A REAL-TIME KNOWLEDGE SCHEME FOR SENSORY-CONTROLLED ROBOT ASSEMBLY TASKS

W. W. Simon*, E. Ersue, H. Gose*** and M. Zoll***

**Department of Control Engineering, Technical University of Darmstadt, 6100 Darmstadt, FRG*
***ISRA Systemtechnik, Mornewegstrasse 45A, 6100 Darmstadt, FRG*
****Siemens AG, 8520 Erlangen, FRG*

Abstract. In the field of assembly automation with industrial robots the research on the application of artificial intelligence techniques is getting increasing importance. This paper describes the hierarchical structure of a knowledge–based sensory–controlled robot assembly system under development which is capable to plan and execute assembly tasks under real–time requirements. The hybrid knowledge representation scheme combining the rule–based and object–oriented approach to represent the assembly domain–specific knowledge is discussed. Furtheron, the knowledge processing concept based upon the representation scheme is explained. A first prototype of the system has been implemented in a real robotic test–bed. Several peg/hole part mating sequences validated the capability of the system to execute assembly tasks in an uncertain environment using sensory information.

Keywords. Artificial intelligence; assembling; knowledge base; knowledge representation; knowledge processing; robots;

1. INTRODUCTION

Today in the field of industrial assembly automation for cost effective small batch manufacturing the use of special purpose parts handling equipment must be minimized. Conform to this requirement, present approaches propose the integration of one or more robots, sensory elements (vision, tactile, force/torque), transport systems and other automation components to flexible assembly cells. For the economical use of these approaches two issues are getting increasing importance. First the task–oriented programming of the robot for obtaining fast adaption capability to new assembly tasks, and second the reliable execution of the given tasks in an uncertain environment.

These issues lead to intelligent robot assembly systems whose characteristics are

- capability of high–level task–oriented robot programming
- self–adaptability to deviations from the nominal task
- reliable task execution in an uncertain environment using various sensory information

Meeting the demands of intelligent behavior the assembly system must be capable to interpret the given assembly task, to plan the task performance and to supervise the execution.

Accordingly, the system has to collect, combine and infer complex symbolic and numeric information about robot, sensors, assembly objects and the assembly performance.

This paper is concerned with an approach for a knowledge–based sensory–controlled robot assembly system being capable to perform assembly tasks with the robot in the real world. In particular the paper discusses the execution module of the system using explicitly represented knowledge about the robot, the working environment, the objects to be manipulated and the specification of the job to be performed, to plan and execute an assembly task under real–time requirements and environmental uncertainties. Moreover, the paper deals with the knowledge representation scheme developed for the knowledgebase realization and the knowledge processing concept used by the execution module. A first prototype of the system under development has been implemented in the *C*–programming language in a real robotic test–bed. A description of the prototype implementation is given at the end of the paper.

2. HIERARCHICAL SYSTEM STRUCTURE

The knowledge based, sensory controlled robot assembly system concept is hierarchically structured and was initially proposed in [Simon,Ersue,Wienand, 1987]. The system structure is illustrated in Fig. 2.1. In the following we will give a brief overview about further developments achieved in the meantime.

The system consists of the following main components.

The *planning module* determines for an assembly task, specified by the user, the nominal sequence of task–oriented and object–specific assembly operations. The input of the planning module is the *directed assembly graph* which describes the number of alternative assembly operation sequences as they result from the construction phase of the assembly product. The *nodes* of the directed assembly graph represent the parts of the product and the *darts* from one node to another are determining the temporal and local dependencies which must be regarded when mating or connecting the related parts. The planning module selects one path in the directed assembly graph, taking the actual workspace conditions and assembly part relations into account. For example, due to a workspace configuration, it may be possible that an assembly part can be gripped only if another part was removed previously. Output of the planning module is the *nominal assembly plan*, consisting of a sequence of object–oriented, task–level assembly operations, e.g.

mate peg–a into hole–b

The assembly task planning is executed in an off–line mode, what means that no robot motions are taking place. The system uses during the planning phase rule–-based knowledge about the assembly task and symbolic information about the objects in the environment, both stored in the *knowledgebase* of the system. The contents and structure of the knowledgebase will be discussed in section 4 later on.

The *execution module* performs the nominal sequence of assembly operations together with the *robot motion control module* in an on–line mode. The operation sequence being executed, can also be commanded by the user. For each operation the execution module determines first a sequence of robot motions, depending on the actual assembly parts, the actual system state, and the environment.

Possible robot motions up to now are

- free space motions
- sensory guarded motions
- sensory controlled motions

For the certain motion the execution module generates the required motion parameters (e.g. goal position, velocity, acceleration, sensory control loop values) and commands the motion execution by the robot motion control.

When the robot motion control returns after the execution of the commanded robot motion, the execution module attempts to classify the actual assembly state reached by the robot motion. For the assembly state classification the available robot, sensor and environment information is interpreted in reference to the executed motion and the actual intended operation.

If the nominal planned assembly state could be verified, the execution module continues with the next robot motion or the next assembly operation. If the planned assembly state was not achieved by the robot motion due to the existing environmental uncertainties the execution module selects, based on the classified assembly state, an error recovery strategy to continue with the assembly task performance (see section 3.4).

Generally the execution of a robot motion changes the object relations in the workspace. According to the motion–specific environmental changes the execution module updates after each assembly state classification the environment model, containing information about locations, dimensions, and relations of the objects handled (see section 4).

The *robot motion control module* performs the execution of the motion planned and parameterized by the execu-

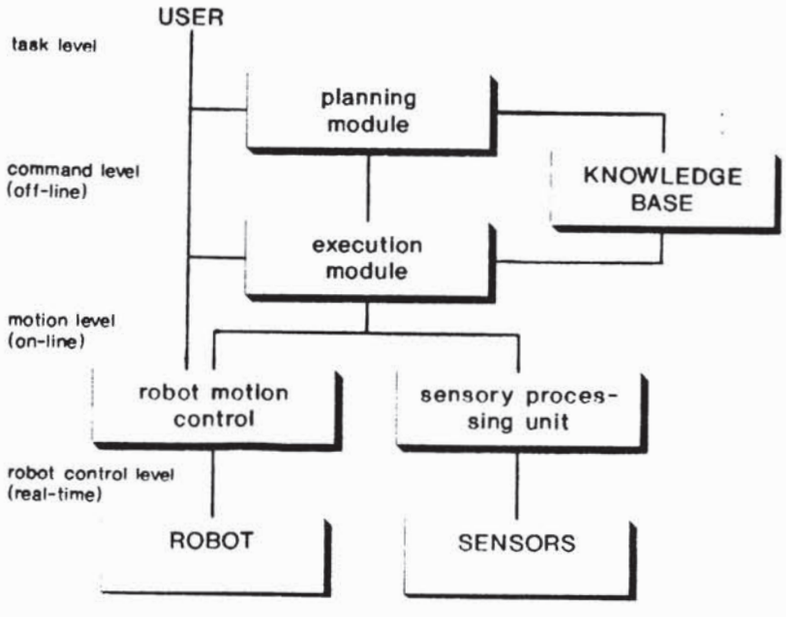

Fig. 2.1: Hierarchical system structure

tion module. The robot motion control module is connected to the robot in a real–time loop. The extend to which sensory information can be intelligently integrated and processed in this low–level motion control loop determines the planning and controlling effort which is required to be performed by the knowledge–based layers of the system. Significant for the execution of assembly tasks is the use of multi–sensory information, especially force/torque and distance information, because this capability defines the dexterity of the robot [Kegel, 1988].

The *knowledgebase* contains the domain–specific information needed by the system modules to perform the assembly task. The intention to represent the assembly–domain specific knowledge explicitly is, that the adaptation to new assembly tasks, the integration of additional system hardware (sensors, tools) or the optimization of the actual task performance can be achieved just by modifying or extending the knowledgebase.

3. EXECUTION MODULE

The main tasks to be carried out by the execution module are

- *assembly operation planning*
- *motion parameter determination*
- *knowledgebase updating*
- *assembly state classification*
- *error handling*

Figure 3.1 shows the performance of the execution module as a flowchart.

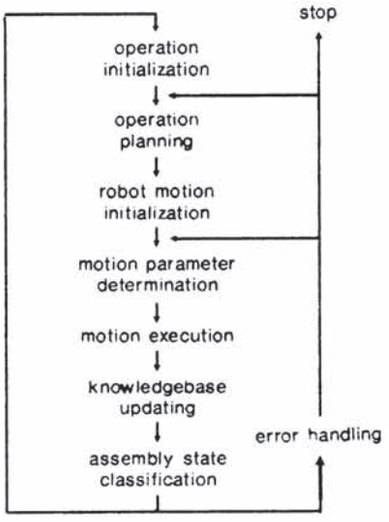

Fig. 3.1: Flowchart of the execution module

3.1 Operation Planning

In the operation planning phase the execution module determines for a given assembly operation with regard to the individual assembly parts and the actual system state the nominal sequence of robot motions for the operation execution.

Input, as mentioned in section 2, is an operation, e.g.

mate peg–a into hole–b

where the operator *(mate)* and the assembly objects *(peg–4, hole–5)* of the operation are specified on a symbolic level. Unknown at this planning level are, e.g., how the peg can be gripped, so his mate surface stays uncovered, or how the peg can be moved collision–free to the hole.

The resulting sequence of robot motions is based on the knowledge about the *mate*–operator which is applied on the actual assembly parts. Referring to the actual system state and environment configuration it can be possible, that in a following assembly step performing the same assembly operator a different motion sequence must be executed. Since there are different solutions for planning one assembly operator the application of a problem–solving strategy is motivated, whereby the actual system state must be included on–line into the problem–solving process. This is very essential for determining the dependencies of the single robot motion upon each other and it enables the system to execute the assembly task under environmental uncertainties.

3.2 Motion Parameter Determination

After the operation planning the execution module determines the motion parameter list corresponding to the type of robot motion. The interface to the robot motion control module is a motion–specific subroutine call, e.g.

LIN-MOVE-WITH-FORCE
(g–pos, v, a, ovr, fc–flag, fc–values)

with

g–pos	–	goal position
v	–	velocity
a	–	acceleration
ovr	–	override–factor
fc–flag	–	force control flag
fc–values	–	contact force values

For the determination of the motion–specific parameter list the execution module processes knowledge about the intended operation and accesses to numerical object data, describing the locations, dimensions and parameters of the assembly parts as stored in the knowledgebase.

3.3 Knowledgebase Updating and Assembly State Classification

Commonly the execution of a robot motion or an assembly operation changes the actual system state. According to the state changes the execution module updates the environment model in the knowledgebase on different information levels. The updating of the numerical location data, e.g., is done if through an assembly operation an assembly part was placed at another position in the workspace. In case of sub–parts are connected to the assembly object to be handled the positions of these parts are also updated by computing geometric relation data with respect to the corresponding part–relations.

After updating the knowledgebase the execution module checks whether the nominally planned system state is obtained by the executed robot motion or based on environment uncertainties, e.g. displaced assembly parts, an unforeseen assembly situation has occurred. The actual assembly state is classified by interpreting the available sensory information in regard with the assembly–specific relationships (operation – motion – objects – previous state).

If the nominal system state can be verified, the next robot motion or assembly operation is initialized and the execution module continues with determining the motion parameters or planning the operation.

If an invalid assembly situation is detected, the knowledgebase updating is according to the classification results repeated and an appropriate error strategy is applied.

If the execution module is not able to classify the unknown assembly situation or no appropriate error strategy is available, the assembly task performance is interrupted and the user has to take over control.

3.4 Error Handling

Corresponding to the detected error situation an appropriate error strategy is selected by the execution module. Generally the following two error strategies are distinguished:

- error–recovery motion execution
- new planning of the operation.

Additional error recovery motions are executed by the system in case of random error situations trying to reach the nominal planned assembly state. For example, if the actual motion was planned to move a peg into a certain hole of an assembly part and the peg was placed beside the hole because the assembly part is displaced, the system executes spiral search motions to find the hole. After detecting that the peg has been slid into the hole, the assembly part position is updated by the execution module and so the environmental uncertainties are reduced for the following assembly sequences.

If in case of systematic errors the active–motion error recovery fails, the system initializes a new planning of the operation using an alternative assembly strategy.

4. KNOWLEDGEBASE

Performing the given task as discussed the modules of the assembly system need symbolic and numeric facts about the environment together with deductive knowledge about the assembly task performance. In the following the structure of the knowledgebase for the assembly domain is generally discussed and in particular an appropriate representation of the knowledge is explained.

4.1 Environment Knowledge

Information about the following objects in the working environment is needed to perform the assembly task in the discussed manner:

- robot and tools
- sensors
- assembly objects

Robots and Tools

When the assembly system is interfaced to a certain robot–system numerical data concerning its geometry and kinematics is needed to calculate the robot motion trajectories and to check the limits of the robot operation space. Depending on the assembly task the robot has a set of tools which are either located at a specific place or mounted at the robot hand. For using or changing the tools their home positions must be known. When the tool is attached to the robot hand additionally to the tool position data are needed concerning the tool corrections, work directions and tool angles.

Sensors

Inevitable for the execution of assembly tasks under environmental uncertainties is the use of sensory information. Generally two sensor types can be distinguished. First sensors which are placed in the working environment, e.g. vision systems, and second sensors which are rigidly mounted to the robot, e.g. force/torque sensors. Accordingly the sensor model of the knowledgebase contains depending on the sensor type either position or geometry data to allow the appropriate information processing. Further the type of sensor information, the range and the resolution is stored.

Assembly Objects

Modelling the assembly objects in an appropriate way is most important for an intelligent assembly system, because the assembly object model can be considered as the cernel of the environment model. It must contain on one hand the symbolic information about the object relations needed to plan the assembly task and on the other hand numerical data, e.g. grip–positions, necessary to execute the single robot motion. The stored object information can be differentiated into

- geometry and position information
- relational information

The shape of the assembly parts is described to the system by using simple geometric, easily defined volume primitives, e.g. blocks or cylinders.

Considering the object relations *vertical* and *horizontal* relations are distinguishable. The vertical relations, e.g. part_of, describe the attachments of the object to superior objects. The horizontal relations are symbolic descriptions, e.g. connected_to, representing the object topology.

4.2 Deductive Knowledge

Besides an adequate environment model the system needs further knowledge about the performance of the assembly task and respective operations. This type of knowledge differs from the previously discussed environmental knowledge. It cannot be formalized as a number of static storable facts but it encloses correlations which can be easily formulated in *IF–THEN* expressions. Commonly this type of knowledge is denoted as deductive knowledge which refers to the facts stored in the environment model and infers new *dynamic* knowledge.

4.3 Knowledge Representation Concept

The computational coding of knowledge is termed as knowledge representation. An overview on knowledge representation methods is given in [Frost, 1986] and [Feigenbaum, 1981], including semantic nets, frame– -based systems and production rules.

The implemented knowledge representation concept defined for the realization of the discussed knowledgebase combines the production rule scheme with the object– -oriented knowledge representation approach to meet the demands of representational adequacy and efficiency.

Production Rules

A production rule is a statement cast in the form

IF	this	*condition*	holds
THEN	this	*action*	is appropriate

The IF–part of the production, called *premise* or *condition–part*, states the facts of a case that must be present for the production to be applicable. The THEN–part, called *conclusion* or *action–part*, is the appropriate action to take. The application of the production rule can result in new facts or initialize operations.

Using production rules for the representation of assembly knowledge has the following advantages:

First the assembly and error strategies determining how to handle the assembly parts in certain assembly states can be formulated in IF–THEN statements, and in doing so also heuristics are representable. Second the knowledge for the assembly state classification and error detection can be formalized in rules. Third the utilization of so called *certainty factors*, described in [Buchanan, 1984], enable the representation and processing of incomplete or uncertain knowledge. Another advantage of the rule–based knowledge representation results from the high modularity and modificability of rules as single knowledge entities.

Finally the rule–based representation concept has a disadvantage. If not decisive knowledge but numerical information about the environment must be represented, the transparency and structure of the knowledgebase get lost.

Object–Oriented Knowledge Representation in Frames

The main characteristics of the object–oriented knowledge representation are object–oriented data storage, hierarchical structured object classes and inheritance mechanisms.

An object–oriented knowledge representation is achieved by using frame–based data structures for the knowledgebase realization. According to Minsky in [Winston, 1975], a frame is a universal data structure permitting the declaration of properties of an object. Generally a frame represents a common property–list where the single property is defined as a *slot*. A slot can contain an arbitrary number of features, called *facets*. The object–specific information is stored in the *facet–values*. Using *type– - slots*, containing references to other frames, an hierarchical structured object class concept can be realized and inheritance mechanisms are applicable.

Representing the Assembly Knowledge

According to the demands of an adequate, transparence and modular representation of the whole required assembly knowledge, both of the representation schemes discussed above have been integrated in one knowledgebase realization, see Fig. 4.1.

Distinguishing between a database and a rulebase the environment information is stored in frame–based data structures and the decisive and strategic knowledge is formulated in rules. Further, the rulebase is structured in argument with the different knowledge domains

- operation planning
- motion parameter determination
- assembly state classification
- error handling
- inference control

Storing the rules by attaching them to different knowledge domains simplifies on one hand the maintenance of the rulebase in regard to keep the rule–based knowledge consistent and complete, and accomplishes on the other hand an effective rule processing.

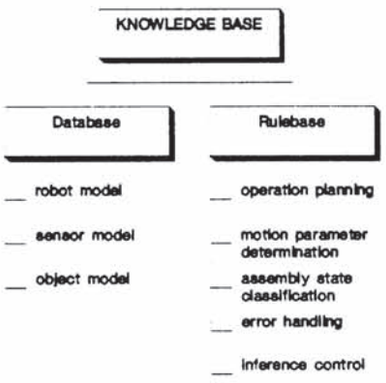

Fig. 4.1: Knowledgebase structure

5. KNOWLEDGE PROCESSING CONCEPT

Following the necessity to apply strategic assembly knowledge, formalized in rules, to an environment and system configuration, modeled in frames, the system infers rule–based knowledge to accomplish the execution of the assembly task.

The concept for the rule–based knowledge processing is derived from the production system approach described in [Nilsson, 1982]. Referring to Nilsson the processing of the rules is obtained in a recognize–act cycle which is decomposable into three phases:

- matching
- selecting
- executing

During the matching phase the system identifies the set of applicable rules. The applicability is classified by matching the condition part of the rules with the facts stored in the object–oriented database. After determining the applicable rule–set one rule is selected from the set. The selection of the rule is based on the interpretation of *rule–priority–values*. In the execution phase the system executes the action part of the selected rule. The execution of the rule–conclusion can modify or delete existing facts or create new facts. Generally the database will be changed and other rules can be applicable in the next cycle. Moreover, the execution of the rule–conclusion can cause robot motions.

According to the discussed knowledge processing concept an inference mechanism has been implemented. Figure 5.1 exhibits the recognize–act cycle. This inference mechanism is used by the execution module to perform the assembly task as discussed in section 3.

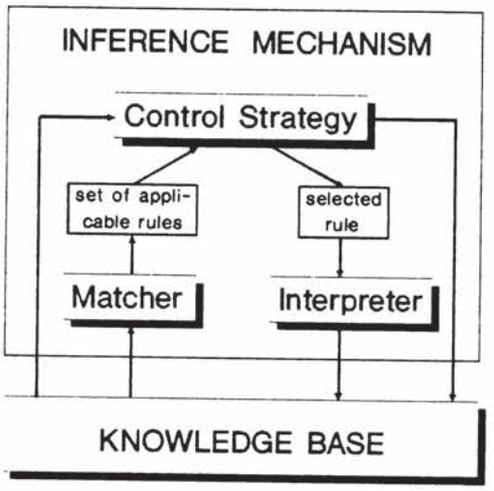

Fig. 5.1: Recognize–act cycle

6. FIRST PROTOTYPE

A first prototype of the discussed knowledge–based assembly system has been implemented in the *C* programming language on an Intel 310 workstation running under the real–time operating system iRMX86. The Intel workstation is connected as a host computer to a standard industrial Siemens RCM3 robot control via dual port memory. According to the demand to process the information in an off–line, on–line and real–time mode the implementation is based on a multi–tasking software concept. Using the assembly system prototype to control a Manutec R3 robot provided with a force/torque wrist–sensor several peg/hole part mating examples have been sucessfully performed.

7. CONCLUSIONS

Using the discussed knowledge representation scheme to realize the knowledgebase of the described robot assembly system an adequate, transparent and modular representation of the whole necessary assembly knowledge has been achieved. The distinction in the knowledgebase between a database containing the static facts and a rulebase representing the deductive knowledge due to the certain knowledge domain facilitates the maintenance of the knowledgebase in regard to consistency and completeness.

The hierarchical structure of the robot assembly system prototypical implemented in the *C* real–time language allows the user to perform assembly tasks under real–time requirements by specifying high level task–oriented commands. Furthermore, the capability of the system to integrate multi–sensory information into the robot motion control and knowledgebase updating permits the assembly execution in face of environmental uncertainties.

8. ACKNOWLEDGEMENT

Part of this work has been supported by the Deutsche Forschungsgemeinschaft –DFG–.

9. REFERENCES

1. Buchanan 1984, *Rulebased Expert Systems*, Addison Wesely, 1984

2. Feigenbaum 1981, *The Handbook of Artifical Intelligence*, Vol.1, Pitman Book Limited, London, 1981

3. Frost 1986, *Introduction to Knowledgebase Systems*, Collins Professional and Technical Books, 1986

4. Kegel 1988, *Vergleichende Untersuchungen an Sensoren zur Messung der Greifparameter in Greiferwerkzeugen fuer Industrieroboter*, VDI Berichte Nr.677, 1988

5. Nilsson 1982, *Principles of Artificial Intelligence*, Springer Verlag, 1982

6. Simon & Ersue & Wienand 1987, A Knowledge–Based Approach for Sensory–Controlled Robotic Assembly Operations, *First European In–Orbit Operation Technology Symposium*, Darmstadt, ESA Publication Division, 1987

7. Winston 1979, *Artificial Intelligence*, Addison Wesely Publishing Company, 1979