deciding on an action and continuing to operate.

To be truly automatic, a system must be able to decide upon recovery alternatives based on information other than simply that available in its sensor data. This additional information may be based, among other possibilities, on the varying degrees of difficulty of each recovery task, the cost of attempting the various tasks, or even the statistical measures of the likelihood of success for a given recovery method.

The fundamental point is that the system must have the authority to try something even if the cause of the error is unknown. A technique must be used which will allow an automated system to respond to unexpected events like a system operator would. The system must be able to try out the different options available to it, and learn from those attempts. By maintaining knowledge about what has been tried, the system can begin to eliminate certain possibilities and finally narrow in on what is actually going wrong. During the completion of these tasks, the problem itself may be resolved. At the very least, some pattern of failures will appear which provide more information about the characteristics of the error.

Once it has been agreed that the system will be allowed to respond, even in the absence of complete knowledge, the task becomes identifying what factors will be considered when choosing between the specific error recovery options. The important factors will change from system to system and, even within the same system, from one moment in the production task to the next. This type of reasoning, shifting the rules and guidelines that apply for varying contexts and situations, avoids the rigidity seen in 'canned' responses, and allows recovery in complex dynamic environments.

Previous research conducted by the authors at the University of Missouri-Rolla has demonstrated the usefulness of hybrid reasoning in supporting the diagnosis and recovery efforts of an automated robotic cell. During that research, a robotic assembly cell was created which assembled a simple mechanical device. The techniques developed during the design and implementation of this cell are described in the following paragraphs.

To support the error recovery task, it was necessary to provide a knowledge-based diagnostic system which was a hybrid of object-oriented representations (model-based reasoning) and production rules (rule-based reasoning). The system was created using NEXPERT Object, and Microsoft C. This hybrid knowledge structure allowed the system to maintain information about the status of the system in the form of an object-oriented model of the cell, while reasoning about the cell using rules and guidelines defined by the system designer. The 'structural' and some 'functional' information
about the cell is contained within the objects and class hierarchy, while 'operational' and additional 'functional' information is captured in the rule base. 'Context-sensitive' rules (those that apply only during certain parts of the assembly process) allow the system to change its response to errors dynamically as the system conditions change.

The control program for the cell was housed within a cell computer which had access to the sensor information in the cell through a direct link with a programmable logic controller (PLC). This PLC (an Allen-Bradley PLC5/15) was responsible for both gathering the sensor data and communicating with the robot controller for the cell computer. The control program took the form of a knowledge base which contained the object-oriented system model, the diagnostic rules and error recovery strategies, as well as the assembly task definitions. To initiate the assembly process, it was simply necessary to 'suggest' the goal 'assembly_completed' and the knowledge base took care of the rest.

The sensory information provided to the knowledge base was fairly simple, all 'sensors' employed in the cell were binary (returning a true or false value), and consisted of such devices as limit switches, photoelectric beams, infrared emitter-detector pairs, and proximity switches. In addition, sensors were placed so as to accomplish 'passive' sensing rather than requiring 'active' sensing. That is, no interruption to the manufacturing process was required to obtain sensor values. There were at least a couple of reasons why this approach was used. The first was based on a desire to minimize the cost of the workcell, the second was motivated by a desire to focus on the main emphasis of the research, which was the structure and/or techniques used to support error recovery. It was felt that while more exotic sensors, such as machine vision systems or force/torque sensors, might alleviate some of the 'errors' which were being studied in this specific system, they simply shifted the domain of 'errors which needed to be solved'. Powerful sensors might eliminate 'alignment' errors or help ensure 'part presence', but there would still be other errors which would develop in those 'sensor-rich' environments and could not be uniquely diagnosed by sensor values. The decision was to develop a knowledge base which could flexibly respond to these 'simpler' errors and examine the resulting structure for guidelines to construct more comprehensive systems in the future.

The overall knowledge base can best be described by a diagram showing each of its 'parts' (Fig. 1). The boxes represent major sub-units of the system and have distinct responsibilities. The circles are information sources which provide the major sub-units with options and descriptions.

![Knowledge base structure](image-url)
Responsibility for control resided in the ‘task control system’ during normal operations. This system was similar to the algorithmic control programs which this ‘knowledge-based control structure’ replaced. It monitored the operation of the cell and utilized the list of instructions provided to it by the ‘assembly instructions’ to guide the cell through the steps necessary to assemble the product. While following these instructions, the task control system monitored ‘sensor data’ to check on the operation of the cell. If the values found in the sensor data were different from those expected, the system would declare an error to have occurred at that specific checkpoint. These checks were made because of the ‘system description’ provided to the task control system.

The types of sensor data checked for by the task control system were: part presence in the gripper, whether or not a part was ready in the feeder, did the fixture clamp properly etc. These ‘checks’ were intended to catch some of the more obvious errors as early as possible. Without using them, it would have been necessary to continue operations until the problem was discovered in some other way. By that item it would normally be quite difficult to pinpoint the problem.

The assembly instructions were simply a manipulator-level description of the movements necessary to accomplish the assembly task. For that project, the robot program was made up of many small ‘movement subroutines’ which consisted of an individual arm movement would could be commanded. The subroutines could be called in any order desired and were combined to make up the compound movements which accomplished the assembly task. The assembly instructions in the knowledge base simply described the order in which the movements were to be called. When an error recovery procedure needed to be run, the knowledge base simply sent a different set of movement calls.

The sensor data was provided through a set of objects whose values were obtained by calling to an external program. That program retrieved (and wrote) values directly from (and into) the data table of the PLC. This provided the knowledge base with information about what the sensors were detecting and allowed it to initiate actions.

The final information source to the task control system is the system description. This structure consisted primarily of objects and object classes which defined ‘what’ the rules were dealing with and where the value of their properties could be obtained.

When a checkpoint was reached and an error was recognized or when normal operation was halted due to an error, the responsibility for task planning shifted from the task control system to the ‘error recovery planner’. This sub-unit was responsible for determining which of the many error recovery options would be exercised. It utilized information gathered by the task control system to identify the specific ‘objects’ which were involved in the error. The names of these objects were given to the ‘doubt management system’ which will be described in a following paragraph. When the objects involved had been identified, the error recovery planner would use the doubt values of the various objects along with the information contained in the ‘additional cell-specific considerations’ to choose between the available ‘recovery options’.

This task, of choosing a specific strategy or recovery option, changed as errors were encountered at different times. This was true because the information contained in the additional cell-specific considerations was indexed to when, during the assembly task, an error occurred. In addition, the values returned from the doubt management system varied depending upon whether or not the objects had been involved in a previous error which occurred immediately before the one then being considered.
The additional cell-specific considerations were comprised of many heuristically defined preferences. For instance, when deciding between recovery procedures, a procedure which could be accomplished with the gripper that was then mounted would be preferred to one which would require a gripper change. This was a result of the fact that a gripper change would likely provide additional opportunity for errors, compounding the problem.

Parts which could be easily discarded and were available in feeders would be discarded, but those which were difficult to discard, or would have required a lot of effort to replace, would be held till last on the discard list. Other recovery procedures which were 'simpler' and could possibly exonerate the 'difficult to replace' part would be attempted first.

The final remaining 'sub-unit' was the doubt management system. This system kept a record of the objects which had been involved in errors, and maintained the cumulative 'doubt load' on a given object. As the control system executed various commands or recovery procedures, the doubt values were updated according to the results of the activities. If objects were involved in successive errors, their doubt values accumulate. If an error recovery procedure 'cleared' one of the objects, then its doubt value was reset to 'zero'. If an error recovery procedure was successful in re-establishing normal assembly operations, then all doubt values were reset, and the doubt management system simply waited until the next error to become active again. The system being described did not incorporate 'learning' in the sense commonly meant by AI researchers, no adaptive systems were used and the software system rules and structure did not change as a result of an error event. However, the system retained the information 'value' acquired during error recovery attempts. An unsuccessful attempt would often exonerate one or more components of the system resulting in a more 'informed' attempt during the next 'iteration'. It is the authors' intent to explore inclusion of adaptive systems into this structure in the future, it is likely that such systems could improve the performance of this module markedly.

The techniques described were implemented on the assembly cell briefly described earlier. The capabilities of this approach were demonstrated by intentionally inserting errors into the process and observing the response of the system. Bad parts were presented, mis-alignments were caused, the system responded by trying various recovery methods until it resolved the problem. If nothing worked, then the system instructed the robot to discard all the pieces, call more down from storage and start over again. The system thus demonstrated a robust ability to continue operation, even in unpredictable surroundings.

5. Summary

Current techniques for controlling robotic workcells are inadequate when dealing with errors that occur during the manufacturing process. To improve the performance of these workcells it is necessary to provide them with the ability to diagnose and recover from errors automatically.

This research has shown that it is possible to provide these capabilities to a system using today's technology and without great expense. The PC running the knowledge base was only a 286-based machine, and the PLC and robot used were standard 'off-the-shelf' devices used in industry. AI-based robotic workcells are not something that is still only practical in the research laboratory. Industry engineers can include these techniques in their 'toolkit' without placing their companies on the 'bleeding edge' of technology.
References


