Achieving Goals Through Interaction with Sensors and Actuators

John Budenske
Maria Gini
Department of Computer Science
University of Minnesota
4-192 EE/CSci Building
200 Union Street SE
Minneapolis, MN 55455

[Accepted by the 1992 IEEE/RSJ International Conference on Intelligent Robotics and Systems]

Abstract - **In order for a mobile robot to accomplish a non-trivial task, the task must be described in terms of primitive actions of the robot's actuators. Our contention is that the transformation from the high level description of the task to the primitive actions should be performed primarily at execution time, when knowledge about the environment can be obtained through sensors. Our theory is based on the premise that proper application of knowledge increases the robustness of plan execution. We propose to produce the detailed plan of primitive actions and execute it by using primitive components that contain domain specific knowledge and knowledge about the available sensors and actuators. These primitives perform signal and control processing as well as serve as an interface to high-level planning processes. In this work, importance is placed on determining what information is relevant to achieve the goal as well as determining the details necessary to utilize the sensors and actuators.**

I. KNOWLEDGE AT EXECUTION TIME

To be useful in the real world robots need to be able to move safely in unstructured environments and achieve their given tasks despite unexpected changes in their environment or failures of some of their sensors. The variability of the world makes it impractical to develop very detailed plans of actions before execution since the world might change before execution begins and thus invalidate the plan.

In this paper we describe our theory for determining the sensor and actuator commands necessary to execute a given abstract-goal and then execute it. The abstract-goal is a single, high level goal of the form that could be produced from a classical planning system. Our theory is based on the premise that proper use of knowledge increases the robustness of plan execution without reducing the ability to react to the environment [1], [2], [3].

Usually, execution requires some amount of information from the external world. If at planning time the robot had a perfect model of the world, the processing to execute the abstract-goal could be greatly reduced and could even be performed in its entirety at planning time. In practice, since the world is dynamic and unpredictable, a perfect model of the world will never exist. Attempting to determine, prior to execution, everything needed to achieve a goal across the domain of abstract-goals, external world states, and sensor availability would be an endless task. The central problem is defining how to transform the given abstract-goal into an explicit representation of what is necessary to achieve the goal. We call this transformation process Explication.

This work was funded in part by the NSF under grants NSF/CCR-8715220 and NSF/CDA-9022509, and by the ATT Foundation.

Executing an abstract-goal is difficult because the abstract-goal implicitly represents a large collection of information and details. To be executed, a plan must explicitly specify the details of how to utilize the sensors and actuators. For the same abstract-goal, different environment situations may require different sensors, different programming and control of the sensors, and different strategies. Explication is also difficult because of the need to adapt to the environment as the command is being executed. Because of this dependency on the current situation it must occur at execution time.

In order to further understand the Explication process, we have broken it down into subprocesses. Explication for different abstract-goals is accomplished through different combinations of the subprocesses. The Explication process consists of:

(a) *determining* the relevant information from the abstract-goal and current world state as sensed by the robot's sensors;

(b) *decomposing* the abstract-goal into sub-goals;

(c) *selecting* the source of the relevant information;

(d) *collecting* the relevant information and executing primitive commands on sensors or actuators;

(e) *detecting and resolving* conflicts which occur between sub-goals;

(f) using the information collected in a *feedback control* loop to control the sensors and actuators;

(g) *monitoring* the relevant information to detect goal accomplishment and error conditions.

Though each of these subproblems is individually solvable, the real challenge is in combining them. For example, an abstract-goal for turning the robot towards a moving object could be defined as a *feedback control* with very little knowledge used within *decomposing, determining, and selecting*. In comparison, an abstract-goal for moving the robot through a doorway can contain detailed knowledge on *decomposing*, *determining*, *detecting* and *resolving conflicts*, etc. Explication knowledge decomposes it into sub-abstract-goals of finding the doorway, and moving towards it, as well as feedback control mapping of sensor-sweeping an area, searching for the doorway until found, and progressively moving through the doorway until clearing it.

Two concepts are crucial for Explication. First is the notion of Relevant Information Need. Explication must deal with the question of "what information is relevant and thus needed in order to execute this abstract-goal?". Explication views such information, and thus the need for it, abstractly. Implicit goal abstraction is replaced by explicit informational need abstraction. To put it another way, the abstract goal: "how to achieve X", is replaced by two things: a) explicit knowledge on "how to achieve" using the sensors and actuators and b) an explicit representation of the needed information. This representation uses abstraction to reduce dependency on the source of the information. Abstract informational needs can then be met by abstract information providers (i. e. sensors). We have utilized the concept of "Logical Sensor" [4], [5] to be such an abstract information provider.

Equally crucial is the second concept of Utilization-Detail. Explication must deal with the question of "what details of using sensors, actuators, and processes are necessary in order to execute this abstract-goal?" Again, abstraction is used to explicitly represent the need of detailed control of sensors and actuators, where control primitives are commands and their parameters for both discrete and continuous actions. We have extended the Logical Sensor concept to allow other logical entities, thus accounting for this explicit abstraction.

Since the abstraction of a goal is hierarchical, Explication uses the above two concepts at each level of the hierarchy. On a single level of abstraction (i.e. for a single abstract-goal), Explication uses knowledge to explicitly represent the information and detail needs. These needs are met by Logical Sensors and other logical entities. These entities either map directly to the corresponding information/details or they each apply Explication hierarchically. Thus, more knowledge is used to explicitly determine further information and detail needs.

We propose a framework for Explication, called the Logical Sensors/Actuators (LSA). The framework is being implemented in an object oriented programming environment resulting in a flexible robotic sensor/control system which we call the Logical Sensor Actuator Testbed (LSAT). The framework consists of the knowledge, mechanisms, and structures used by the Explication process.

Primarily, the framework is a collection of reconfigurable components and flexible component structures which can be combined to achieve abstract-goals through execution. One of the purposes of the framework is to organize/differentiate between the domain dependent and independent mechanisms/structures. This will allow us to implement a platform-independent testbed, and to perform empirical studies on the use of knowledge during execution and the identification of relevant information and Utilization-details.

II. IMPLEMENTATION OF LSAT

The LSAT framework is being implemented on a SUN SPARC 4/330 computer which interacts with a TRC Labmate mobile robot over two separate serial lines. The system is being written in C, Lucid Common Lisp with the Common Loops Object System (CLOS) and LispView windowing system. An object oriented approach has been used in the implementation, where an object corresponds to a Logical Sensor/Actuator (LSA) entity.

The Labmate mobility control is through setting registers for velocity, acceleration, turning radius, and operation modes. It has dead reckoning (wheel and steering counters), bumper sensors, and two active sensors. The first is a ring of 24 polaroid acoustic sensors which have a range of six inches to thirty-five feet. These sensors encircle the robot. The second is a set of infra-red, single beam proximity sensors which can detect the existence (but not range to) of an object up to thirty inches away. These sensors are mounted on the corners of the robot facing forward, to the sides, and to the rear (8 sensors in all).

Within the LSAT framework, we have developed five classes of LSA which are used to implement the framework’s objects. The first, SENSOR, takes raw/processed data as input and outputs data which is further processed (ie. sensor processing). The next class, DRIVER, accepts as input multiple commands for the actual hardware/drivers. This class acts as an interface to the hardware and performs command scheduling, minor command conflict resolution, and routes major conflicts to its controlling LSA.

Another class is the GENERATOR, which accepts sensor data as input and outputs a command meant for a DRIVER. This class can be viewed as a low level, feedback control, looping mechanism between a sensor and the actuator. The MATCHER class is much like the SENSOR class in that it takes as input sensor data and processes them for output. The difference is that it also takes as input a description of a goal or error situation, and the processing consists of matching the goal/error to the input sensor data. The output is simply a measurement of the matching process. The last class is the CONTROLLER. This class accepts processed data from any other class, as well as commands and parameter-values from other CONTROLLERs higher-up in the hierarchy. The output is the control commands and parameter-values (referred to earlier as Utilization-Detail) to its sub-LSAs. CONTROLLERS are the main entity used to implement Explication. They control the other LSAs, and manipulate them through the process of Explication.

The implementation of Explication has been defined as a set of functions which correspond to the subproblems of Explication described earlier. We have developed an algorithm which combines these functions into a complete process and are experimenting with it on the Labmate robot.

III. AN EXAMPLE

To illustrate our approach, we will use an example derived from our lab experiments. We have been developing the knowledge necessary for a robot to robustly find and pass through an arbitrary doorway. This abstract-goal is called: (MoveThrough Door1).

In order to explain the execution of this abstract-goal, we assume that a set of mid-level LSAs have been implemented. This set would include a *MoveToVicinity*, which moves the robot into a designated vicinity; an *Avoid*, which detects and avoids obstacles as the robot is moving; a *ProximitySafe*, which upon detection of a close obstacle (within millimeters) takes control of the robot to guide it away from the obstacle until the MoveToVicinity and Avoid can return to guiding without error. Also available is a *DetectDoor*, a *VerifyDoor*, a *MonitorDoor*, and a *CloseQuartersMove* which guides the robot through the door opening. Each of these mid-level LSAs further decomposes into low level sensor processing, sensor control, robot command generation, and actuator driver LSAs. For brevity we will not discuss the low level details.

When presented with the abstract-goal, the system first determines what information is initially needed. The abstract-goal contains a movement command "MoveThrough" and an object "Door-1". The system sends the abstract goal to a MoveThrough LSA which will then control the execution for that goal.

The overall strategy for moving through a location is to first identify the relative vicinity which can then be searched for the exact location. The abstract-goal is analyzed to determine what information is relative to accomplishing the goal. It is determined that there is an object to detect (door), and that the object has an approximate vicinity for searching for it (from a priori knowledge). Since there is an object involved, the robot will search for the specified object.

This type of command involves movement, and so a movement LSA is needed to propel the robot to and through the door. Movement over a long distance will require the use of the Avoid LSA, while a shorter distance would only require the ProximitySafe LSA. A decomposition is then selected which includes approaching the vicinity of the doorway (MoveToVicinity), in parallel with searching and detecting the doorway (DetectDoor). A controller for the sonar senors (ProximitySonar) spawns sub-LSAs of 24 RangePsonar LSAs and a driver to coordinate their activity (ProximitySonarDrive). This will provide input data for the other two LSAs. This is shown in Fig. 1.

The Sub-LSAs of MoveToVicinity include, the Avoid and ProximitySafe LSAs, which protect the robot from collisions. The ProximitySafe LSA contains knowledge about what type of input is needed (ProximityInfraRed). If the LSAs necessary to provide this data are not yet created, the ProximitySafe LSA will initiate their creation. Since the MoveThrough LSA already created the ProximitySonar LSA, the Avoid determines that its input needs can be resolved through using the ProximitySonar LSA. Once a door-like object is found, the MoveToVicinity (and its sub-LSAs) are deactivated. The DetectDoor LSA is kept to aid in monitoring the next phase.

In the second phase, shown in Fig. 2, a VerifyDoor LSA is activated to make sure that the doorway truly is a doorway, that it is open, and that the robot could fit through it. The process of verifying the doorway involves three sub-phases: a) moving the robot next to the doorway, within the range of the proximity sensors (Touch and Perpendicular), b) move the robot back and forth in front of the opening to verify its existence (Slide), and c) center the robot to the doorway (Slide and Perpendicular). Both proximity and sonar sensors are used. Based on the movements within the Slide LSA, a precise measurement of the doorway opening size can be calculated, and the position of the door frame can be determined. This can be used to line the robot up to the doorway. Each of the sub-phases of VerifyDoor is accomplished through the coordination and control of each of the sub-LSAs. At different points in time during a single sub-phase, each sub-LSA is performing a specific behavior. Subtle changes in behaviors can greatly influence the performance within a sub-phase, and changes in behaviors are controlled by the VerifyDoor LSA through selection of Utilization-Detail values. Thus VerifyDoor coordinates all activities associated with achieving the goal of verifying that the doorway is truly in front of the robot.

It is important to note that the switch from the initial LSAs to the Verify-door LSA is all controlled by the MoveThrough LSA. Within this LSA, the execution of the lower LSAs is monitored and upon completion of that phase of the task, actions are taken by the MoveThrough LSA to switch to, and monitor the next phase.

Once the door frame is verified, the VerifyDoor and the DetectDoor LSAs are deactivated and three new LSAs are activated to complete the achievement of the overall MoveThrough goal, as shown in Fig. 3. The first is an LSA for determining the type of door frame opening the robot is attempting to pass through (DoorType). Doors which open in vs doors which open out will appear differently to many of the sensor processing LSAs and thus a prediction of the door type allows more specific processing to occur. Doors in corners and at end of hallways pose processing challenges as well. Second, a monitor LSA, MonitorDoor, uses the sonar sensors to monitor the robot's placement and movement through the doorway, and also detect any new obstacles which may appear. This serves as a check on the progress of the third new LSA, CloseQuartersMove. Upon activation, the CloseQuartersMove LSA activates and controls a number of parallel sub-LSAs, which accomplish these sub-goals: (1) propel the robot through the doorway to a predicted "other side" (Propel); (2) keep the robot in line with the door opening as the robot moves through the doorway, and monitor for obstacles in the forward path (FrontAlignment); (3) monitor the detection of the doorway frame passing each of the side-mounted proximity sensors (FramePassings); (4) resolve various types of mis-alignments of the robot to the door frame (Shift); (5) monitor the movement of the robot for differentiating between pauses (used by Shift and Propel) and low velocity stagnation (MoveDetect). Often during low velocities, the wheel motor drives will freeze up and require a reset of the velocity register to resolve.

Initially, the robot would be positioned in front of the door frame such that the front sonars would detect the frame, but as the robot passes through the door frame, the sonar beam would become clear of the frame, and could then be used to detect potential obstacles in the robot's path. Synchronization of this sensor usage would occur through feedback control mapping of the sensor data into processes which detect each of these occurrences. Such detection triggers the change in control of the LSAs.

In the lab experiments, we identified both relevant information needs and Utilization-Details which are abstracted by the goals. Within the goal of CloseQuartersMove, there are a number of implicit relevant information needs. These include: location of the left/right sides of the door frame, the size of the door frame, the relative location to other structures (such as walls), whether anything else was obstructing the path, how the robot is aligned to the door frame during the movement, and at what points in time are the various sensors passing the door's frames. All of this information is needed to aid in the selection of various utilization-details and thus in achieving the goal.

Within CloseQuartersMove, each relevant information need is explicitly, and yet abstractly, identified and then matched to a LSA which will then provide the information. As informational needs are resolved, utilization-details are identified as necessary for the execution of the goal. Again, these implicit details are explicitly identified. Some of the important details are: steering directions, speed, acceleration, synchronization of turning vs position within the door frame, and selection of other LSAs to collect data or control sub-processes (such as obstacle avoidance or door frame detection).

At times, movement of the robot is used in conjunction with the sensors to further collect information on the environment (ie. verification of the doorway's existence). The selection of sensors is based on knowledge about them (Utilization-Detail-Knowledge), in this case information on the range and location of the sensor on the robot.

Reactivity to the external world is maintained through constantly managing the sensors, collecting data, and using that data to further guide the robot through its execution. Often, feedback control loops with very short data-flow paths are needed. This example can be augmented to include additional LSAs for monitoring potential errors, and the recovery from those errors upon their detection.

IV. RELEVANT RESEARCH

When the tasks given to a robot are more complex than just reaching a given position while avoiding obstacles, sensor processing becomes more difficult. Although sensing strategies can be developed at planning time and added to an existing plan [Doy86], it is preferable to reason about sensors during the development of the plan [6] or to plan the sensing strategies dynamically when current information about the world becomes available [7].

Albus and his group [8] have developed a hierarchical, highly-synchronized control structure for the sensors which processes the sensor data as they flow though the structure. The highest level decisions are carried out in the top module, and the longest planning horizons exist there as well. The system, thus, is a large, highly-synchronized, static, configuration of routines and subroutines which is not only imposed on the inter-module structure, but on the control and data paths, and the overall goal decomposition strategy.

Recent research efforts have produced a variety of alternatives to classical planning. The most popular and successful approach is the subsumption architecture [1] which avoids planning by using layers of behaviors. The key idea is that layers of a control system can run in parallel to each other and interact when needed. Each individual layer corresponds to a level of behavioral competence. Though the distributed method of control for the robot allows for many behaviors to run in parallel, it also disallows central control of the robot to achieve planned complex tasks.

Georgeff [9] has implemented a planning and control system initially in simulation, and then on the SRI FLAKEY mobile robot. This system is based on a functional decomposition of high level control into primitives, and is being extended to integrate reactive behaviors into the control structure. The REX system [10] decomposes high level goals into mobile robot commands, and then converts them into hardware logic designs. Reactivity is achieved by the augmentation of embedding the planning system into a reactive control system (called Situated Automata Theory).

Henderson [4,5] combines a declarative description of the sensors with procedural definitions of sensor control. This research, termed "Logical Sensors" consists of logical/abstracted views of sensors and sensor processing, much like how logical I/O is used to insulate the user from the differences of I/O devices and operating systems. It provides a coherent and efficient data/control interface for the acquisition of information from many different types of sensors. The specifics of the implementation can change (ie. change sensors, or sensor processing algorithms) without affecting the symbolic-level control system. This allows for greater sensor system reconfiguration, both as a means of providing greater tolerance for sensor failure, and to enhance incremental development of additional sensing and processing devices. Work has been done to extend the concept to include actuator control [11]. Lyons [12] proposes a formal model of the computation process of sensory based robots.

Firby [2] proposes a two component planning system that can react to changes and take advantage of opportunities. In his "Reactive Action Package", there is a library of methods (called RAPs) for accomplishing goals, and an Interpreter which examines a goal and selects methods from a library to apply to the goals. Each RAP in the library contains knowledge on selections of goal decompositions and robot actions which will achieve a goal, but does not address issues of sensor and actuator interactions.

Gat [13] developed a three level architecture for controlling autonomous mobile robots. The top level performs deliberative activities such as planning and world modeling, the middle level draws directly from Firby's RAP system, while the lower level is a stateless reactive control mechanism which controls activities with no decision-making. This design addresses the need to have tighter interaction with sensors and actuators (ie. between the middle and lower levels). Both this research and our work aim at issues of combined reactive and goal-directed behavior, sensor noise, actuator errors, and robust performance. The primary difference is that propose a homogeneous architecture of Logical Sensors/Actuators.

Our work is closely related to the Logical Sensors and to the Perception and Motor Schemas by Arkin [3]. Motor schemas are reactive oriented low level planning components, which can be goal driven, and thus are very similar to the Logical Sensors. The emphasis of the research is on robot navigation via potential field manipulation. Control is achieved by a single control module selecting from a single layer of motor schemas.

V. CURRENT RESULTS AND CONCLUSIONS

Currently, we are implementing the process of Explication within the LSAT framework. We are working on the Move-through example as our first major milestone for abstract-goal achievement. Since the robot only utilizes the dead-reckoning, proximity infra-red, and proximity ultra-sonic sonar sensors, the amount of information obtained from sensing is usually not rich enough to allow confident and complete decision making to occur from a single frame/collection of data. Thus, multiple collections from multiple viewpoints is required. Such a problem domain provides a rich series of decision making, relevant information collections, actuator manipulations, and knowledge utilizations for the robotic system to progress through in order to achieve the goal. Thus, this problem domain exercises the LSAT system's capabilities and allows us to experiment with building robustness in the accomplishment of the goals through the acquisition of knowledge.

Within the lab experiments, the robot is able to perform all the standard maneuvers to achieve the Move-through goal for standard doorways. This includes seeking out the doorway, verifying its existence and passability, and performing the final passage through the door frame.

A great deal of Utilization-Detail-Knowledge has been built into the current system. This includes the knowledge to perform the standard maneuvers for seeking, verifying, and passage through a standard doorway. Since the utility of this theory is on its ability to resolve non-standard and unusual situations through the application of knowledge, current efforts are concentrating on building and structuring this knowledge. We have implemented knowledge on monitoring and resolving situations when the doorway is difficult to find, as well as resolving when the doorway is too small to pass through.

Additional knowledge is being inserted to attempt auxiliary strategies for bumping the door (ie. to open/widen the doorway), and for searching for the most probable location of the door when it is closed (a very difficult task utilizing the given sensor suite). We are also expanding the knowledge base on the types of doorways. Currently the robot deals with doorways of standard size which are not in unusual situations (ie. location in a corner of room, or obstacles directly in front or on other side of doorway). As knowledge is added to monitor for and resolve unusual situations, the goal accomplishment robustness will increase.

Other areas where the role of knowledge is expanding is in the coordination of the sub-LSAs. Though we anticipated conflicts to occur between many of the LSAs during execution, we are finding that applying the knowledge to anticipate and avoid as opposed to detect and resolve such situations has been easier to do and has resulted in better system performance.

One difficult issue is the responsiveness/reactivity of the system. The system is able to customize data-flow paths such to reduce the amount of processing on a single path, thus increasing reactivity for that feedback control loop. Unfortunately, all LSAs and thus all data-flow paths are resident on a single SUN SPARCstation, and so reducing single data-flow paths without reducing the total LSAs does not eliminate the processing bottleneck. Until mechanisms are built into the system to utilize multiple processors (ie. on a Sun Network, or onto multi-processor hardware), knowledge concerning the recognition of time critical activities is being inserted. Thus when an activity requires high reactivity, the system can temporarily shutdown LSAs which are not critical to that activity.

Though still in the initial stages of implementation, we have already discovered some interesting characteristics of the LSAT framework and Explication. For example, many of the LSAs (specially higher level ones) are becoming mini expert systems. They contain a great deal of knowledge on how to achieve a specific goal, including recovering from errors.

As we implement the process of Explication we have improved upon our design. Initially, the knowledge is in a conceptual form of rules. In attempting to implement this knowledge, we devised an expert-system-like inference engine which we augmented for dealing with LSA interactions (ie. the collection of up-to-date information, interaction with sensors, execution of primitives, and the manipulation of, and by, LSAs). This implementation proved to be very costly in processing time, and a further augmentation was used to combine/compile the knowledge from groups of rules into state machines residing in single rules called "E-monitors". These new rules are then manipulated by the augmented inference engine, and with fewer rules, the overall overhead was reduced. This is now leading to a new implementation centered around the primary content of the rules, utilization-details. The new implementation will consist of highly structured rules for the existence of LSAs and their parameter values during specific world situations. Currently, the implementation is driving the design of Explication, and we are still analyzing this effect on the Explication theory.

The development of the LSAT framework, Utilization Detail Knowledge and Explication will provide greater insight to the use of sensors for accomplishing intelligent behavior of mobile robots. We hope to identify how abstract plans should be executed, and what sensor, actuator, and processing knowledge is necessary to achieve them.

REFERENCES

[1] R. A. Brooks, "A Robust Layered Control System for a Mobile Robot," *IEEE Journal of Robotics and Automation*, Vol RA-2, N 1, 1986, pp 14-23.

[2] R. J. Firby, "An Investigation into Reactive Planning in Complex Domains," Proc *Sixth National Conference on Artificial Intelligence*, Seattle, July 1987, pp 202-206.

[3] Ronald C. Arkin, "The Impact of Cybernetics on the Design of a Mobile Robot System: a Case Study," *IEEE Trans on Systems, Man, and Cybernetics*, Vol 20, N. 6, 1990, pp 1245-1257.

[4] T. Henderson and E. Shilcrat, "Logical Sensor Systems," *Journal of Robotics*, Vol 1, N 2, 1984, pp 169-193.

[5] T. Henderson, C. Hansen, B. Bhanu, "The Specification of Distributed Sensing and Control," *Journal of Robotic Systems*, Vol 2, N 4, April 1985, pp 387-396.

[6] S. Hutchinson and A. Kak, "Spar: A Planner that Satisfies Operational and Geometric Goals in Uncertain Environments," *AI Magazine*, Vol 11, N. 1, 1990, pp 31-61.

[7] S. A. Hutchinson and A. Kak. "Planning sensing strategies in a robot work cell with multi-sensor capabilities," *IEEE Trans on Robotics and Automation*, RA-5(6), pp 765-783, 1989.

[8] R. Lumia, J. Fiala, A. Wavering, "The NASREM Robot Control System Standard," *Robotics Computer-Integrated Manufacturing*, Vol 6, N 4, 1989, pp 303-308.

[9] M. Georgeff and A. Lansky, "Reactive Reasoning and Planning," Proc *Sixth National Conference on Artificial Intelligence*, Seattle, July 1987, pp 677-682.

[10] S. Rosenschein and L. Pack Kaelbling, "Integrating Planning and Reactive Control," Proc. *NASA Conference on Space Telerobotics*, Vol 2, G. Rodriguez and H. Seraji (eds), JPL Publ. 89-7, Jet Propulsion Laboratory, Pasadena, Ca, pp 359-366.

[11] Peter K. Allen, "A Framework for Implementing Multi-Sensor Robotic Tasks," Proceedings of the *DARPA Image Understanding Workshop*, February 1987, pp 392-398.

[12] D. M. Lyons and M. A. Arbib "A Formal Model of Computation for Sensory-Based Robotics," *IEEE Trans on Robotics and Automation*, Vol RA-5, N 3, 1989, pp 280-293.

[13] Erann Gat, "Integrating Reaction and Planning in a Heterogeneous Asynchronous Architecture for Mobile Robot Navigation", *SIGART Bulletin*, Vol 2, N 4, 1991, pp 70-74.