An Architecture For Reflexive Autonomous Vehicle Control

David W. Payton

Hughes Artificial Intelligence Center
23901 Calabasas Road, Calabasas, CA 91302

Abstract:
We describe a software architecture to support the planning and control requirements of an autonomous land vehicle. This architecture is designed specifically to handle diverse terrain with maximal speed, efficacy and versatility through the use of a library of reflexive strategies specialized to particular needs. A hierarchy of control is built in which lower level modules perform tasks requiring greatest immediacy while higher level modules perform tasks involving greater assimilation of sensor data. With all levels of the hierarchy operating in parallel, higher level modules perform selective activation and deactivation of lower level responses with limited interference from demands for immediacy, allowing the system to exploit the advantages of both immediate and assimilated data. The reflexive component of this system has been demonstrated both in a detailed real-time simulation and on a small indoor robotic vehicle.

I. INTRODUCTION

Our objective has been to create a software system to govern an autonomous land vehicle capable of moving quickly, efficiently, and continuously over roads and cross-country terrain[3]. The vehicle must be able to overcome difficult obstacles with minimal delay, and command a versatile repertoire of strategies for each unique situation.

In creating such a system, we have predicated our research and derived our conclusions according to a progression of considerations: The autonomous land vehicle must be capable of continuous motion over diverse types of terrain. This creates the need for different strategies to provide acceptable performance under varying conditions. Each different strategy, however, may require that different tradeoffs between immediacy and assimilation be made. The versatility needed in a system architecture to accommodate these different tradeoff requirements may be strongly influenced by the manner in which the architecture is decomposed. A vertical architecture decomposition offers the opportunity to exploit the advantages of a wide variety of immediacy/assimilation tradeoffs without loss of flexibility. We present an architecture based on this approach.

II. VEHICLE PERFORMANCE REQUIREMENTS

A. Short Reaction Time

With continuous vehicle motion, sensor data is subject to quickly becoming obsolete, so the time between receiving data and acting on it is critical. The prospect of continuous vehicle motion places serious constraints on the time available for sensor data processing and plan generation. As soon as sensor data is acquired, vehicle motion begins to make that data obsolete for planning. The farther the vehicle has moved from the point where data was obtained, the less useful that data becomes. Some data may lose value more quickly than other data. For example, a sensor report of a nearby obstacle may only be useful if that report can be processed and acted upon before a collision occurs, while a report of a distant hill or mountain may remain useful for quite some time. In certain instances, data becomes obsolete when different vantage points expose new aspects of the environment that were previously occluded. For example, a sensor report of a clear path ahead may quickly become invalidated by subsequent reports obtained as the vehicle moves in that direction, exposing more detail of the terrain. The problem of data obsolescence must be minimized through timely processing of sensor data and use of efficient methods for plan verification and updating.

B. Repertoire of Strategies

Diverse terrain and adversary threats pose another challenge to planning in that different environmental conditions may require the use of different strategies for effective mobility. A wide variety of planning strategies may also require a variety of sensing capabilities for their support. These requirements may lead to overwhelming computational demands if appropriate tradeoffs between immediacy and assimilation are not made.

Traveling down a road may involve considerably different planning skills and sensing capabilities than traveling cross-country. The primary concerns when following a road are to travel quickly while staying on the road and avoiding obstacles. This is unlike negotiating undeveloped terrain where, in addition to obstacle avoidance, the concerns range from climbing steep slopes to avoiding stoppage by potholes, gullies, and swamps. Different strategies may be needed in these situations to make use of specialized combinations of sensors, and to perform tasks that are specialized to the environmental conditions.

A well diversified set of task capabilities and strategies may also prove useful when adversary threat situations are introduced to the vehicle environment. Under these circumstances, predictable behavior is highly undesirable, as it provides an adversary with advantageous knowledge of one's weaknesses. With the availability of more than one method for accomplishing most tasks, the vehicle planner should be able to vary its strategies in such a way as to reduce its own predictability.

The variety of sensing capabilities needed to support different planning strategies leads to some serious sensor fusion problems. Many types of sensors will all be acquiring data about the environment. Although various sensors will be especially useful for obtaining data about particular attributes of the environment, a great deal of useful information will also be found in the interrelationships between outputs of different sensors. The task of analyzing data from all sensors and constructing a composite view of the world,

Acknowledgments: This work was supported by internal research and development funds from the Hughes Artificial Intelligence Center.

however, may be incompatible with the need for timely information about the environment. At some point, the benefits of data assimilation must be sacrificed in order to achieve the immediacy needed for real-time response.

III. DESIGN TRADEOFFS

A. Immediacy Versus Assimilation

In making the tradeoff between immediacy and assimilation, there is a wide spectrum of options. There are certain advantages and disadvantages at each extreme which are also evident in differing strengths anywhere along this spectrum.

1. Assimilation. On the positive side of assimilation is that the completeness and detail of a constructed world representation can always be enhanced with added processing of sensor data. This clearly has value to planning, since features critical to successful plan execution may not otherwise be discovered. The negative aspects of assimilation come as a result of the time required to obtain it. As a sample of sensor data is undergoing painstaking analysis to detect potential obstacles, the vehicle may have already run into one of them.

As demonstrated by work in several autonomous vehicle projects [2,8,9], thorough sensor processing may allow construction of a scene representation which extends well into the vehicle's projected path, thereby compensating for the time required to construct such a representation. Despite the fact that a portion of the model may become obsolete due to vehicle motion, there are still enough valid parts of the model remaining to properly control the vehicle. Even these projects, however, must face some requirements for immediacy, invoking action as soon as only the most minimal models for their environmental context have been constructed.

2. Immediacy. On the positive side of immediacy is the fact that the faster any sensory data can be used to effect action, the more value it has for control. The combined vehicle perception and planning systems may be considered as a single feedback control system with the environment providing the feedback signal to that system. Since greater stability in a feedback control system is generally achieved by reduction of delays through the system, greater immediacy should simplify the task of maintaining stable control. This simplification may in turn make further reduction in delay possible. Greater immediacy also has value in reducing the likelihood that unexpected changes in the environment may occur between successive data inputs, since these inputs are received more frequently. The problem with immediacy is that critical information may be difficult to obtain from sensory data that has not undergone sufficient assimilation, and it is likely that some extracted information may contain errors or inconsistencies.

Several systems have been developed which use control strategies that are designed specifically to take advantage of data immediacy whenever possible [1,4,10]. These systems have shown that immediate data can be effective in well constrained situations for which there is sufficient a priori knowledge to allow simple disambiguation of sensor data. Data from acoustic range finders, for example, can be adequate for control with very little processing, provided that the environment does not contain hazards which are undetectable to these sensors. Because immediate data loses value in loosely constrained environments, systems exploiting immediacy tend either to limit operation to well constrained environments or to employ complementary methods involving higher degrees of assimilation to establish the needed constraints.

B. Vertical Versus Horizontal System Decomposition

It is possible to view the modularization of the mobile robot problem as one with either a horizontal or a vertical decomposition [1]. In the more traditional horizontal decomposition approach [2,8,9], information flows through a series of assimilation stages, then through a series of planning stages, until finally it results in the issuance of a control signal. Each successive stage in the pipeline requires assimilated data from the preceding stage as its input. In the vertical decomposition approach [1], information is dispersed to a set of parallel layers. Each specialized layer processes data in a manner specific to its own operational goals, and issues control commands accordingly. Although higher level layers are expected to exert influence upon lower level layers, the lower layers are free to exert control over the vehicle without waiting for assimilation to be completed by the upper levels.

1. Horizontal System Decomposition.

Sensor Fusion. In the horizontal approach to system decomposition, the fusion of data from multiple sensors is generally performed through the generation of a composite representation of the local environment. A series of sensor assimilation and integration stages are typically used to produce the representation, which provides subsequent planning stages with data for vehicle control. Such representations are desirable because they can be meaningful over a broad range of situations, and they may allow for fairly general planning approaches to function effectively under a variety of environmental conditions.

Although the generality of a composite representation has its advantages, it also has some serious disadvantages. The primary drawback is that the series of stages through which all sensor data must pass places an unavoidable delay in the loop between sensing and action. Unless the composite representation is bypassed at times, thereby undermining the generality of the approach, the delay can only be reduced by increasing the throughput of some or all of the modules. At times it may occur that specialized circumstances or goals provide sufficient constraints to allow direct disambiguation of sensor data for control, but the specialized functions needed to make this possible may not be easily added if the architecture has not already been organized to accommodate them.

Flexibility. A horizontal decomposition presents some serious problems to system development as well, because of limitations imposed by the need for early interface specification. As a system is partitioned into modules, a set of interfaces must be defined to establish information pathways between modules. These interfaces are important since they specify both the information content and the format for transferred data. The specification of any module, however, depends heavily on the specification of its inputs and outputs, so modules cannot be designed without first obtaining a stable interface specification. As a result, interface specification occurs early in the design cycle of the system, placing serious constraints on immediacy and assimilation requirements before the potential capabilities of each of the modules are completely understood. Development of each of the modules is thus restricted by the interface limitations, discouraging otherwise desirable module enhancements if they cannot be structured to conform to these specifications. Furthermore, should a particular enhancement be found sufficiently useful to warrant modification of an interface, effecting such a change might require significant redesign of the system. At the very least, the modules on each side of the altered interface would need to be changed, and it is likely that these changes would result in the need to change other interfaces and modules as well. To avoid this difficulty, emphasis is often placed on maximizing the throughput of component modules rather than enhancing their functional capabilities. Faster hardware is often seen as the only practical solution to problems which might otherwise be solved with alternate module functionality were it not for the interface constraints.

2. Vertical System Decomposition.

Sensor Fusion. In the vertical approach to system decomposition, multiple sensor data may be fused through vehicle actions as well as within internal composite representations. A parallel set of sensor assimilation and fusion tasks are typically performed, leading to the generation of a variety of specialized

representations for use by a number of concurrent planning tasks. As each planning task employs representations that may be derived from only a small portion of the available sensor data, many distinct, and possibly conflicting, control commands may result. This opens the possibility of bypassing sensor fusion in exchange for command fusion.

Although the use of command fusion lacks the generality obtained from full sensor fusion, it has some definite advantages. Primarily, it allows pathways for primitive sensor data to be exploited for immediate vehicle response. For such specialized pathways to be used effectively, they must be governed on the basis of more thoroughly assimilated data to maintain proper constraints for meaningful sensor disambiguation. This capacity, however, is easily provided from parallel assimilation and planning tasks.

Flexibility. Under the vertical decomposition approach, development is organized around a hierarchical set of layers where each layer may be influenced by its superiors, but is independent from them in performing its own tasks. In this kind of hierarchy, the lower level modules perform more simple and general tasks while the higher level modules perform more complex and situation specific tasks.

The modules of a vertical decomposition may be fairly independent and simple, allowing easy modification to accommodate new approaches and solutions as they arise. With each module having its own specialization and operating goals, each may also have its own specialized interfaces. This delegates the problem of interface specification more appropriately as a subtask of module design. Interfaces can thus be designed according to the unique requirements of each module while remaining sufficiently independent to allow changes to a module's functionality with only a minor impact on other modules.

The added flexibility provided by the vertical decomposition approach allows more effective tradeoffs to be made between immediacy and assimilation. Because the design involves a number of special purpose interfaces rather than a few very general interfaces, a larger variety of different immediacy/assimilation tradeoffs may be made for the overall system. With a different tradeoff selected appropriately for each level of the system hierarchy, the total system may effectively take advantage of a wide range of immediacy/assimilation options simultaneously. Furthermore, this coverage of the immediacy/assimilation spectrum may easily be changed as the system develops and interfaces are modified.

C. Lateral Versus Horizontal Module Decomposition

Although a single layer of a vertical decomposition may itself be decomposed horizontally, it is possible to establish an alternative form of decomposition which incorporates a higher degree of parallelism — we refer to this as "lateral decomposition." The lateral decomposition of individual modules within a vertical hierarchy is performed through the division of each module into a collection of expert agents, each capable of receiving some combination of processed sensor data and communicating with others within the same module over a common blackboard. The output of each module is then formulated by an arbiter which monitors the blackboard and is responsible for arriving at a consensus from the observed activity. This decomposition allows a very fine grained approach to module construction, where small or large enhancements to system performance can be effected through the addition of new experts to various modules without disrupting existing performance capabilities.

IV. SYSTEM ARCHITECTURE

A. Basic Structure

1. Overview. We have developed a system which captures the advantages of both immediacy and assimilation, using a vertical decomposition of the overall system and a lateral decomposition of each individual layer. Through the extension of an earlier multilevel planning system [5], we have specified a four level hierarchy for this architecture, as shown in Figure 1. The distinction between different levels of the hierarchy is based primarily on the need for different degrees of immediacy or assimilation within the system. The major modules in this hierarchy are intended to be operated in parallel, with modules near the top performing tasks requiring a high degree of assimilation and modules near the bottom performing tasks requiring a high degree of immediacy. In this way, the hierarchy covers the entire spectrum of immediacy/assimilation tradeoffs, deriving unique benefits from each level.

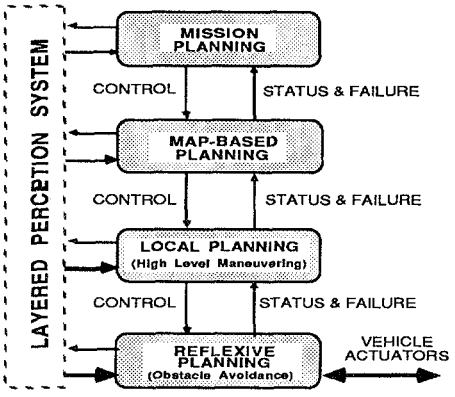

Figure 1: Hierarchical architecture for real-time autonomous vehicle control.

2. System Decomposition. In this vertical decomposition of the mobile robot problem, differing response time requirements and differing needs for programing flexibility form the basis for the segmentation of major modules. The layer of the hierarchy which interacts most directly with vehicle actuators will contain elements which are designed primarily for reflexive performance and will not be required to employ any complex inference mechanisms. Each layer above this in the hierarchy will be less specialized, making greater use of knowledge-based inference mechanisms so as to allow greater flexibility and provide extended capabilities for high level reasoning.

3. Functions of Modules. As illustrated in Figure 1, the planning hierarchy is composed of mission planning, map-based planning, local planning, and reflexive planning modules, each receiving input from a comparably layered perception system. The mission planning module has the function of translating abstract mission goals into a set of geographic goals and mobility constraints appropriate for use in map planning. This function requires reasoning at a fairly high level of abstraction with well assimilated data, and may be allowed a response time on the order of several minutes. At the next level, the map-based planner must translate geographic goals and constraints into specific route plans [7]. This level employs computationally intensive map-based reasoning with the additional use of long-term assimilated data from perception. It, too, is allowed to have a response time of a few minutes. The local planning module has the role of ensuring that a route plan gets executed properly. This module must apply heuristic planning knowledge [6] in the context of local world models constructed from assimilated perceptual data to select reflexive actions that are appropriate to the execution of the desired route plan. This module may be limited to a response time of several seconds. Finally, at the bottom of the hierarchy, the reflexive planning module has the task of maintaining real-time vehicle control. Response times under a second may be required of this module.

4. Intermodule Communication. With assimilation being provided by the upper levels of the hierarchy and immediacy provided by the lower levels, it is necessary for information and control to pass between levels in order for assimilated results to

influence the performance of the reflexive layers. In our hierarchy, constraints and specialization commands are passed downward, and failure and other status reports are passed upward, while perception and vehicle control data flow horizontally through each level of the hierarchy. To minimize the delay through the system, only specialized reflexive procedures are allowed to have direct control of vehicle actuators. These procedures, however, are activated and controlled by higher level modules performing more thorough assimilation of the environment. With a suitably large library of specialized reflexive methods and a significant knowledge base of the limitations and constraints of these methods, higher level reasoning modules are allowed to make complex navigational decisions with little regard to requirements for immediacy.

5. Module Decomposition. The lateral decomposition of individual layers in this architecture allows easy and flexible enhancement of the system and provides for the variety of task capabilities needed for operation under diverse and potentially hostile conditions. As new capabilities are developed, they may be added incrementally to the rest of the system as distinct expert sub-modules, with minimal impact on existing performance standards. As the number of sub-modules grows, simultaneous activation of all sub-modules may not be meaningful. To organize the sub-modules into meaningful groups, "activation sets" are established. An activation set is a pre-defined group of agents which, when operating together, can produce meaningful actions. In the context of the vertical hierarchy, control from upper layers may be exerted upon lower layers through the selection of an appropriate activation set for the current situation. With this grouping of expert sub-modules into activation sets, different modes of action for the planning system can be established at each level of the hierarchy. These modes can provide different operational strategies as required by environmental changes or may be used to reduce predictability in the presence of threat.

6. Vertical Decomposition Strategy. This architecture differs from that proposed by Brooks[1] in that its vertical decomposition is based most directly on immediacy/assimilation considerations rather than on the different classes of behaviors or levels of competence desired for the system. It is not surprising, however, to note that some of the higher levels of competence as described by Brooks will find themselves situated in the higher levels of this architecture as well due to the tendency for most higher levels of competence to also require a higher degree of assimilation. Without a direct mapping between hierarchy layers and competence levels, adding a new level of competence in this system may involve modification of a number of layers. This is not a difficult process, however, since the lateral decomposition of modules allows incremental enhancement at each level. The addition of a new level of competence may involve the addition of some reflexive parts as well as some new high level reasoning parts, but these additions will add to the pool of resources available to all existing expert sub-modules.

7. Focus on Lower Level Modules. Our discussion focuses on the reflexive planning module and its interaction with the local planning module, without further detailing of map-based planning and the mission planning modules. This is because the architecture we describe allows us to establish meaningful levels of competence through the lower levels alone. As the higher levels are added, these established capabilities will remain intact.

B. The Reflexive Planning Module

1. Overview. The reflexive planning module is in the lowest level of the planning hierarchy. It provides an example of how both vertical and lateral system decomposition are used to create an extensible system with diverse functionality. At the bottom of the hierarchy, it is responsible for direct control of vehicle actuators, and therefore is required to provide the highest degree of immediacy. As a laterally decomposed unit, this module consists of a large collection of expert sub-modules, each of which is capable of making decisions about vehicle actions under specialized circumstances. Higher level modules in the hierarchy exert control over the vehicle through the activation and de-activation of select groups of these sub-modules.

This module is critical to the achievement of real-time control because it provides the required high-throughput data paths between sensing and action, while requiring only low bandwidth interfaces to the higher levels of control. This results in the isolation of tight control loops from other reasoning tasks, adding to the flexibility of higher levels of reasoning while permitting real-time optimization at the lower levels.

2. Virtual Sensors and Reflexive Behaviors. Each expert sub-module of the reflexive planning module, as shown in Figure 2, is divided into two distinct elements: a perceptual component called a "virtual sensor" and an action component called a "reflexive behavior" or a "behavior." This division is useful because it allows the sharing of assimilated perceptual data between a variety of different reflexive behaviors. When one of these sub-modules is operating, a portion of the available sensor data flows through the virtual sensor where it is processed and abstracted. This abstracted data is then fed into the reflexive behavior which evokes vehicle control commands based on this data. Under normal circumstances, several virtual sensors and reflexive behaviors are operating asynchronously and in parallel. Communicating through a common blackboard, their vehicle control decisions are fed through a command arbitration unit which selects the highest priority commands and issues them to the vehicle actuators.

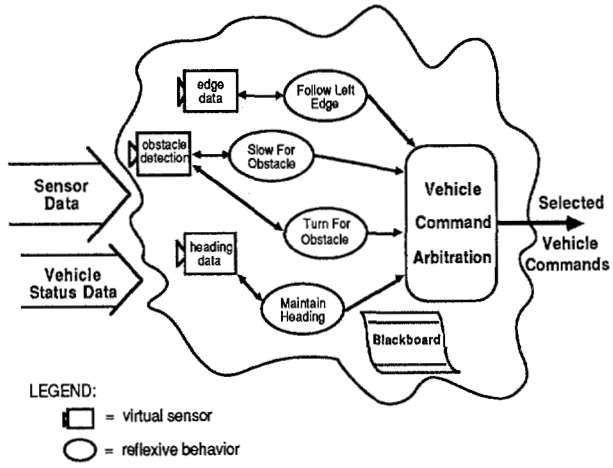

Figure 2: An activation set of reflexive behaviors and their virtual sensors operating within the reflexive planning module.

Definition of Virtual Sensors. Virtual sensors may be regarded as black box sensing devices or perceptual units which can detect very specialized environmental features. Under our use of the term, virtual sensors are purely conceptual entities and need not necessarily be identified as distinct units of hardware. Internally, a virtual sensor is simply a sensing and processing unit which provides assimilated sensor data. The degree of assimilation provided may vary significantly from one type of virtual sensor to the next, depending on the type of features the sensors are intended to detect. A specific process or cascade of processes which assimilate data from one or more sensors may be identified as a virtual sensor. In the context of Figure 1, virtual sensors are most appropriately viewed as part of the layered perception system. In this regard, they represent the tightest coupling between perception and planning for the whole system.

Definition of Reflexive Behaviors. Reflexive behaviors are highly procedural units due to the high demands for immediacy. The pseudo-code example below of a simplified version of the `slow-for-obstacle` behavior provides an illustration of some of the important features of a reflexive behavior definition:

```
BEHAVIOR: slow-for-obstacle
  Parameters: slow-distance = 20;
              stop-distance = 5;
              deceleration = 1
  Virtual sensors:
     (sensor-type: obstacle-detection  update: 2) =>distance
     (sensor-type: speedometer  update: 3) =>vehicle-speed

  DO Forever:
   PAUSE
   Wait For New Data  IF TIMEOUT, Put Command SPEED = 0
   IF distance  < slow-distance
     THEN:
       IF distance  < stop-distance
         THEN: Put Command SPEED = 0
         ELSE: Put Command
                 SPEED = vehicle-speed - deceleration
```

The key elements of this definition are the parameter declarations, the virtual sensor assignments, and the procedure specification. The parameter declarations allow the assignment of a set of default parameter values which may be altered under different activation contexts. The virtual sensor assignments define the sensor inputs to the reflexive behavior. In the example above, inputs are expected to be received from two virtual sensors, one called `obstacle-detection` and another called `speedometer`. During operation, the continuous data flow from these virtual sensors will be assigned to the two local variables *distance* and *vehicle-speed*. The procedure specification for the behavior is written as an infinite loop because it is intended to operate as a continuous independent process. The process is controlled within the loop through a `PAUSE` statement and a command to wait for new data. The `PAUSE` forces the process to wait for an update period which is determined from the update rates requested for the virtual sensors. This is intended to prevent the process from unnecessarily consuming shared processing resources. The "`Wait For New Data`" command forces the process to wait for new data from the virtual sensors. This prevents the re-use of obsolete sensor data and allows the process to handle sensor failure conditions through use of the `TIMEOUT` condition. Within the body of the reflex procedure, the "`Put Command`" operation is used to set `SPEED` to various levels. This is the mechanism through which the behavior controls vehicle action. With a group of reflexive behaviors operating concurrently, prioritized commands for both `SPEED` and `TURN-RATE` are issued and those commands with the highest priority are selected for vehicle control.

Behavior Activation Sets. The most common way to provide vehicle control is through the concurrent operation of groups of reflexive behaviors, called "activation sets." Activation sets are composed from subsets of the total collection of defined behaviors. Figure 3 illustrates an activation set selected from the pool of available reflexive behaviors. During operation, only the behaviors from the current activation set are enabled to process data and elicit vehicle commands. Similarly, only the virtual sensors which support these behaviors need be active as well. In general, a single activation set is fairly specialized in its suitability to different environmental situations. With a sufficiently large library of reflexive behaviors, however, a wide variety of different activation sets may be generated to provide a diverse set of task capabilities suitable to most any situation.

3. Activities. Because activation sets play such a key role in vehicle control, it is useful to have a standardized method for defining meaningful activation sets. The method is based upon describing the unique collection of attributes of the activation set. The method is called "activity definition," and an instance of an activation set thus defined is called an "activity." In addition to identifying the set of reflexive behaviors that must operate together, an activity definition may specify the following attributes of activities:

- **Relative behavior priorities.** It is necessary that there be a way for selecting a single command from among all the commands concurrently issued by different behaviors. With command priority as the primary criterion for this selection, each behavior places an integer priority on all of its commands. This alone, however, is not sufficient to guarantee priority uniqueness among behaviors of an activation set. Therefore, an activity definition places an ordering on its behaviors, establishing relative priorities which are used to select among commands that would otherwise have equal priority.

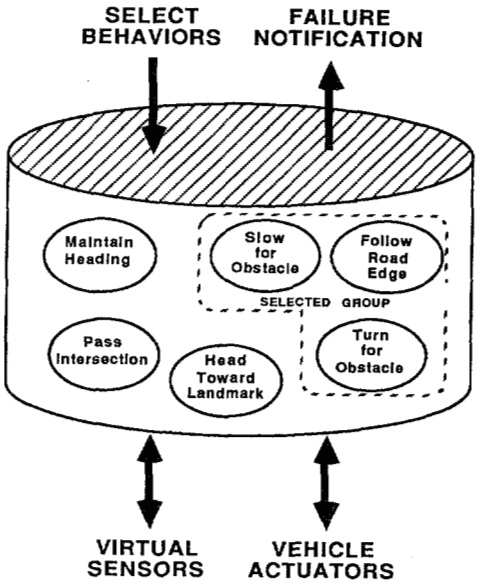

Figure 3: Activation sets are selected from a pool of reflexive behaviors.

- **Behavior parameter constraints.** Each behavior within an activation set is a parameterized procedure. Often, when behaviors are combined in different ways, it is desirable to modify behavior parameters so that cooperating behaviors can function more compatibly. An activity definition may specify parameters for any of the component behaviors, allowing specialized tailoring for maximal compatibility. Two activities composed of the same behavior set may actually be capable of performing entirely different functions if they are given different parameter constraints.

- **Initialization and termination actions.** Both before and after an activation set is enabled, it may be desirable to invoke some specialized procedures for performing certain set up or completion operations. Activity definitions can specify any number of independent initialization and termination actions as needed.

- **Sub-activity sequences.** Often, by stringing together a sequence of activities over time, the vehicle can be made to execute simple scripts. Although normally the task of sequencing activities belongs to the local planner, certain scripts may be so common that they warrant definition as activities themselves. To allow for this, activities can define sub-activity sequences. When an activity containing a sub-activity sequence is invoked, the activity will iteratively invoke the activities within the sequence until one of them causes the iteration to halt. If an activity is listed as a sub-activity of itself, it too will be invoked in sequence along with the other sub-activities. When invoked in this way, however, the activity's behavior set will be activated rather than its sub-activities. This provides a mechanism for the activity to perform transition operations between activation of its different sub-activities.

- **Failure handlers.** Under normal operation of an activation set, there are often a number of typical failure conditions that may arise. Because the actions of most activation sets can be characterized fairly well, it is also possible to determine appropriate response activities for certain failure modes. Should a failure occur, it is desirable for control to temporarily be transferred to the failure handler for corrective action, then returned back to the initial activity after the handler is finished. An activity allows the specification of failure handlers of this type for any number of failure modes.

- **Command arbitration rules.** Although there exists a default method for command arbitration, it is possible that a particular combination of behaviors may function most effectively with some alternate approach. To maintain maximal flexibility, activity definitions support this option by allowing special arbitration rules to be in effect only while the activity is enabled.

Examples of Activity Definitions. The following examples of the *WANDER*, *SEEK-HEADING*, and *BACK-AND-TURN* activities illustrate some of the features of activity definitions:

```
ACTIVITY: WANDER
 BEHAVIORS:  (slow-for-obstacle turn-for-obstacle)
 PARAMETERS: ((turn-for-obstacle turn-distance 25)
              (slow-for-obstacle slow-distance 15))
 SUB-ACTIVITIES:  Nil
 FAILURE-HANDLERS:  ((collision BACK-AND-TURN))


ACTIVITY: SEEK-HEADING
 BEHAVIORS:
   (slow-for-obstacle turn-for-obstacle maintain-heading)
 PARAMETERS: ((turn-for-obstacle turn-distance 30)
              (slow-for-obstacle stop-distance 10))
 SUB-ACTIVITIES:  Nil
 FAILURE-HANDLERS:  ((collision BACK-AND-TURN))


ACTIVITY: BACK-AND-TURN
 BEHAVIORS:  Nil
 PARAMETERS: Nil
 SUB-ACTIVITIES:  (BACKUP-BRIEFLY  TURN-ON-AXIS)
 FAILURE-HANDLERS:  ((collision FORWARD-AND-TURN))
```

The *WANDER* activity is designed to cause the vehicle to wander aimlessly through the environment without hitting any obstacles. The *SEEK-HEADING* activity is similar to the *WANDER* activity, but it also attempts to orient the vehicle toward a specified heading whenever possible. It is interesting to note that the main difference between these two activities is the addition of the `maintain-heading` behavior within the *SEEK-HEADING* activity. Since the behavior ordering within the *SEEK-HEADING* activity gives `maintain-heading` the lowest priority, *SEEK-HEADING* will operate just like *WANDER* except when no obstacles are present to activate the first two behaviors.

The *BACK-AND-TURN* activity is used as a failure handler for collisions in both of the other two activities. This is used with the idea that since both activities are expected to cause the vehicle to travel forward, a logical solution to most collision situations should be to backup slightly and turn. To perform this operation, the *BACK-AND-TURN* activity must sequence through the *BACKUP-BRIEFLY* and the *TURN-ON-AXIS* activities, providing a good example of the use of sub-activities to define a simple script.

4. Intramodule Communication. The critical communication pathways within the reflexive planning module are primarily between virtual sensors and behaviors, between behaviors and their local blackboard, and between behaviors and the command arbitration unit. All other communication, such as that required for activity selection or failure notification between the reflexive planning module and the local planning module, occurs at a very low bandwidth and does not raise serious performance concerns. For communication between virtual sensors and behaviors, specialized high throughput data channels must be available. When a behavior requests a virtual sensor, such a channel must be dynamically allocated to serve as a data pipe between the two processes. Fortunately, external synchronization is not required. The virtual sensor and the behavior will both run at their own rates. If the virtual sensor produces more data than the behavior can handle, then the excess data will simply be flushed from the pipe. If the virtual sensor does not provide data fast enough for the behavior, then the behavior will wait until it reaches its timeout limit.

Communication between reflexive behaviors and their local blackboard is optional. In fact, in an implementation of the reflexive planner on a highly parallel machine architecture, it is likely that communication along this pathway would be avoided altogether. When it must be used, the blackboard is sufficiently specialized to allow efficient information flow. Blackboard entries are simplified such that they may contain only a command and a value. For the sake of efficiency, the set of possible commands is pre-compiled before run time. In this way, the blackboard acts very much like a shared variable space except that the rules of command arbitration described below apply whenever an entry is made.

A special portion of the local blackboard is used for the vehicle control commands of `SPEED` and `TURN-RATE`. Any active behavior may enter a value for either of these commands, but the command arbitration rules determine whether or not the entry will be accepted. To transfer accepted commands to the vehicle control system, an independent process monitors these two special entries of the blackboard, and periodically relays any new values that it finds. The period of this process must be no less than the period of the highest priority behavior if correctly arbitrated commands are to be issued.

5. Command Arbitration and Selection. The default rules for arbitrating between simultaneously received commands are quite simple. Since each command entry may include a priority, commands of higher priority replace those of lower priority. Should two commands of the same priority be issued, then the priority ordering established within the activity definition is used to determine which will prevail. To perform this selection, every command entry must include a reference to the behavior that issued it. Should the same behavior issue a new entry for the same command with the same priority, that new entry replaces the old one.

6. The Control Mechanism. The reflexive planning module features an underlying mechanism that effects the change of activation states. The control mechanism performs operations necessary for efficient transitions in four types of state changes:

- **Initializing an activity.** Whenever a new activity is initialized, each of its component behaviors is initialized as an independent process, with local parameters assigned from the parameter specifications of the activity. In addition, the virtual sensors that feed the behaviors are initialized and data links are established between the behaviors and their virtual sensors.

- **Switching activities.** When a new activity is invoked to replace one that is already in effect, the behaviors of the active one are disabled and their data links are released. For efficiency, the virtual sensors and data links required for the new activation set are initialized in advance of the actual switching operation.

- **Handling failures.** Handling a failure condition through an activity's failure handler is similar to switching activities, except that control is returned to the original activity after the failure is handled. As with activity switching, best performance is obtained if all of the virtual sensors and data links for each of the error handling activities are initialized before any errors occur. Unlike in activity switching, however, the virtual sensors and behaviors of the original activity are merely put on hold while the failure is being handled so that they can readily be re-activated afterwards.

- **Executing simple scripts.** Executing a sequence of sub-activities as part of the execution of a main activity is a means for performing simple scripts. Naturally, this involves the same concerns of activity switching, but also takes into account added knowledge of future virtual sensor and data link requirements. With this knowledge, advance initialization is used to achieve greater efficiency.

C. The Local Planning Module

1. Overview. The local planning module, interacting with the map-based planning module, the reflexive planning module, and the perception system, performs the important function of selecting sets of behaviors that are appropriate for different situations and goals. Although design of this module is not yet complete, we sketch out some of its major features in Figure 4. The local planner receives its goals from the map-based planning system in the form of a route description and path constraints; it receives knowledge of the vehicle environment from scene models provided by the perception system; and it receives operating status and failure information from the reflexive planning module. Internally, the route description and path constraints are translated into specific action goals, the scene models are used to construct a detailed local map of the vehicle's environment, and the behavior status data are monitored to ensure that the current action goals are being satisfied. The primary function of selecting appropriate behavior activation sets is performed through the assimilation of action goals, local map data, and behavior status data with respect to knowledge of behavior performance characteristics.

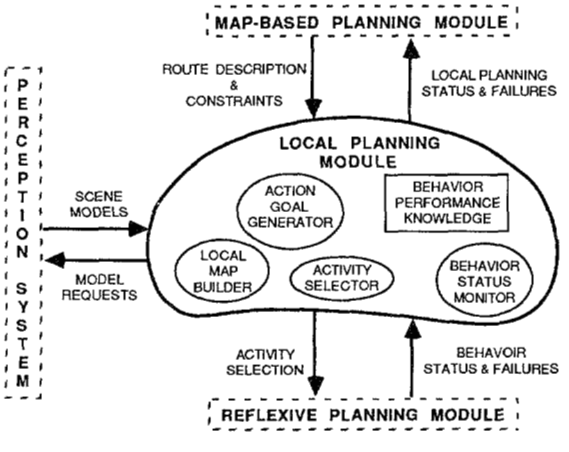

Figure 4: Functional elements of the local planning module.

2. Operating Knowledge. The key to selecting appropriate activation sets is having the ability to make effective use of detailed knowledge of activity capabilities and limitations. Accordingly, as reflexive behaviors are defined and mixed together to form activities, these activities will be given symbolic declarative descriptions which can be interpreted by the local planning module. These descriptions will contain information such as:

- goals that would be satisfied,
- goals that would be thwarted,
- appropriate contexts,
- inappropriate contexts,
- limitations, and
- typical failures.

Use of this knowledge in conjunction with other data assimilated within the local planning module should permit effective selection of different activities for different situations.

V. RESULTS

The reflexive planning module has been implemented along with several reflexive behaviors, and it has been demonstrated to provide meaningful levels of performance competence within both a software simulation environment and a small indoor vehicle testbed environment. These demonstrations are important because they provide a good indication of the ability of the architecture to provide useful real-time performance while sheltering higher levels of planning from concerns of immediacy.

A. The Software Testbed

For simulation purposes, all aspects of the software testbed environment are based on a planar approximation of the real world. Under this approximation, a vehicle is allowed to move in any direction on a plane so long as it does not collide with any obstacles placed on the surface of that plane. Although this simplification fails to capture some of the intricate planning problems related to maneuvering on steep slopes or bumpy terrain, many of the more basic planning problems such as obstacle avoidance and maneuvering through complex obstacle formations may be studied in an environment that allows efficient interaction.

Important aspects of vehicle motion are the vehicle shape and the means for vehicle control. The simulated vehicle is rectangularly shaped, making orientation an important planning issue. Control is maintained through the issuance of speed and rate of turn commands. With a speed and turn-rate provided, the vehicle will follow a fixed trajectory until otherwise commanded. By regularly updating the speed and turn rate commands, the planner can guide the vehicle in much the same way that a person guides a car through use of the steering wheel and accelerator pedal.

In order to place realistic time constraints on the planner, motion of the simulated vehicle is based on a real-time clock so that vehicle motion is not affected by processing time consumed by the planner. This way, the planner must consider its own reaction time when deciding how fast the vehicle should move. If the planner cannot react quickly enough, it must slow the vehicle down. Otherwise the vehicle could easily collide with an obstacle.

The sensors for the simulated vehicle include a compass, an odometer, and a simplified range scanner. In a simulated range scan, rays are traced from the current vehicle position out through the planar environment until either an obstacle is encountered or a maximum range limit is reached. A scan angle and scan depth recorded for each ray describes the result of any given scan. Errors and accuracy tolerances may be simulated for all sensors, to gain maximum realism from the simulation.

B. Simulation Results

The two simple activities described earlier, WANDER and SEEK-HEADING, have both been implemented for control of the simulated vehicle. Figure 5 illustrates a sample path executed by the SEEK-HEADING activity, showing its ability to deal neatly with simple cul-de-sacs as well as other obstacles. In this path, one can see areas where an obstacle was detected by the simulated scan, causing the vehicle to slow and turn away. After turning away from an obstacle and finding open passage, the vehicle attempts to resume its heading to the right. Upon entering the cul-de-sac, the vehicle cannot find open passage despite its attempts to turn away from the obstacle wall. With repeated failures to find open passage, the direction of turn is maintained until the vehicle is free from the obstacle. The primary role a higher level planner would have in this scenario would be to remember the cul-de-sac so that it could be avoided entirely in the future. Because it is composed of many of the same elements as SEEK-HEADING, the WANDER activity performs in much the same way, except that it does not attempt to orient itself toward a heading goal.

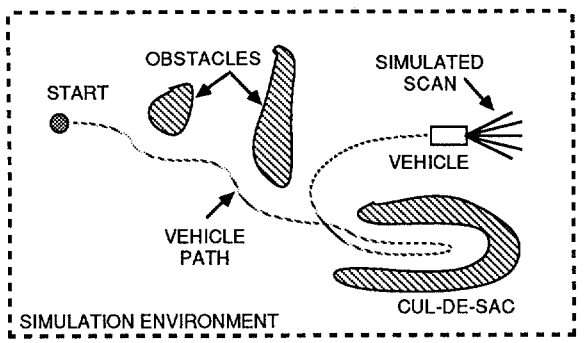

Figure 5: Sample vehicle path through the simulation environment.

C. The Vehicle Testbed

The vehicle testbed is composed of a mobile robot controlled remotely by a LISP machine via a 9600 baud RS-232 radio modem link. The vehicle itself consists of a tracked platform carrying a large storage battery for the propulsion system, a smaller battery for electronics, a velocity controlled interface for driving the two propulsion motors, optical encoder circuitry for measuring the movement of each track, a Polaroid time-of-flight sonar range finder mounted on a rotating platform driven by a stepper motor, four curb feeler type touch sensors, and an STD bus computer for interfacing the vehicle to the control computer. The LISP machine controls this vehicle by sending speed and turn-rate commands, and it acquires data by sending scan requests and requests for feeler and track movement data.

D. Demonstrated Vehicle Performance

Currently, the WANDER activity and an activity called PERSIST have both been implemented for the testbed vehicle. We are awaiting installation of a digital compass on the vehicle before implementing the SEEK-HEADING activity. The PERSIST activity performs the function of closely following the boundary of an obstacle. We expect to use this as a simple means for allowing the vehicle to find its way along the walls of a room until it finds the door.

VI. CONCLUSION

We have presented an architecture which allows the exploitation of a wide range of immediacy and assimilation alternatives for autonomous vehicle control. In selecting this architecture, we have considered the influence that our approach to system decomposition could have on our ability to incorporate a variety of different planning methods as the system evolves. We have found that our vertical decomposition with laterally decomposed layers not only provides the greatest degree of flexibility, but also permits the integration of methods requiring immediacy with those requiring assimilation. We have demonstrated through both software simulation and control of a small indoor vehicle, the potential of the reflexive control component of this architecture to provide a flexible repertoire of activities that may be used as primitive operators by higher levels of planning. The remaining challenges are to develop effective methods for the local planner to select among available activities, and to develop a set of reflexive behaviors suited specifically for the DARPA Autonomous Land Vehicle.

VII. REFERENCES

[1] R. A. Brooks, "A Layered Intelligent Control System for a Mobile Robot," *ISRR; Third International Symposium of Robotics Research*, Gouvieux, France, October 7-11, 1985.

[2] J. L. Crowley, "Navigation for an Intelligent Mobile Robot," *IEEE Journal of Robotics and Automation*, RA-1, pp. 31-41, March 1985.

[3] J. Lowry, and R. Douglass "Autonomous Road Following," *1986 IEEE International Conference on Robotics and Automation*, San Franscisco, California, April 7-10, 1986.

[4] S. Y. Harmon, "USMC Ground Surveillance Robot (GSR) : A Testbed for Autonomous Vehicle Research," *Proceedings of the Fourth University of Alabama Robotics Conference*, Huntsville, Alabama, April 24-26, 1984.

[5] D. M. Keirsey, J. S. B. Mitchell, D. W. Payton, and E. P. Preyss, "Multilevel Path Planning for Autonomous Vehicles," *SPIE Conference on Applications of Artificial Intelligence*, Arlington, Virginia, pp. 133-137, May 3-4, 1984.

[6] J. S. B. Mitchell, "An Autonomous Vehicle Navigation Algorithm," *SPIE Conference on Applications of Artificial Intelligence*, Arlington, Virginia, pp. 153-158, May 3-4, 1984.

[7] J. S. B. Mitchell, and D. M. Keirsey, "Planning Strategic Paths Through Variable Terrain Data," *SPIE Conference on Applications of Artificial Intelligence*, Arlington, Virginia, pp. 172-179, May 3-4, 1984.

[8] H. P. Moravec, "The Stanford Cart and the CMU Rover," *Proceedings of the IEEE*, 71, pp. 872-884, July 1983.

[9] N. J. Nilsson, "Shakey the Robot," *SRI AI Center Technical Note 323*, April 1984.

[10] R. Wallace, A. Stentz, C. Thorpe, H. Moravec, W. Whittaker, T. Kanade, "First Results in Robot Road-Following," *Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, California, pp. 1089-1095, August 18-23, 1985.