The Ground Surveillance Robot (GSR): An Autonomous Vehicle Designed to Transit Unknown Terrain

SCOTT Y. HARMON, MEMBER, IEEE

Abstract—**The Ground Surveillance Robot (GSR) project has proceeded continuously since the Fall of 1980, and in that time an autonomous vehicle design and some degree of implementation has been achieved. The vehicle design has been partitioned into sensor, control, and planning subsystems. A distributed blackboard scheme has been developed which provides the mechanism by which these subsystems are coordinated. Vehicle position and orientation are supplied by vehicle attitude and navigation sensor subsystems. Obstacle avoidance capability has been implemented by fusing information from vision and acoustic ranging sensors into local goals and avoidance points. The influence of these points is combined through potential field techniques to accomplish obstacle avoidance control. Distant terrain characteristics are identified using information from a gray-level vision system, a color vision system, and a computer-controlled laser ranging sensor. These characteristics are used by a general planning engine to develop the desired path to a visible goal in the direction of the final goal. Progress to the final goal consists of a succession of movements from one distant but visible intermediate goal to another. The experience from implementing this autonomous vehicle has indicated the need for an integrated set of debugging tools which make the faults in subsystem hardware and software more distinguishable.**

I. Introduction

THE EXTREMELY hostile conditions imposed by modern combat, outer space, and the deep ocean environments have generated the need for practical autonomous vehicles for military applications and space and ocean exploration. These relatively near-term applications will drive the sophistication and cost of autonomous vehicle technology into the realm where more mundane but more widespread applications such as automated public transportation will be possible. However, significant technology advances will be necessary before even the simplest and most crucial applications can be practically addressed. These advances will only be gained by implementing autonomous vehicle testbeds and gaining experience with the developing technology.

Several previous efforts have prepared the foundation for the autonomous vehicle development including Shaky [1], JASON [2], the RPI Rover [3], the JPL Rover [4], and the Stanford Cart [5], among others. These first generation autonomous vehicles were used to explore basic issues in vision [1], [4], [5], planning [1], [2], [4], [5], and robot control [3], [4]. However, they were all seriously hampered by primitive sensing and computing hardware. More recent efforts have overcome many of these limitations, and very sophisticated second generation autonomous vehicle testbeds have evolved. Some of these efforts include the developments of HILARE [6], the FMC Autonomous Vehicle [7], the Autonomous Land Vehicle (ALV) [8], the various CMU mobile robots [5], and the Ground Surveillance Robot (GSR). This paper will focus on the design and implementation of only one of these recent efforts, the GSR. A more general and complete discussion of autonomous vehicle history and technical issues can be found in other sources [9].

Manuscript received July 29, 1986; revised December 23, 1986. This work was supported in part by the US Marine Corps and the Naval Ocean Systems Center. This paper was presented at the SPIE Mobile Robots Conference, Boston, MA, October 1986.

The author is with Robot Intelligence International, PO Box 7890, San Diego, CA 92107, USA.

IEEE Log Number 8715001.

II. Problem

The design of any autonomous system must begin by defining its task very specifically. A robot's task can be described in terms of its environment and its goals (i.e., those conditions which represent the success, failure, and termination of the task). A task description contains all the information required to specify the sensing, processing, and control components of a robot necessary to accomplish that task. However, the task description only represents a statement of one part of the problem of actually building such a robot. The sensing, processing, and control components require energy handling and mechanical support to maintain their proper function. These components represent part of the reality of implementing an autonomous vehicle testbed and must be addressed with the same seriousness as applied to the more task related components.

A. Task

The GSR is designed to transit from one known geographic location (given in some absolute map coordinates) to another known geographic location over completely unknown natural terrain. The terrain can be any type over which a manned vehicle of comparable capability can traverse. Even though the locations of the starting and finishing points of the journey are well known, the specifics of the intervening terrain are completely unknown. The GSR must therefore develop a terrain map of the territory in the direction of the goal and plan its route from this information alone. However, if a computerized terrain map of the appropriate territory is available, then the GSR should be able to take advantage of this additional information to improve its performance.

Natural terrain can be described in terms of its topography,

its surface composition, the variability of the surface, and the geometric character of the insurmountable obstacles which populate the surface (e.g., trees and rocks). The topography can include planes, continuously varying surfaces (e.g., rolling hills), and discontinuous surfaces (e.g., cliffs, ravines, and precipices). The surface composition can be described in terms of its base composition (e.g., rock, sand, and water) and the biological ground cover (e.g., grass and scrub brush). This terrain description provides sufficient information to identify impassable areas and to estimate the average speed with which a vehicle can transit trafficable surfaces.

In traversing natural terrain the GSR can transit only those areas which support the normal and shear stresses exerted by the vehicle while moving and changing direction. Unfortunately, the Earth's surface is not isotropic in either of those physical properties. Among other hazards awaiting unwary vehicles, the GSR may find ground which will not support its weight, inclined surfaces which it cannot climb or off which it may slide due to insufficient surface friction and flat surfaces with inadequate shear strength to support propulsion and steering actions. Local terrain variations can also change the vehicle's ability to propel and steer itself. Therefore, the GSR must identify traversable areas through its sensors. To complicate this task further, biological ground cover may change base material's ability to bear weight or shear stress.

In reality, other limitations are imposed upon an autonomous vehicle. Vehicles for practical applications must make the desired journey in a limited time and with a limited amount of fuel or other forms of stored energy, and several terrain characteristics can affect the transit time and fuel consumption. For instance, ground cover can impose significant drag upon the moving vehicle, thereby dramatically increasing fuel consumption to an unacceptable level. Furthermore, a field of numerous obstacles can force the vehicle to take such a tortuous path that it cannot meet the limited time criterion. The finite size of the vehicle footprint imposes additional constraints on the terrain which can be efficiently negotiated and must be considered during path identification activities. All these constraints inherent to the task of autonomous transit imply that the vehicle must not only be able to move over a surface but must also maintain continuous knowledge of its position, orientation, and velocity; avoid obstacles and other hazards; and be able to preview enough of the surroundings to identify a suitable long-range path which will take it to its goal in a finite time and with a finite amount of fuel.

B. Vehicle

The task of autonomous transit imposes direct design requirements on the vehicle. It must have sensors to perceive the environment, effectors to move from point to point, and processing which identifies actions from the sensor perceptions which will achieve the desired goals. To be self-contained, the vehicle must also supply the energy to support the processing, sensing, and effecting activities. It must also have structural features to support this equipment and protect it from unfavorable environmental conditions. Additional effectors may also be necessary to control the perspective of sensors with limited fields of view.

The vehicle sensors must perceive the vehicle's instantaneous position, orientation, local obstacles (or, more importantly, free space), and enough of the distant surroundings to plan a route toward the final goal location. The most important vehicle effectors provide its mobility. There are many different forms of mobility which involve different modes of propulsion, steering, and suspension. The choice of an existing manned vehicle which is converted for autonomous operation entirely solves the mobility design problem. However, this decision also imposes all the limitations of an existing vehicle and the difficulties of converting it to automatic control. Nevertheless, the vehicle actually meets the original project goals because its successful performance is compared to the performance of similar manned vehicles performing the same task. In addition, this solution to the mobility problem was within the project budget and has the advantage of demonstrating the concept of retrofitting an existing vehicle thus making this work more attractive to the military sponsors who own a lot of manned vehicles.

The vehicle computing presents a special challenge since it must now reside on a moving vehicle with all the harsh environment conditions that imposes (e.g., shock, vibration, contamination, elevated temperature). The vehicle computing must have sufficient capacity to interpret and integrate sensor data, plan vehicle motions based upon the sensor picture, and control the actuator responses with enough speed to effect stably the desired motion. The computing equipment must still fit within the confines of the vehicle, must not exceed the vehicle's payload, and must not consume all of the vehicle's limited fuel supply.

Furthermore, the vehicle must have an energy-handling system which converts its stored energy into forms to power computers, sensors, actuators, and vehicle locomotion, which transports the energy to where it is needed and regulates it to those conditions demanded by the consumer subsystems and which dissipates any heat generated as a result of the conversion, transportation, regulation, and consumption processes. The vehicle must also provide the structure to protect as much as possible the delicate components from the harsher aspects of environment including moisture, temperature, dust, salt mist, shock, and vibration. This implies that this structure is more than a simple enclosure on some form of mobility. These design considerations represent the reality of implementing a real robot as opposed to simply implementing a simulation of the proper components. All of the design considerations described must be sufficiently developed to implement a successful autonomous vehicle for any application. Even so, this discussion has not addressed the issues of fault tolerance and multiple degrees of freedom.

III. GSR Design

The problem of designing an autonomous vehicle to transit unknown natural terrain is discussed in terms of the basic problems outlined earlier. The GSR sensor capabilities are divided into the horizons. The closest horizon is just under the vehicle. The sensors within this horizon provide vehicle position and orientation. The next horizon identifies the limit of the vehicle's capability to identify and locate nearby

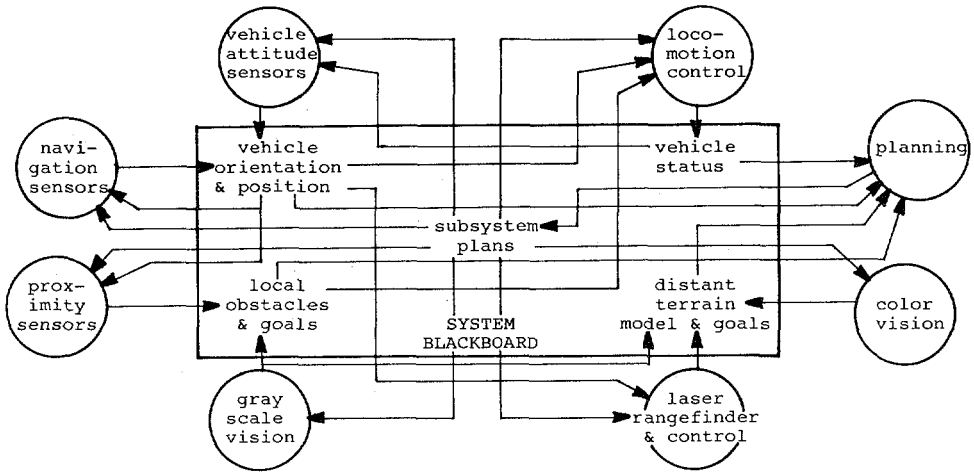


Fig. 1. Blackboard model of subsystem interaction.

obstacles. The final horizon defines the limit of the vehicle's ability to characterize distant terrain. Naturally, one would like to see all the way to the goal but that in practical situations is seldom possible because terrain features and atmospheric conditions dictate limited sensor range.

The discussion of the details of the GSR design begins with consideration of the system architecture and then proceeds to the problems discussed earlier. Each of the sensor subsystems must be tied together in some way, and the system architecture provides a way to do that.

A. System Architecture

The GSR project was started with the knowledge that we did not know precisely how to build a working autonomous system. In addition, we also knew that the project would proceed for several years and that the major component technologies would be evolving greater capability throughout this time. This knowledge significantly affected many of the design decisions. For instance, at each decision point the choice was made which maximized the opportunities for change later on. This means that an architecture was chosen which would enable the change of one part of the system without significantly changing the rest of the system. Furthermore, a choice was made to use as many commercially available resources (both hardware and software) as possible. Fortunately, this effort was undertaken after local area networks, high-speed parallel bus standards, single-board computers, microprocessor multitasking operating systems, and block structured microprocessor programming languages were reasonably well established. These resources enabled layered and modular implementation of both hardware and software.

All software except for a few hardware drivers was implemented in a reasonably high-level programming language, PLM. However, saying all this does not provide a framework within which the components can be integrated. This framework must provide well-defined physical and functional interfaces so the individual components can be implemented and tested independently and in small groups before full-scale integration is attempted. It must also be recognized that many people are going to be involved in the design and implementation of such a complex device, and a well-defined architecture provides a critical basis for communication and coordination. These conditions dictated that the architecture provide for modular implementation. The following discussion of the GSR system architecture is partitioned into the following concepts: subsystem partitioning, the intelligent communications interfaces, and the blackboard representation.

1) Subsystem Partitioning: The first element of an architecture is the model for how subsystems are partitioned. For the GSR, tightly coupled functions must be coupled through a high bandwidth communications path. On the other hand, loosely coupled functions can be partitioned into very high-level modules with considerable shared slowly varying information. The major subsystems were grouped into high-level functions which communicate symbolically by exchanging information about the perceived world. The model of interaction between all subsystems was a blackboard model. This model of interaction is illustrated in Fig. 1. The choice of the major subsystems was made with the awareness of the capabilities provided by existing technology. However, the model supports any likely repartitioning of function. The blackboard is partitioned into the sensor data needed to make the autonomous transit discussed earlier. Vehicle status and system plans are also represented in the blackboard making communication and partitioning of control information easier and more flexible.

We endeavored to distribute as much intelligence to each subsystem as possible to loosen the coupling between them. This minimized intercomponent communications and decreased the real-time processing burden. Intelligent sensor and control components derive information directly from the real world and can exchange information directly between them. The planning mechanism sits above the sensor and control components and monitors the sensor traffic to build a picture of the world and to extrapolate expectations of the future from this current picture. Plans are derived from these expectations and sent to the sensor and control components to coordinate their future interactions. This strategy keeps the planning

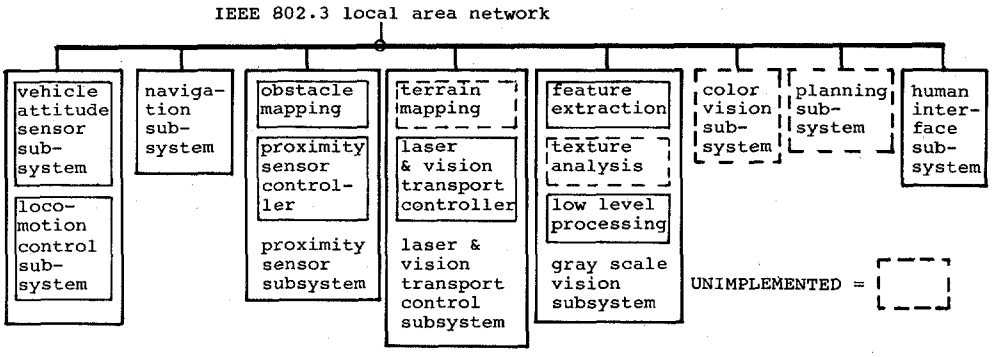


Fig. 2. GSR system architecture.

mechanism out of the real-time loop demanded by the vehicle control subsystems. The planning system, even with the increased throughput provided by advanced computing hardware, is always expected to be unable to deal with the real-time situations.

2) Intelligent Communications Interfaces: While the blackboard interaction paradigm provides many benefits in terms of programming the system, it is not a realistic model for implementation. Thus major subsystems were implemented in tightly coupled multiprocessor groups which communicate through a local area network as illustrated in Fig. 2. The components within each major module communicate through shared memory thus enabling the tight time sensitive coupling required by such processes as vehicle control.

Each major module is coupled to the local area network through a module called the Intelligent Communications Interface (ICI). This device makes each module think that it is effectively sharing an intelligent blackboard memory with all the other modules whether they really share common memory hardware or whether they really are communicating through a local area network. The ICI components are illustrated in Fig. 3, and these concepts for robot subsystem coordination are discussed in detail in [10], [11]. However, a brief discussion of these concepts will be presented below for the completeness of this article.

The ICI's communicate with one another through the ICI Protocol, a taxonomy of which is illustrated in Fig. 4. This taxonomy shows that communication is actually message-based, relying upon the transport layer provided by the local area network specifications (IEEE 802.3 was used). All traffic between modules is divided into world state information handled by reports and control information conveyed by plans. Reports are simply statements of world state in terms of the object attribute value representation chosen for the blackboard. These reports use ASCII coded symbolic descriptions so they could be interpreted directly by the human developers. Plans are effectively production rules which have been extended to accommodate real-time and continuous circumstances. Through plans it is possible to initiate both reporting and controlling actions as well as to control the state of existing plans. The conditions in plans represent either directly or indirectly some situation in the blackboard. Since plans are represented as objects in the blackboard the state of plans can be represented in other plan conditions, therefore making it

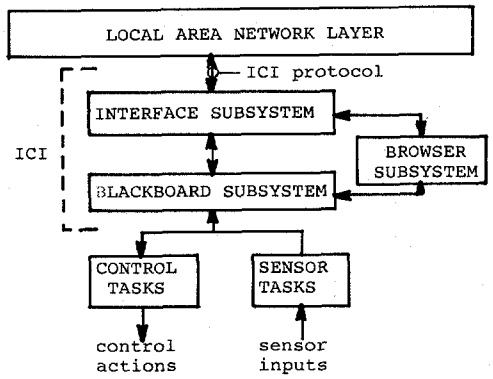


Fig. 3. Intelligent communications interface components.

possible to make plans dependent upon the state of other plans in the blackboard thus providing a very flexible control mechanism.

Fig. 5 illustrates the mechanism of the ICI. This mechanism is very similar to a production system in having components to perform plan parsing and pattern matching. This mechanism also has a component for taking reports broadcast over the network and inserting them into the local copy of the blackboard. Details of this mechanism are discussed elsewhere [10].

3) Blackboard Representation: The blackboard is a conceptual device which allows exchange of information between the GSR's subsystems. The blackboard provides a clear and consistent representation of this information which can be used by individual subsystem component developers. This means that no one person need completely understand the total system. This is certainly a realistic situation for a system as complex as the GSR.

Physically, the blackboard consists of several pieces of shared memory distributed throughout the subsystems. The ICI's provide distributed access to this shared memory as well as consistency control. Subsystems within the same module access the blackboard memory through standard blackboard interface procedures. These procedures provide well-defined mechanisms by which to read and write elements to and from the blackboard.

The blackboard is itself structured as a class tree. Each element in this tree has a list of properties to which are assigned values. This representation paradigm is commonly used in expert systems and other knowledge-based systems.

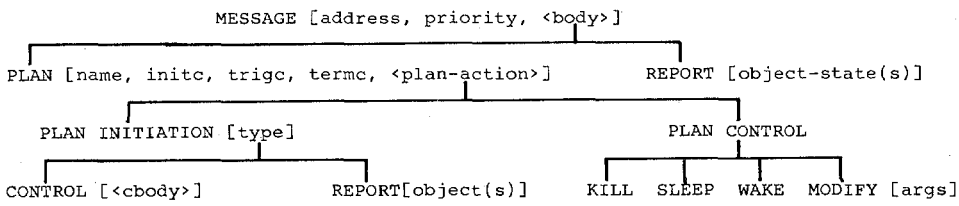


Fig. 4. ICI protocol taxonomy.

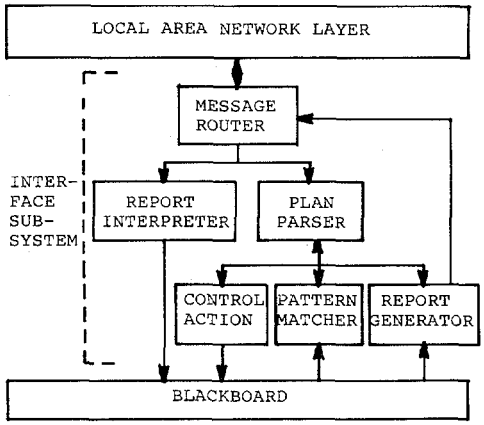


Fig. 5. ICI interface mechanism.

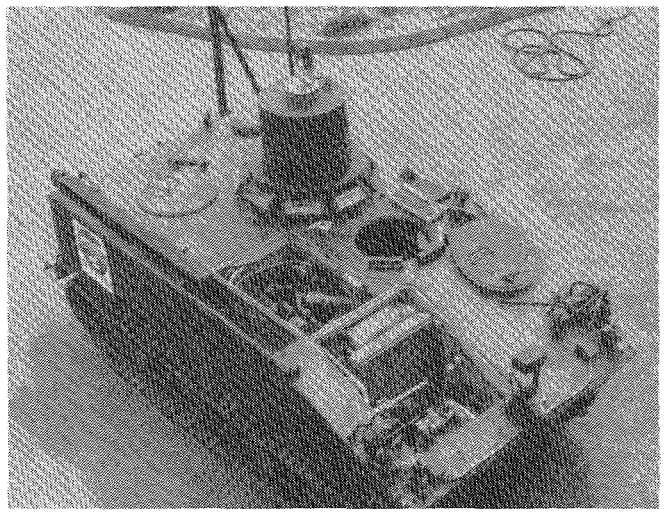


Fig. 6. Photograph of GSR.

The children elements of a class inherit all of the properties associated with the parent class and are distinguished only by the property values unique to that element. Thus the inheritance mechanism of the class tree provides an economical representation method.

Obviously, this blackboard provides the opportunity for much more than simply communicating various data values between subsystems. In fact, the blackboard provides a very powerful mechanism through which to fuse various types of sensor data [11]. Fusion of sensor data requires a consistent representation and various methods for combining the data. The representation discussed earlier must be extended to facilitate sensor data fusion. To this end, the object attribute value tuples of the class tree are expanded to include measures of accuracy, confidence, and timestamp. Most obviously, the computation of a resultant value can be derived by combining two or more sensor values through a functional dependency. This is the simplest technique for fusing data from different sensor sources since the function is predefined by the situation. The blackboard mechanism described here provides a very easy technique for this type of fusion through active functions. Active functions are devices for implementing data driven programming. Each time a value is changed to which an active function is attached, all of the values in the blackboard which are dependent upon that value are also changed to values which reflect the new change in sensor state.

Unfortunately, there are cases when two values which reflect the same value of a property in the blackboard must be fused. This situation of data fusion is much more complex. Several methods can be used to combine overlapping data including filtering, deciding, and guiding. Fusing using filtering can be illustrated by Kalman filtering or linear interpolation. Deciding is the case when some decision is made (based upon some criteria) between which of several competing values should be used. In some situations, the newest information or the data from the most reliable source is chosen. In guiding, the value from one sensor is used to guide the processing of the data from another sensor which can provide more reliable or more accurate information about the prevailing situation. An example of guiding implemented on the GSR is when low-resolution obstacle location data are used to segment the highly complex visual sensor data to provide a more accurate picture of the location of nearby obstacles. In some cases, combinations of these techniques may be chosen.

B. Mobility and Vehicle Control

1) Mobility: The vehicle subsystem is the major effector of any mobile robot. The design of the GSR uses an existing testbed to minimize costly hardware development. Fig. 6 is a photograph of the GSR vehicle. It is an M114 armored personnel carrier which has been converted for automatic control. As can be seen from this photograph, this is a tracked vehicle which is powered by a gasoline engine. Fig. 7 shows a top view drawing of the vehicle and the placement of the various externally mounted sensors.

Several features have been added to this vehicle to permit the installation of on-board sensing and computing. These include a 10-kW auxiliary power generator set, a large air conditioner, and shock mounted equipment racks.

2) Locomotion Control: The locomotion control subsystem controls all of the equipment in the GSR that is associated with its mobility. This includes vehicle throttle, brakes, steering, and transmission shift as well as the steering mode

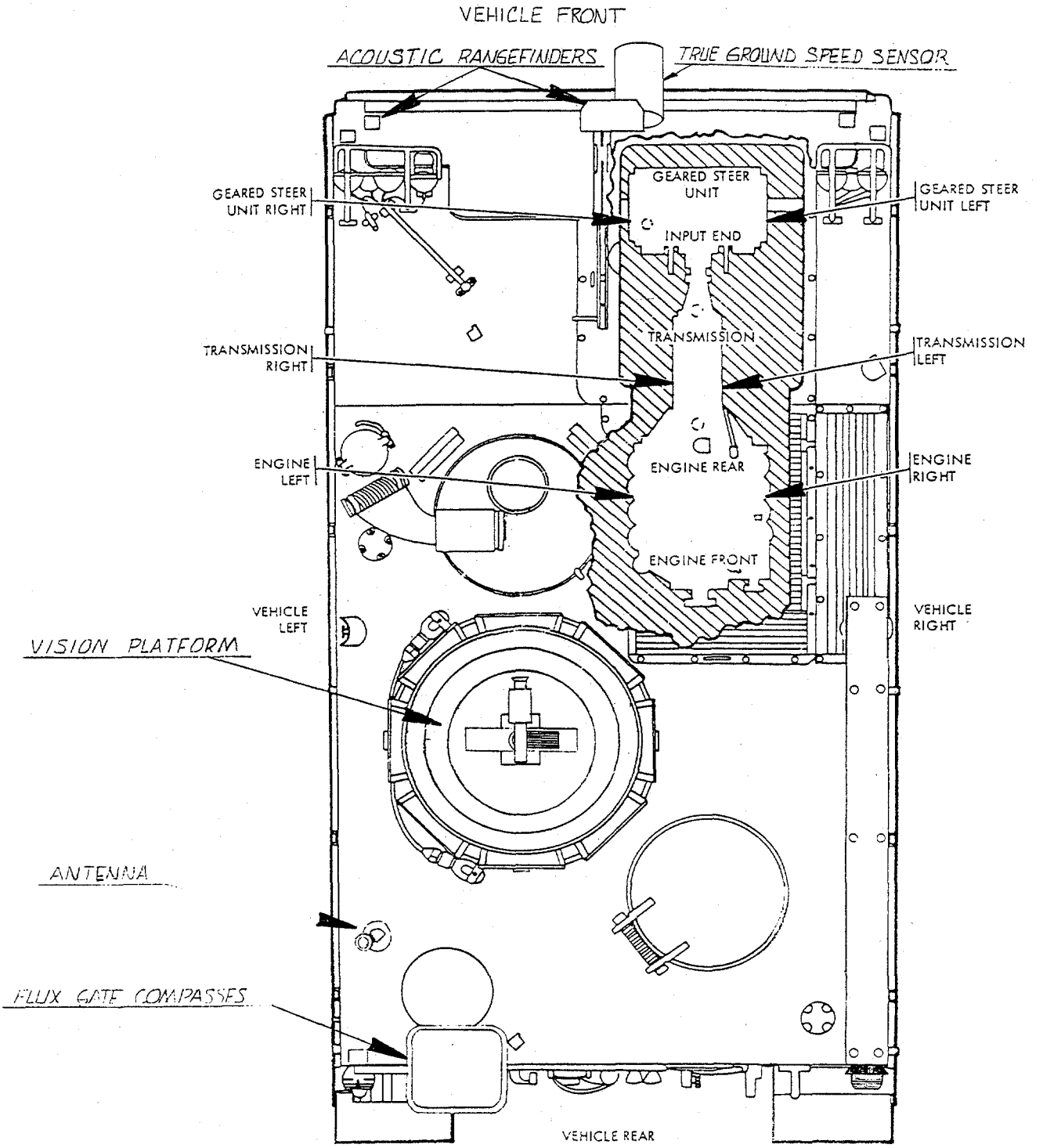


Fig. 7. Distribution of external GSR sensors.

shift, engine starting, and auxiliary power generator starting. The major control axes are actuated using electric torque motors controlled through servo amplifiers. This permits a higher level of computer control and applies well-developed technology to off-load the control computing. Steering, transmission, and throttle servo loops are closed on the actuator position so commands from the computer are given in terms of desired actuator position. The brake servo loop is closed around the applied force measured at the brake level with a load cell. This arrangement makes the brake actuator self-adjusting as well as reduces the possibility of overstressing the mechanical components of the braking mechanism.

The locomotion control computer provides access to the locomotion actuators and, therefore, to the ultimate control of the vehicle's path. The locomotion control computer maps the positions of wall contacts, a target vehicle, local goals, and idealized obstacles (in the form of avoidance points). It also generates short-range vehicle path adjustments and computes

the desired actuator positions to accomplish these changes. The desired vehicle path changes are computed from the wall, target, obstacle, and goal positions using potential field control techniques [12], [13]. In these techniques, goals have attractive fields and obstacles have repulsive fields. Information on the positions of objects such as walls, target vehicle, and other nearby obstacles is used to compute the existing forces upon the vehicle. The changes in the total potential field are computed by superpositioning all the local attractive and repulsive fields to determine a resultant force vector. The resultant vehicle motion changes are computed from this force vector and are used with conventional linear control laws to determine the desired actuator positions.

C. Vehicle Position and Orientation

Vehicle position and orientation are determined using information from the vehicle attitude and the navigation sensor subsystems.

1) Vehicle Attitude Sensor Subsystem: The vehicle attitude sensors provide information on the relative vehicle position, roll, pitch and heading angles, forward speed, and rotational velocity. The complexity in deriving this information comes from using sensors with distressingly finite limitations. For instance, the speed sensors are inaccurate over certain ranges. The track speed sensor provides very good information when the vehicle is traveling slowly and over terrain where the vehicle has very good traction, whereas the Doppler speed sensor provides reliable information only at relatively high speeds (e.g., >2 m/s) regardless of whether the tracks slip or not. Similarly, the gyrocompass drifts with time but magnetic compass is influenced by local magnetic anomalies. Neither of these direction sensors provides completely accurate information all the time. In both of these cases, an algorithm is used to decide when to use which measurement from which sensor.

2) Navigation Sensor Subsystem: The navigation subsystem provides accurate and continuous access to the vehicle's absolute position. This is done using dead reckoning sensors for continuous relative position measurement and a satellite navigation system which provides absolute position intermittently. Like the vehicle attitude sensors, the information from these two sources of position must be fused to provide a continuous and reliable estimate of the vehicle's absolute position. The estimates of relative position gained from dead reckoning sensors are continuously available and highly accurate for short distances traveled. Unfortunately, this position estimate drifts because of the accumulating errors inherent to relative position measurement. Satellite navigation provides quite accurate absolute position measurements (e.g., within 100 m) which do not drift as do the absolute estimates derived from integrating relative position information, but they are not continuously available.

D. Obstacle Avoidance

Obstacle avoidance activity can be divided into perception of obstacles and the vehicle control necessary to avoid them. The burden of local obstacle mapping is handled by the proximity sensor subsystem with some help from the vision subsystem. The proximity sensor controller controls the positions and firing of the active acoustic sensors as well as the preprocessing of the raw returns. The proximity sensor mapper assimilates all the sensor returns provided by the sensor controller into a coherent map of the local obstacles. Actual obstacle avoidance control is accomplished by the locomotion control system which was discussed earlier. Discussion of the proximity sensor subsystem is divided into discussions of the proximity sensor hardware, the sensor controller, and the sensor mapper components.

1) Proximity Sensor Hardware: The proximity sensor subsystem consists of seven Polaroid acoustic ranging sensors with a beam width of approximately 30°, a maximum range of approximately 10 m, and resolution of approximately 0.17 m. The sensors have been concentrated in front of the vehicle since it travels forward most of the time. The arrangement of these sensors is shown in Fig. 7. Three sensors are fixed looking over the front of the vehicle each separated by 30° of azimuth. Four of the proximity sensors are mechanically steered and these are located near the left and right sides of the vehicle. Each of the steered sensors can be rotated through approximately 180° azimuthally.

2) Sensor Controller: The sensor controller processing acts like a small operating system which allocates the resources depending upon the condition of the task at hand. Three tasks are possible: following a moving vehicle, tracking a wall, and finding a way through an obstacle field toward a distant goal. Sensor resources include access to a sensor, access to a region, and blocking of another sensor because of some potential interaction (e.g., through electromagnetic interference). The sensor controller receives the raw echo, filters returns, computes object range and approximate bearing, and estimates the object angular velocity. The sensor controller puts important data directly into the blackboard (i.e., time critical data such as target vehicle and wall positions and relative velocities). It sends all filtered data to the proximity sensor mapper. The sensor controller also schedules sensor coverage for target tracking, wall following, unknown sector mapping (on request from the sensor mapper or other nonlocal sources), and locates free paths in the direction of the goal. The sensor controller must coordinate sensors so no interference is caused by adjacent sensors or sensor firing into the same area.

Various sensor allocation schemes are available to the sensor controller depending upon the task at hand. This flexibility is the primary advantage of having both fixed and steered sensors. For instance, when tracking a target vehicle to execute vehicle following, it is possible to obtain an improved target vehicle position estimate by using opposing steered sensors to triangulate upon the vehicle. The sensor controller can also use multiple returns from the same target to make crude estimates of the relative target speed. These estimates can also be used together with range and angle gates to help differentiate a target vehicle from the surroundings.

The acoustic ranging sensors are notorious for being sensitive to several forms of interference. These include reflected returns from other sensors, electromagnetic interference from other nearby sensors, and electrical noise produced by the electric motors moving the sensors as well as other

sources. All of these forms of interference must be considered when determining the moving and/or firing order of the various sensors. The movement of the vehicle in an obstacle field has been simplified somewhat by using a sensor allocation algorithm which looks only for free space (as opposed to obstacles) in the direction of a distant goal. Thus, if the vehicle is traveling over a relatively uncluttered space, it can move fairly rapidly since its movement is not limited by the need to search the entire space near the vehicle for obstacles. On the other hand, if the space near the vehicle is quite cluttered then the speed must be appropriately reduced to gain sufficient information to avoid nearby obstacles in the direction of the goal.

3) Sensor Field Mapper: The proximity sensor mapper transforms all range and bearing information received from the proximity sensor controller into the absolute reference frame (using position information from the navigation sensor subsystem) and constructs a local obstacle map from that data. The information conveyed by the raw sensor data is fused into a single coherent estimate of nearby obstacle positions by representing each return by a distribution representing the probability that a reflecting obstacle was actually detected and by superpositioning the distributions associated with each sensor measurement. These distributions are formulated from *a priori* knowledge of the sensor behavior. The technique adopted is very similar to that discussed in Moravec and Elfes [14] except that they represented their local map as a grid of cells each representing a particular probability of being occupied, and our technique represents returns as line segments similar to those used by Crowley [15] and Miller [16]. This technique thus combines multiple returns from different vantage points (since the vehicle is constantly moving) to improve the accuracy of the local obstacle map. This improves the map's robustness despite the fact that it is constructed from data generated by sensors which are so susceptible to various types of noise that the measurements are individually unreliable.

The proximity sensor mapper also derives vehicle path suggestions in the form of approach and avoidance points from the existing local obstacle map. Approach points are local goals chosen from knowledge of the direction of the distant goal (formulated by the path planning subsystem) and of the availability of obstacle free corridors large enough to permit vehicle passage. Avoidance points are chosen from the points of obstacles closest to the straight line drawn between the vehicle's current position and the nearest approach point. In this way the vehicle can travel toward the closest approach point and still avoid the most important obstacle positions.

This technique also refines the detailed and constantly changing local obstacle map represented by the proximity sensor mapper to a few points which can be economically represented in the system blackboard. In this way, only a few points need to be updated when new information is received instead of the entire obstacle map which greatly decreases relatively expensive accesses to the blackboard (these are made expensive by the need for operating system context changes and reporting over the local area network). Reporting only new or unexpected deviations from the map or past suggestions greatly improves the performance of the mapper as well as the entire system by significantly limiting communication between subsystems to maintain blackboard consistency. Approach and avoidance points are represented in the blackboard by their absolute positions computed from information on the vehicle's absolute position from the vehicle attitude and navigation sensor subsystems. This choice enables the information from multiple vehicle positions to be easily combined and is thought to be the most economical choice because there is usually only one vehicle and multiple obstacles.

The proximity sensor mapper can also request additional sensor coverage from the proximity sensor controller in the direction of unknown areas. This loose coupling between the mapper and controller limits communication between these elements to a minimum and yet permits the mapper to get further data where existing information is missing or ambiguous. Vision data can also be used to coordinate the proximity sensor resources through the sensor controller to improve the local obstacle map and to enable tracking of major terrain features from the distant terrain mapping information.

E. Terrain Modeling

1) Vision and Rangefinder Sensors: The sensors on the GSR which provide the information for terrain modeling consist of a laser range finder which can make random access range measurements within the field of view of the cameras, a high-resolution (e.g., 512 × 490) gray-scale solid-state camera and a low-resolution color camera. All of these sensors are mounted on a three-degrees-of-freedom vision transport platform.

The problem of developing a terrain model from the data from these sensors is divided into the subproblems of low-level gray-scale processing, low-level color processing, gray-scale texture analysis, ground cover analysis, low-level laser range finder data processing, terrain segment construction, and terrain description integration. In addition to the processing required for terrain modeling a significant amount of processing is required to control the range finding subsystem. This processing consists of a supervisor which integrates information from the vehicle attitude sensors, maps the terrain, and plans the moving mirror paths. The range finder controller controls the data point reference frame resolution, the mirror path generator, and the laser sequencer.

2) Vision Transport Platform: The vision transport platform provides for the best utilization of the terrain modeling sensors which all have relatively narrow fields of view. Fig. 8 shows the construction of the vision transport platform. This platform enables the vision sensors to be elevated 1.5 m and rotated in azimuth approximately 360°. It also permits the camera's elevation angle to be controlled between 30° above and 15° below the horizontal. An additional axis of camera roll control is being contemplated but has yet to be implemented.

3) Representation: A proper representation must be chosen to facilitate integration of the data from the multiple sensors used for terrain modeling. Ideally, all of this different sensor data can be represented in a consistent fashion. A consistent

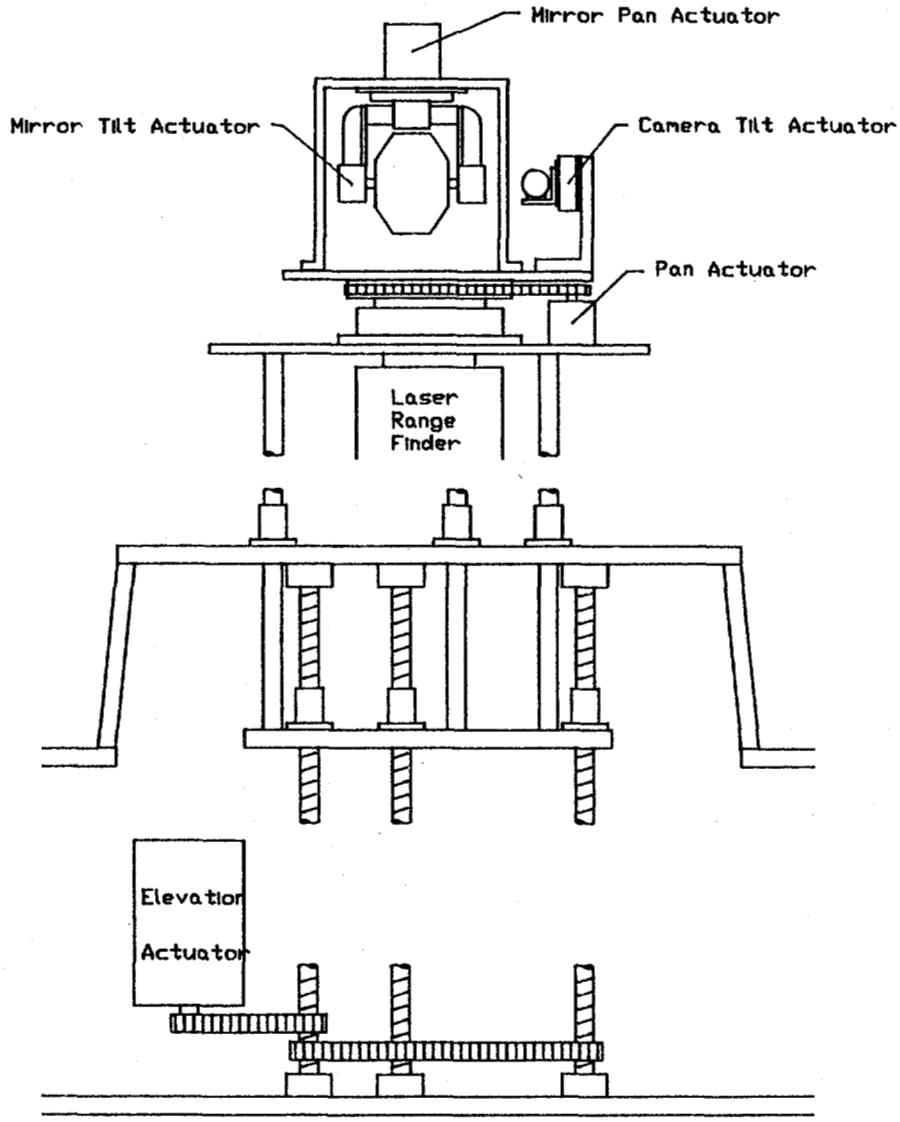


Fig. 8. Diagram of GSR vision transport platform.

representation aids in the distribution and consistency control necessary for a distributed implementation. The blackboard mechanism is used to provide a general format for representation. Within the blackboard model, the world model organizes data into a class tree with inheritance properties. Specifically, terrain data are represented as triangular segments with the properties of absolute position, orientation, adjacency, a terrain modification function, segment variability spectrum, ground cover type, and obstacle population statistics. This representation provides sufficient information to facilitate adequate route planning. This representation also avoids the problems associated with finding convex polygons while providing a very good terrain segment primitive (a triangular segment is already a convex polygon).

F. Path Planning

The planning subsystem just deals with long-range planning. Local obstacle avoidance planning is left to the control system. This decreases the direct coupling between the relatively slow high-level planning activity and the demanding vehicle controller. Actually, the experience in constructing and testing planners for outdoor situations is very limited [3], [7], [8]. Path planning in the GSR is constructed using a generic planning engine to support the specific task of vehicle route planning over unknown terrain. This approach enables the planning problems to be separated from the specifics of the route-planning problem.

1) Route Planning: Route planning in unknown situations has the two interacting subgoals of collecting sufficient information to enable fruitful planning and of actually making progress toward the final goal. The distribution of robot activity between these competing goals must be optimized to arrive at the goal as quickly as possible.

In general, the route planner minimizes a cost function based upon the energy requirements, the estimated unknown hazards, the estimated vehicle transit speed, and the distance to the goal. This function is very similar conceptually to many other approaches to a similar route-planning problem. The GSR's planner performs this function from the present vehicle location to the final goal location if sufficient map information

is available internally. However, if the terrain between the present vehicle location and the final goal is unknown, then sensor plans are generated to collect the terrain information within the sensor horizons. If the final goal is not within the sensors' ranges, then an intermediate goal is chosen which provides the best trade-off between approaching the final goal, gaining new information from the best possible vantage point, and minimizing energy consumption and travel time. In this way, the process of transiting unknown terrain consists of transiting from one intermediate goal within the sensor horizons to another intermediate goal toward the final goal location. The process of finding a path to a goal location outside of immediate sensor range over unknown terrain effectively describes the body of human knowledge called orienteering which provides various heuristics for position finding, route planning, and route following.

2) Planning Engine: Development of a planning engine for an autonomous vehicle which is appropriate for route planning in unknown natural terrain presents several challenges [17]. Ideally, orienteering knowledge can simply be encoded into the representation of some generic planning engine while the actual planning activity occurs as a natural function of the engine. This approach maximizes the flexibility of the system and readily enables the programming of competing goals. The planning mechanism should plan for the future using the present sensor information about the surroundings so that it does not interfere with the function of tight looped control systems. Since terrain information is supplied incrementally by the vehicle sensors, it should also plan from evidential, incremental and, thus, uncertain information. To make the best use of limited computational resources, the planning mechanism should be able to focus its processing demands to concentrate on the most important issues in a rapidly evolving situation. Unlike expert systems' reasoning mechanisms, this planning mechanism need not have an explanation function since once it is debugged no one will likely ask how it works, and maintaining the reasoning trace needlessly burdens the efficiency of the mechanism. In addition, the planning engine must reason in time, plan in widely different time scales (i.e., seconds to hours), and still be able to keep pace with the evolution of events in real time. It must simultaneously model several different environmental situations, approach planning as a process of continual replanning to be able to handle continuously failing plans, plan using both goals and constraints, and plan in spite of uncertainty, inaccuracy, and the unknown in the sensor data. Ideally, the planning mechanism should be completely independent of the domain, thus permitting easy modification and debugging. Finally, the planning mechanism should be as simple and uniform a mechanism as possible while still operating efficiently on a large data base. This implies that any search of the complete data base should be minimized during normal planning operation.

The requirements described present a considerable challenge to any existing planning concept. Many concepts address a small subset of these requirements but fail to approach the performance described by all of these criteria together. For this reason a completely new planning mechanism was designed for the GSR, although it could be applied to any planning problems with similar requirements (e.g., almost any robotics situation). The first generation of this engine has been implemented, but it has not been installed on the vehicle. However, detailed discussion of the design of this system is out of the scope of this article.

IV. Implementation Status and Experience

Like many projects of this grand a nature, a significant portion of the aforementioned design remains only partially implemented. Fig. 2 illustrates with broken lines the subsystems described above which have not been implemented. However, this project is very incremental in nature, and all of the present subsystems could use various degrees of improvements. In spite of the lack of completion of this project, much useful experience has been gained. This experience is described in terms of resolved and unresolved issues.

A. Project Status

The vehicle presently has the ability to follow a target vehicle and to avoid those obstacle conditions which it can sense with the proximity sensor subsystem. The locomotion control system presently uses the potential field obstacle avoidance approach. However, its performance could be improved through various tuning actions. All proximity, locomotion, vehicle attitude, and navigation sensors have been installed aboard the vehicle, and a significant portion of the processing for these sensors has been implemented and tested. The terrain analysis and planning capabilities have not been implemented on the vehicle, although various degrees of development and prototype implementation have occurred. All of the existing modules have been integrated through the ICI's and the function of the blackboard paradigm has been demonstrated.

B. Resolved Problems

Solutions to several major technical problems inherent to autonomous systems implementation were derived and tested during the GSR's development. In particular, the solutions addressed overcoming sensor processing bandwidth limitations, coordinating planning and control processes, fusing sensor data, representing system knowledge, and coordinating the interactions between the various sensor and control subsystems.

1) Sensor Processing Bandwidth Limitations: The major sensors on the GSR provide a high volume of data flow which must be processed to extract critical information. The processing, if approached in a straightforward way, would in most cases exceed the time available to make the necessary control corrections. As a result, several steps were taken to shortcut this processing: 1) using high-resolution sensors to concentrate on the aspects of the environment most distant from the robot, 2) using low-resolution sensors closer to the robot where the quickest responses are necessary, and 3) using the information from low-resolution sensors to guide the processing of high-resolution data.

Processing in the vision sensors was structured to concentrate on the distant scenes. Concentrating on the distant both

increases the time available before a particular situation is encountered and decreases the environmental detail available to the sensor which must be processed. This is not to say that the near field is ignored. Low-resolution sensors such as the acoustic proximity sensors provide the primary source for this information. This allocation proved ideal since the major necessary near-field information was used primarily for obstacle avoidance and required only coarse spatial resolution. The processing of the proximity sensor information was also broken into two parts: very nearby and more distant. Of course, the very nearby information was processed first and the more distant and, in most cases, the more processing intensive data were processed later. This processing was allocated to separate processors which were running asynchronously. The architectural approach described greatly facilitated this partitioning.

One additional way to cope with the sensor bandwidth limitations was to use the low-resolution sensors to guide the processing of the high-bandwidth sensors (such as vision). In particular, the information from the acoustic sensors was designed to aid in the segmentation of the gray level vision data.

2) Control and Planning System Coordination: Control and planning systems have historically and probably will continue to function with widely different situational constraints. Control systems must typically operate very closely to the environmental circumstances and, therefore, must have the tightest coupling to the real-time situation. Their responses must occur with very little delay. Also, in general, specific types of control processing work well within very limited circumstances and their performance degenerates when the circumstances deviate from the ideal. At this point, different types of control approaches are needed. Planning systems can usually deal with a wide variety of situations, but they impose very large time delays especially when large context changes are needed. In addition, planning systems can or should be able to function when uncertainty or inadequate sensor information is present. Control systems need fairly well-defined situations with adequate sensor information for stable control. Some difficulty arises because complex robots need the abilities of both of these systems, but they must somehow be coordinated.

The approach taken in this effort, which has proved successful, is to decouple the control and planning systems. Each is fed with the same sensor information but uses it for entirely different purposes. The control system uses the blackboard information for tight loop control of the vehicle functions while the planning system uses the prevailing sensor picture (as represented on the blackboard) to make predictions of the situational trends. It then develops future plans (for both control and sensor allocation) to deal with these expected situations. Once the initial planning has been completed and the robot has actually started executing the task the planning system continually assesses the validity and progress of the prevailing plans. If the situation seems to be deviating from one in which those plans are appropriate or effective, then replanning is done to generate an expected situation more consistent with the overall system goals. In other words, the planner is continuously replanning throughout the task. This strategy keeps the planning mechanism out of the real-time loop demanded by the control systems. The planning system even with the increased throughput provided by advanced computing hardware is always expected to be unable to cope with the demands of real-time situations so some kind of decoupling will always be necessary.

This activity and system functional decomposition is facilitated by the distributed blackboard mechanism where the best assessment of the existing circumstances is continuously available to all robot subsystems and where control information to the subsystems can be provided through the ICI plan structure. The control system therefore operates using the desired control law until the situation changes to make that control strategy ineffective. At this point a new plan is activated, and the old plan is flushed. The ICI plan mechanism permits a whole series of interconnected contingency plans to be placed in direct access to the target subsystem. This organization enables the real-time constraints of the prevailing situation to be eased from the shoulders of the planner and placed into the domain of the coordination system.

3) Knowledge Representation: The blackboard is a conceptual device which enables exchange of information between the GSR's subsystems. The blackboard provides a clear and consistent representation of this information which can be used by individual subsystem component developers. This means that no one person participating in system implementation need completely understand the total system. This is certainly a realistic situation for a system as complex as the GSR.

Physically, the blackboard consists of several pieces of shared memory distributed throughout the subsystems. The ICI's provide distributed access to this shared memory as well as consistency control. Subsystems within the same module access the blackboard memory through standard blackboard interface procedures. These procedures provide well-defined mechanisms by which to read and write elements to and from the blackboard. The blackboard is itself structured as a class tree. Each element in this tree has a list of properties to which are assigned values. This representation paradigm is commonly used in expert systems. The children elements of a parent class inherit all of the properties associated with the parent class and are distinguished only by the property values unique to that element. Thus the inheritance mechanism of the class tree provides an economical representation method.

4) Sensor Data Fusion: A significant amount of data is fused in the GSR. This ability provides a critical flexibility enabling repeated reorganization of the sensor processing functions.

The blackboard scheme provides the opportunity for much more than simply communicating various data values between subsystems. In fact, the blackboard provides a very powerful mechanism through which to fuse various types of sensor data [12]. Fusion of sensor data requires a consistent representation and various methods for combining the data. To this end, the object attribute value tuples of the class tree must be expanded to include measures of accuracy, confidence, and timestamp. The computation of a resultant value can be derived by combining two or more sensor values through a functional

dependency. This is the simplest technique for fusing data from different sensor sources, since the function is predefined by the situation. The blackboard mechanism described here provides a very easy technique for this type of fusion through active functions. Active functions are devices for implementing data-driven programming. Each time a value is changed to which an active function is attached, all of the values in the blackboard which are dependent upon that value are also changed to values which reflect the new change in sensor state.

Unfortunately, there are cases when two values which reflect the same value of a property in the blackboard must be fused. This situation of data fusion is much more complex. Several methods can be used to combine overlapping data including filtering, deciding, and guiding. Fusing using filtering can be illustrated by Kalman filtering or linear interpolation. Deciding is the case when some decision is made (based upon some criteria) between which of several competing values should be used. In some situations, the newest information or the data from the most reliable source is chosen. In guiding, the value from one sensor is used to guide the processing of the data from another sensor which can provide more reliable or more accurate information about the prevailing situation. An example of guiding implemented on the GSR is when low-resolution obstacle location data are used to segment the highly complex visual sensor data to provide a more accurate picture of the location of nearby obstacles. In some cases, combinations of these techniques may be chosen.

5) Subsystem Coordination: The blackboard mechanism provides a very powerful tool for flexible subsystem coordination. As much intelligence was distributed to each subsystem as possible to minimize intercomponent communications and decrease the real-time processing burden. Intelligent sensor and control subsystems used blackboard sensor information directly. The system planner sat above the sensor and control components and monitored the sensor traffic to build a picture of the world and to extrapolate expectations of the future from this current picture. Plans were derived from these expectations and sent to the sensor and control components to coordinate their future interactions.

Since the optimal subsystem partitioning of the GSR was not known at the time the initial design was conceived advance identification of the best interaction paths between subsystems was not possible. This situation eliminated the possibility of using a strictly hierarchical design and coordination strategy. Certainly some form of system hierarchy evolved during the implementation, but many of the details of the final result changed en route. The blackboard mechanism enabled this evolution to occur unfettered by uninformed decisions made early in the design process.

The ability to distribute plans and thus intelligence to the subsystems dynamically enabled the subsystem organization and interactions to be reconfigured at system runtime. As a result, the subsystem coordination strategy could be changed in response to varying environmental circumstances. Distributed plans also provided an additional and powerful debugging aid since system developers could make minor changes in the subsystem organization without expensive downloading each time the change was desired.

C. Unresolved Problems

Unfortunately but realistically, satisfactory solutions were not discovered for all the problems encountered during the GSR development. Interestingly, these were, for the most part, not strictly technical problems although they had technical components. These major unresolved issues include managing the complexity inherent to an autonomous system development, monitoring and debugging the system during development, dealing with the situation of exploratory system implementation and addressing the coupling of system testing with safety.

1) Complexity Management: The implementation of the GSR was a very complex process, and it produced a very complex device by all present standards. Managing this complexity proved to be very difficult. The complexity of the device came, in part, from the very flexible implementation strategy because almost any connection between any of the subsystems was possible. This provided a staggering number of possible interaction combinations. Further complexity was introduced by having a system with a large number of sensor, processing, and control components. This generated a considerable number of possible failure modes and made debugging the system a very complex and time-consuming process. Needless to say, significant complexity was introduced by the amount of software in the vehicle. When the project was initiated a considerable amount of thought was devoted to keeping track of this complexity. However, as the implementation proceeded many of these procedures were abandoned because they imposed unacceptable burdens upon the developers working in an exploratory situation where design changes were daily occurrences.

Additional complexity arose from the need to have several separate people implement the system. The work started with only three full-time people and evolved to a situation where the efforts of 12 people were dedicated to the project. This transition as well as the use of part-time personnel (such as students) produced many loose ends; many of which were never tied. Much of the system remains undocumented, and ancient bugs still remain hidden in the system despite considerable testing.

Some systematic approach to the management of this complexity is needed before sophisticated autonomous systems will evolve to practical utility. All the existing complexity management tools and techniques were either used consistently or tried and discarded in the course of this effort. Despite these measures, this problem remained a significant stumbling block throughout the effort.

2) System Monitoring and Debugging: The complexity of this system made system monitoring and debugging very difficult. The ICI mechanism proved to be the most valuable system monitoring and debugging tool available, but it was still inadequate. An integrated set of debugging tools is sorely needed which makes the dissociation of hardware and software problems easier and less time consuming. Unfortunately, the ICI mechanism did not provide adequate capability as it was implemented to enable little more than coarse-grained system performance analysis. Further tools for this purpose would

greatly assist the implementation process. These tools were not developed during this effort because of the limited time and financial resources available. Thus much potential research remains in the development of system debugging and performance monitoring tools for the implementation of complex autonomous robots.

3) Exploratory Implementation: Projects which have contributed the most experience to the collected knowledge of complex hardware and software systems development usually began with well-defined design goals and plans. These are projects in which the developers knew beforehand which design elements would work and which would not. Unfortunately, autonomous systems do not have very much collected implementation experience at all. At this time, it is not clear which design elements are the most useful. Thus developers of today's autonomous systems must be prepared to alter their design ideas throughout the entire development of the autonomous system. This situation is similar to that encountered by developers of artificial intelligence systems. For these systems a new type of programming has evolved called exploratory programming for which such languages as LISP are ideally suited. The exploratory programming model is one where the developer builds and tests the system a little at a time until a final system implementation is achieved which exhibits the desired behavior.

Very sophisticated knowledge-based programming tools have evolved to support exploratory programming (e.g., expert systems shells, hot editors, and incremental compilers). However, autonomous systems developers do not have the luxury of having a set of tools already available to them to make their jobs easier. Existing complex system design tools assume that the designer knows at design time what works and what does not and very little of the entire autonomous system needs the very sophisticated tools provided by artificial intelligence programming environments. Existing concepts for system configuration management do not apply to this situation largely because the system design is sometimes changing on a daily basis. Cumbersome but very useful system documentation procedures quickly degenerate to disuse because of the rapid changes and the sizable investment of developer time required to follow them. Rapid hardware and software changes make any attempts to keep consistent system documentation very expensive to impossible. Clearly, some techniques are required to help autonomous systems designers over the hump until they can rely on sufficient past experience to have some confidence in their designs and thus to take advantage of existing proven design techniques and tools.

The lack of available design tools and theory for the development of autonomous systems also means that there is no way to design an autonomous system to achieve some complex task reliably. This means that the system must be implemented and tested incrementally throughout the design and implementation process. However, at no point along the way can the developer have any well-founded confidence that the system he is developing will meet the design goals. Once the system has been implemented, there is no way to test that system reliably to be sure that it meets the original design goals. This situation does not inspire confidence in the customer community and must be resolved before practical autonomous systems will be widely accepted.

4) Safety and System Testing: Preserving safety during autonomous system testing presents a particularly difficult situation because these are systems which are made to operate without human assistance. This is not to say that they must be tested without human supervision. However, provisions must be made in the design of the system to permit such supervised testing. This complicates the system design and implementation often requiring radio links or other forms of command links. The supervising humans must watch what is going on in the system and be able to identify any failure before it causes the system to damage itself or its surroundings. They must also be able to take sufficient control of the system to rescue it in the event of misbehavior. This situation is further complicated by the existence of additional dangerous effectors that the system has (e.g., large vehicle, dangerous lasers). Very careful testing procedures must be instituted and controlled. During the several years of GSR development we were both very careful and very lucky. Not a single accident occurred in several hundred hours of testing and demonstrations, but the potential of serious injury was omnipresent throughout this time.

V. Conclusion

This project has involved several years' work, and many lessons have been learned in that time. However, most of these lessons have involved implementation techniques as opposed to system design. Overall, the chosen design has proven quite successful, and very few deviations from the original design ideas have been taken. The most important contribution of this effort has been the development of a very flexible integration scheme which is applicable to many different robot system applications. The choice of an integration technique which forces most of the specifics of the subsystem interfaces into a database has been instrumental in the success of the implementation of the GSR to date. This has enabled the quick changes and the incremental development which are critical to any effort dealing with hitherto unknown development areas.

One area which has continued to hamper this development has been the nearly total lack of a comprehensive set of debugging tools which deal with both system hardware and software. Often one is reduced to pursuing parallel paths of software and hardware debugging techniques just to identify where the offending problem hides. A significant portion of this development effort has been devoted to pursuing system bugs. When this problem has been revealed to other robot system developers, a profusion of advice has been received. However, the most common failure mode is for all subsystems to work independently, and when they are integrated, the system performance just goes away with no indication as to the area of fault. This failure mode involves all of the subsystem developers who all simultaneously believe that their efforts are flawless. Again, this situation was the most common situation experienced during this development. An integrated set of debugging tools designed specifically for robot systems would be a truly valuable contribution to the development of future autonomous systems.

ACKNOWLEDGMENT

This work was performed while the author was employed by the Naval Ocean Systems Center, Code S442, San Diego, CA 92152. The author is extremely grateful to the many people who have contributed to this program for the past five years.

REFERENCES

[1] N. Nilsson, "A mobile automaton: An application of artificial intelligence techniques," in *Proc. 1969 Int. Joint Conf. Artificial Intelligence,* Washington, DC; May 1969, pp. 509–520.

[2] M. Smith *et al.*, "The system design of JASON, A computer controlled mobile robot," in *Proc. IEEE Conf. Cybernetics and Society,* San Francisco, CA, Sept. 1975, pp. 72–75.

[3] S. Yerazunis, D. Frederick, and J. Krajewski, "Guidance and control of an autonomous rover for planetary exploration," in *Proc. IEEE Milwaukee Symp. Automatic Computation and Control,* Milwaukee, WI, Apr. 1976, pp. 7–16.

[4] J. Miller, "A discrete adaptive guidance system for a roving vehicle," in *Proc. IEEE Conf. Decision and Control,* New Orleans, LA, Dec. 1977, pp. 566–575.

[5] H. Moravec, "The Stanford cart and the CMU Rover," *Proc. IEEE,* vol. 71, pp. 872–884, July 1983.

[6] G. Giralt, R. Chatila, and M. Vaisset, "An integrated navigation and motion control system for autonomous multisensory mobile robots," in *Proc. 1st Int. Conf. Robotics Research,* Bretton Woods, NH, Aug.–Sept. 1983.

[7] J. Nitao and A. Parodi, "A real-time reflexive pilot for an autonomous land vehicle," *IEEE Contr. Syst. Mag.,* vol. 6, pp. 14–23, Feb. 1986.

[8] J. Lowrie *et al.*, "Autonomous land vehicle," Annual Report, MCR-85-627, ETL-0413, Martin Marietta, Denver Aerospace, Denver, CO, Dec. 1985.

[9] S. Harmon, "Autonomous vehicles," in *Encyclopedia of Artificial Intelligence.* New York: Wiley, 1986.

[10] S. Harmon *et al.*, "Coordination of complex robot subsystems," in *Proc. IEEE Conf. Artificial Intelligence Applications,* Denver, CO, Dec. 1985, pp. 64–69.

[11] S. Harmon, G. Bianchini, and B. Pinz, "Sensor data fusion through a distributed blackboard," in *Proc. IEEE Conf. Robotics and Automation,* San Francisco, CA, Mar. 1986, pp. 1449–1454.

[12] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proc. IEEE Conf. Robotics and Automation,* St. Louis, MO, Mar. 1985, pp. 500–505.

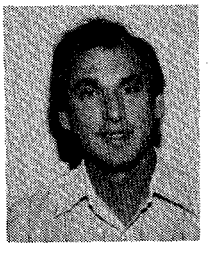
[13] B. Krogh, "A generalized potential field approach to obstacle avoidance control," in *Proc. SME/RI Robotics Research Conf.,* Bethlehem, PA, Aug. 1984.

[14] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *Proc. IEEE Conf. Robotics and Automation,* St. Louis, MO, Mar. 1985, pp. 116–121.

[15] J. Crowley, "Navigation of an intelligent mobile robot," *IEEE J. Robot. Automation,* vol. RA-1, pp. 31–41, Mar. 1985.

[16] D. Miller, "A spatial representation system for mobile robots," in *Proc. IEEE Conf. Robotics and Automation,* St. Louis, MO, pp. 122–127, Mar. 1985.

[17] S. Harmon, "Comments on route planning in unknown natural terrain," in *IEEE Conf. Robotics and Automation,* Atlanta, GA, Mar. 1984, pp. 571–574.



Scott Y. Harmon (M'78) is presently the owner of a small robotics research and development firm, Robot Intelligence International. Prior to that he pursued robotics research with the Naval Ocean Systems Center, both in San Diego, CA. He has over 25 publications in the areas of robot communications, robot computing architecture, autonomous vehicles, and mobile robot planning.

He is a member of ASME, APS, SPIE, AAAI, SME/RI, and Sigma Xi.