

Behavior-Based Planning

Julio Rosenblatt

The Robotics Institute
Carnegie Mellon University
Pittsburgh PA 15213
jkr@cmu.edu

Introduction: 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.

Distributed Decision-Making: Layered Control

As with any complex system, tradeoffs must be made between the coherence, correctness, and relative straightforwardness of a centralized system on 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.

This research was partly sponsored by DARPA, under contract DAAE07-96-C-X075, "Technology Enhancements for Unmanned Ground Vehicles", monitored by TACOM, and by a Hughes Doctoral Research Fellowship.

Centralized architectures perform sensor fusion in order to construct a coherent world model which is then used for planning actions; this approach has the advantage of being able to produce optimal action under the assumptions of complete and perfect knowledge, 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.

In contrast, behavior-based architectures do not need to create a central world model; instead, the perceptual processing is distributed across multiple independent modules. Each action-producing module, or *behavior*, is responsible for a particular aspect of vehicle control or for achieving some particular task; it operates asynchronously and in parallel with other behaviors, sending its outputs to the arbiter at whatever rate is appropriate for that particular function. Each behavior requires only fragmentary knowledge of the world and receives exclusively that sensory data which is directly relevant to its particular decision-making needs, so there is no need to fuse all available data into a single model. Instead of performing sensor fusion, behavior-based systems must arbitrate command inputs to determine an appropriate course of action. Frameworks such as the Subsumption Architecture (Brooks, 1986) and Gapps (Rosenschein & Kaelbling, 1986) achieve this 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 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, thus providing varying tradeoffs between assimilation and immediacy (Payton, 1986), and thus between long-term correctness and completeness and short-term survival and relevance. Traditional hierarchical architectures use a homogeneous functional decomposition, with each level

constructed of the same modules, albeit at different levels of reasoning (Albus, McCain & Lumia). 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. In particular, each layer in an hierarchical architectures controls the layer beneath it and assumes that its commands will be executed as expected. 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, Lin & Fedor, 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.

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 modules concurrently share control of the robot by sending votes to be combined rather than commands to be selected and executed. DAMN is a behavior-based architecture similar in some regards to reactive systems such as the Subsumption Architecture (Brooks, 1986). However, 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, 1987) 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.

Figure 1 shows the organization of the DAMN architecture, in which individual behaviors such as road following or obstacle avoidance send votes to the command arbitration module; these inputs are combined and the resulting command is sent to the vehicle controller. 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.

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, Rosenblatt & Keirse, 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

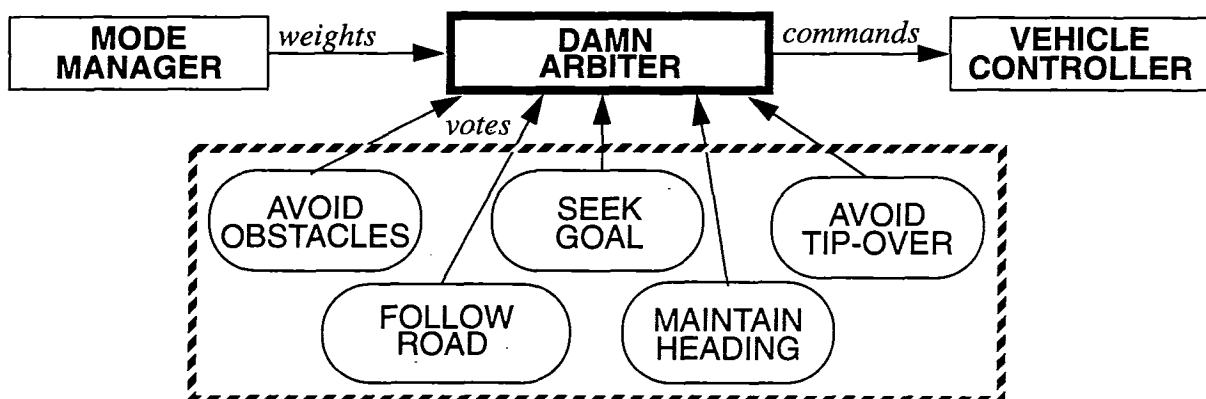


Figure 1: Overall structure of DAMN.

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.

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.

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.

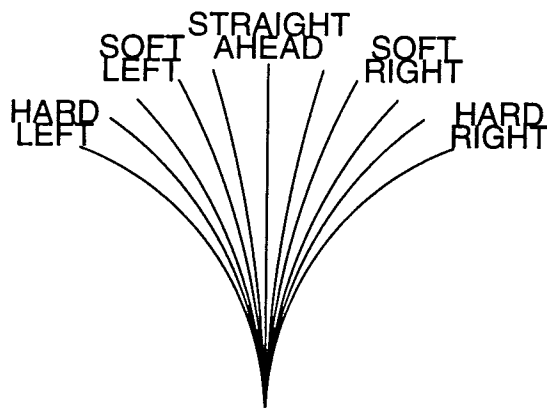
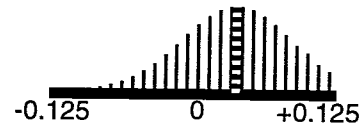


Figure 2: Curvature-based turn command space

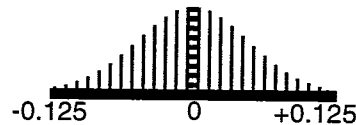
The arbiter collects the new votes from each behavior that has sent them, and performs a normalized weighted sum to find the turn command with the maximum vote value. In order to avoid problems with discretization such as biasing and “bang-bang control” (i.e., alternating between discrete

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, and (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, Naoi & Goto, 1990); indeed an architecture has been implemented which recasts this type of DAMN arbitration into a Fuzzy Logic framework (Yen and Pfluger, 1992).

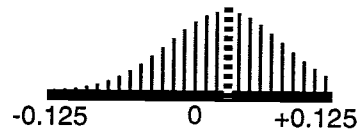
a) Behavior 1, weight = 0.8, desired curvature = 0.04



b) Behavior 2, weight = 0.2, desired curvature = 0.0



c) Weighted Sum, max vote curvature = 0.035



d) Smoothed & Interpolated, peak curvature=0.033



Figure 3: Command fusion process

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 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, that dynamics do not need to be considered. In addition, it is assumed that a new set of votes is received from a behavior before vehicle motion has rendered that behavior's previous set of votes obsolete or erroneous, and that votes are received often enough from all behavior that synchronization of their votes is not a concern.

Field of Regard Arbiter

In another DAMN arbitration scheme, each behavior votes for or against various alternatives in an abstract command space, i.e., they vote for the desired effect of the mechanism being controlled rather than the direct control of the mechanism's actuators. The arbiter effectively synchronizes the votes by maintaining a consistent command space in which those votes are initially represented, so that votes are counted correctly even after significant vehicle motion by re-mapping them into the current actuator frame of reference, and obsolete votes are not counted at all. However, once a behavior's votes have become obsolete, that behavior has no effect on the decision-making process until it issues a new set of votes.

For example, a field of regard arbiter and its associated behaviors have been implemented and used for the control of a pair of stereo cameras on a pan/tilt platform. Field of regard refers to the camera field of view mapped on to the ground plane. Behaviors vote for different possible field of regard polygons, as shown in Figure 4, based on considerations such as not looking in the direction of known obstacles (since travelling in that direction is impossible), looking toward the goal, and looking at a region contiguous to already mapped areas.

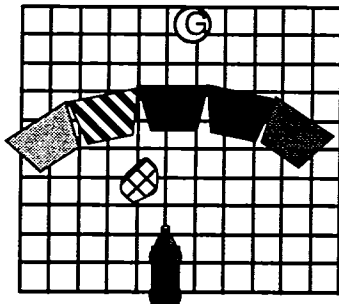


Figure 4: Field of regard voting

The arbiter receives the votes for locations at which to point a steerable camera and maintains a local map of these votes. and transforms them as the vehicle moves. At each iteration, these votes are mapped into a pan-tilt space and arbitration takes place within the actuator space. As the vehicle moves, the arbiter updates its commands so that the camera continues to point at the desired location, thus accounting for vehicle movement

Path Arbiter

In a third 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 5. 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.

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.

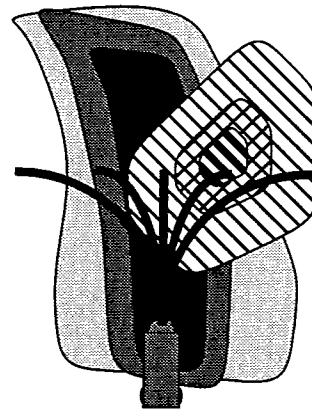


Figure 5: Map-based path arbiter

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.

In one DAMN arbitration scheme, behaviors vote for and against each of a set of candidate commands, either directly in actuator space. This method has the benefit of being very simple and straightforward, so that many different types of modules may easily be incorporated within the architecture. However, this simplicity comes at the cost of some assumptions that must hold if the system is to operate correctly; vehicle dynamics and system latencies must be negligible, and behaviors must be capable of updating their votes before the previous votes become incorrect and before synchronization becomes a problem. Uncertainty is dealt with in an *ad hoc* manner using techniques similar to those used in fuzzy logic systems.

In another arbitration scheme, behaviors vote for desired effects rather than directly for actions. This voting space is not time-dependent, so that an arbiter using such a representation is capable of effectively synchronizing and maintaining a consistent interpretation of the votes received from asynchronous behaviors, thus providing coherent reasoning in a distributed system

Another means of action selection in DAMN is to perform map-based utility arbitration. Instead of voting for actions, behaviors indicate the utility of various possible world states. The arbiter can then use models of the robot and of the environment to determine which states are actually attainable, and to provide greater accuracy and stability of control. Decision-making based on the maximization of expected utility provides a unified conceptual framework for defining the semantics of votes

and for reasoning about uncertainty.

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, Rosenblatt & Hebert, 1994).

References

- Albus, J., McCain, H. & Lumia, R. (1987), NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM), Tech. Note 1235, Gaithersburg, MD.
- Arkin, R., (1989) *Motor Schema-Based Mobile Robot Navigation*, in International Journal of Robotics Research, Vol. 8(4), August 1989, pp. 92-112.
- 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.
- 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, D., Rosenblatt, J. & Hebert, M. (1994), *A Behavior-Based System For Off-Road Navigation*, IEEE Journal of Robotics and Automation, vol. 10, no. 6, pp. 776-782, December 1994.
- Lee, C. (1990), *Fuzzy Logic in Control Systems: Fuzzy Logic Controller -- Parts I & II*, IEEE Transactions on Systems, Man and Cybernetics, Vol. 20, No. 2.
- Payton, D. (1986), *An Architecture for Reflexive Autonomous Vehicle Control*, ICRA, San Francisco, CA, April 7-10, 1986, pp. 1838-1845.
- Payton, D., Rosenblatt, J. & Keirse, D. (1990), *Plan Guided Reaction*. IEEE Transactions on Systems Man and Cybernetics, 20(6), pp. 1370-1382.
- Rosenblatt, J. & Thorpe, C. (1995), *Combining Multiple Goals in a Behavior-Based Architecture*, IROS, Pittsburgh, PA, August 7-9, 1995.
- Rosenschein, S. & Kaelbling, L. (1986), *The Synthesis of Digital Machines with Provable Epistemic Properties*, Theoretical Aspects of Reasoning about Knowledge, pp. 83-98.
- Simon, H. (1967), *Motivational and Emotional Controls of Cognition*. Reprinted in *Models of Thought*, Yale University Press, 1979, pp. 29-38.
- Simmons, R., Lin, L.J., Fedor, C. (1990) *Autonomous Task Control for Mobile Robots*, in Proc. IEEE Symposium on Intelligent Control, Philadelphia, PA, September 1990.
- Yen, J., Pfluger, N., (1992), *A Fuzzy Logic Based Robot Navigation System*, AAAI Fall Symposium, 1992.