Behavior-Based Planning for Intelligent Autonomous Vehicles

Julio K. Rosenblatt*
University of Maryland Institute for Advanced Computer Studies
A.V. Williams Building, College Park, MD 20742, USA
julio@cs.umd.edu, http://www.cs.umd.edu/~julio

Abstract

An architecture is presented in which distributed decision-making processes cooperatively determine a mobile robot's path by sending votes to a centralized arbiter. One type of arbiter defined within this DAMN architecture performs *command fusion* to select that action which best satisfies the combined behavior votes, allowing multiple goals and constraints to be considered simultaneously. Another type of arbiter performs action selection via *utility fusion*. Within the utility fusion framework, behaviors indicate the utility of various possible world states and the arbiter determines the next action based on the maximization of expected utility. Thus, DAMN performs centralized arbitration of votes from distributed behaviors and in so doing provides coherent, rational, goal-directed behavior while preserving real-time responsiveness to its immediate physical environment

Keywords: behavior-based architectures, autonomous vehicles, distributed planning, navigation

1 ARCHITECTURAL ISSUES

In domains such as mobile robot navigation, the dominant characteristics which must be addressed by an intelligent agent are the incomplete and uncertain knowledge of its environment, uncertainty in the current state of a complex system, as well as uncertainty in the effects of the agent's own actions. In order to function effectively in unstructured, unknown, and dynamic environments, planning systems cannot generate a plan *a priori* that can be expected to perform reasonably in the face of such uncertainty, nor can they anticipate all contingencies that may arise. Planning systems must be reactive in the sense that their decisions must take into account current information and state at all times, proceeding in a data-driven manner, rather than attempting to impose unrealizable plans in a top-down fashion.

Some key issues to be considered in the design of a planning and control architecture are whether the architecture should be centralized or distributed, whether the reasoning should be reactive or deliberative, and whether control should be top-down or bottom-up. In addition, there is a fundamental choice to be made in the method by which information from multiple sources is combined, via sensor fusion or command arbitration. These issues, which are of course interrelated, should not be treated as dichotomies, but rather as continuous spectra to be considered as design trade-offs to be combined for the desired capabilities of the system. The question then becomes not "which one?" but rather "how much of each?" for a particular class of domains.

As with any complex system, tradeoffs must be made between the coherence, correctness, and relative straightforwardness of a centralized system one the one hand and the responsiveness, robustness, and flexibility of a distributed system on the other. Some form of layered architecture is highly desirable for the development and use of a complex, versatile robot control system.

Centralized mobile robot systems operate by gathering all available sensory data, creating a complete model of its static environment, planning an optimal series of actions within the context of that model, and then executing that plan [Durrant-Whyte, 1986; Moravec, 1990; Nilsson, 1984; Shafer *et al.*, 1986]. The robot would then gather more information and the process would repeat. This approach has the advantage of being able to combine evidence to overcome ambiguities and noise inherent in the sensing process [Moravec & Elfes, 1985], but has the disadvantage of creating a computationally expensive sensory bottleneck; all sensor data must be collected and integrated before it can be acted upon. A single monolithic world model is also more difficult to develop, maintain, and extend. In addition to introducing potentially harmful delays, a centralized architecture also leads to brittleness because the system may fail entirely if any single part of it is not functioning properly.

Another difficulty with sensor fusion is that information from disparate sources such as maps, sonar, and video, are generally not amenable to combination within a single representational framework that is suitable for planning such dissimilar tasks as following roads and avoiding obstacles. For example, ALVINN [Pomerleau, 1992] uses an artificial neural network to associate video images of roads with appropriate steering directions and has been one of the most

* *New address as of November 3, 1997:*
Dept. of Mechanical & Mechatronic Engineering
University of Sydney, NSW 2006, Australia

successful road following systems to date, yet it has been less successful than other systems such as Smarty [Langer *et al.*, 1994] which use range data for the purpose of obstacle avoidance. Thus, by requiring a single representation for all sensor and map data, a centralized architecture does not allow specialized modules to use other representations and algorithms best suited to the task at hand.

In contrast, in behavior-based architectures, the perceptual processing is distributed across multiple independent modules. Each action-producing module, or *behavior*, operating asynchronously and in parallel with other behaviors, is responsible for a particular aspect of vehicle control or for achieving some particular task, based on only that sensory data which is directly relevant to its particular decision-making needs. A behavior encapsulates the perception, planning and task execution capabilities necessary to achieve one specific aspect of robot control. Thus, in such an architecture, not only the sensing and planning, but the actual control of the robot itself is distributed across multiple independent modules. Therefore, in a behavior-based system, it is necessary to select among or combine the actions suggested by the various behaviors to produce an action that meets the needs of the overall system. By appropriately combining behavior commands through arbitration, a robot control system can respond to its environment without suffering the problems inherent in sensor fusion such as bottlenecks; however, command arbitration runs the risk of losing information valuable to the decision-making process. A careful balance must be struck between completeness and optimality on the one hand versus modularity and efficiency on the other.

Frameworks such as the Subsumption Architecture [Brooks, 1986] and Gapps [Rosenschein & Kaelbling, 1986] arbitrate among behaviors by explicitly or implicitly assigning priorities to each behavior; of all the behaviors issuing commands, the one with the highest priority is in control and the rest are ignored. However, one of the requirements for an intelligent planning and control system is that it be capable of satisfying multiple, possibly conflicting goals [Simon, 1967]. While priority-based schemes are effective when choosing among incompatible commands, they do not provide an adequate means for dealing with multiple goals that can and should be satisfied simultaneously [Rosenblatt & Thorpe, 1995).

Hierarchical architectures [Albus *et al.*, 1987] are another type of distributed system in which the modules are organized into multiple control levels that operate at varying granularities, levels of abstraction, and time scales. They are composed of multiple control levels, each of which have the same type of structure as a centralized system; however, each level operates in parallel at a different rate, so that the lowest levels are free to respond to immediate stimuli without having to wait for higher level reasoning processes. While this framework effectively bypasses the sequential bottlenecks of purely centralized systems, this recursive decomposition imposes a rigid structure which has been found in practice to be overly constraining; no structure has been found to be appropriate for all levels.

Hybrid architectures have also been proposed as a means to combine the complementary strengths and weaknesses of deliberative and reactive architectures [Payton, 1986). Hybrid architectures consist of layers, each composed of different reasoning elements operating within various paradigms. At the top level is a deliberative planner which assimilates all available information and creates long-term global plans to be used by the lower levels. The lowest level consists of a behavior-based reactive planner which responds in real-time to sensory stimuli, but when possible also takes into account the higher level considerations or constraints passed down to it from above. In most hybrid architectures, an intermediate level also appears between the high level symbolic reasoning of the deliberative planner and the low level numerical computations of the reactive planner operating at the actuator control level [Gat, 1992).

In hierarchical and hybrid architectures, the design and operation of the robot control system are expected to proceed in a top-down manner; each level controls the level beneath it and assumes that its commands will execute as anticipated. Since expectations are not always met, there is a need to monitor the progress of desired actions and to report failures as they occur [Simmons *et al.*, 1990]. In an unstructured, unknown, or dynamic environment, this approach introduces complexities and inefficiencies which could be avoided if higher level modules participated in the decision-making process without assuming that their commands will be strictly followed [Payton *et al.*, 1990]. In addition, only one module in a given level is activated by the level above and allowed to participate in the decision-making process, and a decision made at a higher level severely constrains the range of possible solutions which may be found by lower levels, often without sufficient information to warrant such a restriction. In contrast, DAMN combines reaction and deliberation within a single level; there are no layers of control.

2 THE DISTRIBUTED ARCHITECTURE FOR MOBILE NAVIGATION

Rather than imposing a top-down structure to achieve this desired symbiosis of deliberative and reactive elements, the Distributed Architecture for Mobile Navigation (DAMN) takes an approach where multiple behaviors concurrently share control of the robot. It consists of a group of distributed behaviors such as road following or obstacle avoidance which send votes to a centralized command arbiter, as shown in Figure 1.

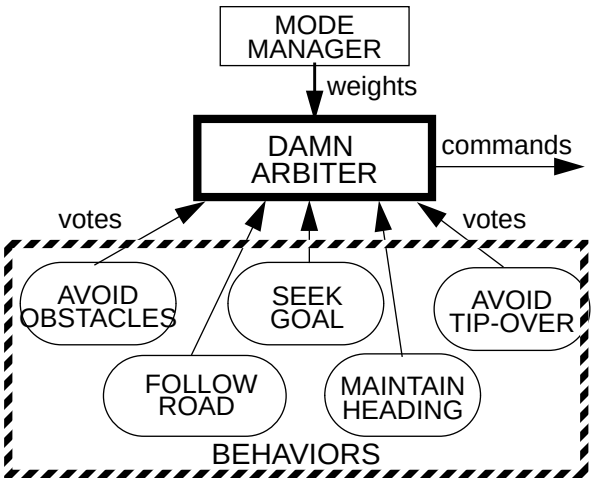

Figure 1: DAMN framework consists of centralized arbitration of votes from distributed behaviors

Each behavior is assigned a weight reflecting its relative priority in controlling the vehicle. A mode manager may also be used to vary these weights during the course of a mission based on knowledge of which behaviors would be most relevant and reliable in a given situation. The arbiter is then responsible for combining the behaviors' votes and generating actions which reflects their objectives and priorities; the appropriate commands are then sent to the vehicle controller. The behaviors are developed independently and function separately; each behavior functions without any explicit knowledge of or communication with any module other than the central arbiter, thus promoting and facilitating evolutionary system development. The distributed, asynchronous nature of the architecture provides real-time responsiveness to its immediate physical environment and allows multiple goals and constraints to be fulfilled simultaneously, while the centralized command arbitration provides a framework capable of producing coherent, rational, goal-directed behavior.

Unlike other behavior-based systems that use priorities to effect a traded control system, DAMN takes a shared control approach where several modules concurrently have some responsibility for control of the robot. So that multiple considerations may concurrently affect decision-making, DAMN uses a scheme where each behavior votes for or against each of a set of possible vehicle actions. An arbiter then performs *command fusion* to select the most appropriate action. Although all votes must pass through the command arbiter before an action is taken, the function provided by the arbiter is fairly simple and does not represent the centralized bottleneck of more traditional systems. While the Motor Schema framework [Arkin, 1989] also offers a means of fusing commands from multiple behaviors, it suffers from the well known problem of local minima in potential fields. Another, perhaps more serious problem, is that arbitration via vector addition can result in a command which is not satisfactory to any of the contributing behaviors. DAMN arbiters do not average commands, but rather select the command which has the most votes from the behaviors.

Higher-level planners are instantiated as behaviors and send votes to the arbiter just as any other behavior would. Thus, plans are not used in a top-down fashion but rather as a source of advice, so that the flexibility of the reactive level is preserved [Payton *et al.*, 1990]. The distinction made in DAMN is not in the level of abstraction of a given module, but rather whether its domain is represented and acted upon in a discrete or continuous manner; all continuous servo-like activity is instantiated as a voting behavior without regard for the time or space scale in which it operates; sequential activity and discrete mode changes are controlled by a meta-level mode manager which ensures that behaviors with mutually exclusive goals do not operate simultaneously.

Since both deliberative and reflexive modules are needed, DAMN is designed so that behaviors can issue votes at any rate; for example, one behavior may operate reflexively at 10 Hz, another may maintain some local information and operate at 1 Hz, while yet another module may plan optimal paths in a global map and issue votes at a rate of 0.1 Hz. The use of distributed shared control allows multiple levels of planning to be used in decision-making without the need for an hierarchical structure. However, higher-level reasoning modules may still exert meta-level control within DAMN by modifying the voting weights assigned to behaviors and thus controlling the degree to which each behavior may influence the system's decision-making process and thus the robot's actions.

3 DAMN ARBITERS

In order to preserve the respective advantages of centralized and distributed architectures and provide for effective shared control, sufficient information must be communicated from the behaviors to allow the arbiter to make intelligent decisions, but the arbiter must not be so complex as to become a bottleneck for the system. Various points along this trade-off spectrum have been explored within DAMN, using different types of arbiters and vote structures.

3.1 Turn Arbiter

In one DAMN arbitration scheme, each behavior votes for or against various alternatives in the actuator command space; for example, the turn arbiter receives votes for a fixed set of curvatures which represent the possible steering commands for vehicles with Ackerman steering, as shown in Figure 2. Each behavior generates a vote between -1 and +1 for every possible steering command, with negative votes being against and positive votes for a particular command option. The votes generated by each behavior are only recommendations to the arbiter.

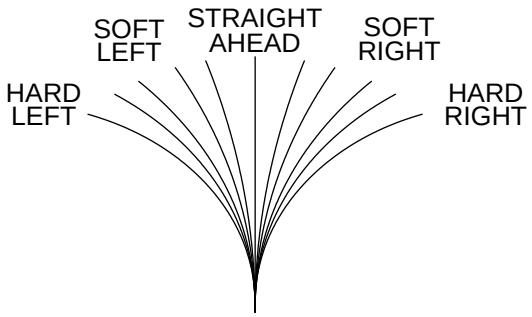

Figure 2: Curvature-based turn command space

The arbiter collects the new votes from each behavior that has sent them, and performs a normalized weighted sum to find the turn command with the maximum vote value. In order to avoid problems with discretization such as biasing and "bang-bang control" (i.e., alternating between discrete values in order to achieve an intermediate value), the arbiter performs sub-pixel interpolation. The arbitration process is illustrated in Figure 3, where: (a & b) the votes from behaviors are received, (c) a weighted sum of those votes is computed, nd (d), the summed votes are smoothed and interpolated to produce the resulting command sent to the vehicle controller. This is similar to defuzzification in Fuzzy Logic control systems [Lee, 1990; Kamada *et al.*, 1990); indeed an architecture has been implemented which recasts this type of DAMN arbitration into a Fuzzy Logic framework [Yen & Pfluger, 1992).

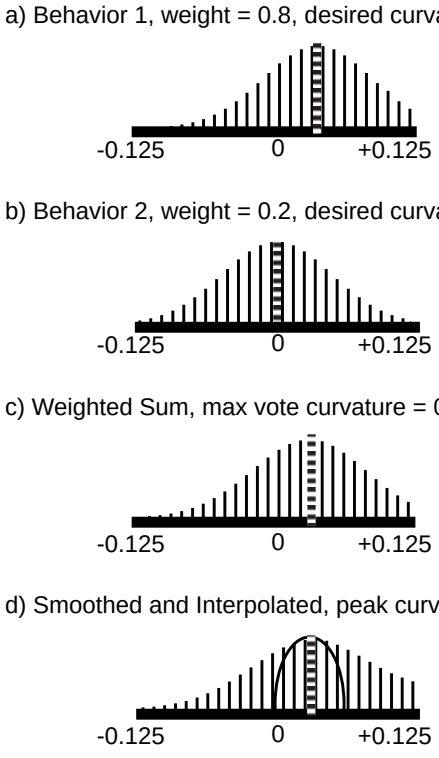

Figure 3: Command fusion

This arbitration scheme provides a means by which commands can be combined, unlike action selection schemes that choose a single behavior's command to be used in controlling the robot. However, the information supplied to the arbiter is somewhat minimal so that it is unable to take into consideration the dynamics of the plant being controlled, i.e., the vehicle's speed and turn radius; it is assumed that the vehicle is moving slowly enough that dynamics do not play an important role. In addition, it is assumed that behaviors will be able to update their votes at a sufficiently fast rate compared to vehicle speed and that those votes will be acted upon quickly enough by the system such that the arbiter receives a new set of votes from each behavior and acts on it before vehicle motion has rendered that behavior's votes obsolete or erroneous. Furthermore, it is assumed that votes are received often enough from all behaviors that synchronization of their votes is not a concern. These assumptions were reasonable for the slow vehicle speeds used in experimentation, and indeed the turn arbiter worked quite well; however, as the speed of image acquisition and processing techniques improves, these assumptions lose their validity.

3.2 Path Arbiter

In another DAMN arbitration scheme, behaviors do not vote for commands but instead express the *utility* of possible world states which may be represented within a map, and it is the responsibility of the arbiter to determine which states are actually attainable and how to go about achieving them. This type of arbiter is no longer performing command fusion, nor is it performing sensor fusion; it is combining utilities to perform *evidence fusion*. With this scheme, plant dynamics may be fully accounted for, and vote obsolescence only becomes an issue if the vehicle is moving faster than information can be collected and processed by the behavior, which is an unavoidable limitation of any control system. This new approach strikes a balance between the extremes of action selection and sensor fusion and has been found to yield many benefits.

For example, a map-based path arbiter has been implemented as a very different means of voting for and producing steering control. Behaviors communicating with the path arbiter vote on the desirability of various possible vehicle locations, and the arbiter maintains a local map of these votes, as indicated in Figure 4. Based on the vehicle's current state, the path arbiter evaluates the possible trajectories which may be followed, and selects that one for which the total utility is the greatest. This external location-based scheme is capable of maintaining a consistent interpretation of the votes received and correctly coordinating votes received at different times and from different locations, and updating them as the vehicle state changes, i.e., as it moves. Behaviors can function without knowledge of the system dynamics, thus increasing their reusability for other systems.

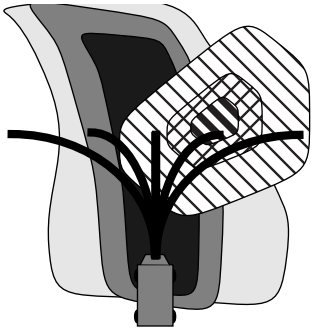

Figure 4: Map-based path arbiter

The voting scheme for this class of arbiter is cast within the framework of utility theory so that uncertainty within the system is explicitly represented and reasoned about within the decision-making processes. Each behavior votes for the subjective utility of the vehicle being in the various particular locations of concern to that behavior, e.g. obstacle locations or road locations. The behavior may also express any uncertainty associated with the perception process. The arbiter can then use utility theory to reason explicitly about the uncertainty in position and control of the vehicle and the *Maximum Expected Utility* (MEU) criterion can be applied to select the optimal action based on current information.

4 CONCLUSION

Because reactivity is essential for any real-time system, we must eschew the sensing and planning bottlenecks of centralized systems, but if we are to avoid sensor fusion, the system must combine command inputs to determine an appropriate course of action. However, priority-based arbitration only allows one module to affect control at any given time. Command fusion provides a mechanism for the concurrent satisfaction of multiple goals, and allows modules to be completely independent, thus allowing incremental, evolutionary system development.

The Distributed Architecture for Mobile Navigation is a planning and control architecture in which a collection of independently operating behaviors collectively determine a robot's actions. A command arbiter combines the behavior outputs and selects that action which best satisfies the prioritized goals of the system. The distributed, asynchronous nature of the architecture allows multiple goals and constraints to be fulfilled simultaneously, thus providing goal-oriented behavior without sacrificing real-time responsiveness. Unlike other behavior-based architectures, DAMN is designed so that behaviors provide both deliberative and reflexive capabilities; the use of distributed shared control allows multiple levels of planning to be used in decision-making without the need for an hierarchical structure.

DAMN has been used to combine various systems of differing capabilities on several mobile robots, and has also been used for active sensor control. Various subsystems developed at CMU and elsewhere have been integrated within this architecture, creating systems that perform road following, cross-country navigation, map-based route following, and teleoperation while avoiding obstacles and meeting mission objectives [Langer *et al.*, 1994].

REFERENCES

[Albus *et al.*, 1987] Albus, J., McCain, H. & Lumia, R. NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM), Tech. Note 1235, Gaithersburg, MD, 1987.

[Arkin, 1989] Arkin, R., (1989) *Motor Schema-Based Mobile Robot Navigation*, in International Journal of Robotics Research, Vol. 8(4), August 1989, pp. 92-112.

[Brooks, 1986] Brooks, R. (1986), *A Robust Layered Control System for a Mobile Robot*, IEEE Journal of Robotics and Automation vol. RA-2, no. 1, pp. 14-23, April 1986.

[Durrant-Whyte, 1986] Durrant-Whyte, H., Integration, Coordination, and Control of Multi-Sensor Robot Systems, Ph.D. dissertation, University of Pennsylvania, Philadelphia, PA, 1986.

[Gat, 1992] Gat. E., Integrating Planning and Reacting in a Heterogeneous Asynchronous Architecture for Controlling Real-World Robots, in proceedings of Eleventh AAAI, 1992.

[Kamada *et al.*, 1990] Kamada, H., Naoi, S., Goto, T. (1990), *A Compact Navigation System Using Image Processing and Fuzzy Control*, IEEE Southeastcon, New Orleans, April 1-4, 1990

[Langer *et al.*, 1994] Langer, D., Rosenblatt, J. &. Hebert, M. A Behavior-Based System For Off-Road Navigation. In *IEEE Journal of Robotics and Automation*, vol. 10, no. 6, pp. 776-782, December 1994.

[Lee, 1990] Lee, C. *Fuzzy Logic in Control Systems: Fuzzy Logic Controller -- Parts I & II*, IEEE Transactions on Systems, Man and Cybernetics, Vol. 20, No. 2, 1990.

[Moravec, 1990] Moravec, H., *The Stanford Cart and the CMU Rover*, in Cox, I. and Wilfong, G., Autonomous Robot Vehicles, Springer-Verlag, 1990.

[Moravec & Elfes, 1985] Moravec, H., and Elfes, A., *High Resolution Map From Wide-Angle Sonar*, in proceedings of the IEEE International Conference on Robotics and Automation, pp.116-121, 1985.

[Nilsson, 1984] N. Nilsson, *Shakey the Robot*, SRI Tech. Note 323, Menlo Park, Calif., 1984.

[Payton, 1986] Payton, D. (1986), *An Architecture for Reflexive Autonomous Vehicle Control*, ICRA, San Francisco, CA, April 7-10, 1986, pp. 1838-1845.

[Payton *et al.*, 1990] Payton, D., Rosenblatt, J. & Keirsey, D. (1990), *Plan Guided Reaction*. IEEE Transactions on Systems Man and Cybernetics, 20(6), pp. 1370-1382.

[Pomerleau, 1992] D. Pomerleau, *Neural Network Perception for Mobile Robot Guidance*, Ph.D. dissertation, Carnegie-Mellon Technical Report CMU-CS-92-115, 1992

[Rosenblatt and Thorpe, 1995] Rosenblatt, J. & Thorpe, C. Combining Multiple Goals in a Behavior-Based Architecture. In *Proceedings of 1995 International Conference on Intelligent Robots and Systems*, Pittsburgh, PA, August 7-9, 1995.

[Rosenschein & Kaelbling, 1986] Rosenschein, S. & Kaelbling, L. *The Synthesis of Digital Machines with Provable Epistemic Properties*, Theoretical Aspects of Reasoning about Knowledge, pp. 83-98, 1986

[Shafer *et al.*, 1986] Shafer, S., Stentz, A., Thorpe, C., An Architecture for Sensor Fusion in a Mobile Robot, Proceedings of the IEEE International Conference on Robotics and Automation, pp. 2002-2011, San Francisco, CA, April, 1986.

[Simon, 1967] Simon, H. Motivational and Emotional Controls of Cognition. Reprinted in Models of Thought, Yale University Press, 1979, pp. 29-38, 1967.

[Simmons *et al.*, 1990] Simmons, R., Lin, L.J., Fedor, C. *Autonomous Task Control for Mobile Robots*, in Proc. IEEE Symposium on Intelligent Control, Philadelphia, PA, September 1990.

[Yen & Pfluger, 1992] Yen, J., Pfluger, N. *A Fuzzy Logic Based Robot Navigation System*, AAAI Fall Symposium, Cambridge, MA, 1992.