

# An On-Line Planner for MARIE

Frank Terpstra, Arnoud Visser and Bob Hertzberger

August 2001

## Abstract

In this paper we introduce an on-line planner implemented on MARIE, a real autonomous robot. Planning occurs at a high level, the planner controls other modules such as the path planner, which contain planners themselves. Plans are represented in a tree structure. The on-line planner has a library of partial trees to choose from when replanning is needed. We think that our research is unique in applying on-line planning at such a high level on a real robot. The on-line planner uses a simple algorithm based on elements of POP [10] in combination with random choice. The random choice is essential for real world applications with information errors, because it prevents the robot from taking the 'best' choice repetitively. Despite being simple the algorithm performs very well in practice. Tests show that the on-line planner gives a major increase in robustness and reliability while not being significantly slower than perfect algorithms.

## 1 Introduction

In planning research most projects using high level planning are “theoretical” planners for the STRIPS [6] domain [10] [1] [2]. Most of the planning research being done on real robots[8] deals with the lower levels of planning.

The restriction to the real robot domain [5] is important because assumptions made for theoretical planners such as Atomic Time<sup>1</sup>, Deterministic Effects<sup>2</sup>, Omniscience<sup>3</sup> and Sole cause of Change<sup>4</sup> cannot hold in the real world. The goal in our research has been to implement a high-level planner on a real robot leading to an improvement in reliability and robustness.

The robot used is MARIE (Mobile Autonomous Robot in an Industrial Environment) which originated from the Esprit-MARIE project started in 1989. The workings of MARIE and the MARIE ar-

---

<sup>1</sup>An action is uninterruptable. Simultaneous actions are impossible.

<sup>2</sup>The effects of an action are a deterministic function of the action and the world at the time this action takes place.

<sup>3</sup>The planner has complete knowledge of the world and the effects of its own actions.

<sup>4</sup>The world only changes through actions performed by the planner.

chitecture will be explained in section 2, while section 3 is dedicated to plan representation because of its particular relevance to on-line planning.

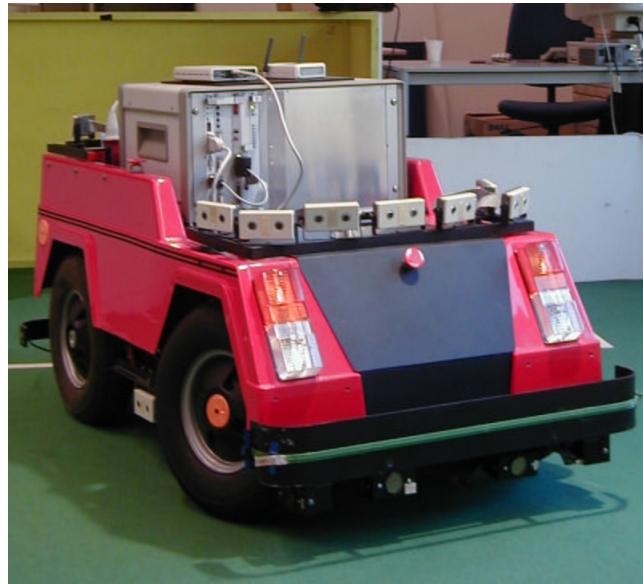


Figure 1: MARIE

Originally MARIE used an off-line planner; the plans were generated prior to execution. With the current on-line planner, only a skeleton plan is generated off-line, which represents the goals to be fulfilled. This skeleton plan is sent to MARIE, and further extended during the execution based on actual status information. This and other aspects of planning will be discussed in section 4, followed by a section describing the implementation. Finally the results of our experiments are given, and we will discuss whether our goals of improved robustness and reliability have been achieved.

## 2 MARIE Architecture

Here we will give a short description of the hardware and software that MARIE uses. MARIE is an autonomous robot based on an electric cart for the disabled. It is equipped with wide beam ultrasonic sensors for collision avoidance and narrow beam sensors for wall following. Positioning is done by dead reckoning using shaft encoders on all wheels. The robot is controlled by an on-board computer with 4MB of ram and a Motorola 68030 processor running VxWorks. Finally there is a wireless network connection which among other things allows some control modules, including the planner, to run remotely.

For the MARIE software a hybrid architecture is used, this means that the software tries to combine the best aspects of functional and behavioral architecture. This results in a system with the mission planning capability of a functional approach and the fast reactive behavior of a behavioral approach.

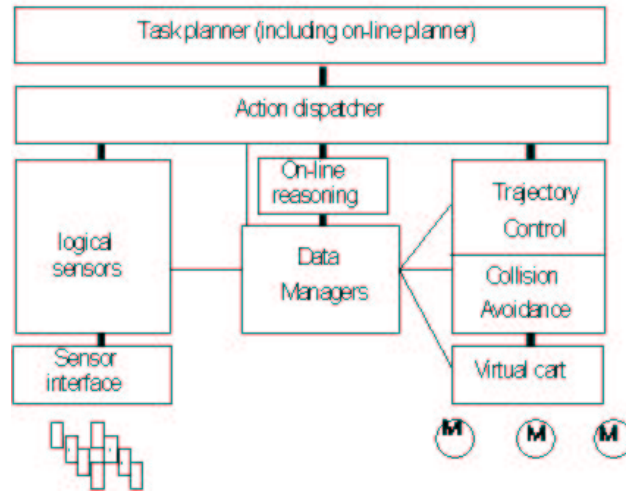


Figure 2: The MARIE architecture

On MARIE several tasks are running in parallel, specialized in sensing, planning and driving the vehicle. The datastreams between the tasks can be flexibly arranged via data managers, central to the MARIE architecture (see figure 2). Through the data manager all tasks communicate and share data. Control of the tasks is provided by the action dispatcher, as described in [9]. The action dispatcher is steered by the plan information that is provided by the task generator which now also includes the on-line planner.

### 3 Plan representation

As this paper focuses on the on-line planner we will now take a detailed look at how plans are defined within MARIE. Plans are represented by a tree structure consisting of nodes. The following figure illustrates the different attributes of the nodes, grouped in three classes.

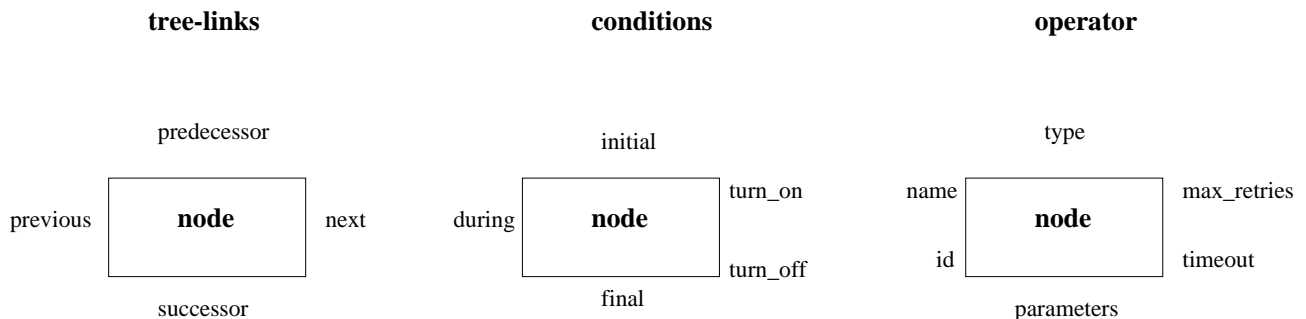


Figure 3: Some of the attributes of a node

Here we will describe the characteristics of the different classes of attributes:

**tree-links** The node can contain up to four pointers to other nodes in the tree.

**conditions** The node contains three different conditions and two lists. The three conditions are called initial, during and final. The first and latter indicate the status flags to be true respectively before and after the node is successfully ended. The during condition flags are inherited as initial condition by all the successors of the node. The two lists, turn\_on and turn\_off, indicate the changes of the status-flags.

**operator** The node also contains the description of the operator it represents. The operator is defined by a number of parameters. The actual parameters used depend on the type of the node.

There are several types of nodes, AND, OR, Action, 'elementary operation' and 'simple operation' nodes. The elementary- and simple operation are the leaf nodes. The leaf nodes have no successors, and represent the actual actions that could be performed by the robot. The difference between the elementary- and simple operations is that elementary operations can be executed concurrently. Several elementary operations can be grouped in an Action-node which defines the type of concurrent behavior. An Action-node can have only elementary operations as successors.

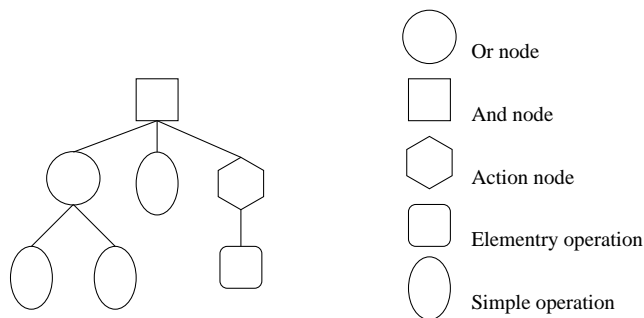


Figure 4: Example of a plan-tree structure

The AND- and OR-nodes are used to make sure actions and simple operations are performed in the correct order. The difference between AND- and OR-nodes is the order of execution. An OR-node executes the first successor it can find with its initial-conditions fulfilled. The successors are always tested in the same order. An AND-node executes its successors one at a time, in a fixed order. When for one of its successors the initial-conditions are not fulfilled, the AND node fails. Once executed a successor is not tried again.

The execution continues (for both the AND- and OR-node), as long as:

- the during-conditions hold,
- the final-conditions are not fulfilled,

- the list of successors is not at its end,
- the number of tries is less than the max\_retries (OR-node only)

Execution of a plan starts at the root (top) node (see figure 4) and then continues through the tree until the root node's final conditions are met (success) or until there are no options left (failure).

## 4 Planners

This section looks at what constitutes an on-line planner. First we will look at the difference between on- and off-line planning, then at the possible variations in on-line planning. Finally we will look at what methods can be used for planning in general. An off-line planner generates a complete plan before the task is performed. An on-line planner on the other hand generates at least part of its plan while the task is being executed. This makes an online planner much more appropriate for dynamic environments such as the one MARIE is operating in. On-line planners exist in various forms. There are those that generate a plan off-line and then adjust it during execution. There are also those that start generating a plan when execution starts. Another possible variation is prediction. In this case the planner tries to predict the next problem and makes a contingency plan so that when the exception occurs the planning is already done. For the actual planning we looked at several possibilities. For instance genetic programming/algorithms [8] [7] have the advantage that they have learning ability, the downside is that they don't integrate very well in the existing MARIE architecture. Algorithms such as POP [10] are very interesting because they have great similarities with the existing architecture. This method starts with the final goals and then searches for actions which satisfy these goals, the initial conditions of these actions are then added to the final goals. This repeats itself until an action is found which is compatible with the current state and satisfies one or more of the final goals. Several constraints are used to minimize the search space by eliminating counter productive actions. Both POP and the existing plan architecture use actions which have pre and post conditions. Furthermore the existing architecture uses during conditions as a measure to prevent counter productive actions. This is why our implementation uses similar techniques to POP but with adaptations for real world use.

## 5 Implementation

What follows next is a description of the actual implementation of the on-line planner for MARIE. The mission goals are specified in the root node of a skeleton plan. This skeleton plan is used by the

planner as a basis to insert new forks into an existing tree structure.

The planner's search space consists of a library of partial plan trees or macros. These macros perform actions like calling the path planner or driving a path or even following a wall for a defined number of metres. Macros are programmed by hand, this is done because they consist mainly of initializations which are not of the high level at which our planner operates.

If during the execution of either an AND or an OR node the final conditions have not been met and all child nodes have been tried then the on-line planner is called. The on-line planner then uses the algorithm as described in figure 5.

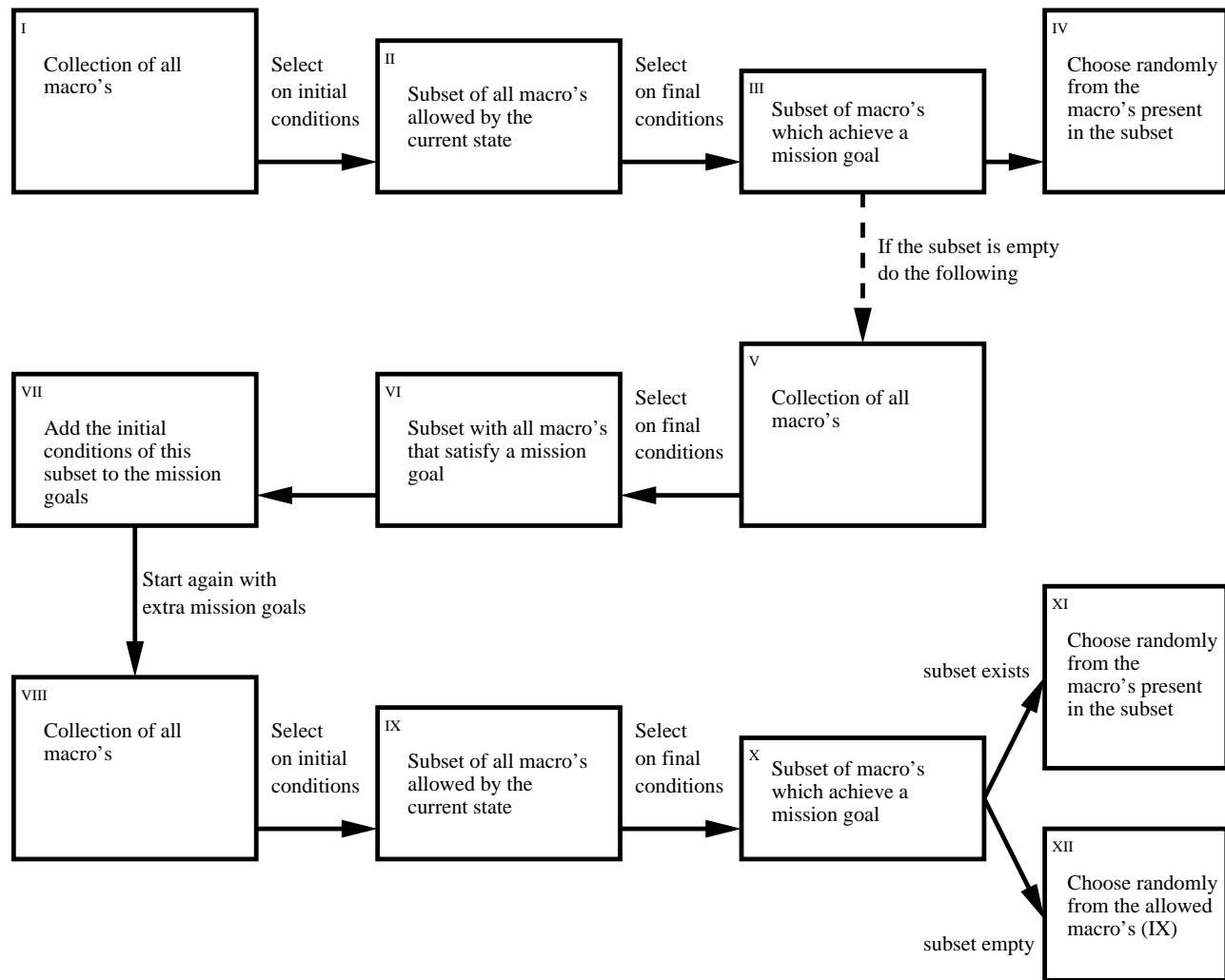


Figure 5: Algorithm of the on-line planner

First it looks at which macros are allowed by the current status of MARIE. Then it looks to see if any of these allowed macros satisfy a mission goal. If this is the case a random selection is made from these macros. When there are no macros which satisfy a mission goal a regressive method is used to create new goals. The planner searches for macros which do satisfy a mission goal but are not allowed by the current robot status. When such macros are found their pre conditions are added to the mission

goals.<sup>5</sup> If these new goals can't be satisfied either, a random selection is made from the macros that are allowed by the current state.

At this point theoretical planners such as POP [10] take a breadth first approach. Yet, for real world operations it is more important to have some variation in the sequence in which actions are performed to allow for non-deterministic effects and information errors. To achieve the variation mentioned above random choice was used for selection of nodes instead of for instance breadth first.

## 6 Experiment

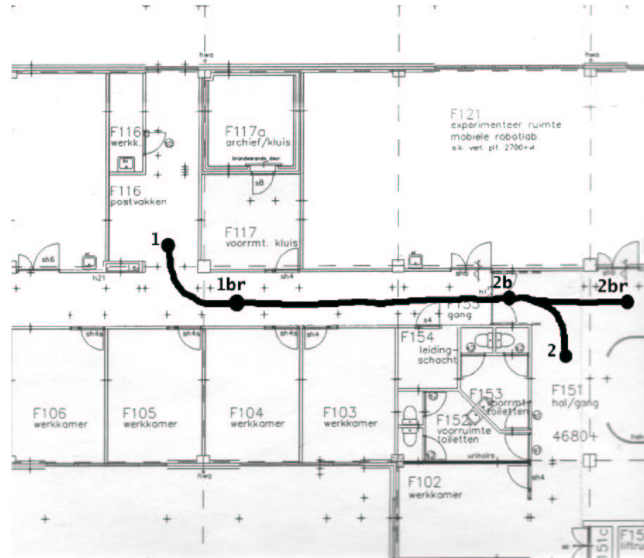


Figure 6: The test begins at 1 and ends at 2 or 2br

To test the capabilities of the on-line planner a test was designed where MARIE would have to travel approximately 20 metres through the hallways of the university. This test has many difficult situations for a robot using sonar as its only sensor. There are irregular brick walls, wooden doors with glass windows, highly reflective metal doorposts and fire extinguishers. All of these elements have the potential to generate exceptions which are difficult to plan for using an off-line planner. As can be seen in figure 6 the test starts with a narrow curve leading on to the main hallway. Through the hallway MARIE has to follow the wall until she comes to a hall where she has the option of driving to either one of two end points.

With these results we will show that the on-line planner was indeed quite successful in providing better reliability and robustness, in order to do this we will give a detailed explanation. Before starting with the experiments we expected that the on-line planner would only be capable of completing partial

---

<sup>5</sup>Unlike POP [10] this isn't done recursively at the moment see section 7

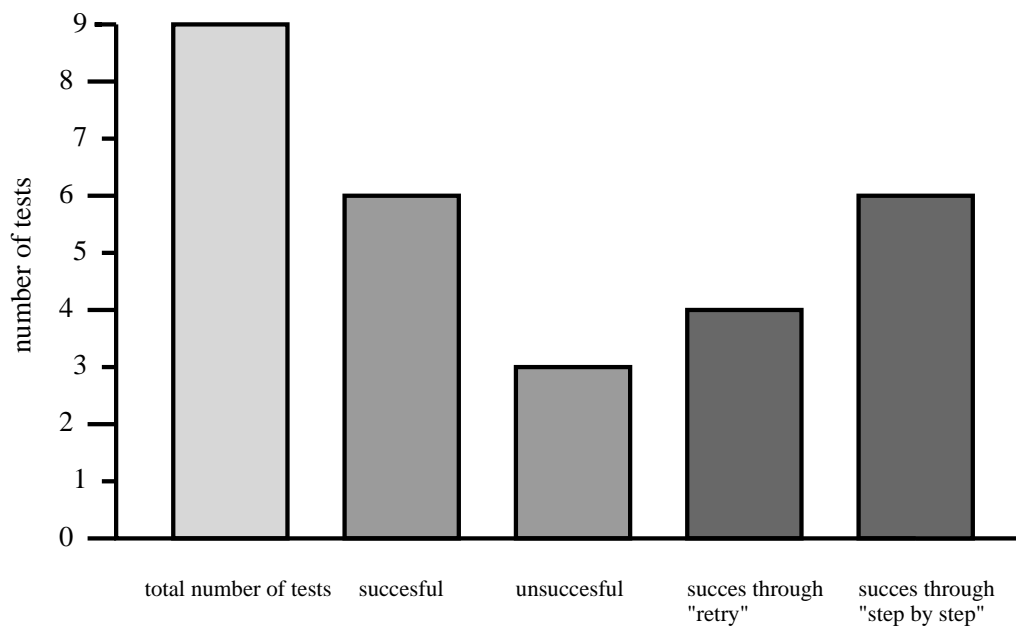


Figure 7: Test results

plans; during testing it became obvious that it could also construct complete plans if it was given a skeleton plan to start with.

The detailed test logs showed three tests failed. All three failed at the first turn of the test, this turn is very tight and has metal doorposts nearby. The latter is the largest problem; the metal doorposts are very reflective and result in inaccurate measurements by the ultrasonic sensors. The result is that during the failed tests MARIE didn't notice the wall on the opposite side of the hallway until she ran into it.

Now for the successful tests. First it has to be noted that an off-line planner using the same search algorithm would have failed at all tests. This is because the on-line planner was needed in all 6 successful tests to get to an end position. The successful tests can be divided in two categories, success through "retry" and success through "step by step". In the case of "retry" it was always the path-planner that failed to find a path, but after the addition of one or more macros by the on-line planner a path was found. "Step by step" always occurred in the last part of the test where MARIE has to drive from point 2b to either point 2 or 2br (see fig. 6). What happened in these cases was that the during conditions of the driving module were violated resulting in MARIE only driving part of the planned path. The on-line planner solves this by repeatedly calling the driving module until the planned path is completed. Because of the random nature of the algorithm MARIE won't always pick the right macro. The average over the tests was that two picks were needed before the right macro was found. This means that during tests MARIE is spending 4 seconds choosing and executing useless macros. Since tests consist largely of driving and take 1:30 min to 2:00 min, 4 seconds is not a great problem.

## 7 Discussion

The tests show that the algorithm performs very well in practice, although several possibilities exist to optimize the algorithm. In this section we will discuss three of them.

One possibility is to add new goals recursively (box VII), instead of the random choice in box XII of fig. 6. The algorithm then becomes a regressive partial order planner similar to POP [10]. Doing this would reduce the choosing of useless nodes. As indicated in the results, 50% of random choices were useless, which allows room for improvement. Yet, in practice the execution time of the useless nodes is insignificant when compared to the time taken up by modules other than the planner, which makes this improvement unnecessary for the time being. Adding a recursive loop creates the possibility of an endless loop. Additional precautions are then required to prevent this.

The on-line planner isn't implemented on MARIE. Instead it's a remote module that uses the wireless network connection to communicate with MARIE. This was mainly done to ease development but also to avoid running into the limits of MARIE's on-board computer. It is also possible to execute the on-line planner on the MARIE-robot locally. This possibility is only attractive for increasing robustness. From the perspective of performance it is not necessary because the amount of exchanged data is not enough to be a bottleneck.

Another option is to let the on-line planner work ahead, which makes it possible to have a plan available at the moment that it is needed. To implement this option we have to add prediction (as described in section 4) to the planner. To be able to predict the plan that is needed by the robot, the planner has to have knowledge about the non-nominal path through the plan. At this moment this knowledge is not available. Also from the perspective of performance it is not needed yet, because the response time of the workstation is nearly instantaneous, due to the small search space.

Although several optimizations of the on-line planning are possible, they are not yet needed. MARIE is held back more by other modules than the planning module. For instance: the positioning module only uses dead reckoning. This works fine to indicate the travelled distance, but the angular information is becoming inadequate for longer distances. The implemented algorithm performs very well in the MARIE architecture in its current state. Changing it will have to be considered only when significant alterations are made to the size of the search space.

## 8 Conclusions

The tests clearly show that MARIE can reach her goals even with only a skeleton plan. This means that MARIE has a great robustness for incomplete plans. Also through the use of the on-line planner

MARIE has greatly improved her reliability. This is shown in the tests by the fact that the on-line planner is used in all successful tests to rescue a plan that would have failed if an off-line planner had been used. This research has demonstrated that the use of a high level planner implemented on a real robot can offer significant benefits.

## References

- [1] Avrim L. Blum, Merrick L. Furst. Fast Planning Through Graph Analysis. *Artificial Intelligence*, 90:281-300, 1997.
- [2] Avrim L. Blum and John C Langford. Probabilistic Planning in the Graphplan Framework.
- [3] G.A. den Boer, G.D. van Albada, L.O. Hertzberger, C. Koburg, M. Mergel, "The MARIE Autonomous Mobile Robot". International Conference on Intelligent Autonomous Systems: IAS-3, 15-19 Februari 1993, Pittsburgh PA, USA.
- [4] G.A. den Boer, G.D. van Albada, L.O. Hertzberger, G.R. Meijer, J.B. Thevenon, P. LePage, E.J. Gaussens, F. Arlabosse, "An exception handling Model applied to Autonomous Mobile Robots", International Conference on Intelligent Autonomous Systems: IAS-3, 15-19 Februari 1993, Pittsburgh PA, USA.
- [5] R.A. Brooks. Intelligence without reason. MIT AI Memo No. 1293, 1991.
- [6] R. Fikes and N. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3/4), 1971.
- [7] Jing Xiao, Zbigniew Michalewicz, Lixin Zhang, Krzysztof Trojanowski. Adaptive Evolutionary Planner/Navigator for Mobile Robots. *IEEE transactions on evolutionary computation*, vol. 1, no 1, April 1997.
- [8] Peter Nordin, Wolfgang Banzhaf. An On-Line Method to Evolve Behavior and to Control a Miniature Robot in Real Time with Genetic Programming. 20-1-1997
- [9] A. Visser, G.D. van Albada and L.O. Hertzberger, "Data and event handling for the MARIE vehicle", in J.I. Soliman and D. Roller, editors, 28th International Symposium on Automotive Technology and Automation ISATA, Proceedings of the dedicated conference on robotics, motion and machine vision in the automotive industries, pages 403-410, (Automotive Automation Limited, Croydon, England), Sep. 1995.

[10] Daniel S. Weld. An Introduction to Least Commitment Planning. AI Magazine Summer/Fall 1994.