Utility Fusion: Map-Based Planning in a Behavior-Based System

Julio K. Rosenblatt
Australian Centre for Field Robotics
Dept. of Mechanical & Mechatronic Engineering
University of Sydney, NSW 2006, Australia
julio@mech.eng.usyd.edu.au
http://mecharea.mech.eng.usyd.edu.au/people/julio

Abstract

A new means of action selection via *utility fusion* is introduced here as an alternative to both the bottlenecks of centralized systems and the incoherence of distributed systems. Within the utility fusion framework, distributed, independent, asynchronous behaviors indicate the utility of various possible world states and the uncertainty associated with them. A centralized arbiter then combines these utilities and probabilities to determine the next action based on the maximization of expected utility. The construction of a utility map allows the kinematic constraints of the system being controlled to be modeled and compensated for; experimental results verify that the resulting system provides greater stability.

1 INTRODUCTION

In unstructured, unknown, and dynamic environments, such as those encountered by outdoor mobile robots, an intelligent agent must adequately address the issues of incomplete and inaccurate knowledge; it must be able to handle uncertainty in its sensed data, in any maps which may exist, in the current state of the robot itself, as well as in the effects of the agent's actions. In order to function effectively in such 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.

Nonetheless, deliberative planning and reactive control are equally important for mobile robot navigation; when used appropriately, each complements the other and compensates for the other's deficiencies. Centralized architectures provide the ability to coordinate, in a coherent fashion, multiple goals and constraints within a complex environment, while decentralized architectures offer the advantages of reactivity, flexibility, and robustness. However, sensor fusion creates a bottleneck, and command arbitration runs the risk of losing information valuable to the decision-making process; therefore a careful balance must be struck between completeness and optimality on the one hand versus modularity and efficiency on the other. In addition, when dealing with a physical system such as a mobile robot, it is important to consider aspects of control such as stability and the limitations and constraints of the physical plant, for example the non-holonomic constraints of a vehicle.

In order to achieve this desired symbiosis of deliberative and reactive elements, the Distributed Architecture for Mobile Navigation (DAMN) consists of a group of distributed behaviors communicating with a centralized command arbiter, as shown in Figure 1. This information may take various forms, such as votes for and against possible actions or effects (as described in [Rosenblatt, 1997a]). 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 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.

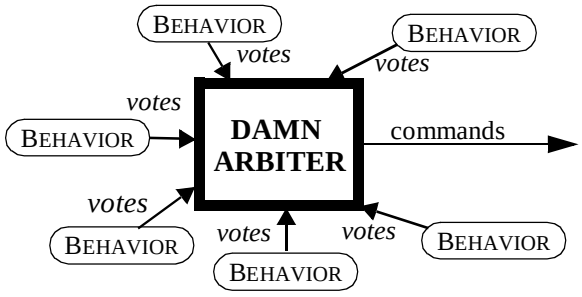

Figure 1: DAMN framework consists of centralized arbitration of votes from distributed behaviors.

Within the framework of DAMN, a new means of action selection is introduced here as alternative to both the sensor fusion and command fusion approaches. Information concerning the desirability and probability of possible world

In *Field and Service Robotics*, Springer-Verlag, 1998.

states is obtained from multiple independent sources and combined via *utility fusion*. The determination of what the next action taken should be is then based on this combined evidence. In this paradigm, behaviors do not select or express preferences for actions but instead determine the utility of possible world states. It is then the responsibility of a central arbiter to determine which states are actually attainable and how to go about achieving them.

2 BACKGROUND

Centralized mobile robot systems operate by gathering all available sensory data, creating a complete model of its environment, planning a series of actions within the context of that model, and then executing that plan [Moravec, 1990; Nilsson, 1984; Shafer *et al.*, 1986]. The robot then gathers more information and the process repeats. This approach has the advantage of being able to combine evidence to overcome ambiguities and noise inherent in the sensing process [Durrant-Whyte, 1986], 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 fails entirely if any single part of it is not functioning properly, particularly when the real world deviates significantly from the models employed.

In architectures which employ priority-based arbitration such as the Subsumption Architecture [Brooks, 1986] and GAPPS [Rosenschein and Kaelbling, 1986], action selection is achieved by assigning priorities to each behavior; the active behavior with the highest priority is in control and the rest are ignored. This allows for quick responses to new situations and stimuli, although, by definition, prioritization only allows one module to affect control at any given time. While this is an effective scheme for choosing among incompatible commands, it does not provide an adequate means for dealing with multiple goals that can and should be satisfied simultaneously. A compromise between behaviors cannot be achieved in such an all-or-nothing scenario; whenever one behavior's output is overridden by another, the information and knowledge represented by that behavior is completely lost to the system.

Architectures that perform command fusion combine the commands from individual behaviors so that decisions may be made based on multiple considerations while preserving the modularity and reactivity of distributed systems. 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.

Motor schemas [Arkin, 1989] provide a general framework for command fusion, but because they are implemented as potential fields [Khatib, 1990] they are subject to drawbacks such as local minima, and the averaged command that is the result of vector addition can produce an action that is not satisfactory to any of the contributing behaviors; for example, a robot cannot pass through closely spaced obstacles such those surrounding a doorway, and there also exist conditions under which potential fields suffer from oscillations and instability [Borenstein and Koren, 1991]. The root of these limitations is that, like priority-based architectures, each behavior simply outputs a single command, which is insufficient for effective command fusion.

Many control systems have been constructed using fuzzy logic [Lee, 1990], including a fuzzy control system developed for mobile robot navigation [Kamada *et al.*, 1990]. A previous type of arbiter implemented within DAMN operates in a similar manner and in fact has been recast into a fuzzy logic framework and used for control of a mobile robot [Yen and Pfluger, 1992]; however, the DAMN "actuation-space" arbiter is somewhat more general in that it allows for an arbitrary distribution of votes rather than being restricted to take on the shape of a particular function or rule output [Rosenblatt and Thorpe, 1995]. As we shall see, this system also suffered from some of the limitations shared by all command fusion systems. In order to preserve the respective advantages of centralized and distributed architectures, 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.

2.1 Limitations of Command Fusion

System Dynamics

Whether summing vectors in potential fields, combining rules by fuzzy logic, or combining votes in the DAMN actuation-space approach, the arbiter is fusing commands proposed by the behaviors which are not physically realizable; for example, a vehicle turn command may require a change in the commanded curvature which exceeds the steering wheel actuator's torque capabilities. Command fusion methods do not in general account for vehicle dynamics and kinematic constraints; thus they produce commands which are not executable by the system being controlled. Another important difference between the commanded and actual vehicle trajectories is due to the system delays arising from latencies in data acquisition, data processing, intermodule communications, and actuator response. Given the continuous motion of the vehicle, these delays imply that by the time the command is being executed the vehicle is no longer in the state at which the behavior commands were generated. Stable control requires that the system anticipate these latencies and allow the behaviors to consider actions that are kinematically achievable and which originate from the point where the vehicle will actually be when the command is executed [Kelly, 1995].

Synchronization

Synchronization allows reasoning to be coordinated and therefore coherent, but reduces the throughput of the system as modules must wait for a signal from each other in order to remain synchronized. Allowing the modules in a distributed architecture to operate asynchronously, each at the greatest rate of which they are capable, maximizes throughput and therefore reactivity. However, if there is no synchronization between behaviors, then their votes are produced based on different system states, so that the semantics of combining the two sets of votes is ill-defined and may yield unpredictable results.

Uncertainty

Domains such as mobile robot navigation necessarily contain a great deal of uncertainty. There exists uncertainty in the sensing of internal state such as vehicle position, uncertainty in perception such the location or shape of an object, and uncertainty in the effects of actions, e.g., due to slippage. These uncertainties are accounted for in an *ad hoc* manner in behavior-based systems, for example by "growing" the size of observed obstacles by some fixed amount or by "fuzzifying" the inputs to a system (including the DAMN actuation arbiter) and using fuzzy reasoning to determine an approximately appropriate output [Kamada *et al.*, 1990].

3 UTILITY FUSION

A new means of distributed action selection via utility fusion is introduced as an alternative to priority-based and command-fusion arbitration schemes. Instead of voting for actions, behaviors indicate the utility of various possible world states. The arbiter combines these utilities and determines the next action based on the maximization of expected utility, thus providing a unified conceptual framework for defining vote semantics and for dealing with uncertainty. As illustrated in Figure 2, behaviors process sensory input to determine the utilities of possible states, which the arbiter collects into a utility-space map. The arbiter then evaluates candidate actions within this map, using not the current state of the system but the predicted state the system will be in when the action is actually performed. The action which maximizes expected utility is then chosen and sent as a command to the controller.

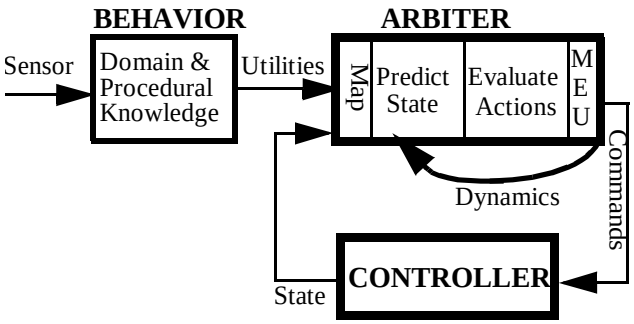

Figure 2: DAMN Arbiter evaluates predicted candidate actions using utility map information.

Utility fusion does not to create a world model as sensor fusion systems do. The information combined and stored by the utility fusion arbiter does not represent sensed features of the world, as in certainty grids [Moravec and Elfes, 1985], but rather the desirability of being in a particular state according to some criterion defined by the behavior. The processing of sensory data is still distributed among behaviors, so the bottlenecks and brittleness associated with sensor fusion are avoided.

Unlike command arbitration or command fusion systems, the utility fusion arbiter does not simply select among or combine actions proposed by behaviors. Instead, the arbiter is provided with much richer evaluation information from behaviors, thus allowing for intelligent decision-making. The arbiter accumulates utility and probability evaluations from the behaviors and bases its decision-making on the combined evidence, so that the limitations of command fusion systems may be overcome.

For example, a utility map-based path arbiter for steering control has been developed. 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. Figure 3 shows polygons of positive utility that a road-following behavior has sent based on detected road location, with the greatest value being the polygon closest to the centre of the detected road, and polygons of negative utility that an obstacle avoidance behavior has sent, with the greatest value being the polygon closest to the centre of the detected obstacle. Based on the vehicle's current state, the path arbiter evaluates the possible trajectories which may be followed, shown in the figure as arcs emanating from the vehicle. The expected utilities are summed along each arc, and the arbiter selects that one for which the total is the greatest.

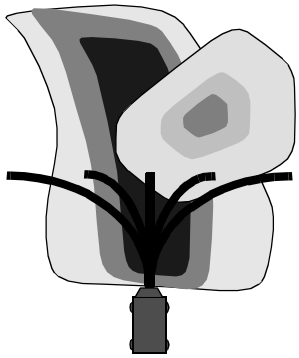

Figure 3: Map-based path arbiter voting. Darker polygons reflect higher vote utility values; striped polygons indicate negative utilities. Arcs indicate trajectories evaluated by the arbiter.

3.1 Advantages of Utility Fusion

System Dynamics

One advantage of map-based utility fusion over command fusion is that the dynamics of the system being controlled can be fully modeled and accounted for by the central reasoning process, providing greater control accuracy and stability. For example, because the arbiter is evaluating candidate actions rather than the behaviors, it can use knowledge of its own processing latencies as well as delays inherent in the system and compensate for them via predictive control. Using current vehicle state, a history of recently issued commands, and knowledge of the effects of those commands, the arbiter can determine approximately where the vehicle will be when the next command is actually executed and assess feasible trajectories originating from that position, as indicated by the lighter vehicle outline in Figure 4.

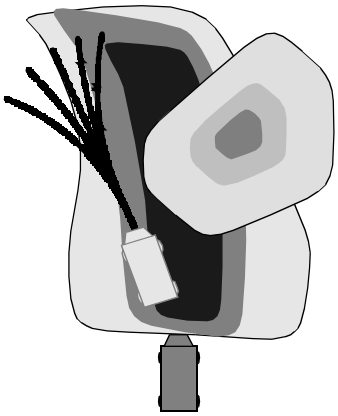

Figure 4: Map-based path arbiter voting with dynamics. Trajectories evaluated by the arbiter are clothoids emanating from the predicted vehicle position, indicated by the lighter vehicle.

Non-holonomic and kinematic constraints can also be modeled and accounted for by the arbiter. For example, the steering mechanism of the vehicle imposes a constraint of continuous curvature along the path. A clothoid is a curve whose curvature varies linearly with path length: $\kappa = k(s) + \kappa_0$, where k specifies the *sharpness*, or rate of change of curvature, and s is the total distance travelled [Kanayama and Miyake, 1985]. Because it has sufficient state information, the map-based utility arbiter can evaluate clothoids rather than simple arcs, as suggested in the diagram above, thus evaluating those actions which may be followed faithfully by the vehicle.

If behaviors were to be able to account for system dynamics, they would require a great amount of state information, both from the vehicle and from the command fusion arbiter. In addition, each behavior would need to have a vehicle model and apply it to the state information, representing a considerable duplication of effort. Such a system would be difficult to develop and maintain, with the possibility of introducing inconsistencies in the various models. However, because the utility arbiter is evaluating the candidate actions, behaviors need not know which actions the system is capable of; a behavior can express the utility of a desired world state independently of which actions would need to be taken to achieve it. A behavior only contains the domain and procedural knowledge needed for evaluating possible world states in the context of the task for which it is responsible. This provides greater modularity and interchangeability of behaviors; for example, a behavior developed for a vehicle with Ackerman steering could be reused as is for a system to control an omnidirectional robot.

As described in Section 4, experiments were conducted comparing the map-based utility arbiter to the previously implemented actuation-space turn arbiter. At slow vehicle speeds, both arbiters were able to successfully achieve their mission, but at higher vehicle speeds the effects of system latency and dynamics became very apparent, and the path arbiter with predictive control performed much better than the turn arbiter under those conditions [Rosenblatt, 1997b].

Synchronization

A map-based utility arbiter also solves the problem of unsynchronized behaviors because the information received from them is not time dependent. Command fusion involves combining behavior outputs which are only valid for a brief interval, so that their semantics are ill-defined unless executed immediately, but the utility arbiter receives votes for external world states whose meaning is well-defined independent of the current vehicle state. The use of a map allows synchronization of the votes to occur within the arbiter without imposing timing constraints on the behaviors, which would reduce system responsiveness. The external location-based scheme used in the map-based path arbiter is capable of maintaining a consistent interpretation of the votes received and correctly coordinating votes received at different times and from different locations. The trajectory evaluation process is repeated as the vehicle moves, so that action selection is based on the most recent information available, without any new utility information from the behaviors being immediately necessary.

Uncertainty

Utility theory provides a unified conceptual framework for defining votes and weights and dealing with uncertainty. Because we are attempting to decide which among a set of possible actions to take, it is natural to make judgments on the usefulness of each action based on its consequences. If we assign a utility measure *U(c)* for each possible consequence of an action *a*, then the *expected utility U(a)* is:

$$U(a) = \sum_c U(c) \cdot P(c|a, e)$$

where *P(c|a,e)* is the probability that consequence *c* will occur, given that we have observed evidence *e* and taken action *a* [Pearl, 1988]. Thus, if we can define these utilities and probabilities, we can then apply the *Maximum Expected Utility* criterion to select the optimal action based on our current information.

By casting the voting scheme for this class of arbiter within the framework of utility theory, uncertainty within the system is explicitly represented and reasoned about within the decision-making processes. Utility theory teases apart the value of the consequence of an action from the probability that the consequence will occur and provides a Bayesian framework for reasoning about uncertainty [Berger, 1985]. 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 also express any uncertainty associated with the perception process as covariances in a multi-dimensional normal distribution. The arbiter can then use utility theory to reason explicitly about the uncertainty in position and control of the vehicle and apply the Maximum Expected Utility criterion to select the optimal action based on current information. By explicitly representing and reasoning about uncertainty within the decision-making processes, a system can be created whose effects are well-defined and well-behaved.

4 EXPERIMENTAL RESULTS

In this section we present some results from experiments conducted on the Navlab II HMMWV Vehicle at Carnegie Mellon University, and in simulation, demonstrating the benefits of utility fusion in compensating for vehicle dynamics. Figure 5 shows an example of the environment in which the vehicle experiments took place; the terrain in the "slag heap" test area included hills, rocks, and ditches.

Figure 5: Experimental area.

4.1 Performance Metrics

Mean Obstacle Proximity

An important metric for a vehicle path is the average distance to obstacles along that path. The distance to that obstacle which is closest to the vehicle at any given moment provides a measure of safety clearance; when inverted, it provides a measure of proximity to obstacles which is to be minimized. The square of distance is used to reflect the increasing relative importance of obstacle proximity when the vehicle is closer to an obstacle. Thus, the mean obstacle proximity metric for a path is defined by the inverse square of the distance l_o to the closest obstacle, integrated along the path and normalized by the total number of path points n:

$$l_o = min(\forall o\left(\sqrt{(x_v - x_o)^2 + (y_v - y_o)^2}\right))$$

$$\text{mean obstacle proximity} = \frac{\int_0^l \left(\frac{1}{l_o}\right)^2 ds}{n}$$

Lower mean obstacle proximity means that the vehicle was on the average further away from the nearest obstacle, and therefore the path was safer.

Roughness

Roughness is defined by the square of the change in vehicle curvature κ with respect to time, integrated along the path and normalized by the total time t:

$$roughness = \frac{\int_0^l \left(\frac{d\kappa}{dt}\right)^2 ds}{t}$$

A lower roughness measure means that curvature changed less over the course of the path, and therefore that the vehicle path was smoother.

4.2 Path Arbiter: Vehicle Run Results

For the following vehicle experiments, two behaviors sent utilities to the path arbiter, *OBSTACLE AVOIDANCE* and *FOLLOW PATH*. As input to the first behavior, an ERIM laser range finder generated a polar map of sensed depth every 500ms. This range image was processed to create a Cartesian depth map that was then used to determine the presence and location of obstacles [Langer *et al.*, 1994]. For each obstacle detected, the *OBSTACLE AVOIDANCE* utility behavior sent to the arbiter a large negative utility associated with the obstacle *per se* to avoid collision and another negative utility with a smaller value was also assigned to the obstacle location to reflect the problems associated with getting too close to an obstacle, i.e., constrained mobility, occlusion of unknown areas, etc. At each iteration, the behavior sent utilities for all obstacles within the map, overriding ones previously sent to the arbiter so that the most recent information was always used. The entire process of processing the image, identifying obstacles, and reporting utilities to the arbiter took approximately 250ms to complete.

The *FOLLOW PATH* behavior associated a positive utility with each of the subgoals to be reached by the vehicle, and a line utility between subgoals was also defined so that a corridor was effectively created between consecutive goals. The line utility was given a higher utility than the goal point so that the vehicle would be drawn back to the corridor when it strayed from it, rather than heading directly towards the subgoal, e.g. after avoiding an obstacle. This behavior sent all utilities to the arbiter at once, and each goal attracted the vehicle in turn as it got closer; if a goal was unreachable or inadvertently bypassed, then the utilities defined by the next corridor attracted the vehicle to the next goal

The path arbiter combined the utilities from these behaviors into its vehicle-centered utility map, evaluated candidate trajectories within that map, and issued the steering command that maximized the expected utility. This process took 200ms on average, depending upon the number and type of utilities within the map.

Path Arbiter without Predictive Control

This experiment was run with the predictive control capability of the path arbiter turned off so that its effect could be observed and compared to the subsequent experiment conducted using predictive control. As can be seen in Figure 6, the vehicle oscillated quite a bit, yielding a roughness measure which we will see is high compared to subsequent runs using predictive control.

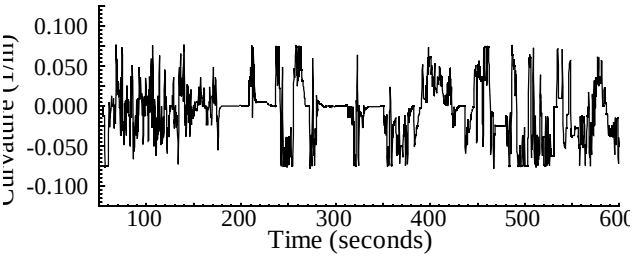

Figure 6: Vehicle curvature along vehicle run, without prediction.

The metrics for this run are:

- Mean obstacle proximity = 0.41487
- Path roughness = 0.00432

Figure 7a shows the path taken by the vehicle as it wound its way between the dense obstacle field indicated by the cross marks. The total length of the path was almost 400 meters, and the average vehicle speed was 0.7 meters/second. Figure 7b and Figure 7c show close-up snapshots

of the vehicle in on-road and off-road portions of the path, respectively, as indicated by the arrows.

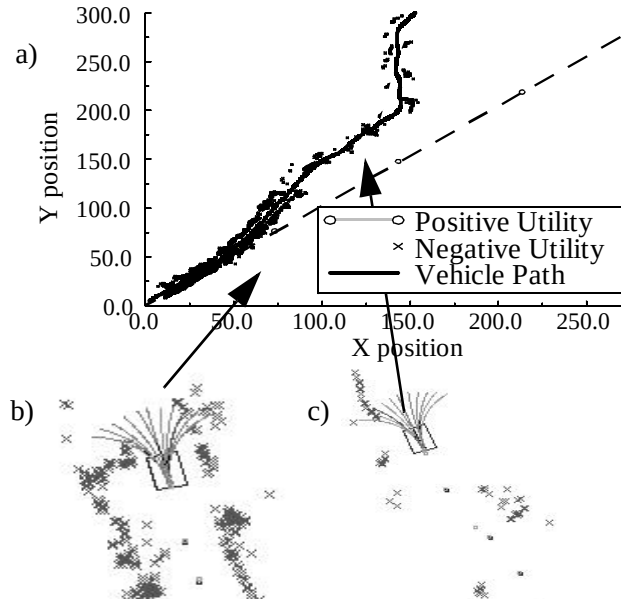

Figure 7: Vehicle run using path arbiter without prediction: a) trace of full path, b) close-up of on-road portion of path, c) close-up of off-road portion of path

Path Arbiter with Predictive Control

Further experiments were conducted with the path arbiter, this time with the predictive control capability in use. The previous experiment could not be duplicated due to several problems with the testbed vehicle and sensors. The laser range finder was replaced with a pair of stereo cameras, and depth map construction was sparser and more time consuming. In addition, the runs were conducted without a goal-based behavior in operation. In spite of these complications, significantly better results were obtained by using predictive control. Two separate short runs were made in the slag heap, as shown in Figure 8. The first run was 55 meters in length and the average speed was 0.9 m/s. The second path was approximately 150 meters long and the average speed was 0.8 m/s.

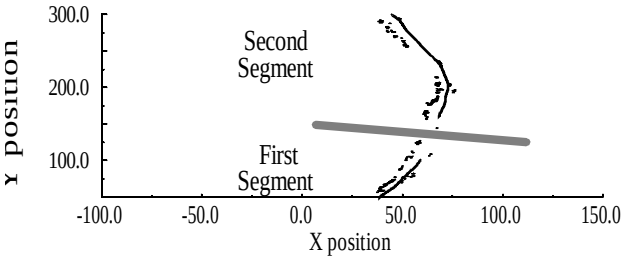

Figure 8: Path arbiter vehicle runs with predictive control.

The path metrics for the first segment were:

- Mean obstacle proximity = 0.15770
- Path roughness = 0.00007

and for the second segment:

- Mean obstacle proximity = 0.14090
- Path roughness = 0.00002

The curvature profile for these runs is shown in Figure 9. It can be seen to be substantially smoother than the curvature generated without predictive control shown in Figure 6. This is borne out by the very low values of the roughness metric for these runs, achieved while also maintaining a low obstacle proximity value.

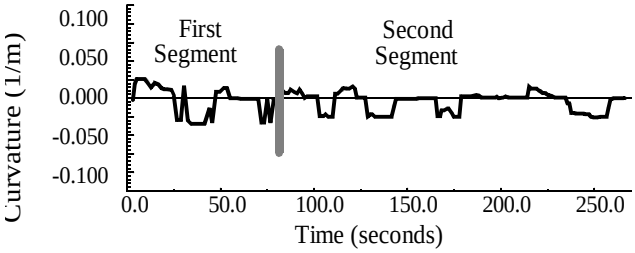

Figure 9: Vehicle curvature for path arbiter with prediction.

4.3 Simulation Experiments

A vehicle simulator was used for experiments where conditions could be carefully controlled and higher speeds could be used without risk of damage. The path arbiter with and without predictive control, as well as the turn arbiter used in previous systems [Rosenblatt, 1997a; Langer et al., 1994], were compared at various vehicle speeds and system latencies. For experiments at slow speeds, all arbiters were able to successfully achieve their mission, both in actual vehicle experiments and in simulation. However, the effects of system latency and dynamics became very apparent at higher vehicle speeds, and the path arbiter with predictive control performed much better than the path arbiter without predictive control or the turn arbiter under those conditions.

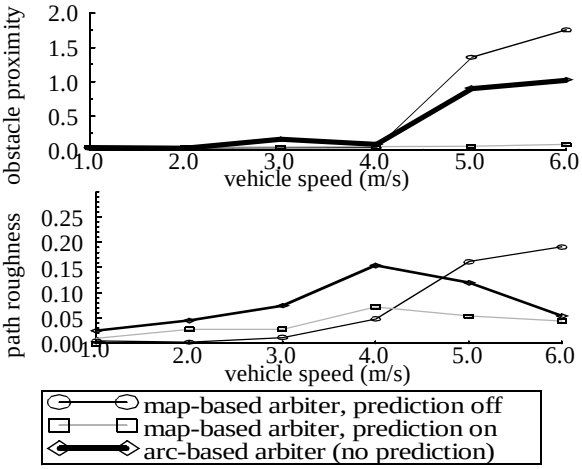

Figure 10: Path metrics as a function of speed: a) mean obstacle proximity, and b) path roughness.

The graphs of mean obstacle proximity as a function of speed in Figure 10a and of path roughness vs. speed in Figure 10b show that the turn arbiter does very badly at higher speeds; these runs are shown in Figure 11a. The graphs also show that, at higher speeds, the path arbiter without predictive control performed even worse than the turn arbiter, possibly due to the path arbiter's greater complexity; these runs are shown in Figure 11b. However, when the path arbiter made use of its predictive control capabilities, it was still able to go through this narrow

corridor and reach the goal, in spite of the fact that a delay of 2 seconds at a speed of 6 meters/second meant that the vehicle travelled 12 meters between the time that a command was issued and the time that it would actually be executed. These successful path traces are shown in Figure 11c, along with the trace of the position of the vehicle as predicted by the arbiter, which coincided well with the actual path taken.

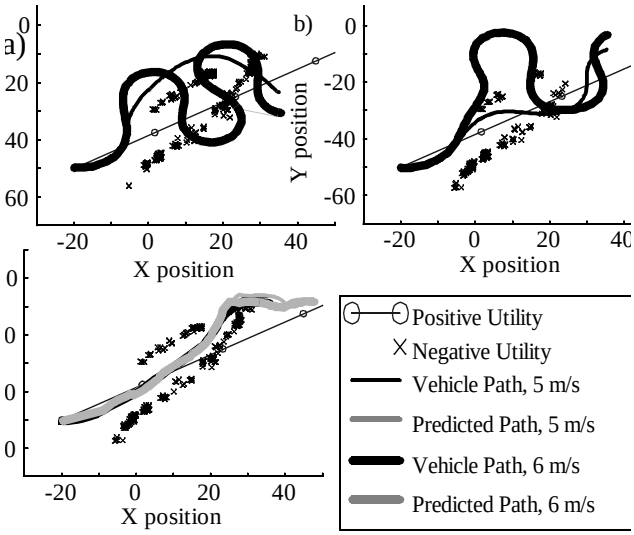

Figure 11: Paths executed at high speeds with 2 second latency: a) turn arbiter, b) path arbiter w/o prediction, c) w/ prediction.

5 CONCLUSION

Because reactivity is essential for any system operating in a dynamic, uncertain environment, it is necessary to avoid 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 evolutionary system development. However, existing command fusion techniques deal with uncertainty in an *ad hoc* manner, and they do not take system constraints into consideration when deciding upon a proper course of action.

Within DAMN, behaviors operate in a distributed fashion to generate votes for actions based on domain-specific knowledge, while a central arbiter combines their results to generate reasonable behavior which obeys all constraints and simultaneously satisfies as many objectives as possible by choosing that action which maximizes a function of the behavior votes. DAMN performs centralized arbitration of votes from distributed, independent, asynchronous decision-making processes and in so doing provides coherent, rational, goal-directed behavior while preserving real-time responsiveness to its immediate physical environment.

A new means of action selection, via utility fusion, was introduced as a solution to some observed shortcomings of behavior-based systems. Instead of voting for actions, behaviors indicate the utility of various possible world states, and it is the responsibility of the arbiter to determine which states are actually attainable and how to go about achieving them. This new approach strikes a balance between action selection and sensor fusion and has been found to yield many benefits.

The utility fusion arbiter determines the next action based on the maximization of expected utility, thus providing a unified conceptual framework for defining the semantics of votes and for dealing with uncertainty. For example, a map-based path arbiter has been implemented as a means of voting for and producing steering control. The path arbiter maintains a utility map and evaluates candidate trajectories within it, and selects that action for which the total expected utility is the greatest.

The utility 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. Behaviors can function without knowledge of the system dynamics, thus increasing their reusability for other systems. The utility arbiter can use models of the system being controlled to determine which states are actually attainable, and to increase the accuracy and stability of control. In particular, the map-based utility arbiter gathers information from behaviors about the desirability of possible vehicle locations and then evaluates candidate trajectories to determine appropriate actions. The arbiter can then use kinematic models of the robot to determine which actions can be commanded without violating non-holonomic constraints, and use of the system to provide greater stability.

DAMN has been used to combine various systems of differing capabilities on several mobile robots, at various sites; in addition to its use on the CMU Navlab vehicles, DAMN has also been used at the Lockheed Martin Corporation, the Hughes Research Labs, and the Georgia Institute of Technology. DAMN arbiters have been used to integrate navigation modules for the steering and speed control of single as well as multiple vehicles at these sites, and have also been used to select field of regard for the control of a pair of stereo cameras on a pan/tilt platform. Vehicles under the control of DAMN have driven at highway speeds, navigated across stretches of off-road terrain some kilometers in length, cooperated with other robotic vehicles, and performed teleoperation, all while providing for the safety of the vehicle and meeting mission objectives.

ACKNOWLEDGMENTS

This research was supported in part by grants from ONR (N00014-J-91-1451), ARL (DAAH049610297), and ARPA (N00014-94-1090, DAST-95-C003, F30602-93-C-0039). Work done at Carnegie Mellon University under the UGV project was supported by ARPA under contracts DACA76-89-C-0014 and DAAE07-90-C-R059, and by the National Science Foundation under NSF Contract BCS-9120655. The author was supported by a Hughes Research Fellowship.

REFERENCES

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

[Berger 85] James Berger. *Statistical Decision Theory and Bayesian Analysis*, 2nd ed. New York: Springer, 1985.

[Borenstein and Koren, 1991] Johann Borenstein and Yoram Koren. Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation. In *Proceedings of the International Conference on Robotics and Automation*, 1991.

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

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

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

[Kanayama, 1985] Yutaka Kanayama and N. Miyake. Trajectory Generation for Mobile Robots. In *Proceedings of Third International Symposium on Robotics Research*, pp. 333-340, Gouvieux, France, 1985.

[Kelly, 1995] Alonzo Kelly. *An Intelligent Predictive Control Approach to the High-Speed Cross-Country Autonomous Navigation Problem* (Ph.D. dissertation). Carnegie Mellon University Robotics Institute Technical Report CMU-RI-TR-95-33, 1995.

[Khatib, 1990] Omar Khatib, Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. In *Proceedings of the International Conference on Robotics and Automation*, 1990.

[Langer *et al.*, 1994] Dirk Langer, Julio Rosenblatt, and Martial Hebert. 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] C. Lee. Fuzzy Logic in Control Systems: Fuzzy Logic Controller -- Parts I & II. In *IEEE Transactions on Systems, Man and Cybernetics*, Vol 20 No 2, March/ April 1990.

[Moravec and Elfes, 1985] Hans Moravec and Alberto Elfes. High Resolution Map From Wide-Angle Sonar. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pp.116-121,1985.

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

[Pearl, 1988] Judea Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, 1988.

[Rosenblatt, 1997a] Julio Rosenblatt. The Distributed Architecture for Mobile Navigation. In *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 9, no. 2/3, pp.339-360, April-September, 1997.

[Rosenblatt, 1997b] Julio Rosenblatt. *DAMN: A Distributed Architecture for Mobile Navigation* (Ph.D. dissertation). Carnegie Mellon University Robotics Institute Technical Report CMU-RI-TR-97-01, Pittsburgh, PA, 1997.

[Rosenblatt and Thorpe, 1995] Julio Rosenblatt and Charles Thorpe. 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 and Kaelbling, 1986] Stanley Rosenschein and Leslie Kaelbling. The Synthesis of Digital Machines with Provable Epistemic Properties. In *Proceedings of Theoretical Aspects of Reasoning about Knowledge*, pp 83-98. 1986.

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

[Yen and Pfluger, 1992] J. Yen and N. Pfluger. A Fuzzy Logic Based Robot Navigation System. In *Proceedings of AAAI Fall Symposium*, 1992.