DAMN: A Distributed Architecture for Mobile Navigation

Julio K. Rosenblatt
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
jkr@cmu.edu

Abstract

An architecture is presented where a collection of distributed task-achieving modules, or *behaviors*, cooperatively determine a mobile robot's path by expressing their preferences for each of various possible actions. An arbiter then performs *command fusion* and selects that action which best satisfies the prioritized goals of the system, as expressed by the behaviors and their associated weights. Examples of implemented systems are given, and future research directions in command fusion are discussed.

Introduction

In order to function in unstructured, unknown, or dynamic environments, a mobile robot must be able to perceive its surroundings and generate actions that are appropriate for that environment and for the goals of the robotic system. To function effectively, an architectural framework for these sensing and reasoning processes must be imposed to provide a structure with which the system may be developed, tested, debugged, and understood. The system must also deal with uncertainty and incomplete knowledge of its environment and of the effects of its own actions. Another crucial consideration is the ability to respond to potentially dangerous situations in real-time while maintaining enough speed to be useful.

In addition, mobile robots need to combine information from several different sources. For example, the CMU Navlab vehicles are equipped with sensors such as video cameras, laser range finders, sonars, and inertial navigation systems, which are variously used by subsystems that follow roads, track paths, avoid obstacles and rough terrain, seek goals, and perform teleoperation. Because of the disparate nature of the raw sensor data and internal representations used by these subsystems, combining them into one coherent system which combines all their capabilities has proven to be very difficult. Many architectures espousing diverse principles of design methodology have been proposed over the years, but few have proved capable of integrating subsystems that have each been developed independently using whichever paradigm best achieves the task for which it is intended.

The earliest work in robot control architectures attempted to reason by manipulating abstract symbols using only pure logic (Nilsson, 1984). The limitations of this top-down AI approach led to a new generation of architectures designed in a bottom-up fashion to provide greater reactivity to the robot's surroundings, but sacrificed generality and the ability to reason about the system's own intentions and goals (Brooks, 1986; Agre & Chapman, 1987; Arkin, 1987).

It has been argued that a hierarchical approach is needed which allows slower abstract reasoning at the higher levels and faster numerical computations at the lower levels, thus allowing varying trade-offs between responsiveness and optimality as appropriate at each level (Payton, 1986; Albus, McCain & Lumia, 1987). While such an approach provides aspects of both deliberative planning and reactive control, the top-down nature of hierarchical structures tends to overly restrict the lower levels so that newly received information cannot be fully taken advantage of (Payton, Rosenblatt & Keirsey, 1990). In hierarchical architectures, each layer 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.

Experience over the years with different architectures and planning systems for mobile robots has led me to a distributed approach where an arbiter receives votes for and against commands from each subsystem and decides upon the course of action which best satisfies the current goals and constraints of the system. The architecture is designed with the underlying belief that centralized arbitration of votes from distributed, independent decision-making processes provides coherent, rational, goal-directed behavior while preserving real-time responsiveness to its immediate physical environment. Furthermore, a framework for developing and integrating independent decision-making modules communicating with such arbiters facilitates their development and leads to evolutionary creation of robust systems of incrementally greater capabilities.

The Distributed Architecture for Mobile Navigation has been successfully used to integrate the various subsystems mentioned above, thus providing systems that perform road following, cross-country navigation, or teleoperation while avoiding obstacles and meeting mission objectives. In addition to its use on the CMU Navlab vehicles, DAMN has also been used on outdoor test vehicles at Martin Marietta and on indoor robots and simulated environments at the Hughes Research Labs..

The Distributed Architecture for Mobile Navigation

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. Reactive components provide the basic capabilities which enable the robot to achieve low-level tasks without injury to itself or its environment, while deliberative components provide the ability to achieve higher-level goals and to avoid mistakes which could lead to inefficiencies or even mission failure. But rather than imposing an hierarchical structure to achieve this symbiosis, the Distributed Architecture for Mobile Navigation (DAMN) takes an approach where multiple modules concurrently share control of the robot. In order to achieve this, a common interface is established so that modules can communicate their intentions without regard for the level of planning involved (Langer, Rosenblatt & Hebert, 1994).

A scheme is used where each module votes for or against various alternatives in the command space based on geometric reasoning; this is at a higher level than direct actuator control, but lower than symbolic reasoning. This reasoning at the geometric level creates a bridge between the high-level goals of an AI planner and the low-level motor skills of a controller and is crucial to the successful operation of a robotic system in the real world, and yet it is the least understood.

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 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 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.

DAMN is a behavior-based architecture similar in some regards to reactive systems such as the Subsumption Architecture (Brooks, 1986). In contrast to more traditional centralized AI planners that build a centralized world model and plan an optimal path through it, a behavior-based architecture consists of specialized task-achieving modules that operate independently and are responsible for only a very narrow portion of vehicle control, thus avoiding the need for sensor fusion. A distributed architecture has several advantages over a centralized one, including greater reactivity, flexibility, and robustness (Payton, Rosenblatt & Keirsey, 1990). However, one important distinction between this system and purely reactive systems is that, while an attempt is made to keep the perception and planning components of a behavior as simple as possible without sacrificing dependability, they can and often do maintain internal representations of the world. Brooks (1993) has argued that "the world is its own best model", but this assumes that the vehicle's sensors and the algorithms which process them are essentially free of harmful noise and that they can not benefit from evidence combination between consecutive scenes. In addition, disallowing the use of internal representations requires that all environmental features of immediate interest be visible to the vehicle sensors at all times. This adds unnecessary constraints and reduces the flexibility of the overall vehicle system

The DAMN architecture is designed to provide the basic capabilities essential to any mobile robot system, or *first level of competence* in the parlance of the Subsumption Architecture. In DAMN, this consists of safety behaviors which limit turn and speed to avoid vehicle tip-over or wheel slippage, obstacle avoidance behaviors to prevent collisions, as well as various auxiliary behaviors (see DAMN Behaviors section). As new functions are needed, additional behaviors can be added to the system without any need for modification to the previously included

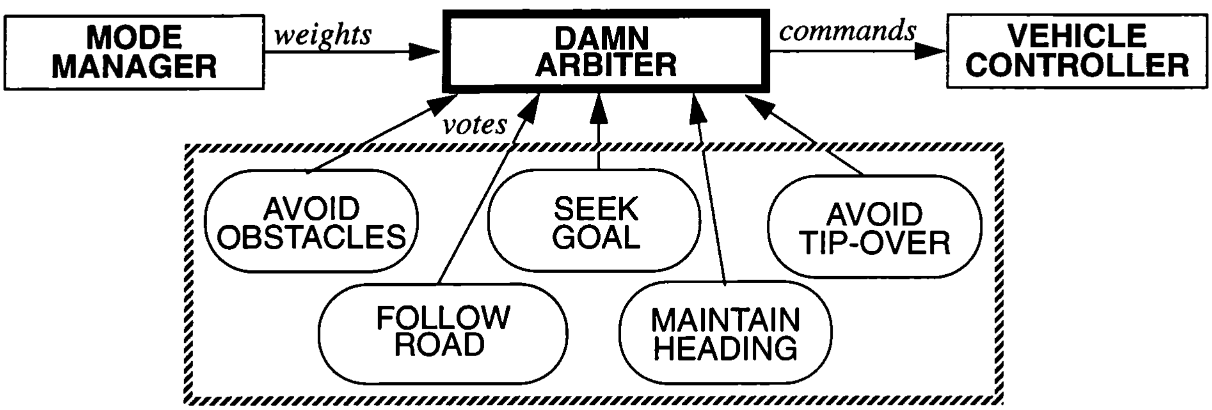

Figure 1: Behaviors sending votes to arbiter

behaviors, thus preserving their established functionality.

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.

DAMN Arbiters

In a distributed architecture, it is necessary to decide which behaviors should be controlling the vehicle at any given time. In some architectures, this is achieved by having priorities assigned to each behavior; of all the behaviors issuing commands, the one with the highest priority is in control and the rest are ignored (Brooks, 1986; Rosenschein & Kaelbling, 1986). In order to allow multiple considerations to affect vehicle actions concurrently, DAMN instead uses a scheme where each behavior votes for or against each of a set of possible vehicle actions (Rosenblatt & Payton, 1989). An arbiter then performs *command fusion* to select the most appropriate action. While 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.

Turn Arbiter

In the case of the turn arbiter, 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. The arbiter computes a weighted sum of the votes for each steering command, with the weights reflecting the relative priorities of the behaviors. The steering command with the highest vote is sent to the vehicle controller.

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, the arbiter performs sub-pixel interpolation. This is done by first convolving the votes with a Gaussian mask to smooth the values and then selecting the command option with the highest resulting value. A parabola is then fit to that value and the ones on either side, and the peak of the parabola is used as the command to be issued to the controller. This process is illustrated in Figure 2, where the votes from two behaviors (a & b) are linearly combined (c), and then smoothed and interpolated to produce the resulting command (d).

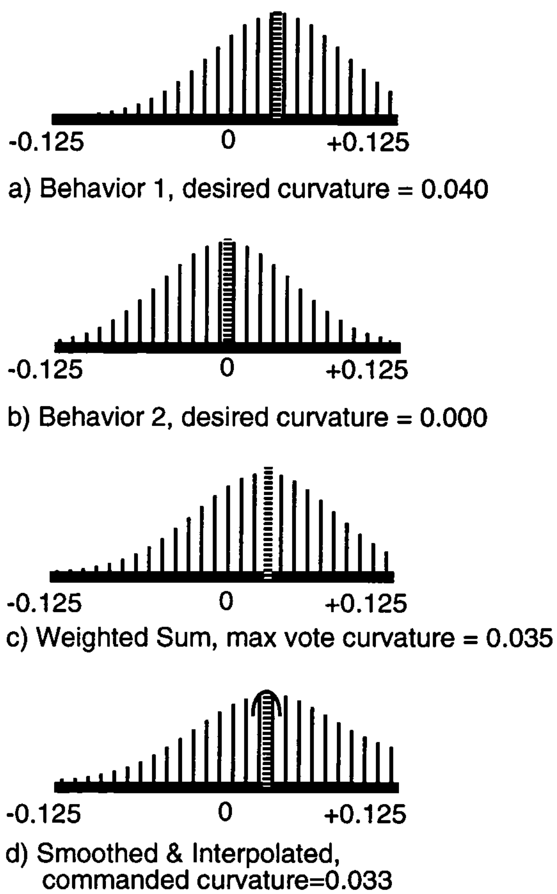

Figure 2: Command fusion process

Speed Arbiter

The emphasis in the research thus far has been in command fusion for the control of vehicle steering; until recently the commanded speed was decided in a very simplistic fashion based upon the commanded turn radius. The user-specified maximum vehicle speed was multiplied by the normalized weighted sum for the chosen turn radius; the result was the speed command issued.

An entirely separate speed arbiter with its own set of associated behaviors has now been developed. Thus, the turn behaviors can vote for turn commands without concern that the absolute magnitude of their votes will affect vehicle speed. At present each speed behavior votes for the largest speed possible which meets that behavior's constraints, and the arbiter simply chooses the minimum of those maxima, so that all speed constraints are satisfied.

Coordination of Arbiters

Because the choices of turn and speed commands are not completely independent and therefore must be coordinated, many of the speed behaviors have as one of their inputs the output of the turn arbiter, so that the choice of an appropriate speed is influenced by the currently commanded turn radius. Other speed behaviors instead use the estimated actual turn radius of the vehicle so that they operate in a closed-loop fashion, albeit with greater delays. Likewise, some turn behaviors use the current vehicle speed in deciding upon allowable turn options.

DAMN Behaviors

Within the framework of DAMN, behaviors must be defined to provide the task-specific knowledge for controlling the vehicle. Each behavior runs completely independently and asynchronously, providing votes to the arbiter each at its own rate and according to its own time constraints. The arbiter periodically sums all the latest votes from each behavior and issues commands to the vehicle controller.

Safety Behaviors

A basic need for any mobile robot system is the ability to avoid situations hazardous to itself or to other objects in its environment. Therefore, an important part of DAMN is its "first level of competence" (Brooks, 1986), which consists of behaviors designed to provide vehicle safety. In contrast to priority-based architectures which only allow one behavior to be effective at any given moment, the structure of DAMN and its arbitration scheme allow the function of these safety behaviors to be preserved as additional levels of competence are added.

Obstacle Avoidance The most important behavior in the context of vehicle safety is the *Obstacle Avoidance* behavior. In order to decide in which directions the vehicle may safely travel, this behavior receives a list of current obstacles in vehicle-centered coordinates and evaluates each of the possible command options, as illustrated in Figure 3. The source of these obstacles may be intraversable regions of terrain determined by range image processing or stereo vision, by sonar detection of objects above the ground plane, or any other means of obstacle detection as appropriate to the current task and environment (Daily et al, 1986; Langer, Rosenblatt & Hebert, 1994).

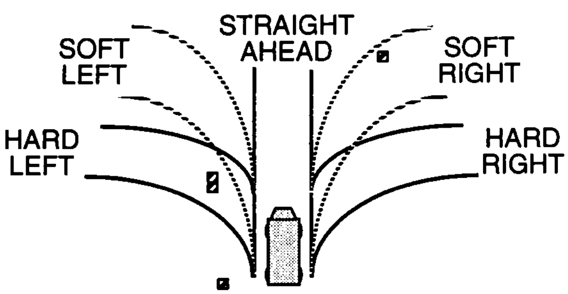

Figure 3: Arc evaluation in the Obstacle Avoidance behavior

If a trajectory is completely free of any neighboring obstacles (such as the Straight Ahead or Hard Right turns shown in Figure 3), then the obstacle avoidance behavior votes for travelling along that arc. If an obstacle lies in the path of a trajectory, the behavior votes against that arc, with the magnitude of the penalty proportional to the distance from the obstacle. Thus, the *Obstacle Avoidance* behavior votes more strongly against those turns that would result in an immediate impact (Hard Left in the figure) and votes less strongly against those turns which would only result in a collision after travelling several meters (Soft Right). In order to avoid bringing the vehicle unnecessarily close to an obstacle, the behavior also votes against those arcs that result in a near miss (Soft Left), although the evaluation is not as unfavorable as for those trajectories leading to a direct collision.

Vehicle Dynamics Another vital aspect of vehicle safety is insuring that the commanded speed and turn stay within the dynamic constraints of the vehicle as it travels over varying terrain conditions. The most important of these constraints is the one that insures that the vehicle will not tip over. Given a velocity of magnitude v, the maximum positive and negative curvatures κ to avoid tip-over would be:

$$\pm\kappa_{max} = \frac{\pm(\eta \cdot g \cdot \cos\rho) + g \cdot \sin\rho}{v^2}$$

where η is the ratio of the distance between the vehicle's center of gravity (c.g.) and the wheels to the c.g. height, g is the acceleration due to gravity, and ρ is the vehicle roll with respect to the gravity vector, as illustrated in Figure 4. Likewise, for a given vehicle turn curvature, the maximum velocity is:

$$v_{max} = \mathrm{Min}\left|\frac{\pm(\eta \cdot g \cdot \cos\rho) + g \cdot \sin\rho}{\kappa}\right|^{1/2}$$

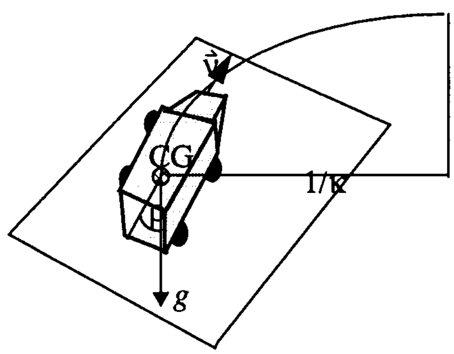

Figure 4: Vehicle dynamics

Similar constraints can be imposed on vehicle turn radius and speed in order to avoid tire slippage. The limit on curvature for slippage is:

$$\pm\kappa_{max} = \frac{(\mu \cdot g \cdot \cos\rho) \pm (g \cdot \sin\rho)}{v^2}$$

where μ is the dynamic coefficient of friction between the tire and the terrain, and for speed:

$$\pm v_{max} = \left| \frac{(\mu \cdot g \cdot \cos\rho) \pm (g \cdot \sin\rho)}{\kappa} \right|^{1/2}$$

Two behaviors, *Limit Turn* and *Limit Speed* send votes to the arbiter that implement these constraints, voting against commands that violate them.

Road Following

Once vehicle safety has been assured by the obstacle avoidance and dynamic constraint behaviors, it is desirable to add additional behaviors that provide the system with the ability to achieve the tasks for which it is intended., such as road following; one of the behaviors that have been implemented within DAMN to provide this function is ALVINN.

The ALVINN road following system is an artificial neural network that is trained, using backpropagation, to associate preprocessed low resolution input images with the appropriate output steering commands (Pomerleau, 1992). In the case of ALVINN, creating a behavior that independently evaluated each arc was relatively straightforward. The units of the neural network's output layer each represent an evaluation of a particular turn command, with the layer trained to produce Gaussian curves centered about those turns that would follow the road ahead. These units are simply resampled to the DAMN voting command space, using a Gaussian of the appropriate width. This process is illustrated in Figure 5.

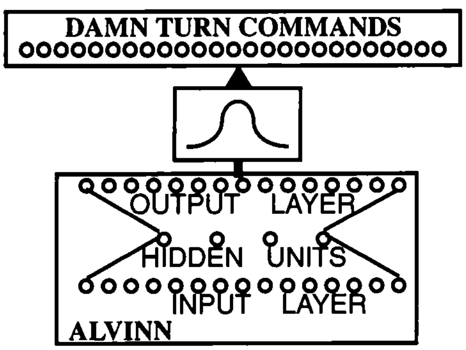

Figure 5: Resampling of ALVINN output layer

Goal-Directed Behaviors

Another important level of functionality that should be present in any general purpose robotic system is the ability to reach certain destinations using whatever global information is available. While the low-level behaviors operate at a high rate to ensure safety and to provide functions such as road following and cross-country navigation, high-level behaviors are free to process map-based or symbolic information at a slower rate, and periodically issue votes to the arbiter that guide the robot towards the current goal.

Subgoals The *Goal Seeking* behavior is one way to provide this capability. This simple behavior directs the vehicle toward a series of goal points specified in global coordinates either by the user (Langer, Rosenblatt & Hebert, 1994) or by a map-based planner (Keirsey, Payton & Rosenblatt, 1988). The desired turn radius is transformed into a series of votes by applying a Gaussian whose peak is at the desired turn radius and which tapers off as the difference between this turn radius and a prospective turn command increases. A goal is considered satisfied once the vehicle enters a circle centered at the goal location; then the next goal is pursued. Because of errors in goal placement and accumulated errors in vehicle positioning, a goal point may not be reachable. For this reason, an ellipse is defined with the current goal and the subsequent goal as foci; if the vehicle enters this ellipse, the current goal is abandoned and the next one becomes the current goal instead, thus allowing progress to continue.

Dynamic Programming Some more sophisticated map-based planning techniques have also been integrated and used within the DAMN framework. These planners use dynamic programming techniques based on the A* search algorithm (Nilsson, 1980) to determine an optimal global path. However, an important point is that they do not hand a plan down to a lower level planner for execution, but rather maintain an internal representation that allows them to participate directly in the control of the vehicle based on its current state. A* yields a set of pointers within the map grid that point toward the goal, as depicted by the small arrows in Figure 6. During execution, this grid may be indexed by the current vehicle position to yield a path towards the goal which is optimal based on the information available in the map at that time.

The Internalized Plans (Payton, 1990) approach uses a detailed map to perform an A* search from the goal(s) back toward the start point to create a "Gradient Field" towards the goal. The type and slope of the terrain, among other factors, is used to estimate the cost of traversal between grid cells. During run-time, the grid cell containing the current vehicle location is identified, and the Gradient Field pointers are followed forward to the point **G'** in Figure 6; the desired heading to reach the goal is that from the current location **S** to **G'**, and a series of votes with its peak at that value is sent to the turn arbiter.

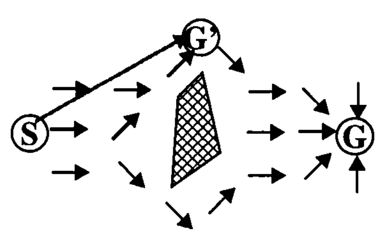

Figure 6: Following Gradient Field to determine intermediate goal heading

The D* planner (Stentz, 1993) also creates a grid with "backpointers" that represent information on how best to reach the goal from any location in the map. The map may initially contain no information, but is created incrementally as new information becomes available during the execution of a mission, and the arc traversal costs and backpointers are updated to reflect this new knowledge. The resulting global plan is integrated into DAMN as a behavior by determining, for each possible turn command, the weight w of reaching the goal from a point along that arc a fixed distance ahead (the squares designated collectively as **S'** in Figure 7). If w_{max} and w_{min} are the maximum and minimum values of w, then the vote for each turn command is determined as: $(w_{max} - w) / (w_{max} - w_{min})$. In the case that a point S' is not represented on the grid, or if the goal cannot be reached from it, then the vote for that arc is set to -1.

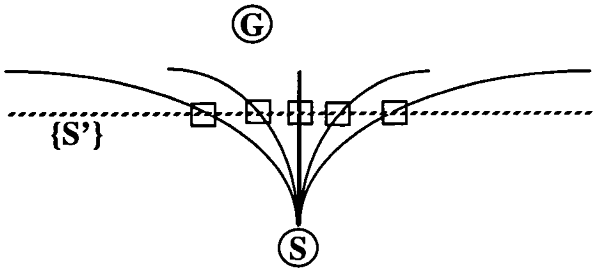

Figure 7: Using D* to evaluate distance to goal for each arc

Teleoperation

Teleoperation is another possible mode in which a robotic system may need to operate. The STRIPE teleoperation system (Kay & Thorpe, 1993) provides a graphical user interface allowing a human operator to designate waypoints for the vehicle by selecting points on a video image and projecting them on to the surface on which the vehicle is travelling. STRIPE then fits a spline to these points and uses pure pursuit to track the path. When used in isolation, it simply sends a steering command to the controller; when used as a DAMN behavior, it sends a series of votes representing a Gaussian centered on the desired command. This allows the dynamic constraints and obstacle avoidance behaviors to be used in conjunction with STRIPE so that the safety of the vehicle is still assured.

Auxiliary Behaviors

Various other auxiliary behaviors that do not achieve a particular task but issue votes for secondary considerations may also be run. These include the *Drive Straight* behavior, which simply favors going in whatever direction the vehicle is already heading at any given instant, in order to avoid sudden and unnecessary turns; and the *Maintain Turn* behavior, which votes against turning in directions opposite to the currently commanded turn, and which helps to avoid unnecessary oscillations in steering, the *Follow Heading* behavior which tries to keep the vehicle pointed in a constant direction, as well as various behaviors which allow user input to affect the choice of vehicle turn and speed commands.

Combining Behavior Votes

The voting strengths, or weights, of each behavior are specified by the user, and are then normalized by the arbiter so that their sum equals 1. Because only the relative values are important, and because the magnitude of each behavior's votes vary according to their importance, DAMN is fairly insensitive to the values of these weights and the system performs well without a need to tweak these parameters. For example, the *Obstacle Avoidance* behavior has been run in conjunction with the *Seek Goal* behaviors with relative weights of 0.75 and 0.25, respectively, and with weights of 0.9 and 0.1, and in both cases has successfully reached goals while avoiding obstacles. The vote weights of each behavior can also be modified by messages sent to the arbiter from a mode manager module. It can reconfigure the weights according to whatever top-down planning considerations it may have, and potentially could use bottom-up information about the effectiveness and relevance of a behavior (Payton et al, 1993). Different modes of operation that exclude some behaviors can be constructed by setting the weights those behaviors to 0. A Mode Manager was developed at the Hughes Research Labs to be used with DAMN for this purpose, and at CMU Annotated Maps were integrated with DAMN to provide this capability (Thorpe et al, 1991).

As a simple example to illustrate the manner in which votes are issued and arbitrated within DAMN, consider the case in Figure 8 where two behaviors are active, one responsible for obstacle avoidance and the other for goal seeking (only five turn options are shown for simplicity). The magnitude of a vote is indicated by the size of a circle, with a large unfilled circle representing a vote of +1, a large striped circle a value of -1, and a small circle a value near 0. Thus, the goal-seeking behavior is voting most strongly in favor proceeding straight and less favorably for a soft left turn, and voting against hard left or any right turns; the obstacle avoidance behavior is voting against a hard left or soft right, and allowing the other turns as acceptable, with soft left being the most favorable.

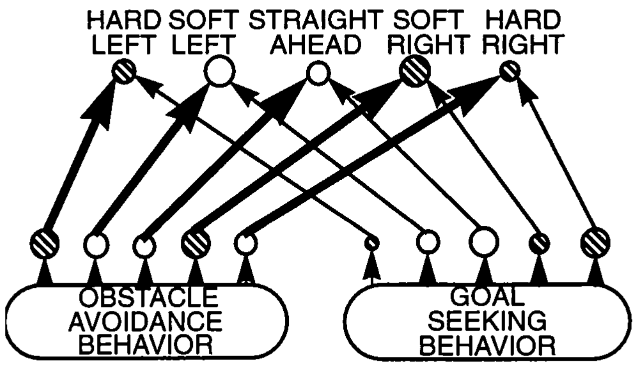

Figure 8: Command fusion in DAMN

Because avoiding obstacles is more important than taking the shortest path to the goal, the obstacle avoidance behavior is assigned a higher weight than the goal seeking behavior, as indicated by the thicker arrows in the diagram. The arbiter then computes a weighted sum of the votes it has received from each behavior, and the command choice with the highest value is selected and issued to the vehicle controller. In this case a soft left turn would be executed, since its weighted sum is the greatest, thus avoiding any obstacles while still more or less moving toward the goal. The favorableness of the selected turn command may also used to determine vehicle speed, so that, for example, the vehicle would slow down if a command is issued which will take the vehicle too far from the path to the goal point. Another possibility is to have a separate speed arbiter that would receive commands from behaviors that, given the current vehicle radius, determine the maximum speed that would satisfy their objectives.

User Interface

A simple text interface which optionally outputs informational messages from each running module is provided for debugging and logging purposes. The user may also interactively start and halt the arbiters and the vehicle, vary parameters such as maximum speed, and toggle debugging output and data recording.

A Graphical User Interface has also been integrated into the DAMN arbiters. It outputs the votes issued by each active behavior, as well as their current weights, and allows the user to modify those weights. The summed weighted votes and commands chosen by the arbiters are also displayed. A screen dump of this display is shown in Figure 9.

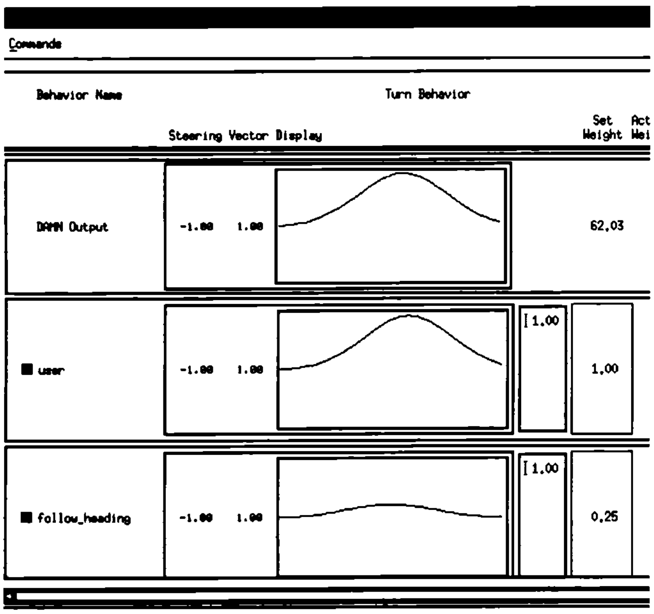

Figure 9: DAMN Graphical User Interface

Results

DAMN is designed so that various behaviors can be easily added or removed from the system, depending on the current task at hand. Although the modules described above all use very different paradigms and representations, it has been relatively straightforward to integrate each and every one of them into the framework of DAMN. Sensor fusion is not necessary since the command fusion process in DAMN preserves the information that is critical to decision-making, yielding the capability to concurrently satisfy multiple objectives without the need for centralized bottlenecks. A detailed description of an implemented system and the experimental results achieved can be found in (Langer, Rosenblatt & Hebert, 1994).

All of the behaviors described in the DAMN Behaviors section have been used in conjunction with each other in various configurations, yielding systems that were more capable than they would have been otherwise. Conceptually, three levels of competence have been implemented in DAMN thus far, as shown in Figure 10. These levels of competence are convenient for describing the incremental manner in which the system's capabilities evolve; however, it is important to note that all behaviors co-exist at the same level of planning. The importance of a behavior's decisions is reflected by the weighting factor for its votes, and is in no way affected by the level of competence in which it is described.

The safety behaviors are used as a first level of competence upon which other levels can be added. Movement is the second level of competence that has been implemented; road following, cross-country navigation, and teleoperation behaviors have all been run together with the obstacle avoidance behavior to provide various forms of generating purposeful movement while maintaining safety (Thorpe et al, 1991). The third level of competence is comprised of the various map-based goal-seeking behaviors. Cross-country behaviors have been combined with goal-oriented behaviors to produce directed off-road navigation (Keirsey, Payton & Rosenblatt, 1988; Stentz, 1993).

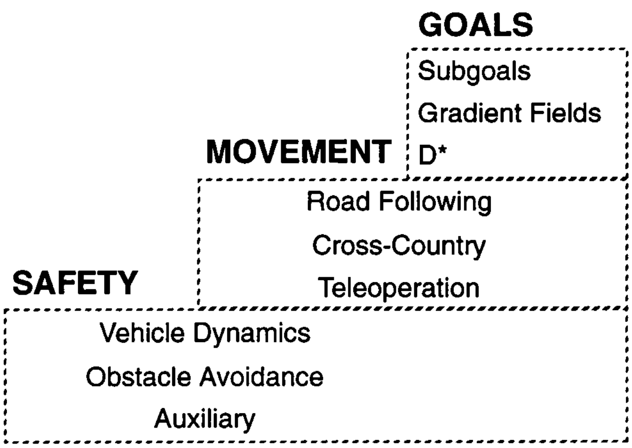

Figure 10: Evolutionary system development in DAMN

Future Work

DAMN has proven to be very effective as an architecture which greatly facilitates the integration of a wide variety of different vehicle navigation subsystems; however, the methods used to date have largely been of an ad hoc nature. As DAMN is used for increasingly complex tasks where many behaviors may be issuing votes concurrently, there will be a greater need to have the semantics of the voting process carefully defined. 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.

The various approaches to reasoning with uncertainty can be classified as either *extensional* or *intensional* (Pearl, 1988). One extensional approach currently in vogue is Fuzzy Logic systems (Zadeh, 1973; Lee, 1990), and indeed the voting and arbitration scheme described in the DAMN Arbiters section bears some obvious similarities to such systems. Fuzzy Logic is typically used within the framework of rule-based systems, as in (Kamada, Naoi & Goto, 1990), but behavior-based systems are generally procedural in nature and reason at the geometric level, and therefore do not readily lend themselves to the use of if-then rules which operate at the symbolic level of reasoning. Yen and Pfluger (1992) propose an architecture based on DAMN that uses Fuzzy Logic, but it restricts each behavior to determining a unique desired steering direction which is then voted for using fuzzy membership sets. The scheme described here is more general in that it allows for an arbitrary distribution of votes.

Intensional approaches, also referred to as *declarative* or *model-based*, attach uncertainties not to individual assertions but rather to possible states of the world. The classical intensional system is Bayesian probability (Bayes, 1763), which is founded on Bayes' Theorem for inversion of conditional probabilities:

$$P(H|e) = \frac{P(e|H) \cdot P(H)}{P(e)}$$

which states that the posterior probability of hypothesis H being true given supporting evidence e is equal to the prior probability of H multiplied by the probability of e occurring given H, normalized by the prior probability of e. Practical problems with the use of Bayesian probabilities arise because all of the prior probabilities $P(e)$ and conditional probabilities $P(e|H)$ must be specified, and the truth value of all relevant evidence e must be known before any conclusions can be drawn. The Dempster-Shafer Theory is a non-Bayesian method which allows the use of incompletely specified probabilistic models (Shafer, 1976). Rather than specifying probabilities of assertions being true, it uses partially specified models of possible worlds to reason qualitatively about the probability that a given assertion is provable. While this solves some of the problems of Bayesian reasoning, Dempster-Shafer theory only provides estimates of the probability of various partially specified possible worlds that may include contradictions, and therefore is poorly suited for the purpose of action selection.

Because we have control over some of the variables involved (actuator values) and are attempting to decide which among a set of possible actions to take, it is more natural to make judgments on the usefulness of actions based on their consequences than it is to judge the likelihoods of various statements being true. If we assign a utility measure $U(c)$ for each possible consequence of an action, then the *expected utility* for an action a is:

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

where $P(c|a,e)$ is the probability distribution of the consequence configuration c, conditioned upon selecting action a and observing evidence e (Pearl, 1988). Thus, if we can define these utilities and probabilities, we can then apply the *Maximum Expected Utility* (MEU) criterion to select the optimal action based on our current information.

If utility theory is to be applied to the problem of evidence combination and action selection for mobile navigation tasks, the first and foremost issue that must be resolved is how to define the utility functions. Behaviors are defined in order to achieve some task, so it is fair to assume that there must be at least an implicit measure of "goodness" or utility with respect to that task. For example, an obstacle avoidance behavior's task is to maximize distance to obstacles, so that the distance from the vehicle to the nearest obstacle could be used as a measure of goodness. Likewise, proximity to the current goal could be used for the goal-based behaviors, as proximity to the center of a road (or lane) could be used by road following modules.

Evaluation

Objective quantitative evaluation of mobile robot systems has historically proven to be very difficult, and the task is doubly difficult when attempting to compare entire architectures because there are so many degrees of freedom in the design and implementation of a mobile robot system. The most straightforward means of evaluation is to attempt to define measures of the quality of the path which a vehicle has traversed under autonomous control.

Utility Measures One means of evaluating architectures is to use measures of "goodness" like those used to define the utility of each action for the various behaviors. Although evaluating the performance based on the utility measures defined in the Future Work section would appear to be circular, they are the best measures available. Furthermore, their evaluation is not vacuous given that uncertainty exists in all aspects of a system: in the declarative knowledge, sensed information, procedural knowledge, and in the effects of its actions and the actions of other agents; thus the expected utility will in general not agree with the actual utility that can be measured for those actions that are taken.

Let the goodness value for behavior b at a state s be computed by the function $g_b(s)$. Then one possible measure for the utility, with respect to behavior b, of consequence c is the change in this measure of goodness with respect to the current state s_0:

$$U_b(c) = \Delta g_b = g_b(c) - g_b(s_0)$$

However, the value of this measure can be arbitrarily large, so it must be normalized and assigned a consistent semantic meaning for all behaviors. One means of normalization would be to divide by the larger of the two goodness measures:

$$U_b(c) = \Delta g_b / Max(g_b(c), g_b(s_0))$$

This would bound the absolute value of $U_b(c)$ to be no greater than 1. It would also have the effect that, for the same Δg_b, the utility of an action would be greater when the goodness measure is small than when it is large. This appears to be desirable for behaviors such as obstacle avoidance whose importance should grow as the situation worsens; however, the suitability of this measure is less clear for other behaviors such as goal-seeking.

Another possibility for a normalized utility measure is analogous to the one used by the D* behavior described in the Goal-Directed Behaviors section. After computing $g_b(c)$ for each possible consequence c, we determine the maximum and minimum values g_{max} and g_{min}, and use the following measure:

$$U_b(c) = (g_b(c) - g_{min}) / (g_{max} - g_{min})$$

While this measure has the desirable property of being bounded by the interval [0,1], it has the potentially undesirable property that the range of utility values will always completely span that interval. Thus, for example, if all potential actions have exactly the same utility except for one that has a slightly higher value, then the normalized utility values will be 1 for that one action and 0 for all the rest.

Path Smoothness Another general measure of the quality of the path is the smoothness with which it is controlled. Integrating the squared derivative of curvature along the vehicle's actual path provides a measure of smoothness that can be applied to the architecture (Kamada, Naoi & Goto, 1990), and the derivative of acceleration, or *jerk*, provides a measure of smoothness in the vehicle's speed as well as steering. Integrating the squared vehicle curvature along the vehicle's path may also be useful as a measure of smoothness, which reflects consistency in decision-making and the ability to anticipate events.

Path Accuracy The accuracy of the path, i.e. the extent to which the commanded path matches the actual path taken by the vehicle, can also provide an important means of evaluating an architecture and its planning subsystems. Path accuracy can be measured by integrating the squared error between estimated and actual vehicle poses. If the system commands trajectories which are not physically realizable, the actual path taken may deviate significantly. Likewise, if large latencies exist in the system and are not adequately compensated for, the commanded path will only begin to be executed well after the system intended it.

Conclusion

The Distributed Architecture for Mobile Navigation has been successfully used to create systems which safely follow roads or traverse cross-country terrain while avoiding obstacles and pursuing mission goals. Like other behavior-based architectures, it avoids sensory and decision-making bottlenecks and is therefore able to respond in real-time to external events; however, it imposes no constraints on the nature of the information or the processing within a behavior, only on the interface between the behavior and the command arbiter. Furthermore, the behaviors are not subject to any timing constraints; each behavior operates asynchronously and in parallel.

Non-reactive behaviors may use plans to achieve goals and coordinate with other agents; thus, like centralized or hierarchical architectures, DAMN is able to assimilate and use high-level information. Finally, unlike architectures with prioritized modules, DAMN's vote arbitration scheme allows multiple goals and constraints to be fulfilled simultaneously, thus providing goal-oriented satisficing behavior without sacrificing real-time reactiveness.

Acknowledgments

This research was partly sponsored by ARPA, under contracts "Perception for Outdoor Navigation" (contract number DACA76-89-C-0014, monitored by the US Army Topographic Engineering Center) and "Unmanned Ground Vehicle System" (contract number DAAE07-90-C-R059, monitored by TACOM), and partly supported by a Hughes Research Fellowship. The author would like to acknowledge the support and guidance of Dave Payton at the Hughes Research Labs and Chuck Thorpe at Carnegie Mellon University, as well as the shared efforts and technical support of the Robotics Institute at Carnegie Mellon University.

Appendix 1: Responses to workshop questions

1. **Coordination**-- *How should the agent arbitrate/ coordinate/cooperate its behaviors and actions? Is there a need for central behavior coordination?*

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. The DAMN architecture takes the position that some centralization is needed, but the right level must be chosen so that it does not create a bottleneck, and the interfaces must be defined so as to avoid being overly restrictive.

Rather than imposing an hierarchical structure or using a prioritized behavior-based systems to effect a traded control system, the Distributed Architecture for Mobile Navigation takes a shared control approach where several modules concurrently have some responsibility for control of the robot. In order to achieve this, a common interface is established so that modules can communicate their intentions without regard for the level of planning involved.

Votes from all behaviors are used in determining what the next action should be, so that compromises are made when possible; however, if two behaviors suggest actions that can not be reconciled, then one of the two must be chosen. Meta-level control may be exerted so that behaviors with mutually exclusive goals do not operate simultaneously; however, 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.

The hypothesis proposed here is that centralized arbitration of votes from distributed, independent decision-making processes provides coherent, rational, goal-directed behavior while preserving real-time responsiveness to its immediate physical environment.

2. **Interfaces**-- *How can human expertise be easily brought into an agent's decisions? Will the agent need to translate natural language internally before it can interact with the world?*

Natural language is not needed for human-computer interaction, and in many domains would actually be more difficult for an operator to use than other modes of interaction. For example geometric information or mission goals can be more easily specified via a Graphical User Interface (GUI), and direct control of the robot can be better effected through the use of a joystick, for example.

In DAMN, the user may specify which behaviors are to be active and what their relative weights should be, either a priori or during run-time via a GUI, thus providing meta-level control. Behaviors that support teleoperation have also been implemented, so that the user may specify the robot path via waypoints in a video image, by using a joystick, or by typing simple keystroke commands. As with all other behaviors, the user's input is expressed as votes which are then combined with the votes of other behaviors and arbitrated.

3. **Representation**-- *How much internal representation of knowledge and skills is needed? How should the agent organize and represent its internal knowledge and skills? Is more than one representational formalism needed?*

The DAMN system only requires votes from each behavior, so that each module is free to use whatever representation and paradigm best serves for its particular task. However, the selection of which behaviors should be active at any given time is currently done in a fairly rudimentary way; an explicit representation of each behavior's skills would allow for more dynamic and flexible responses to unforeseen situations, and may also facilitate learning.

4. **Structural**-- *How should the computational capabilities of an agent be divided, structured, and interconnected? How much does each level/component of an agent architecture have to know about the other levels/ components?*

The capabilities of an agent should be divided up as finely as is practical among task-achieving behaviors which operate asynchronously. They should be completely modular and independent, so that new capabilities may be added in an evolutionary fashion without a need to disrupt or modify existing functionality.

Ideally, higher level components that exert meta-level control should only need to know which skills are provided by the lower levels, without any knowledge of how those skills are implemented whatsoever.

5. **Performance**-- *What types of performance goals and metrics can realistically be used for agents operating in dynamic, uncertain, and even actively hostile environments?*

The most straightforward means of evaluation is to attempt to define measures of the quality of the path which a vehicle has traversed under autonomous control. The measures I suggest within the body of the paper are the path's utility, smoothness, and accuracy; please refer to the appropriate section for further details.

6. **Psychology**-- *Why should we build agents that mimic anthropomorphic functionalities? How far can/should we draw metaphoric similarities to human/animal psychology? How much should memory organization depend on human/animal psychology?*

Existing systems, i.e. humans and other animals, should serve merely as an inspiration, never as a constraint. Furthermore, good software engineering practices dictates that robotic systems must evolve in a more orderly fashion than their biological counterparts.

7. **Simulation**-- *What, if any, role can advanced simulation technology play in developing and verifying* *modules and/or systems? Can we have standard virtual components/test environments that everybody trusts and can play a role in comparing systems to each other? How far can development of modules profitably proceed before they should be grounded in a working system?*

Simulation technology can be a very important tool for developing systems which are eventually realized in a physical implementation, but the former must not replace the latter. In my experiences in developing navigation systems, simulation has often played a key role in their success. Simulation provides a means to develop and extensively test the system with minimal resources and without risking physical damage to a robot or its surroundings. Subsequent testing on a real robot then provides not only a means of validating the system, but also a means of discovering the flaws in the simulation so that its fidelity may be improved for future use.

Any attempt to standardize the virtual components of a robotic system would be premature at this point, as there is still much debate as to what form the decomposition of the architecture should take. Also, different needs in terms of complexity, responsiveness, and optimality will require different architectures which make different trade-offs.

Standardized test environments, however, are something that is sorely needed so that architectures may be compared to each other in a systematic way. Again, different environments with different demands in complexity, responsiveness, and optimality will be needed to highlight the relative strengths and weaknesses of each architecture in various domains. While a real robot provides the best testbed, simulation can also play an important role because of its accessibility, its low cost, and because it provides the ability to record every aspect of a trial for analysis and for reproducing results.

8. **Learning**-- *How can a given architecture support learning? How can knowledge and skills be moved between different layers of an agent architecture?*

There are many different forms of learning that can be supported in many different ways. There currently is no learning within the DAMN architecture, but reinforcement learning would be a natural means of acquiring information regarding the utility of various actions in different circumstances. The relevance and usefulness of each behavior in particular situations might also be learned through techniques such as genetic algorithms. Simulation would most likely play a significant role in such learning schemes.

References

Agre, P. & Chapman, D. (1987), *Pengi: An Implementation of a Theory of Activity*, Proc. of Sixth AAAI, pp. 268-272, Seattle, WA.

Arkin, R. (1987), *Motor Schema Based Navigation for a Mobile Robot: An Approach to Programming by Behavior*, in Proceedings of the International Conference on Robotics and Automation.

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

Bayes, T. (1763), *An Essay Towards Solving a Problem in the Doctrine of Chances*, in Philosophical Transactions, vol. 3, pp. 370-418.

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.

Brooks, R. (1993), *Intelligence Without Reason*, in Proceedings of the International Joint Conference on Artificial Intelligence.

Daily, M., Harris, J., Keirsey, D., Olin, K., Payton, D., Reiser, K., Rosenblatt, J., Tseng, D. and Wong, V. (1988), *Autonomous Cross-Country Navigation with the ALV*, in IEEE Conference on Robotics and Automation, Philadelphia, PA, April, 1988. (Also appears in DARPA Knowledge Based Planning Workshop, December, 1987 pp 20-1 to 20-10).

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

Kay, J., Thorpe, C. (1993), *STRIPE Supervised Telerobotics Using Incremental Polygonal Earth Geometry.* In Proc. Intelligent Autonomous Systems Conference.

Keirsey, D.M., Payton, D.W. & Rosenblatt, J.K. (1988), *Autonomous Navigation in Cross-Country Terrain*, in Proceedings of Image Understanding Workshop, Cambridge, MA, April, 1988.

Langer, D., Rosenblatt, J. &. Hebert, M. (1994), *A Behavior-Based System For Off-Road Navigation*, in IEEE Journal of Robotics and Automation, vol. 10, no. 6, pp. 776-782, December 1994; (also appears as *An Integrated System For Autonomous Off-Road Navigation*, in the Proceedings of the IEEE International Conference on Robotics and Automation, San Diego, May 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, March/April 1990.

Nilsson, N. (1980), *Principles of Artificial Intelligence*, Tioga Pub. Co., Palo Alto, Calif.

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

Payton, D. (1986), *An Architecture for Reflexive Autonomous Vehicle Control*, in IEEE International Conference on Robotics and Automation, San Francisco, CA, April 7-10, 1986, pp. 1838-1845.

Payton, D. (1990), *Internalized Plans: A Representation for Action Resources,* Robotics and Autonomous Systems, 6(1), 1990, pp. 89-103. (Also in Designing Autonomous Agents, ed. Pattie Maes, MIT Press, Cambridge, Mass. 1991, pp. 89-103.)

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

Payton, D., Keirsey, D., Kimble, D., Krozel, J. & Rosenblatt, J. (1993), *Do Whatever Works: A Robust Approach to Fault-Tolerant Autonomous Control*, Journal of Applied Intelligence, Volume 3, pp 226-250.

Pearl, J. (1988), *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers.

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

Rosenblatt, J. & Payton, D. (1989), *A Fine-Grained Alternative to the Subsumption Architecture for Mobile Robot Control*. in Proc. of the IEEE/INNS International Joint Conference on Neural Networks, Washington DC, vol. 2, pp. 317-324, June 1989 (also appears in 1989 AAAI Spring Symposium Series).

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

Shafer, G. (1976), *A Mathematical Theory of Evidence*, Princeton University Press.

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.

Stentz, A. (1993), *Optimal and Efficient Path Planning for Unknown and Dynamic Environments*, Carnegie-Mellon Technical Report CMU-RI-TR-93-20.

Stentz, A. & Hebert, M. (1994), *A Complete Navigation System for Goal Acquisition in Unknown Environments*, Carnegie-Mellon Technical Report CMU-RI-TR-94-7.

Thorpe, C., Amidi, O., Gowdy, J., Hebert, M. & Pomerleau, D. (1991), *Integrating Position Measurement and Image Understanding for Autonomous Vehicle Navigation*. Proc. Workshop on High Precision Navigation, Springer-Verlag Publisher.

Yen, J. & Pfluger, N. (1992), *A Fuzzy Logic Based Robot Navigation System*, AAAI Fall Symposium.

Zadeh, L. (1973), *Outline of a New Approach to the Analysis of Complex Systems and Decision Processes*, IEEE Transactions on Systems, Man and Cybernetics, Vol 3 No 1, January 1973