Sensor based control for autonomous robots

Conference Paper · February 1989

DOI: 10.1109/ICCON.1989.770581 · Source: IEEE Xplore

Geleyn Meijer
Amsterdam University of Applied Sciences
35 PUBLICATIONS **166** CITATIONS
SEE PROFILE

Frans C. A. Groen
University of Amsterdam
277 PUBLICATIONS **4,500** CITATIONS
SEE PROFILE

Bob Hertzberger
University of Amsterdam
138 PUBLICATIONS **1,033** CITATIONS
SEE PROFILE

SENSOR BASED CONTROL FOR AUTONOMOUS ROBOTS [1]*

G.R. Meijer, G.A. Weller, F.C.A. Groen, L.O. Hertzberger
Computer Science Department (FWI), University of Amsterdam
Kruislaan 409, 1098 SJ Amsterdam, The Netherlands

Summary. In this paper the work of the computer systems group on robotics and sensor systems is presented. Based on a functional decomposition of actuator and sensor modules, a control system for autonomous robots is developed. The interface level between actuator- and sensor systems is described in terms of high level sensor modules. The sensor modules themselves are able to monitor their performance and adjust internal model parameters to increase the reliability of the sensor readings. A model is developed in which the sensor output is used for monitoring, diagnosis and exception handling for the control of the autonomous robot. To make efficient use of this control system, off-line programming and simulation techniques have been realized. The sensor based control system is both tested on assembly applications and a mobile robot.

Keywords. Sensors; Sensor data processing; Sensors modules; Off-line programming; Error handling; Robots; Artificial intelligence; Assembling; Manufacturing processes.

1. INTRODUCTION

In the area of flexible production automation increasing emphasis is put on the requirements of autonomy. The objective of this interest is to extend the control capabilities of robot systems to cope with changes of the external conditions of the robot during run-time. The bottleneck for generating more flexible robot systems is often not a shortcoming of the mechanical capabilities of the manipulator or the physical sensor system, but lies in the control software and the proper interpretation of the sensor data. Current state of the art control software lacks the general decision making capabilities to handle abnormal situations in an adequate manner.[2,3,4]

This paper addresses the problem of equipping a robot system with a control- and sensor system that realizes autonomous behavior of the robot. That is, the robot system is capable of handling deviations from a priori defined operating conditions, which changes the pre-planned flow of actions. Our solution to this problem is to add three capabilities to the control system. These capabilities are monitoring, diagnosis and recovery. These modules do not only appear at the level of actuator control, but are also essential for the processing of sensor data.

Reliability of the sensor system is of great importance since all the actions of the robot will essentially be based on the information that is acquired and interpreted by the sensor system[5,6,7]. Equipping the sensor system with a means to maximize its reliability will increase the overall performance of the robot system. Techniques that are in use are for example statistical techniques in which, on basis of more than one measurement, a statistically best measurement is chosen [8,9]. These techniques use the concept of redundancy to refine their measurements or even to reject non-fitting measurements [10]

In our control model, the sensor system is split into sensor modules, each having their own functionality. This idea has been proposed by Henderson et. al.[11,12] Each sensor module is equipped with tests to verify certain characteristics in its input and with tests to verify derived properties in its output. A negative test result implies a sensor module failure.

We are using two types of tests. The first are tests that verify demands on the input of the sensor module that must be fulfilled at all times. The other type of test verifies demands on the input of the sensor module that only need to be fulfilled under certain conditions. These conditions originate from restrictions in the sensed environment. The latter tests are latent present and are

[1] * This work is supported by: -Esprit project 623: "Operational Control for Robot Systems Integration into CIM, Systems planning, Implicit and Explicit Programming", - NWO/SION project: "Intelligent Sensors".

activated only in those specific situations. A mechanism is introduced that allows activation of these latent tests on basis of the environmental conditions. When a test gives a negative result the sensor module starts a recovery procedure. This procedure is based on adjusting the parameters of the algorithm and/or demanding adjustments of the input to the sensor module. These adjustments are available in the form of rules. Every sensor module is equipped with its own set of rules.

We have developed a control model which contains the required monitoring, diagnosis and recovery planning functions. For this purpose a functional decomposition of the actuator and sensor modules is made. This control model is described in the next section. A detailed description of the actuator modules is found in section 3 and the sensor system with its own mechanism for handling exceptions is presented in section 4. Section 5 contains a discussion and presentation of the results.

2. MODEL OF THE CONTROL SYSTEM

The aim of the control system is to provide a robot system with autonomous capabilities. An off-line programming and simulation environment is needed to create the off-line program. The correctness and functionality of this program can be tested and adapted in simulation. The simulation environment should be transparent to the available robot hardware. This allows the program, when it is sufficiently tested in simulation also to be downloaded to the actual robot controller and executed on-line. To execute the robot program, most current robot controllers run an interpreter which executes the language statements of the robot program one by one. Executing the program the control system makes use of sensor modules which provide the actual value of the environment variables.

An off-line program is generated by transferring a global task description into a sequence of subtasks. The structure of the off-line program reflects the precedence of the subtasks. A subtask consists of elementary operations of the robot.

In this off-line programming system, it is often very hard to model the work environment of a robot or to predict the value of relevant environmental variables. Many environmental variables have a discrete nature and complex relations with others. As a result, the internal model of the robot environment often only partly reflects the actual status of the environmental variable.

At the subtask level small deviations of the environmental variables can be handled in a closed loop control. All environmental variables which are not handled in the various control loops, but are of importance to the successful execution of a robot task are considered to be constant. They are ranging from part positions and sizes in an assembly robot system to obstacle locations in an inspection robot.

If the actual environmental variables are out of the boundary values of the operating conditions for a subtask, an exception has occurred. These exceptions may prevent the robot from achieving its task. To handle these exceptions autonomously, the control system needs three additional functions.

In the first place the control system must be capable of detecting deviations of environment variable values from their expected value. This monitoring activity is performed in parallel to the execution of the off-line program. The sensor system is thus used both for the realization of pre-planned closed control loops and for detecting the occurrence of exceptions.

Once a fault condition is detected, a diagnosis of the current state of the environment is needed to reveal the cause of the fault condition and to classify the exception. For this diagnosis activity the control system again makes use of the sensor system. A successful classification of the exception opens up the way for a recovery planning module to plan corrective actions. The aim of these actions is to restore the environment such that the pre-planned program can be continued. In figure 1 the model of the control system is graphically represented. The internal mechanisms of the actuator modules are described in the next section.

The sensor system provides all sensory data for the active feedback loops, the monitoring module and the diagnosis module. It is our view that the processing of sensor data so that the system produces reliable information, is an all but trivial matter because of the sensor data interpretation. Because the control system modules are often only concerned with higher level sensor data abstractions, the sensor data processing is concentrated in a separate sensor system.

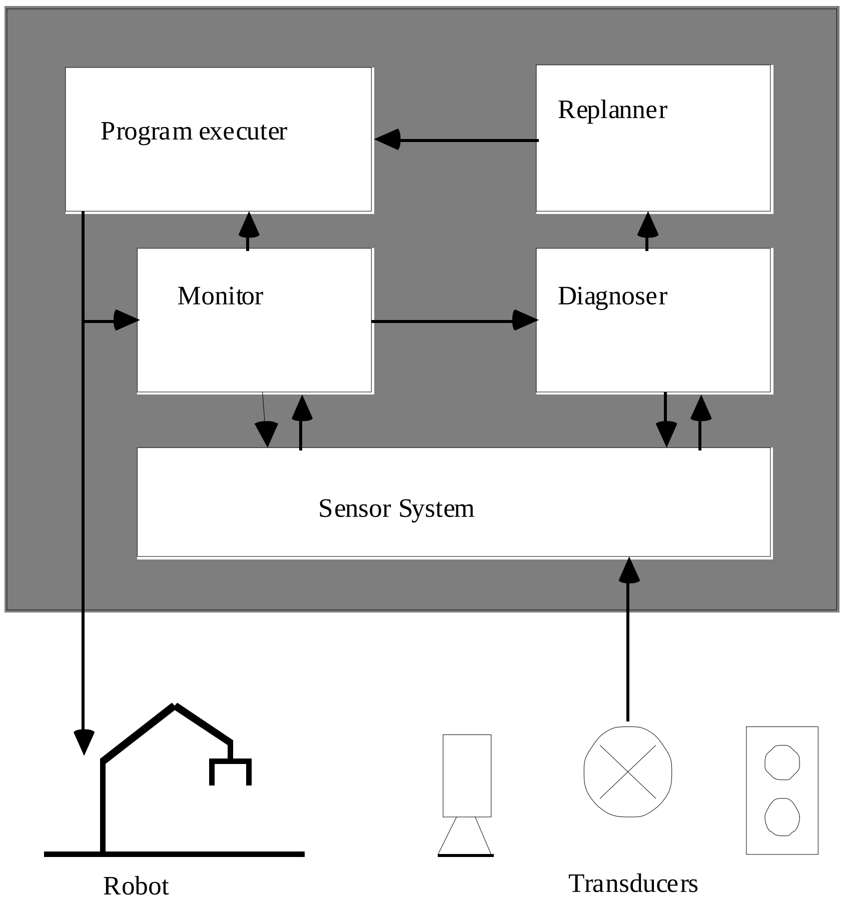


Figure 1. Model for sensor based control

A consequence of the modular sensor system approach is that different sensor data processing strategies can easily be incorporated in the overall control model. The sensor system itself controls the acquisition of sensor data through transducers and the subsequent processing and analysis of the data. During this process the sensor data passes various levels of abstraction due to the concept of sensor module that we apply. Similar to the situation with robot programming, exceptions or fault conditions can occur. Therefore a similar mechanism of monitoring, diagnosis and recovery functions is realized within the sensor modules.

3. ACTUATOR MODULES

The aim of the actuator modules is to execute a robot program and to handle exceptions in an appropriate manner. A production task like part welding or assembly consists of a number of subtasks which have a dependency relation given by the application. The dependency relations indicate the necessary order in which subtasks have to be carried out. The order constraints can be of a geometrical nature like the assembly problem of stacking several parts upon each other. Also the availability of parts or production resources can impose constraints on the allowable order of the tasks.

A possible representation for the decomposition of an assembly task into subtask and their dependencies, is a precedence graph. In a precedence graph, the subtasks are represented by nodes and the dependency of two subtasks is represented by an arc, connecting the two nodes. The dependency arc indicates that the node which is at the top end of the arc needs to be successfully performed, before the node at the lower end can be executed. In the case of assembly applications, the precedence graph is also called 'assembly graph'. We use as a benchmark a simple mechanical assembly kit that consists of two side plates with a pendulum like structure, the lever, in between. The parts are connected by locker pins and spacers.

A robot program is generated by transferring a global task description into a sequence of subtasks. These subtasks are detailed enough to be directly expressed in the operating primitives of the robot. These operating primitives are called elementary operations. In table 1 the elementary operations of a robot arm for assembly operations are listed.

Elementary action	Parameters
MovePtP	goal, speed
MoveCs	goal, speed
Grip	part
UnGrip	part
Insert	goal, forces, speed
Approach	goal, forces, speed
Depart	goal, forces, speed
Comply	goal, forces, speed

Table 1. Elementary operations

The resulting task structure of a robot program and the corresponding elementary operations are graphically represented in figure 2.

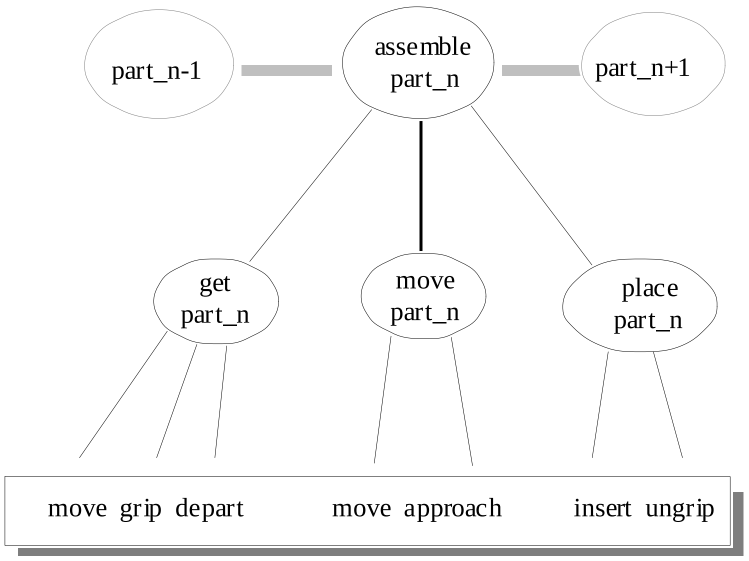


Fig. 2. Elementary operations and task knowledge for an assembly operation.

The off-line planned sequence of elementary operations is called the normal program and when the robot controller is executing the normal program without exceptions, we say the controller is in normal operation.

Monitoring and diagnosis.

The robot control system must be capable of detecting deviations of environment variables from their expected value. To realize this, for each elementary operation, a list of monitoring conditions is specified. This list instructs the monitoring module which environmental variable should be checked when the corresponding elementary operation is executed. The sensor system delivers the values of the environmental variables. An exception is detected by thresholding on the relevant sensor module output. The values for the threshold operation are provided by the parameters of the elementary robot operation which is monitored.

After detection of an exception, a diagnosis is performed to gain additional information on the nature of the exception and to update the internal model of the robot environment. In general it is a problem to decide which environment variables need updating and which do not. We are using fault tree structures to guide the diagnosis activity. The input for the diagnosis is the set of sensor module output values which exceeded their boundary values (as detected by the monitor). For each entry in this list, a corresponding entry is given for the fault trees. The fault tree for a specific exception entry consists of a number of arcs and nodes. Each node represents a query to the sensor system to measure the value of an environment variable. Based on the result of the query, a new arc is followed which either leads to another sensor system request, or ends in a leave of the tree. The leaves of the fault trees represent the possible outcome of the diagnosis.

We have analyzed the possible exceptions which can occur during an assembly process. For each measurable environmental variable, a fault tree was constructed. The possible results of the diagnosis activity is given in table 2. The exceptions are split in 5 groups and each group is further divided in several variants.

COLLISION	- Collision with unknown object
	- Collision with unknown object at position P
	- Colision with object O at position P
OBJECT ERROR	- Object lost at unknown position
	- Object lost at postion P
	- Object moved in gripper
NOT REACHED GOAL	- not reached goal in time, now at position P
OBSTRUCTION BY FORCE	- Obstruction with unknown object
	- Obstruction with unknown object at position P
	- Obstruction with object O at position P
GRIPPER ERROR	- Gripper containing unknown object
	- Gripper containing object O

Table 2. Exceptions

Exception handling

Knowledge about the task structure of a robot program enables two approaches to exception handling: recovery planning and task rescheduling. In the first approach, an attempt is made to recover from the exception by replanning the robot actions. The goal of the replanning activity is to restore the operating conditions in such a way that the off-line generated or normal program can continue. One research direction to tackle this problem is to apply automatic planning systems. These planning systems require a precise description of the environment and a detailed description of the effects on the environment of planning operators. An alternative is to use a more heuristic approach in which the exception handling mechanisms are specified as a sequence of planning primitives for a particular application.

In (Meijer,1988a) two mechanism are described, one based on recovery planning and one based on task rescheduling for using exception handling strategies to guide the exception handling activity.

4. THE SENSOR SYSTEM

The communication with the sensor system is realized issuing a request to measure an environmental variable. These requests are the equivalent of the elementary operations of the actuator modules. The sensor system supplies an answer to the request. For our work we have defined a number of environmental variables, relevant for the assembly application. These environmental variables are used in the monitoring and diagnosis modules described in the former section and are listed in table 3.

Sensor modules	Description
Check Motion (CM)	Measure movement of robot
Robot Forces (RF)	Measure forces of the robot
Robot Free (RFR)	Measure contact with objects
Plan Position (PP)	Measure current position
Object Available (AO)	Measure availability object in gripper
Object Orientation (OO)	Measure orientation object in gripper
Find Object (LO)	Find location, orientation of object
Identify Object (IO)	Identify object using database

Table 3 Sensor primitives.

The sensor module

The sensor module concept opens up the way to create many different sensor modules through combinations of others, leading to a flexible sensor system structure. The sensor system is built up hierarchically. (Figure 3)

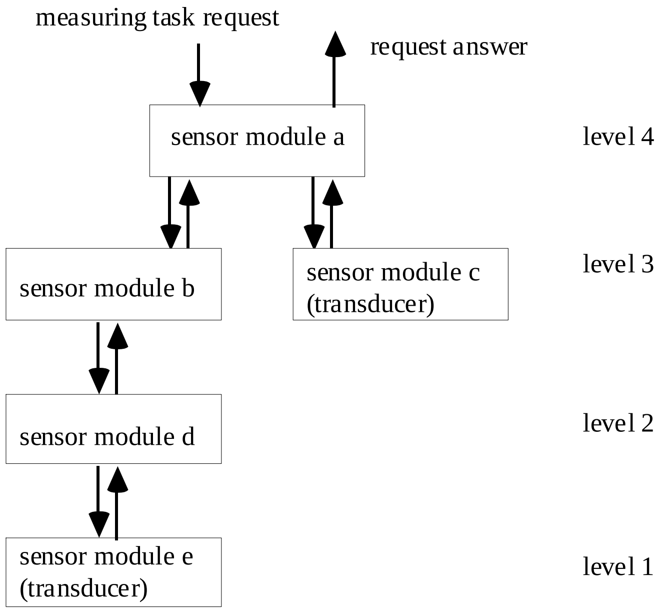


Figure 3. A hierarchical build-up sensor system. The transducers are at the bottom level and at the top level we find the application specific sensors modules to measure the environment variable values that are to be known for the assembly task.

A mechanism has been incorporated to handle an 'erroneous' input to the sensor module and to cope with it. In our approach we want to trace the origin of the failure and try to recover from it by adjusting the parameters of the sensor module algorithm and/or the input to the sensor module. The scope of the tests in a sensor module is not restricted to the input signal. During the execution of the algorithm processing the input, certain tests will also be activated, increasing the ability to pinpoint the exact location and nature of a sensor module failure. A negative test result will initiate a recovery stage. The recovery strategy is based on a set of rules that are available locally to the sensor module of which the test is a part. These rules determine the explicit recovery steps. In case no recovery is possible a final attempt may be to activate an alternative sensor module in the sense as proposed by Henderson [11].

Sensor module tests

The tests may be either active or latent. An active test will always be executed. A latent test will only become active if certain environmental conditions justify such a test. If, for example, a sensor system contains a sensor module that measures what colors are present in a camera picture, this sensor module may have a latent test available on the distribution ratio of these colors in the picture. This test need only be activated when such a distribution ratio is actually known. Latent tests that have not been activated play no further role.

The previous example showed how a test can be activated based on information concerning the environment within which a measurement is performed. We will call this additional information, such as the distribution ratio, environmental information. Activating latent tests to verify such environmental information increases the number of active tests in a sensor module and therefore the ability to locate and identify a possible sensor module failure more precisely. Obtaining the demands that activate tests to verify them is illustrated in the next example.

For the calibration of the cameras that are used during assembly, a sensor module called 'foreground' is activated. This sensor module measures from a grey-valued input picture the foreground that can be discriminated from the background on basis of a thresholding technique. In this application the input consists of an image of the calibration picture made up of straight dark lines on a bright background. This sensor module determines the proper threshold and outputs a binary image. The output is used by a subsequent sensor module that measures straight lines in a binary input picture and outputs the algebraic equations. These equations are input to a next higher in hierarchy positioned sensor module that measures the coordinates of the cross points of the lines which in their turn are input to the highest in hierarchy positioned sensor module that uses these cross points coordinates together with the knowledge about the calibration picture and the position of the camera to determine the camera parameters. This last sensor module is the 'measure camera parameters' module. After activation of this module the aforementioned sequence of sensor modules is activated in reverse order. At the level of the 'measure camera parameters' module knowledge is available on the geometry of the used calibration picture. Furthermore there may be knowledge available on the illumination conditions under which the measurement is performed. All this knowledge makes up the environmental information. On basis of this information demands can be formulated on the data at the various levels of data abstraction in the activated sensor modules. Two problems can now be distinguished:

- Demands derived from environmental information that is available on a high logical sensor level, must be applied to data only available on a low logical sensor level.
- Even if the relevant data are still available on a high logical sensor level, it is advantageous to verify their validity immediately when they are derived on a lower level. So the environmental conditions have to be translated to the data representations at lower levels.

We want the demands on environmental information, that are available on a high sensor module level, also to be available on the low level sensor modules where they can be verified by tests. This way we increase the total number of tests and the ability to measure a sensor failure in an early stage.

After the execution of a test three situation can be distinguished:

situation 1- The input fulfils the demands prescribed by the sensor module. The output fulfils the demands prescribed by the logical sensor.

situation 2- The input fulfils the demands prescribed by the sensor module. The output doesn't fulfil the demands prescribed by the sensor module.

situation 3- The input doesn't fulfil the demands prescribed by the sensor module.

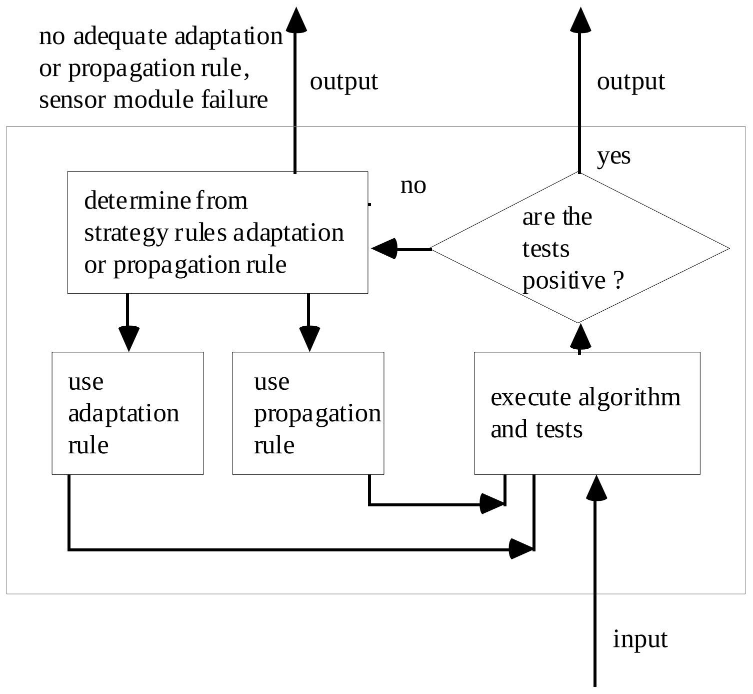
Recovering from sensor module failures

On basis of a negative test result a sensor module error is detected. The sensor is then in situation 2 or situation 3. The next step is to try to recover from this error. There are three types of rules which govern this recovery process

The strategy rules relate a negative test result to required changes in the output or other internal available input derivatives that the test when executed again will lead to a positive result.

The adaptation rules translate these required changes to algorithmic parameter changes.

The propagation rules translate required changes to demanded input signal changes. A depiction of the recovery based on these rules is given in the next figure.



no adequate adaptation or propagation rule, sensor module failure
output
output
yes
no
determine from strategy rules adaptation or propagation rule
are the tests positive ?
use adaptation rule
use propagation rule
execute algorithm and tests
input

Figure 4. Control in the sensor module.

5. DISCUSSION AND RESULTS

In this paper we have presented the work on sensor based control for autonomous robots currently being carried out in our research group. We have argued that a basic prerequisite for autonomous behavior is the ability of the control system to react appropriately to unexpected interruptions of the pre-planned actions. To handle these exceptions the actuator modules of the control system are expanded with monitoring, diagnosis and replanning functions. After the occurrence of an exception a rescheduling scheme is used to select an alternative task. This activity can only performed if knowledge concerning the tasks to be performed and their common precedence relations is given.

For the sensor system a similar mechanism for detection and handling of sensor failures is developed. Detection of sensor failures is based on tests on the signal of the sensor. We have divided the tests into two groups, environment dependent- and environment independent tests. The first group of tests verifies demands on the input of a sensor module. The environment dependent tests verify the demands on the input and output of the sensor module that are determined by the context within which this module is used. We have shown how the latter group of tests increases the ability of a sensor module to find a failure in an early stage. We incorporated rules in each sensor module as a means to recover from a sensor module failure. The recovery scheme uses these rules to adjust parameters and/or the sensor input.

To realize the computational environment for the robot control system with a real robot and sensors, we have implemented a control scheme with symbolic computations expressed in Prolog and numerical processing defined in the language C. This system offers the possibility to evaluate the functionality of monitoring, planning and replanning modules and provides an easy access to the underlying control mechanisms. A first version of the domain knowledge for an assembly application is specified. The modules are tested by using simulated sensor module output. The lower level sensor modules are being realized and will consist of a wrist force/torque sensor and a 2D vision system using various filter techniques.

The control system is based on an object oriented programming approach and we have used the workbench tool ART. It provides a symbolical trace of the execution behavior of the robot programs when exceptions are generated (Wonderen,1988).

Also the user interface for the off-line programming of exception handling strategies has been realized. The system is based on an extension of an existing off-line programming system ROSI, developed at the University of Karlsruhe (Dillmann, 1986).

Acknowledgements

We would to thank G. M. van Wonderen and P. Mul for stimulating the research work described in this paper.

REFERENCES

1- Dillmann,R., Hornung,B., Huck,M. (1986). Interactive programming of robots using textual programming and simulation techniques. Proceedings of 16th ISIR conference, Brussel, 1986.
2- Gini,M. (1985). The role of knowledge in the architecture of a robust robot control. Proceedings of IEEE 1985, 561.
3- Gini,M. (1986). Symbolic and Qualitative reasoning for error recovery in robot programs. In: Rembold U, Hörmann K, (eds), Proceedings of the NATO International Advanced Research Workshop on Languages for sensor based Control in Robotics, Italy september 1986.
4- Meijer,G.R, Hertzberger,L.O.H. (1988). Exception handling for robot manufacturing process control. Proceedings of CIM Europe conference, Madrid may 18-20 1988, IFS publications.
5- Pugh(1985) Robot sensors - A personal view.pp. 521-532, ICAR 85.
6- Bogler (1987) Shafer-Dempster reasoning with applications to multi sensor target identification systems.IEEE Transactions on Systems, Man and Cybernetics,vol. SMC-17, no. 6, November/December 1987.
7- Kak A.C., Roberts B.A., Andress K.M. and Cromwell R.L.(1987) Experiments in the integration of world knowledge with sensory information for mobile robots.IEEE Int. Conf. on Robotics and Automation.pp 734-740 1987
8- Ayache N. and Faugeras O.D.(1987) Maintaining representations of the environment of a mobile robot Int. Symposium on Robotics ResearchSanta-Cruz, August 1987
9- Luo R.C., Lin M.H. Scherp R.S. (1987) The issues and approaches of a robot multi-sensor integration IEEE Int. Conf. on Robotics and Automationpp 1941-1946, 1987
10- Chow E.Y.and Willsky A.S. (1984) Analytical redundancy and the design of robust failure detection systems.IEEE Transactions on Automatic Control. pp 603-614, vol. AC-29, no. 7, July 1984
11- Henderson and E. Silcrat. (1984) Logical sensor systems. pp. 169 - 193, Journal of Robotic Systems, 1(2) 1984.
12- Henderson, Hansen and Bhamu (1985) The specification of distributes sensing and control.2(4), 387-396, Journal of Robotic Systems. 1985.
13- Wonderen,G.M. van. (1988). AI and high level robot control and a preliminary implementation of a High Level Interpreter. Technical report, Faculty of Mathematics & Computer Science, department of Computer Systems, July 1988.