A HYBRID TASK PLANNER ARCHITECTURE FOR PICK AND PLACE SEQUENCING

Şule Yıldırım[*], Turhan Tunalı[*] and Pavel Petrovic[**]

ABSTRACT

Robotic task planning for a class of problems involving sequences of pick-and-place operations is examined. Major planners developed in the literature are overviewed from the point of view of the introduced class of problems. A new architecture with reactive aspects is proposed. The basic properties of the architecture are discussed. Finally, an implementation of the architecture is explained.

Keywords: robotics, unstructured robotic environments, operation safety and efficiency, planners, reactive systems, hybrid systems.

1. INTRODUCTION

Present robot arms carry out predefined tasks in well defined environments where all opreations are completed successfully. However, if an unexpected event occurs in the environment, an operator has to interfere with the robot operation and typically reprogram the robot arm manually. If mechanisms can be developed to tackle with the unexpected events, then robot arms can be used safely in unstructured environments with uncertainties. Bearing this fact in mind, robotics and AI people work on building up safe and efficient robotic architectures to solve task planning problem in presence of unexpected events.

This study concentrates on a high level planning domain where a planner receives an initial and a target configuration of the environment and generates a sequential plan of operations that transform the initial state of the environment to the target state. We consider a class of pick-and-place problems which fit some predefined constraints. The planner is expected to cope with a class of unexpected events occuring in the environment.

Unexpected events or actions caused by external intruders change a random part of the environment at a random time. They differ from actions in a plan sequence. Unexpected events cause the generated plans to become obsolete. Consequently, either the current state of the environment can be taken back to the state before the unexpected event occurred or a completely new plan can be generated and executed [1]. The second option is chosen when the previous experience suggests that generating a new plan would take less total time than moving back to the previous state and continuing with the original plan. However, the state of the environment after an unexpected event might be closer to the target configuration and the complete replanning might be still too time consuming. In such a case, the

[*] International Computing Institute, Ege University, Bornova Izmir Turkey,
yildirim, tunali@ube.ege.edu.tr

[**] Department of Computer and Information Sciences, Norwegian University of Science and Technology, Trondheim Norway,
pavel.petrovic@idi.ntnu.no

system finds the “most” time-saving sequence of operations that will move the state of the environment to a state that is on the path of the original plan.

This paper is organized as follows: Section 2 explains and provides examples of planners, reactive systems and hybrid systems in literature. Section 3 discusses related planners further from our problem’s point of view. Section 4 defines our general problem, for which a general architecture is shown in section 5. An implementation is given in section 6. Finally, in section 7, concluding remarks are made.

2. DELIBERATIVE, REACTIVE AND HYBRID SYSTEMS

Current robotic systems are deliberative (plan and use knowledge representation), reactive or hybrid [2]. Deliberative systems rely mainly on symbolic reasoning and world representation whereas reactive systems are reflexive. The speed of a response of a robotic system increases as it becomes more reactive. On the other hand, the predictive capabilities of the systems increase while the systems get more deliberative. Also deliberative systems depend on accurate and complete world models while reactive robotic systems don’t tend to use world models at all.

In common sense, the word “planning” refers to the process of computing several steps of a problem solving procedure before executing any step. There exists vast amount of literature about planners. Basic types of planners are non-hierarchical, hierarchical, skeletal and opportunistic [3]. Non-hierarchical planners use goals directly to find operators as in NOAH [4] and STRIPS [5] planners. In hierarchical planners, planning begins at an abstract level, but later abstract goals are expanded into more detailed subgoals as in ABSTRIPS [6], NONLIN [7] and DEVISER [8]. Skeletal planners store successful earlier plans in a plan database. Before planning begins, goals are compared against the skeletal plans. If one or more plans in the database satisfy the current goals and the current world model, the best plan will be chosen as in MOLGEN [9]. Opportunistic planners develop plans in two stages: parts of a plan may be arranged with backtracking and later parts are linked together and enlarged as opportunities become available.

Reactive systems were developed in response to several apparent drawbacks of deliberative reasoners including a perceived lack of responsiveness in unstructured and uncertain environments due both to the requirements of world modelling and the limited communication pathways. Inaccurate information can cause the deliberative reasoner’s reasoning to be totally incorrect. In a dynamic world with arbitrarily moving objects (e.g. in a crowded corridor), it is potentially dangerous to rely on past data that may no longer be valid. Representational world models are therefore generally constructed from both prior knowledge about the environment and sensory data in support of deliberation. Some researchers view planning only as a way of how to avoid figuring out what to do next.

In the real world of biological agents, the conditions favoring purely deliberative planners generally do not exist [2]. To cope with the problems of real world environments, methods like behaviour-based reactive control are necessary [2]. To make use of best of two methods, hybrid deliberative/reactive robotic architectures have recently emerged combining aspects of traditional AI symbolic methods and their use of abstract representational knowledge, but maintaining the

goal of providing the responsiveness, robustness, and flexibility of purely reactive systems.

3. EXISTING ROBOTIC TASK PLANNERS

While the existing task planners try to solve planning problems for a wide variety of areas, we prefer to limit the planning problem to sequences of pick-and-place operations. We think that, the simplicity of the case can further help us in developing the idea. On the other hand, these type of operations are typical in many areas, for example, assembly planning. An assembly plan is an ordered list of instructions specifying what components to be assembled, what fixtures and tools to be used, etc., such that a product can be successfully assembled with the equipment available in an assembly cell [10]. The ordering of instructions in an assembly plan is important because placing a component to a certain location might prevent another component from being placed to its destination. [11,12] are two approaches which represent assembly plans as an ordered list of components and subassemblies so that generated assembly plans are feasible. On the other hand, knowledge-based approaches [10], where the knowledge involved in the problem can be represented by predicate calculus is necessary for the specification of fixture and tool configurations.

[13] also designs a task-level programming kernel where related domain knowledge, scheduling knowledge, world model parameters and constraint knowledge are entered into the system by a domain expert via a knowledge editor.

In another work, Hwang and Ho propose to use a real time Petri nets (RTPN) to model events, actions, states and temporal constraints [14].

[15] aims at developing agents that can achieve complex tasks in dynamic environments with many unexpected happenings. The agent synthesizes new plans at run time in order to achieve its goals. The dynamic nature of the environment requires the agent to be able to deal with changes in its world in a timely manner and so it has the ability to modify the plan during execution in critical domains where it is infeasible to halt the activities while replanning.

Atlantis [16], a three-level hybrid system incorporates a deliberator that handles planning and world modelling, a sequencer that handles initiation and termination of low-level activities and addresses reactive-system failures to complete the task, and a reactive controller that manages collection of primitive activities (Figure 1). In sequencing layer, conditional sequencing occurs upon the completion of various subtasks or the detection of failure. *Cognizant failure* [17] emphasizes the situation when it cannot complete the assigned task. Task specific or more general monitor routines detect changes in the environment and then interrupt the system if cognizant failure occurs.

The work in [18] uses CAD databases and user queries to define the world to the assembly planning system. In assembly planning, CAD modelling and user queries are inevitable but a vision supported assembly planning system will help getting rid of some of the work now carried out by modelling and queries.

In [19], the mobile robot builds a global world model based on sensory information and uses it for path planning. This approach guarantees global

convergence to the target. However, the reliance on a global model for navigation requires frequent localization of the robot relative to the model, a process which is difficult to attain due to the inherent uncertainties of practical sensors.

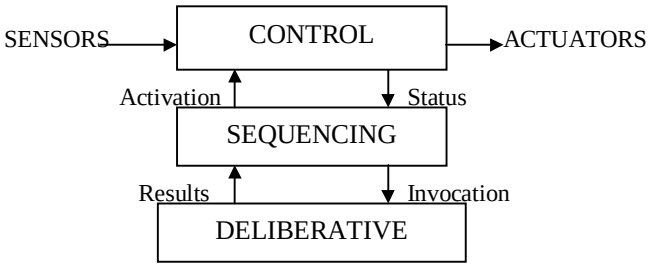

Figure 1. The Atlantis Architecture.

[20] is a work in concurrent assembly sequencing and fixturing. CAD tools are used for the geometric description of the product. In [21], Blythe attempts to describe a planner for domains with external events beyond the control and prior parts of the planning knowledge. He assumes an agent that knows the types of external events and their consequences on the world. He works with a planning domain of an oil tanker that has run aground near the coast and began to leak oil into the sea. The goal is to stop the water and coastline pollution. The available actions include pumping the oil from the tanker into a secondary vessel, surrounding the tanker with booms to contain the oil, using booms and other equipment to prevent the oil from reaching the shore and cleaning up oil from either the sea or the shore. The amount of spilled oil and the path of travel are stochastic processes represented as exogenous events. A branching plan is claimed to be successful.

In [22], behaviours provide an abstract interface for humans to enter plans, and fuzzy logic provides a method for dealing with inexact variable values.

In [23], in order to plan for external events Blythe uses Markov Decision Process models to predict the probability of an event to occur from its pasts occurrences.

The planner in [24] presents action representation methods for actions with uncertain outcomes. In this work, the cause of the changes in the world are defined as unknown initial world states, contingencies or exogenous events. Contingencies occur as a result of the uncertain outcomes of the actions in the plan whereas exogenous events are the events changing the environment independent of these actions.

In [25], the complete integrated planning, executing and learning robot ROGUE, which analyzes execution experience to detect patterns in the environment that affect plan quality is presented. ROGUE extracts learning opportunities from massive, continuous and probabilistic execution traces and correlates them with environmental features to detect patterns in the form of situation dependent rules.

4. GENERAL PROBLEM ASSUMPTIONS

We summarize the basic assumptions for our case as follows: The objects in the environment are cubic blocks located in square cells that form a square grid. The blocks are identified with a type and a number. The world model is kept as a list of entries in a data file. As an example, an entry (*a*, 2, 3, 5) indicates block 2 of

type *a* is on coordinates (3, 5). The number of blocks is user-defined. Each grid cell can contain only one block at any time.

It is assumed that there is a vision system that has the capability to capture the state of the world model whenever requested. The system is capable of generating status information from the captured image. The objects are well positioned within the cells so that there is no grasping problem. There's no occlusion problem as far as the vision system is concerned.

We address a class of cases where an initial and the goal state of the square grid are given as the inputs to the planner, which finds a plan to carry the pieces from their initial state positions to goal state positions. While the plan is being executed, unexpected events that change the state of the environment randomly are allowed. Both the pieces that have been carried to their goal positions and the ones that haven't been carried yet can be mixed-up. This causes the original plan to become obsolete.

The above problem is similar to the Travelling Salesman Problem (TSP), which is an NP-complete problem. In TSP after a city is visited, a new city is selected. In our problem, there are two types of selections when the initial and the goal states of the board are given as inputs to the planning algorithm. The first type of selection is made among a possible set of goal destinations for a piece. More than one piece for a certain type of piece is allowed on the board. If, at the time of planning, there is more than one piece of a certain type, then there will be more than one possible goal destinations, which the planning algorithm has to choose among. In this selection type, the distance between a piece and the possible goal destinations are considered for distance optimization. The second type of selection is made among a set of pieces (which can be all of the same type, all of different types or a mixture of same and different types) to be placed into their goal positions. In this selection type, the distance between the robot arm and the possible pieces is considered for distance optimization. TSP problem only makes a selection of the second type where a city resembles a piece. However the TSP problem doesn't allow more than one city of the same type whereas we allow more than one piece (more than one city) for a certain type of a piece (city). With these limitations and without considering selection type 1, TSP problem turns out to be a more constrained problem than our problem, which makes us consider our problem as NP-complete too. Both of the selections in our problem use the minimum distance heuristic from a piece to a goal destination or from the robot arm to a piece.

The operators used in this problem are pick-and-place commands. An example pick and place command can be pick-and-place (5, 3; 2, 7) which commands that the block is to be moved from location (5, 3) to location (2, 7).

In our study, in addition to the hard physical constraints on the order of pick and place commands [10], we attempt to generate time (energy) sub-optimal sequences of pick and place commands. Since ours is not a pure assembly planning work, some concepts such as specification of fixture and tool configurations are not relevant. However, we aim at a general planning architecture, extensible to as many pick-and-place problems (including assembly

planning) as possible. Such a planner might use a knowledge-based structure as in [10] so that assembly planning constraints on tools and fixtures can easily be defined within this knowledge base. That is, if this planner is also used for assembly planning operations, the database can be configured differently to adapt to different environments.

5. PROPOSED ARCHITECTURE

Our proposed architecture is given in Figure 2. In this architecture, the planner generates an initial plan by taking user defined target configuration and current world model as input. The current world model is supplied to the planner by a vision recognition module. The vision recognition module takes the images of the environment, recognizes objects in the environment and produces a current world model as an initial configuration. Then the generated plan is put into execution.

During the execution of the plan and even at the generation step of this plan, a reactive vision module receives images from the environment. When it is detected that there's an unexpected event occuring in the environment, the robot arm stops execution and moves sideways. Then the controller activates replan decision module which receives current configuration from the recognition vision module. Improve plan module receives the original plan from the planner and generates a better plan when possible.

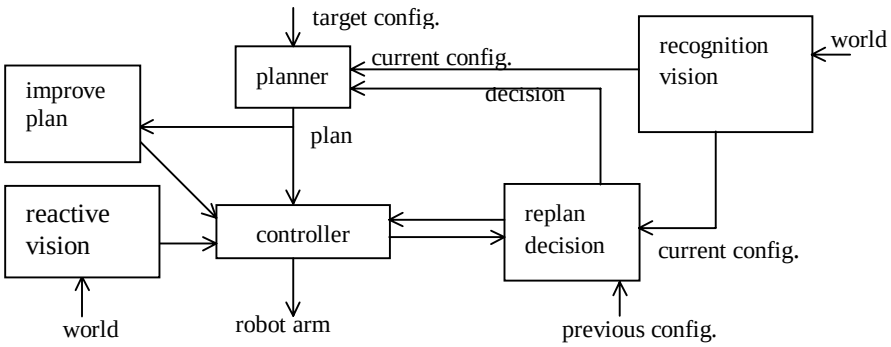

Figure 2. Proposed architecture

The following attempts to compare the existing systems with our approach:

In [13], a knowledge base is used for monitoring and system control operations. We follow the same approach however, our system uses vision information to enter knowledge into the system whereas [13] uses a domain editor.

The Cypress system in [15] becomes aware of an unexpected happening after a run time execution error whereas our system is expected to detect these events before a run-time execution error with a continuous monitoring of the environment via a vision system. Although we use a vision system, there still might be situations that we can't predict some unexpected happenings previously so reactive response and failure recovery might be necessary in our system as well.

In [17], monitor routines for sensing the environment are added to Atlantis architure to perform complex navigation tasks. In our architecture, we both use vision sensing and aim at optimizing the time (energy) of stationary robot arms.

In [21], testing the oil's position to see if it's reaching the shore is made at certain time intervals but in the architecture we propose the monitoring of an unexpected event is made continuously which makes the detection of unexpected events fast. However this work doesn't give many hints about how this test is done. When comparing with our work, a vision system that takes the picture of the sea from above at certain time intervals will detect the spreading of the oil towards sea very fast and then the branching plans will be useful.

In our architecture, we define some simple behaviours as in [22], but contrary to [22], we don't need fuzzy logic because we don't have inexact variable values. As an example, a piece is assumed to be either in a given location or not.

Contrary to [23] and [24], we neither have need for the dependence on the previous occurrences of an event to predict its probability of occurrence in future nor we deal with contingencies.

6. EXAMPLE PROBLEM: LEGO Chess Robot

We implemented our proposed architecture on an example problem which we call it mixed chessboard pieces problem [1]. The square grid mentioned in general problem definition is a chessboard and the pieces are the chessboard pieces.

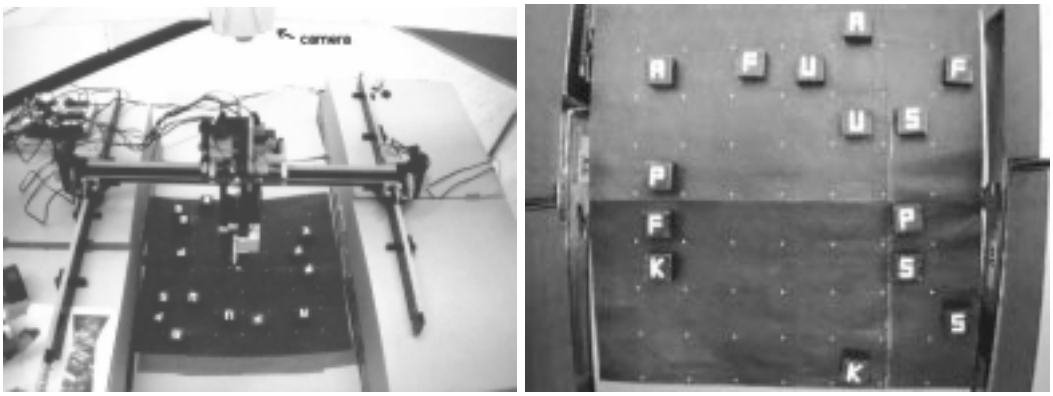

Figure 3. Gantry robot (left), view from the camera (right).

Our experimental gantry robot was built from 2 LEGO Mindstorms robotic construction sets and few additional parts (Figure 3). Two LEGO computer bricks receive commands from the workstation through Interrupt Request (IR) tower and control 5 motors (2 for row axis, one for column and vertical axes, and two to close and open the gripper), and 4 rotation sensors (2 for row axis, one for column and vertical axes). Rotation sensors are used as a reactive feedback for positioning the gripper. The camera is fixed above the scene. During the robot operation, parts of the chessboard are hidden.

The hybrid reactive control architecture for the chessboard example problem with planner is inspired by the Behavior-Based Robotics [2] and is an instantiation of the general problem architecture. It is built bottom-up from independent behavior modules that can run in parallel, have direct access to the robot's sensors and actuators, and send signals/data to each other (Figure 4). According to the initial *plan* that propogates from *heuristic planner* [1] to *plan sequencer*, three primitive behaviors: *arm mover*, *grasper*, *releaser* are activated. Both primitive behaviors and *replan decision* module [1] receive an *unexpected event* signal generated by

the *bitmap comparator* behavior module. Initial *plan* is sent also to a Genetic Algorithm (*GA) plan improver*, which is notified about the progress of execution (*next command*) by *plan sequencer*. Each time a more efficient plan is generated, a *plan update* signal is sent to the *plan sequencer* and *replan decision* module, which receives also the initial *plan* so that it can keep track of the current expected situation on the chessboard. Before the decision is taken, *replan decision* module uses *move arm sideways* primitive behavior and asks the *Neural Network (NN) recognition vision* module to obtain current situation.

Heuristic planner receives the target configuration in a data file from the user and current configuration from the *NN recognition vision* module. It generates a sequence of plan steps which carries the chessboard pieces from their initial positions to their final positions. Replan decision module gives a decision of what to do in case of an unexpected event making a selection among decision alternatives [1].

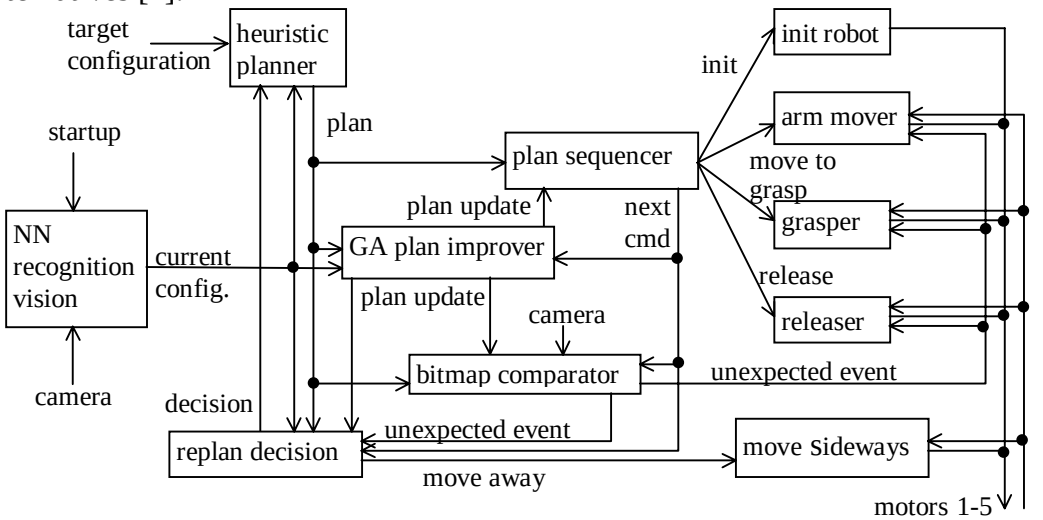

Figure 4. Control architecture for chess example problem.

NN recognition vision module consists of an *image capture module* (implemented using a standard web camera with TWAIN support), *image preprocessing module*, which clips and scales the grid cells out of the bitmap image so that 8x8 gray-scale images are obtained and sent as an input vector to a *3-layer feedforward neural network*. Neural network classifies each grid cell as either empty, one of the block types or unknown. Network is trained with Backpropogation learning algorithm on training images during the design of the system. In order to extend the set of block types, the network has to be re-trained. In our practical experiments, we used network with 7 output categories, 24 hidden layer neurons, learning rate 0.3, and momentum 0.1.

Reactive vision module (*bitmap comparator*) continually obtains an image from the camera and compares the area of borders of the chessboard to the previous frame. If any disturbance is detected besides the movements of the robot that are predicted according to the current plan execution, *start of the unexpected event* is detected and signal is sent to other behavior modules. When the disturbance vanishes, the *end of the unexpected event* is sent out after a short timeout.

The *plan improvement* module is a process running in the background, using up the remaining CPU resources. The module is a Genetic Algorithm [26] that searches the space of all possible plans. Members of population are individual plans that reach the target configuration from the current state. While the plan is executed by the robot, the beginning of the genotype becomes fixed and only the remaining commands are affected by crossover and mutation operators. A special crossover operator is used so that only feasible plans are generated from the two parent plans.

7. CONCLUSION

In this study, a vision supported hybrid architecture is proposed for a class of robotic pick-and-place problems and is implemented in an experimental robotic environment. Our work concentrates mainly on the architectural issues where the reactive aspect of the control is combined with high level planning. In future work, we will elaborate more on the high-level planning side and enhance our algorithmic approach with knowledge base containing rules and constraints of a particular domain.

REFERENCES

[1] Ş. Yıldırım and T. Tunalı, "A new methodology for dealing with uncertainty in robotic tasks", XIV. Int. Symp. on Comp.& Inf.Sci., Kuşadası, TURKİYE, 1999.

[2] R. C. Arkin, "Behaviour-Based Robotics", The MIT Press, Cambridge, Massachusetts, 1999.

[3] H. Jack, "A Historical Review of Artificial Intelligence Planning", 1998, Grand Valley State University.

[4] E. Sacerdoti, "A Structure for Plans and Behaviour", Elsevier, North-Holland, New York, 1977.

[5] R. Fike, P. Hart and N. Nilsson, "Learning and Executing Generalized Robot Plans", Readings in Artificial Intelligence, Nilsson and Webber, eds., Tioga Publishing, Palo Alto, California, 1981, pp. 231-249.

[6] E. Sacerdoti, "Planning in a Hierarchy of Abstraction Spaces", Artificial Intelligence, V. 5, pp. 115-135, 1974.

[7] A. Tate, "Generating Project Networks", Proceedings IJCAI-77, Cambridge, Massachusetts, 1977, pp. 888-893.

[8] S. Vere, "Planning in Time: Windows and Durations for Activities and Goals", IEEE Transactions on Pattern Analysis and Machine Intelligence, V. 5, pp. 246-267, 1983.

[9] M. Stefik, "Planning and Metaplanning", Readings in Artificial Intel., Nilsson and Weber, eds., Tioga Publishing, Palo Alto, California, 1981, pp. 272-286.

[10] Y. F. Huang and C. S. G. Lee, "A Framework of Knowledge-based Assembly Planning", Proceedings of the IEEE International Conference on Robotics and Automation, Sacramento, California, April 1991.

[11] J. D. Wolter, “On the Automatic Generation of Plans for Mechanical Assembly”, Ph.D thesis, Univ. of Michigan, Dept. of Computer, Information and Control Engineering, Sept. 1988.

[12] L. S. Homem de Mello, “Task Sequence Planning for Robotic Assembly”, Ph.D thesis, Canegie Mellon Univ., Dept. of Electrical and Computer Engineering, May 1989.

[13] C.P. Hwang and C. S. Ho, “ Development of a Task-Level Programming Kernel for Robots using RTSDE”, Proceedings of 1st Chinese World Congress on Intelligent Control and Intelligent Automation, Beijing, Aug. 26-30, 1993.

[14] C.P. Hwang and C. S. Ho, “ RTPN-based Task Plan Modeling and Verification for Manufacturing Cells”, Proceedings of International Symposium on Artificial Intelligence, Monterrey, Mexico, Sep. 20-24, 1993.

[15] D. E. Wilkins, K. L. Myers, J. D. Lowrance and L. P. Wesley, “Planning and Reacting in Uncertain and Dynamic Environments”, SRI International, 333 Ravenswood Ave., Menlo Park, Ca. 94025, 1994.

[16] E.Gat, “Reliable Goal-Directed Reactive Control of Autonomous Mobile Robots,” Ph.D thes., Virginia Polytechnic Inst. & State Univ., Blacksburg, 1991.

[17] E. Gat and G. Dorais, “Robot Navigation by Conditional Sequencing,” Proceedings of the IEEE International Conference on Robotics and Automation , 1994, pp. 1293-99.

[18] R. H. Wilson, “Minimizing User Queries in Interactive Assembly Planning”, IEEE Transactions on Robotics and Automation, Vol. 11, No. 2, April, 1995.

[19] I. Kamon, E. Rivlin, “Sensory-Based Motion Planning with Global Proofs”, IEEE Transactions on Robotics and Automation, Vol. 13, No. 6, December 1997.

[20] B. Romney, “Atlas: An Automatic Assembly Sequencing and Fixturing System”, Proc. Intl. Conf. on the Theory and Practice of Geometric Modelling, Tübingen, Germany, October 1996.

[21] J. Blythe, “A Representation for Efficient Planning in Dynamic Domains with External Events”, CMU, 1996.

[22] D. S. Blank and J. O. Ross, “Learning in a Fuzzy Logic Robot Controller”, the Proceedings of 1997 Meeting of the American Association of Artificial Intelligence, 1997.

[23] J. Blythe, “Planning under Uncertainty in Dynamic Domains”, Ph.D Thesis, CMU, 1997.

[24] L. Pryor and G. Collins, “Planning for contingencies: A decision-based approach”, Journel of Artificial Intelligence Research Vol. 4, AI Access Foundation and Morgan Kaufmann Publishers, 1996.

[25] K. Z. Haigh, “Situation-Dependent Learning for Interleaved Planning and Robot Execution”, Ph.D thes., School of Com. Sci., Carnegie Mellon Univ., 1998.

[26] D E Goldberg, ”Genetic Algorithms in Search, Optimization, and Machine Learning”, Addison-Wesley, Reading, MA, 1989.