A TECHNIQUE FOR COORDINATING AUTONOMOUS ROBOTS

S. Y. Harmon, W. A. Aviles & D. W. Gage

Autonomous Systems Branch, Naval Ocean Systems Center
San Diego, CA 92152

ABSTRACT

This paper describes a technique for coordinating the subsystems of autonomous robots which takes advantage of a distributed blackboard mechanism and a high degree of functional distribution between subsystems to minimize communications and simplify the interfaces. Distributed blackboard memory contains a world model which represents knowledge about itself and its surroundings as collections of objects important to the task and the relations between them. Objects or instances are represented as lists of object-attribute-value-accuracy-confidence-timestamp tuples which are organized into a class tree with inheritance properties and active functions. Intelligent Communications Interfaces maintain the consistency between blackboard memory elements distributed through loosely coupled subsystems by exchanging reports which represent world state information and plans which represent control information. These concepts can be extended for coordinating multiple autonomous robots by making independent robot world models intersect to enable communications and by broadening the interpretation of plans and reports. Coordinating autonomous robots through distributed blackboards makes potentially complex programming simpler by providing a well defined but flexible framework for module interfaces.

INTRODUCTION

Implementation of autonomous robots for difficult tasks is complicated by the need for sophisticated sensor input and by the numerous interactions possible between complex subsystems. Little serious research has addressed the problems limiting the integration of practical autonomous robots much less the problems associated with coordinating multiple robots. Yet these problems must be solved before individual autonomous robots and groups of autonomous robots can become cost effective options for such tasks as strategic defense and space construction. Significant future effort is likely to be focused on coordinating multiple interacting robots because of their widespread potential applications. One technique has been proposed for coordinating the complex subsystems of a single autonomous robot which uses distributed blackboards for information structuring and exchange [1]. This discussion reviews this technique and extends it to address the problems of coordinating multiple autonomous robots.

This discussion assumes that communications bandwidth will always be more expensive and less accessible than processing capability. While this situation is not universally true, communications bandwidth remains a significant limitation and design consideration in most practical applications of autonomous robot systems. This paper begins by examining the system structuring, world model, interactions and implementation of the complex subsystems which comprise an autonomous robot. The distributed blackboard concepts presented are then extended to the problems of coordinating multiple distributed robots.

SINGLE ROBOT COORDINATION

As robot capability increases, the number of sensor, effector and processing components that must be integrated and coordinated also increases. Distributed computing offers a number of advantages to aid in coping with the significant design and implementation complexity inherent to sophisticated robot systems. In many cases, a single computer which is capable enough to meet the a complex robot's processing demands cannot fit within the size, weight and power constraints imposed by task demands and construction realities. In addition, distributed computing implementations are often cheaper and more resilient than the uniprocessor alternatives. Also, multiple, possibly redundant, computing elements provide the opportunity to take advantage of parallelism for improved throughput and fault tolerance. System designs striving to meet the realtime constraints accompanying most robot implementations can benefit considerably from the concurrency inherent to distributed computing solutions. The modularization necessary for a distributed implementation often simplifies the difficulties of complex hardware and software implementation and debugging as well. Fortunately, bus and local area network (LAN) standards have improved commercially available hardware and software products for multiprocessor implementations.

Blackboard concepts have been employed for many

years in artificial intelligence research where knowledge is represented by expert sources and problems are solved through the cooperation of these experts [2]. The cooperating expert paradigm has provided the inspiration for the coordination technique described herein. In this concept, the blackboard is a uniform interface to the system world model. This interface makes the development of system modules much easier since each may be implemented and tested separately and changes in one module do not necessarily imply changes to another. The blackboard also provides a simple, flexible and proven model for exchanging information between modules which is easy to implement on distributed computing resources.

System Structuring

The two primary system structures important to this coordination technique are the functional partitioning of each module and the structure of knowledge important to the system. The system's knowledge is represented by its world model.

Functional Partitioning

In the proposed coordination technique, individual modules are made as functionally independent as possible. Partitioning in this manner implies that much sensor information is processed close to the source. Likewise, planning and control processing is pushed deeply into the control hierarchy. This greatly improves the response time to complex control situations. So far, processing load is permanently assigned to dedicated resources instead of being dynamically assigned at runtime although this is not a restriction inherent to the technique. Each subsystem is treated as an autonomous entity with all the processing capability necessary to raise its level of abstraction to the highest possible level. This functional partitioning reflects the minimum communications assumption by requiring that all communications take place at the highest level and thus in the tersest practical representation. Although each module should be as independent as possible, function should be assigned so each module's capabilities overlap in those areas where some degree of system level fault identification and recovery is desired.

World Model

Content. In order to distribute sufficient function to a module, sufficient knowledge must also be distributed to that module. This implies that each module must represent a considerable portion of the world model locally. This degree of information distribution also provides an additional line of fault tolerance through data redundancy. In this scheme, a sensor module does all the processing and represents all the knowledge necessary to transform its raw sensor data all the way to symbolic level. This means different things to modules with different functions. For instance, a vision sensor certainly requires significantly more resident processing and a larger portion of the total system world model locally than a satellite based absolute navigation system to transform the raw sensed signals into symbolic form.

The world model is divided into knowledge of the SELF and knowledge of the SURROUNDINGS. The SELF context is defined by the physical boundaries of the individual robot and it represents all of the important internal state information of the robot. The SURROUNDINGS context represents all those objects in the environment which are important to the task yet not defined by the robot system itself. Although of quite different nature, the SELF and SURROUNDINGS contexts can be represented within a single data structure [3].

Representation. The distributed implementation described here requires that the world model data structure be constructed to access both locally and remotely supplied information in a uniform manner. This permits information to be used without the regard for where or how it was obtained. The blackboard concept is a data storage and structuring paradigm that can be designed to meet these constraints. Data in the blackboard can be organized as a class tree which provides a useful set of auxiliary abstractions which promote economical communications and processing. This discussion of the blackboard will be limited to its use for domain model representation although its use for the other purposes is not precluded. Although blackboard concepts are in fairly common use in expert systems, robots have requirements not encountered in these classical blackboard applications. Complex robots must often use data from multiple local and remote sources in a timely manner. The proposed blackboard based system is designed to meet these needs.

The blackboard concept described here consists of a distributed data structure, a set of interface procedures to access those structures and a set of active functions which relate one data element to another. Through these, the blackboard supports both a hierarchical data organization and a data driven programming model. All information available to the robot concerning the SELF and SURROUNDINGS contexts is represented as a collection of blackboard objects. Each object is an abstract data type consisting of an object name and one or more named attributes. Each attribute consists of an attribute packet with four fields:

<value, accuracy, confidence, timestamp>

The value field holds the current state of an attribute and can be a number (either integer or real), a string, or a token representing another object. The accuracy field represents a measure of the value's potential deviation from the measured value and the confidence field represents the statistical confidence in the correctness of the interpretation of the attribute's value. The timestamp field contains the time when the attribute packet was last updated and permits independent assessment of the data's timeliness. Objects in the blackboard are organized hierarchically into a class tree. The class tree has both inheritance properties and active

functions. The inheritance properties imply that all terminal objects or instances of a class possess all of the attributes of the parent class. Thus, class objects can provide structural templates and default values for the attribute packets of all class instances. Inheritance properties ensure an economy of representation and permit class instances to be created dynamically at runtime.

A robot's sensor and effector capabilities must be completely represented in its world model. Sensor capabilities include actual sensors as well as logical organizations of these sensors for high level information processing and abstraction. Analogously, effector capabilities include individual actuators as well as logical effectors derived from combinations with sensor capabilities and modes of data processing. All sensor and effector capabilities are represented as blackboard objects. The blackboard's structure allows the dynamic creation of compound sensor and effector objects from the simpler sensor and effector capabilities. A blackboard object maps some area of sensor or effector space onto a particular symbol. The most primitive sensor objects are the readings directly from the sensors themselves and the most elementary effector objects are the simplest actions of which the effectors are capable. Additional objects are derived either directly or indirectly from the most primitive objects. Sensor based objects often represent perceivable things in the task environment and are therefore directly interpretable in terms of the physical world. Effector based objects can be considerably less obvious since they can represent intended actions or changes in robot processing.

Interaction Mechanisms

The modules of a robot can interact either through blackboard memory or through communications. The blackboard memory supports interactions between tightly coupled functional modules and communications on a LAN enable interactions between loosely coupled modules.

Blackboard Memory

Shared Memory. Subsystem modules can be tightly coupled through memory shared on a high speed parallel bus (e.g., IEEE 796) with the processing components. A module's blackboard resides in this memory. Standard Blackboard Interface Procedures (SBIPs) provide uniform access to the information contained in the blackboard structure and provide the concurrency control mechanisms necessary to be effective in a dynamic multitasking and multiprocessor computing environment. Classes and instances may be dynamically created and changed using these procedures. As initially implemented, dynamic access was limited to the processes of creating and deleting instances of a class and of reading and writing attribute value packets. At that time, object classes were defined by software engineers at system creation time. A recent enhancement permits object classes to be created dynamically by downloading blackboard structures from another source. Ultimately, the robot should be able to create new object classes itself to help optimize use of its processing and communications resources. Dynamic class handling creates the problem of finding the set of abstractions whose implementation in the blackboard can support an optimal balance between minimizing communications and minimizing program size and execution time. This capability when coupled with the capability to generate the appropriate programs that manipulate blackboard symbols constitutes a sophisticated learning machine.

Active Functions. Active functions also provide another path, albeit indirect, in the blackboard memory through which subsystems can interact. The active function mechanism is an implementation of the data-oriented programming paradigm and can be used to change the state of object attributes when the values of other attributes are read or changed. Both instances and classes can be coupled through active functions. This very powerful data driven programming paradigm is implemented in the SBIPs through which attribute value accesses (i.e., reads and writes) can trigger actions. Active functions are represented by function lists associated with each object attribute value. The active function lists of class instances are inherited from the class object at instance creation.

Internal Communications

Communications between modules loosely coupled through the LAN (e.g., IEEE 802.3) consists of plans and reports.

Plans. Plans communicate control information and take the form of production rules extended to control of realtime situations. A plan has the following fields: name, initiation condition, trigger condition, termination condition and action. The plan name is a symbolic token which permits high level access to the plan. Conditions are arbitrarily complex conjuncts and disjuncts of attribute fields and of predicates applied to attribute fields. The initiation condition specifies the conditions after which the plan action can first be invoked. The trigger condition specifies the conditions which must be true for the plan action to be taken (i.e., both initiation and trigger conditions must both be true for the plan action to be taken). The trigger condition is continually evaluated after the initiation condition becomes true and until the termination condition becomes true. The termination condition specifies the condition after which the plan action will never be invoked. If the termination condition becomes true before the initiation condition becomes true then the plan action is never taken. Plan actions can initiate either control actions or report actions. Plans require virtual circuit communications services since the production of a plan implies source receipt of a number of acknowledgements from the destination module which identify the success or failure of different aspects of a plan's progression.

Reports. Reports communicate world model information used to maintain the consistency of distributed blackboards. As a result of this purpose and for the economy of communications, all reports are broadcast to every module on the network. Reports represent assertions in the form of the object-attribute-value-accuracy-confidence-timestamp tuples used to represent world model information in the blackboard. Reports are generated in response to active plan actions. Report plan conditions can be used to filter the transmission of world model data in both time and value space within a robot.

Intelligent Communications Interfaces. The blackboard can be distributed over the LAN by connecting loosely coupled modules through Intelligent Communications Interfaces (ICI) [1]. This mechanism maintains the local consistency of the blackboard by monitoring the report traffic on the network. It also supports the distribution of control by enabling the downloading and triggering of contingency plans for each module. The blackboard concept provides a well defined interaction mechanism through which all modules can cooperate. Since all representation in the blackboard is symbolic, communications can be limited to very terse representations. Further details of the ICI design and implementation can be found in Refs. [1 & 3].

Implementation

Programming. The distributed blackboard concept described here requires a minimum of software support consisting of the ICI, the SBIPs and a multitasking operating system kernel which supports memory control, task switching and access to the network. This compactness makes this coordination concept very accessible to programmers. The layering of the SBIPs and the ICI hides much of the complexity inherent to interacting sophisticated subsystems. The SBIPs provide a simple and well defined interface to the capabilities of the blackboard to represent the world model. The blackboard's object representation structure, class tree structure, inheritance properties and active functions all form a powerful collection of flexible tools necessary to represent both the static and dynamic aspects of the task environment. All of these features make a tractable and capable programming environment. The decoupling effect provided by the blackboard enables software modules to be developed independently and to be integrated with a minimum of effort. Programming efficiency can be increased by building software modules which are built up from small reusable pieces.

Examples. This distributed blackboard concept has been implemented on an autonomous vehicle testbed, the Ground Surveillance Robot (GSR). This experimental vehicle has permitted exploration of a variety of interactions between complex sensor and control subsystems. Interactions between the navigation sensor and the proximity sensor subsystems are used to determine obstacle position in absolute coordinates. In two way interactions between the vehicle attitude sensor and the locomotion control subsystems, the locomotion subsystem uses vehicle attitude information for informed adaptive vehicle control and the vehicle attitude subsystem uses locomotion state information to anticipate and filter incoming sensor data. Absolute vehicle position estimates are improved through cooperative interactions between the vehicle attitude sensor and navigation sensor subsystems. Recently, cooperation between the proximity sensor and the vision sensor subsystems has been implemented to improve target identification and location. The proximity subsystem uses the vision subsystem to locate an out-of-range target and the vision subsystem uses proximity information to narrow its search for the target vehicle. From this experience, the coordination technique described above has demonstrated that it supports the diverse cooperation between subsystems required by a complex autonomous robot system.

MULTIPLE ROBOT COORDINATION

Multiple robots can be coordinated to achieve greater diversity and to obtain greater parallelism. Diversity provides multiple robots with the abilities for spatial extension and resource tailoring. Spatial extension permits multiple robots to address tasks which require simultaneous coordinated actions at two or more separate locations. Resource tailoring lets a mix of tasks with differing requirements to be most economically addressed by assigning to each task only the resources required. This capability could lead to the implementation of robot families with various specialized skills and the assignment of the appropriate robot "team" to perform each task. Cooperative behavior between redundant robots can be used to detect and correct subsystem failures and processing errors. This ability enables multiple robots to ensure the success of critical tasks where maintenance and repair services are not available despite the occurrence of repeated equipment failures (e.g., extended space missions). Parallelism also provides the opportunity for multiple robots to increase overall system throughput. Even though a single robot may be entirely capable of performing a large task, the application of multiple robots to the same task can speed its completion. In this situation, coordination is necessary to ensure that efforts are not duplicative or counterproductive. Eventually, multiple robots will be employed for any and all of the above purposes.

Several factors must be considered when designing a cooperating collection of autonomous robots including interaction spontaneity, available communications bandwidth, robot domain coupling, robot complexity and the number of robots employed. The coordination mechanism employed should be flexible enough to support both centralized and decentralized automated planning. This flexibility permits greater spontaneity of interaction and thus provides greater ability to cope with circumstances unanticipated by the system designer. The communications bandwidth

available between multiple robots can affect their organization. If the channel connecting the robots is as large as the total sensor-effector bandwidth then the robots can be coupled so tightly as to function as a single virtual robot. On the other hand, if the channel bandwidth is much smaller then the coordination must take place at higher symbolic levels to maximize communications effectiveness. The coupling between robot sensor and effector domains affects the degree of cooperation required between multiple robots. Less coupling implies decreased communications requirements and coupling complexity. The coordination problem among robots becomes more complex as the number of participating robots is increased and as the sophistication of the participating robots increases. The distributed blackboard concept described above can be extended to address the requirements imposed by coordinated multiple robots. In this extension, each robot is modelled as an intelligent component of a larger system. The proposed coordination technique can support a variety of static and dynamic system organizations.

System Structuring

Unlike the problem of coordinating a single robot where some guidelines for system structuring could be suggested, coordination of multiple robots is by its very nature an expansive and difficult problem. The organization of cooperating robots depends primarily upon the nature of the application. However, some general observations can be made. Multiple robots should be designed to benefit from both diversity and parallelism. This implies that each robot's sensor and effector domains should somewhat overlap another's domains to accommodate inevitable device failures and to be able to exploit task parallelism when it is available. Further, complex tasks usually demand more capability than available from a single robot so complete functional overlap, while desireable, is often impossible. These observations support the conclusions drawn from consideration of the design factors discussed above. Coordination of multiple robots for practical applications requires a very flexible system structure. This requirement places challenging demands upon the world model representation.

World Model

In a collection of multiple robots, each with its own resident knowledge, each robot must model only that portion of the task environment which is relevant to its functions. Robot world models must logically overlap only when the function of one robot depends upon the function of another. For cooperating robots the additional awareness of the SYSTEM context must be added to the world model [4]. Now information represented by a single robot's world model may be either local (i.e., sensed directly by SELF) or remote (i.e., sensed by some other robot in the SYSTEM). The SYSTEM context is much like that of SELF but it represents the perceptions and intentions of the other robots in the collection. Obviously, the SYSTEM context includes only those robots with which there are communications. No cooperation is expected from the objects from the SURROUNDINGS. Thus, if noncommunicating robots are present in the task environment, they are simply represented as components of the SURROUNDINGS. Cooperating robots can be viewed as set of capable knowledge based entities acting on the entire SELF, SYSTEM and SURROUNDINGS knowledge base. Knowledge in all of these contexts can be represented as collections of blackboard objects. No substantial changes are needed in the single robot representation described above to cope with the demands of multiple coordinated robots.

As discussed earlier, a robot can be modelled as a collection of sensor, effector and reasoning capabilities. Sensor knowledge includes all the physical and logical state information available to the robot about the SELF, SYSTEM, and SURROUNDINGS contexts. Effector knowledge includes information about all the physical and logical actions available to the robot in the SELF and SYSTEM. Reasoning capabilities include the knowledge based expertise and planning available to the robot from both the SELF and the SYSTEM contexts. Active functions and plans associate blackboard object accesses with actions of sensors and effectors. Local sensor and effector knowledge is provided by drivers writing to the local blackboard through the SBIPs.

Remote sensor and effector objects, while treated no differently than local objects, require communication with the sources and destinations of their information. This communication can be transparent to the individual robot but the world model of each cooperating entity should have the same structure and must have enough overlap in content to permit communications (i.e., cooperating robots must communicate in common languages). The amount of overlap between robot world models is determined by the organization through which the communications is conducted, by the difference in levels between the communicating robots, by the relationship between the communicating entities and by the structural differences in the world models. If necessary, effector objects can represent plans just as well as past actions and the history of results. No longer must these spaces represent only the possible actions which might be appropriate for the prevailing sensed situation. This modification permits multiple robot cooperation as well as enabling learning behavior in single robots.

Communications

The message passing paradigm employed to coordinate the subsystems of a single autonomous robot can be applied almost directly to the coordination of multiple robots. One small difference is that messages must now be sent through a distributed message routing system. For coordinated robots, a message consists of source and destination addresses (either local procedures, local modules or separate robots) and an informational body. A distributed routing mechanism designed into the ICI mechanism provides

the necessary routing functions transparent to the individual robots. As described above, the body can be either a report or a plan and it can be limited to very terse symbolic representations. Structuring the world model as a class tree permits very sparse communications since the greatest level of generalization can be used by taking advantage of inheritance properties and active functions. For distributed robots, plan actions now represent alterations to world models as well as control actions.

Implementation Issues

Programming. This technique provides a high level set of tools and, more importantly, a set of standard interface specifications which allow the system designer to concentrate on WHAT information needs to be represented by the robots' world model and WHAT information is needed for communication not HOW to store the information or HOW to communicate that information. The designer may validate his system first using only the robot's local blackboard and later integrate the robot into the SYSTEM. This technique gives the designer freedom from exhaustively programming the robot for every possible situation. This is a key advantage in programming distributed robots since their inherent flexibility makes deterministic programming a very difficult job. This technique also permits much of the system behavior to be changed very easily by simply changing the plans which are communicated between robots. Software must be designed as modules which accomplish a certain task usually in parallel with other activities and whenever the situation demands.

Examples. This technique for implementing distributed robot systems has been applied to the design of two different multiple robot situations: a concept for integrating multiple robots and other intelligent components of an automated factory and a concept for integrating multiple teleoperated vehicles driven by multiple interacting operators. The first of these systems, the Integrated Flexible Welding System (IFWS), consists of an automated welding workcell with several complex sensors and several associated planning components. The ICIs and the distributed blackboard are used to coordinate the activity of the planners toward generating a final welding plan. This plan is then communicated to an execution controller which decomposes the welding plan into subplans for each of the sensor and control components in the workcell.

The other distributed robot system is known as the Teleoperated Vehicle System (TOV) and consists of three remotely controlled land vehicles operated from a control vehicle. In this system, the actions of the operators and information from the vehicles are coordinated through distributed blackboards by ICIs. This arrangement permits multiple operators to coordinate combined function on a single vehicle as well as facilitating multiple vehicle coordination by a single operator. The concepts introduced in this paper have proven invaluable for making the design and implementation decisions for these complex multiple robot systems.

CONCLUSIONS

A technique has been introduced which provides a flexible integration scheme for a single complex robot system composed of interacting capable subsystems. This technique has been extended to support the interactions of multiple cooperating autonomous robots. A distributed blackboard is used as the communications paradigm both within a single robot and between cooperating robots. This concept provides a clean and consistent interface between subsystem developers and individual robot developers alike. This blackboard approach is flexible and provides the initial progress toward the implementation of practical self programming and learning machines.

ACKNOWLEDGEMENTS

The research upon which this paper is based was sponsored by the United States Marine Corps.

REFERENCES

[1] S. Harmon, D. Gage, W. Aviles & G. Bianchini, "Coordination of Complex Robot Subsystems", IEEE Conf. on Artificial Intelligence Applications, Denver, CO, Dec. 1984, p64-69.

[2] F. Hayes-Roth, D. Waterman & D. Lenat (eds.), Building Expert Systems, Addison-Wesley Publishing Company Inc., Reading, MA, 1983, p308-314.

[3] W. Aviles, S. Harmon, D. Gage & G. Bianchini, "An Architecture for the Coordination and Control of Complex Robotic Subsystems", 1985 Conf. on Intelligent Systems and Machines, Rochester, MI, Apr. 1985, p40-46.

[4] S. Harmon & D. Gage, "Protocols for Robot Communications: Transport and Content Layers", 1980 Int. Conf. on Cybernetics and Society, Boston, MA, 8-10 Oct., 1980, p1090-1097.