AIAA-94-1281-CP

A MULTITASKING BEHAVIORAL CONTROL SYSTEM FOR THE ROBOTIC ALL TERRAIN LUNAR EXPLORATION ROVER (RATLER)

P. Klarer*
Sandia National Laboratories
Albuquerque, New Mexico

Abstract

The design of a multitasking behavioral control system for the Robotic All Terrain Lunar Exploration Rover (RATLER) is described. The control system design attempts to ameliorate some of the problems noted by some researchers when implementing subsumption or behavioral control systems, particularly with regard to multiple processor systems and real-time operations. The architecture is designed to allow both synchronous and asynchronous operations between various behavior modules by taking advantage of intertask communications channels, and by implementing each behavior module and each interconnection node as a stand-alone task. The potential advantages of this approach over those previously described in the field are discussed. An implementation of the architecture is planned for a prototype Robotic All Terrain Lunar Exploration Rover (RATLER) currently under development, and is briefly described.

Introduction

One of the more interesting problems in the fields of robotics and artificial intelligence research is the autonomous navigation and control of ground vehicles. The capability for a vehicle to autonomously perceive, plan, and navigate through obstacle fields in realistic conditions has potential application in a wide variety of areas, from automated warehouse delivery carts to planetary exploration platforms. Since the advent of modern digital computers the typical approach to solving the autonomous navigation problem has been through Artificial Intelligence (AI) techniques, where a systematic procedure of perception, modeling, planning, action and feedback (see Figure 1) are applied in a manner reminiscent of human being's problem solving techniques[1,2].

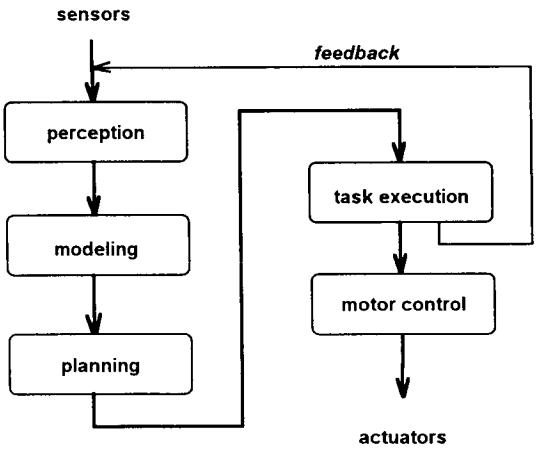

Figure 1. Traditional AI Approach
(from Brooks, 1986)

The computational requirements for such systems have proven to be significant, and alternatives have been developed over the past few years which employ techniques that are less anthropomorphic, including neural networks, fuzzy logic, and behavioral control. Behavioral control was first proposed by Brooks[3] as a simple and extensible architecture that could provide fully autonomous systems without the high

* Member of the Technical Staff, Advanced Vehicle Development Department, Robotic Vehicle Range

This work is sponsored by the Laboratory Directed Research and Development (LDRD) Program at Sandia National Laboratories and the US Department of Energy.

computational requirements of the traditional AI approach. The subsumption approach is based on an entirely different paradigm which is less 'sequential' in nature than traditional AI, and is more 'layered' (see Figure 2).

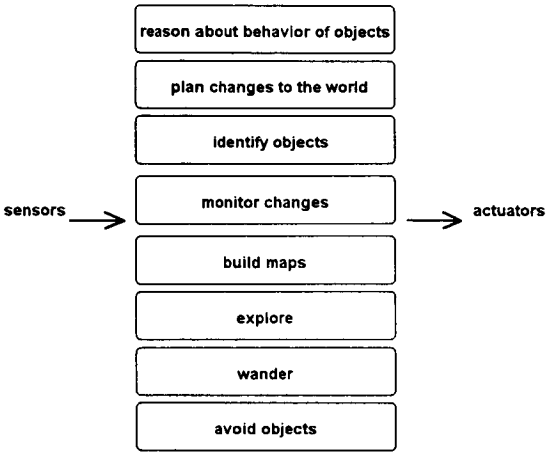

Figure 2. Subsumption Approach
(from Brooks, 1986)

Some observations by researchers who have developed these behavioral or 'subsumption' based systems have pointed out pitfalls in the subsumption paradigm that can make implementation problematic[4]. The use of multiple processors that are physically wired together to produce the subsumption system resulted in a system that was mechanically complex, required custom designed hardware, and was somewhat prone to failure. Although these problems may be alleviated through the miniaturization and consolidation of the 'macro-sized' multiprocessor system to a 'micro' or even 'nano' sized custom silicon chip[5], the basic weakness of these systems is their inability to provide internal models or representations of the environment. This lack of 'state' information makes interaction with the system by a user somewhat like attempting to converse with a very simple lifeform. It must be stated that the elimination of internal models or 'state' information was one of the objectives of the subsumption architecture's proponents, in that it is precisely this requirement for accurate world models that drives the traditional AI techniques' high computational requirements, and the argument has been made that an insect requires very little 'computation' power to function autonomously[6]. While it is true that insects can function autonomously in a real world environment, it is difficult to imagine a technique that allows one to interactively direct the behavior of an insect to suit one's purpose, and that is after all, one of the important criteria for robotic systems in the service of humankind. One potential approach that not only circumvents those problems but may actually facilitate the implementation of a subsumption architecture is to employ a multitasking framework as the basis for system design. This approach employs conventional computer processors to alleviate the mechanical complexity issue, allows software modules to operate independently in real time much as the subsumption architecture's 'behaviors' do, and provides many intrinsic features that facilitate the communication and interactions between independent modules. In addition, a real-time multitasking system allows the use of traditional AI problem solving techniques in concert with a subsumption architecture, resulting in a hybrid system that takes advantage of the benefits of both approaches. The remainder of this paper is devoted to describing how such a system could be implemented, and discusses some concepts for specific implementation on a mobile robotic system under development at Sandia National Laboratories' Robotic Vehicle Range. It should be noted that detailed explanations of the subsumption approach are referenced at the end of this paper, and are therefore not covered here, however Dr. Brooks' notation is used herein to describe the use of the subsumption architecture's connections and features.

Generalized Hybrid System Description

Figure 3 illustrates a generalized hybrid system architecture for robotic system control, employing both a subsumption system (surrounded by a dashed line), and a conventional set of real-time input\output tasks and associated data stores. Data acquisition from the sensor suite is handled by one or more specialized tasks, which typically run at a fixed rate. The fixed, regular rate of data sampling is an important characteristic of real-time systems, in that the application of some digital filtering or

post processing techniques often requires that data from several disparate sensors be acquired at fixed, repeatable intervals. One of the major advantages of a real-time multitasking system in this context is the ability to add, modify or delete major modules without affecting the timing characteristics of the system as a whole, or the iteration rate of other task modules in the system. As shown in the figure, data acquisition is completely independent of the subsequent tasks in the system that make use of it, as are the control output tasks.

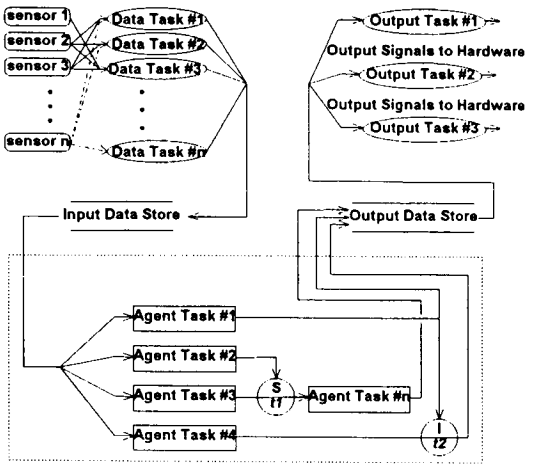

Figure 3. Multitasking Behavioral System Architecture

By using a set of common data stores between the I/O tasks and the subsumption control system, the architecture allows for the addition of other 'high level' control or 'AI-like' elements, as shown in Figure 4. For example, interfacing a set of tasks which perform perception and feature extraction, the maintenance of a data-based world model, and the planning of actions based on the model to the system is straightforward, and only affects the system at two points. The input data store must accommodate any new sensors required for perception, and the motion planner's output is hooked into one of the subsumption system's outputs via a new subsumption node. The world model data store is entirely independent of any other features of the system, and is only interfaced to the new tasks which require it.

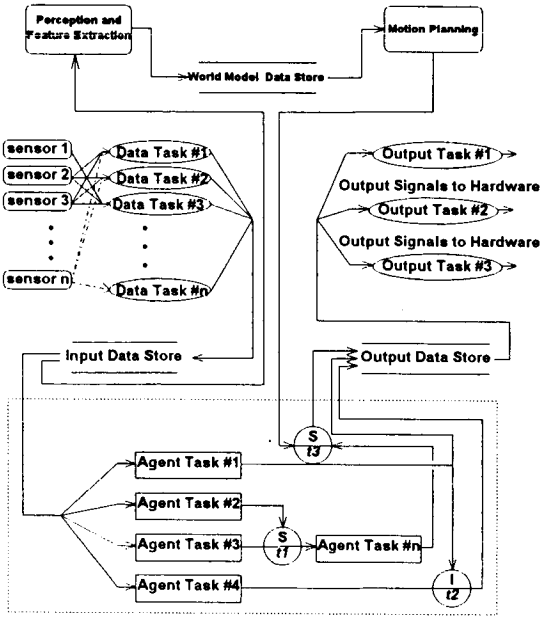

Figure 4. Hybrid System Architecture

The previous example is included only to illustrate the method by which a subsumption system can be integrated with traditional AI approaches, and how a real-time multitasking architecture facilitates this. In an actual implementation, the addition of traditional AI techniques may be obviated if the subsumption system is fully implemented, resulting in a fully autonomous system which is capable of initiating motion, avoiding obstacles, pursuing some goal directed behavior, perception and\or reasoning about disparate sensory input, and exploration of its surrounding environment.

As noted in Brooks' description of the subsumption architecture, communications between each of the modules and the method of interconnecting the various modules through special nodes is the key to implementation[3]. The built-in features of most multitasking systems that allow intertask communication and synchronization provide the methods for implementing a subsumption system such that more than a single specific mechanism may be used. The most common feature available in multitasking systems is message passing. Message passing involves packaging data into a special data structure specific to the operating system, and tagging the data structure's 'destination'

variable (or pointer) so that the multitasking kernel knows which task is the intended recipient. The kernel is invoked with a special call by the sender, and the message arrives at the destination task at some time in the future. In a real-time multitasking system, the message 'delivery' time is guaranteed to be no more than some specific interval. For the purposes of a subsumption system, the contents of the message would be the pertinent data from one module's 'output line' as described by Brooks, which corresponds to an 'input' line to either another module or to a subsumption node. In either case, the embodiment of behavioral modules and nodes as separate tasks makes them appear to be identical to the multitasking kernel, and are treated in exactly the same way by the system. Thus, message passing provides an immediate interconnection tool to create a subsumption system using tasks to implement behavior modules and connection nodes, and potentially allows both synchronous and asynchronous functionality, depending on the details of the multitasking kernel's features.

An additional, complementary method for performing intertask communications involves the use of a user-defined data structure located in global memory (or accessed via a pointer) and a real-time multitasking feature called an 'event'. Events are implemented in a variety of ways across different kernels, but usually involve a special bitfield or kernel variable that is monitored by the kernel for changes. Changes in the state of the event bitfield are initiated by tasks that wish to record the fact that something of importance has occurred, and some other task needs to respond. The kernel notes a change in the event bitfield at some defined, minimum time after a task changed the state of the bit, and responds by triggering or 'kicking' the second task which is hooked to the event bit. Although somewhat clumsy when compared to message passing, the use of real-time 'events' and an associated data structure operates much faster than the passing of a message. This is due to the fact that message passing usually involves much more in the way of data operations by the kernel than the simple setting or clearing of a single event. Either way, the intrinsic features of a real-time multitasking system provide the means by which synchronous and asynchronous interactions between independent tasks may be applied to implementing a subsumption architecture.

Example Implementation for RATLER

In the previous section some generalized descriptions of multitasking features were used to illustrate how a subsumption system could be implemented within such a framework. In this section, an example implementation for a specific mobile robot system is described. The example is taken from Brooks' description of a subsumption system[3], and has been modified to fit the multitasking approach to implementation for the Robotic All Terrain Lunar Exploration Rover (RATLER)[7]. The RATLER II vehicle shown in Figure 5 is a four wheeled, skid steered articulated chassis design, with the body divided into right and left halves, two wheels on each side. The halves are joined together such that they may rotate along the lateral axis to enhance the mobility and stability of the platform. The RATLER is intended for use as a teleoperated vehicle, with semiautonomous navigation capabilities selectable by a remote operator. The RATLER will eventually carry a suite of instruments to perform mission-specific tasks, and will also be fitted with a multi-DOF manipulator arm. The primary mission of the RATLER is planetary exploration, but the platform may find terrestrial applications as well.

Figure 5. RATLER II Prototype

Although not yet implemented, some form of proximity sensing will be required on the vehicle to provide obstacle detection input for the software control system. This will most likely take the form of a series of infrared or microwave range sensors arrayed around the vehicle's periphery, which output a signal if an object is detected within a preset minimum range limit or set of range bins.

The subsumption architecture as originally described by Brooks does not include any provisions for a user to interact with the system directly. Rather, the focus on minimalism to achieve autonomy ignores the possibility of teleoperation as a control mode. For certain applications, i.e. the exploration of a planetary surface too far removed from a human teleoperator for real-time remote control to be feasible, a strictly autonomous operations mode makes sense. However, there are many other applications of robotic vehicles that would benefit from a capability to teleoperate, including both terrestrial and possibly lunar surface missions[8,9]. Two implementation concepts for a subsumption system with a teleoperation capability are briefly described below. The first assumes teleoperation to be the 'lowest' functional layer in the subsumption architecture, whereas the second concept assumes teleoperation to operate at the 'highest' level. Precisely which of the two approaches, or some other as yet undefined approach, is best will the subject of future work.

Teleoperation as the 'lowest' level:

As illustrated in Figure 6, the subsumption system proposed for the RATLER is based on the real-time multitasking approach described previously. A set of dedicated tasks perform data acquisition from a sensor suite, and run at a fixed, periodic rate between 10 and 100 Hz. The timing interval is controlled by the multitasking kernel via inherent timer interrupts, and is transparent to the user. The sensor data is stored in a common memory structure and is available to the subsumption system's modules as required. Each module accesses only the data it needs in order to perform its function, which is to operate as a finite state machine. The data store is accessed via pointers, and does not require any event bits or message passing. Note that this system differs from Brooks' original description in one significant regard; that is, the lowest functional element is a module called 'teleoperate'. This is assumed to be the lowest possible level of functionality, considering a teleoperation system to be of a lower order than a self guided or autonomous system.

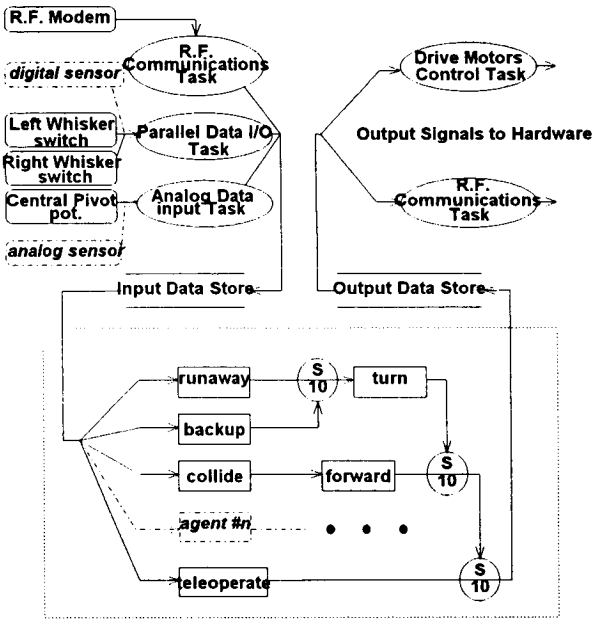

Figure 6. Teleoperation as 'lowest' level

As the lowest level of functionality in the subsumption system, teleoperation is always active if commands from the user (located at a remote control console) are arriving and are being stored in the data structure. If any of the 'higher' modules illustrated in Figure 6 are active, their outputs will subsume the teleoperator's outputs in a manner identical to that described for Brooks' subsumption architecture. This simple change from the original system description allows a direct user interface for teleoperation, and does not adversely affect the operation of the overall subsumption design. The higher subsumption modules may be switched 'on' or 'off' by the user via a set of flags in the data structure, so that the system can be operated in either a straight teleoperation mode, or in any combination of autonomous modes, depending on which behaviors are switched on and which are switched off.

Each behavior module is an independent task which runs at a specific periodic rate, typically between 10 and 100 Hz, and communicates with other modules via either the event bit method or message passing method as previously described. The subsumption nodes are designated as circles with an 'S' in the figure, and are also implemented as independent tasks which run at fixed periodic rates. In order to perform their function as subsumption nodes, they must maintain an internal record of the latest inputs from their associated behaviors, and perform a finite state machine transition based on their simple set of internal rules. The result of the machine state switch determines what the node's output signal will be, and that is sent via either a message or event bit signal to the next behavior or node as illustrated above. After the signals have passed through the subsumption system and have arrived at the output data store, placed there via pointer access, a set of dedicated output control tasks access the data structure for setpoints and translate them into output signals to the robot's hardware. Functionally, this system is no different than more conventional implementations of the subsumption architecture, except it relies on the inherent facilities of a real-time multitasking kernel to perform the communications required between modules and nodes, and to control the real-time preemptive triggering of those modules according to a real-time clock.

Teleoperation as the 'highest' level:

In contrast to the concept previously described for a teleoperation capability in a subsumption system, the assumption of teleoperation as occupying the 'highest' level in the subsumption system leads to a very different implementation scheme. The system shown in Figure 7 is the basic multitasking subsumption architecture described in earlier sections of this paper, with the addition of a teleoperation module *outside the subsumption subsystem*. The teleoperation module receives its inputs from a remote control station where a human operator is located, whose commands are in the form of motor\axis setpoints in position, velocity, or torque. Global commands in the form of a desired vehicle heading or destination waypoint may also be generated from the remote console.

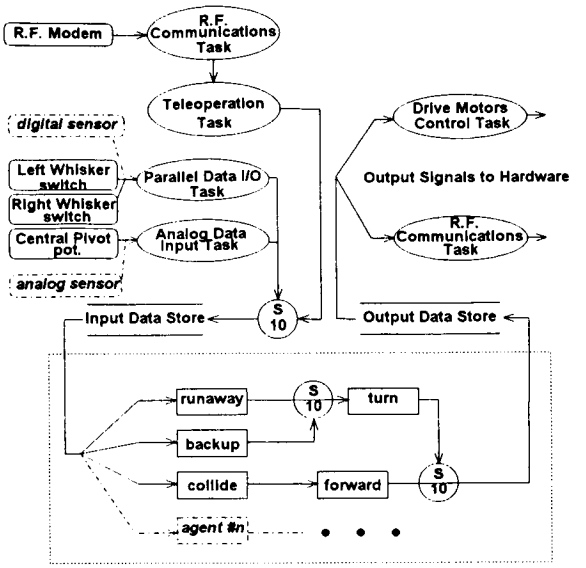

Figure 7. Teleoperation as 'highest' level

The desired setpoints or global state commands received by the teleoperation task are used to generate a set of synthetic alternate signal inputs from appropriate sensors, which are then placed in the input data store. They may simply be used to modify real sensor data, or to replace the real data altogether much in the same fashion as the subsumption subsystem operates on its data flows. The act of teleoperating then becomes a matter of 'faking out' the subsumption system by providing it input signals that will cause the subsumption system to react in the desired manner. A simple example would be to cause the vehicle to move forward, the generation of a synthetic obstacle signal from one of the rearward facing obstacle sensors by the teleoperate task should cause the 'runaway' behavior to activate, and the subsumption system would then move the vehicle away from the offending signal, in other words, forward. This approach may have advantages over the previously described 'approach, in that it relies on the subsumption system to perform all control outputs and is implemented outside of the subsumption subsystem, whereas the 'teleoperation is lowest' method requires the teleoperation task to be an

integrated part of the subsumption architecture.

Summary

This paper has described an approach to implementing the so-called 'subsumption' or 'behavioral' control scheme for robotic systems within the framework of a real-time multitasking architecture. The potential advantages of this system result from taking advantage of timing and communication features available with most multitasking kernels. The proposed architecture also allows for the construction of hybrid systems which employ both subsumption and traditional AI techniques, and easily provides for a teleoperator's interface as well. The proposed system is well suited to operating on conventional general purpose computing hardware, and should allow development with existing software tools designed for real-time multitasking systems.

Future Work

Although only two multitasking features were described, event scheduling and message passing, there are undoubtedly other inherent multitasking system features that may be employed to some advantage. The benefits of employing teleoperation as either the 'highest', 'lowest' or some intermediate level with respect to the subsumption system remains to be fully evaluated. Pursuing those and other issues will be part of future efforts by Sandia National Laboratories in the development of the RATLER control system software.

Acknowledgements

Special thanks go to P. Heerman, R. Byrne, B. Pletta, and W. Amai for their assistance in developing the ideas in this paper. Their insights and unconventional approaches to problem solving will hopefully help bring these ideas to fruition in the future, as they already have helped to bring the basic concepts into communicable form. The author also acknowledges the work of the people of the Artificial Intelligence Laboratory at MIT, without whose groundbreaking efforts this paper would not have been inspired.

References

1. CROWLEY, J. L., "Navigation for an intelligent mobile robot.", IEEE Journal of Robotics & Automation, vol. RA-1, no. 1, March 1985.

2. MORAVEC, H. P., "The Stanford cart and the CMU rover.", Proceedings, IEEE, vol 71, July 1983.

3. BROOKS, R. A., "A Robust Layered Control System for a Mobile Robot.", IEEE Journal of Robotics & Automation, vol. RA-2, April 1986.

4. CONNELL, J. H., "Minimalist Mobile Robotics, A Colony-style Architecture for an Artificial Creature", Academic Press Inc., San Diego, CA, 1990.

5. BROOKS, R. A., "Micro-Brains for Micro-Brawn; Autonomous Microbots", Proceedings, IEEE Micro Robots and Teleoperators Workshop, November 1987

6. BROOKS, R. A., "Engineering Approach to Building Complete, Intelligent Beings", SPIE Vol. 1002 on Intelligent Robots and Computer Vision, 1988

7. PURVIS, J., and KLARER, P, "R.A.T.L.E.R.: Robotic All Terrain Lunar Exploration Rover," Sixth Annual Space Operations, Applications, and Research Symposium, Johnson Space Center, Houston, TX, 1992.

8. MEYER, C., "Goals and Requirements for Scientific Lunar Rovers", Proceedings, ASCE, SPACE 94: Conference on Robotics for Challenging Environments, February 1994.

9. HOFFMAN, S.J. and WEAVER, D.B., "Results and Proceedings of the Lunar Rover/Mobility Systems Workshop", Exploration Programs Office document #EXPO-T2-920003-EXPO, NASA Johnson Space Center, Houston Tx, 1992