A Behavior-Based System For Off-Road Navigation

D. Langer, J. K. Rosenblatt, and M. Hebert

The Robotics Institute
Carnegie Mellon University
Pittsburgh PA 15213

Abstract

In this paper, we describe a core system for autonomous navigation in outdoor natural terrain. The system consists of three parts: a perception module which processes range images to identify untraversable regions of the terrain, a local map management module which maintains a representation of the environment in the vicinity of the vehicle, and a planning module which issues commands to the vehicle controller. Our approach is to use the concept of "early traversability evaluation," in which the perception module decides which parts of the terrain are traversable as soon as a new image is taken, and on the use of a behavior-based architecture for generating commands to drive the vehicle. We argue that our approach leads to a robust and efficient navigation system. We illustrate our approach by an experiment in which a vehicle travelled autonomously for one kilometer through unmapped cross-country terrain. The system used in this experiment can be viewed as a core navigation system in that other modules, such as a map navigation module, can be easily added to the system.

1 Introduction

Autonomous navigation missions through unmapped open terrain are critical in many applications of outdoor mobile robots. To successfully complete such missions, a mobile robot system needs to be equipped with reliable perception and navigation systems capable of sensing the environment, of building environment models, and of planning safe paths through the terrain. In that respect, autonomous cross-country navigation imposes two special challenges in the design of the perception system. First, the perception must be able to deal with very rugged terrain. This is in contrast to more conventional mobile robot systems which operate in simpler structured environments. Second, the perception system must be able to reliably process a large number of data sets over a long period of time. For example, even a relatively short navigation mission of a few kilometers may require processing thousands of images. Furthermore, because manual rescue of the vehicle from a serious failure is impossible, the perception system must be able to process a large number of images without any errors, or, at least, the system must be able to identify and correct for errors in time to avoid catastrophic failure of the vehicle. Although the basic computer vision and planning technologies exist and have been demonstrated in the laboratory, achieving such a level of reliability is still a significant challenge.

Several approaches have been proposed to address these problems. Autonomous traverse of rugged outdoor terrain has been demonstrated as part of the ALV [17] and UGV [16] projects. JPL's Robby used stereo vision [15] as the basis of its perception system and has been demonstrated over a 100 m traverse in outdoor terrain. Other efforts include: France's VAP project which is also based on stereo vision [6]; the MIT rovers which rely on simple sensing modalities [1]; and several Japanese efforts[11].

In this paper, we argue that relatively simple algorithms for obstacle detection and local map building are sufficient for cross-country navigation. Furthermore, when used in the context of a behavior-based architecture, these algorithms are capable of controlling the vehicle at significantly faster speeds than would be possible with a system that planned an optimal path through a detailed, high-resolution terrain map. Moreover, we argue that an accurate map is not necessary because the vehicle can safely traverse relatively large variations of terrain surface.

The underlying principle of this work was to keep things as uncomplicated as possible. We opted for a simple yet effective system rather than using more complex methods which often look good on paper yet produce problems in the field.

To illustrate our approach, we will describe a set of perception and navigation modules which constitute the core of a cross-country navigation system. The goal of this system is to enable the vehicle to travel through unmapped rugged terrain at moderate speeds, typically two to three meters per second. We arranged the system modules in a self-contained navigation system which we demonstrated on a one kilometer path through unmapped open terrain. In the next sections, we will use this result as the main reference to illustrate our approach and to discuss the system performance and the implementation details of each module.

The perception and navigation system was developed as part of the Unmanned Ground Vehicle (UGV) project. The support vehicle is a retrofitted HMMWV capable of cross-country navigation (Figure 1). The sensor is the Erim laser range finder which acquires 64x256 range images at 2 Hz. An estimate of vehicle position can be obtained at all times by combining readings from an INS system and from encoders.

Figure 2 shows a high level view of the architecture of the system. A perception module computes a list of untraversable regions and sends the region description to a local map management module. The local map module is responsible for gathering information from the perception module over time and for maintaining a consistent model of the terrain around the vehicle. (We will elaborate on the description of the perception and local map modules in Sections 2 and 3, respectively.) The description of the untraversable regions is sent by the local map module to a planning module at regular intervals. Untraversable regions are terrain features such as high slopes, ditches, or tall objects over which the vehicle cannot drive safely.

Based on a set of driving behaviors, the planning module generates arcs which steer the vehicle such that it remains clear of the untraversable regions. (We describe the planning module in detail in Section 4.) The three logical modules, perception, local map, and arc generation constitute the core of the system. Although it is divided into three logical units, the core system is implemented as a decentralized set of seven software modules. The software modules exchange information using the TCX communication system [5].

In order to be used in a real mission, this core system must be embedded in a larger navigation system so as to carry out a specific task. As will be explained in Section 4, the planning module is capable of arbitration between the steering directions generated by an external module and the steering directions generated by the core navigation system. For example, the external module can be a module that forces the vehicle to drive to a specific goal point or to follow a specific direction. We will show in Section 5 an example in which an additional module drives the vehicle through a set of intermediate goal points.

2 Perception

The range image processing module takes a single image as input and produces a list of regions which are untraversable. The initial stage of image filtering resolves the ambiguity due to the maximum range of the scanner, and removes outliers due to effects such as mixed pixels and reflections from specular surfaces (see [7] for a complete description of these effects). After image filtering, the (x,y,z) location of every pixel in the range image is computed in a coordinate system relative to the current vehicle position. The coordinate system is defined so that the z axis is vertical with respect to the ground plane, and the y axis is pointing in the direction of travel of the vehicle. It is convenient to center the coordinate at the point used as the origin for vehicle control, in this case between the two rear wheels, rather than at the origin of the sensor. The transformation takes into account the orientation of the vehicle read from an INS system. The points are then mapped into a discrete grid on the (x,y) plane. Each cell of the grid contains the list of the (x,y,z) coordinates of the points which fall within the bounds of the cell in x and y. The size of a cell in the current system is 20 cm in both x and y. The choice of the grid resolution is based on the angular resolution of the sensor, in this case 0.5^o, and on the size of terrain features which need to be detected, 20cm in the case of the HMMWV.

2.1. Terrain classification algorithm

The terrain classification as traversable or untraversable is first performed in every cell individually. The criteria used for the classification are:

- the height variation of the terrain within the cell,
- the orientation of the vector normal to the patch of terrain contained in the cell,
- and the presence of a discontinuity of elevation in the cell.

To avoid frequent erroneous classification, the first two criteria are evaluated only if the number of points in the cell is large enough. In practice, a minimum of five points per cell is used. Once individual cells are classified, they are grouped into regions and sent to the local map maintainer. It is necessary to use a slope criterion instead of a simple test on elevation for two reasons. First, the vehicle has limitations on the type of slopes on which it can drive independently of any elevation discontinuity. Second and most importantly, a test on absolute elevation would be very unreliable due to the potentially high variation of elevation from the near range to the far range of the field of view. Also, a small error in the estimation of vehicle pitch may induce a large error in the elevation at the far range of the field of view.

- For every range pixel $p = (\rho, row, col)$:
- Convert p to a 3-D point $P = [x\ y\ z]$ with respect to the vehicle position at the time the image was taken.
- Compute the location of the cell C which contains P.
- Add P to the list of points in C.

- For every non-empty cell *C*:
- Compute the elevation statistics: $h_{\min}$, $h_{\max}$, σ_h, and the slope *v* by doing a weighted least-squares estimation using the list of points in C.
- If $h_{\min}$ - $h_{\max}$ and *v* are outside of the acceptable bounds for the current vehicle configuration in terms of undercarriage and tipover constraints, classify the cell as untraversable.
- Send the list of untraversable cells to the local map manager along with the pose of the vehicle at the time the image was taken.

In general, the density of points in the map is not uniform. As a result, many cells of the map may end up being empty. A dense map without these gaps could be produced by first interpolating the map. This would be necessary in order to evaluate slope by using a neighborhood of each of the grid points. However, with the algorithm above, it is not necessary to interpolate the map because slope is evaluated at each cell individually without using its neighborhood. All that is required in order to compute the slope at a given cell is that enough data points fall in that cell. As result, the slopes cannot be evaluated at those cells of the map which have low or no data content. This is acceptable assuming that the data acquisition processing are fast enough compared to the speed of the vehicle so that the regions of the map with insufficient data can be covered in subsequent images. Although this solution relies on fast perception rate, it is in practice preferable to interpolating the map for two reasons. First, interpolation does increase the computation time substantially, thus increasing reaction time and map update time. Second, the interpolation may smooth out important local details of the terrain which are left untouched in our algorithm.

2.2. Example

Figure 3 shows an example of terrain classification at one position along the path of Figure 8. A video image of the terrain is shown in Figure 3 (a); the corresponding range image is shown in Figure 3 (b). In this example, a large part of the terrain on the left side of the vehicle is untraversable because of either high slope or high elevation.

Figure 3 (c) shows the grid built internally by the perception module. The (x,y) plane is the reference ground plane and *z* is vertical. The *z* value displayed at each element of the grid is the highest elevation of the set of data points within that cell. The scaling factors in (x,y) and in *z* are different so that the height of the hill on the left is exaggerated in this display. The elevation varies by approximately one meter across the terrain in this example.

Figure 3 (d) shows the result of the terrain classification. The points that are classified as obstacles because they are part of an untraversable region of the terrain are indicated by non-zero values, the rest of the terrain is set to zero. The cells in the region indicated by the label *A* are not classified as obstacles because the terrain slope within these cells is still within the bounds of the slopes on which the vehicle can travel. The cells in the regions indicated by the label *B* are not all classified as untraversable because the number of data points within each of these cells is not sufficient.

It is clear from this example that the portions of the grid that are not visible are not reported to the local map manager because only the untraversable cells are explicitly detected. This can occur for two reasons: insufficient density of data points at long range or occlusions from other parts of the terrain. The former occurs only at long range and the timing of the system is adjusted so that new scans of the same area are taken before the vehicle reaches that area. Specifically, assuming that a minimum of *n* points per cell is necessary with cells of size *l* meters on the side, a cell on a the ground will have a number of data points too small when it is a range greater than $R = \sqrt{(hl)/(n\theta)}$, where *h* is the height of the sensor above the ground and θ is the angular field of view of a single pixel. With l = 40cm, n = 3, and θ = 0.01, R is approximately 6.5m. At a speed of 3m/s, three scans of this cell will be taken before the vehicle reaches it, using the current image acquisition rate of 2Hz. This result corresponds to the worst case of the ground plane because cells on slanted surfaces have a higher density of points. This analysis shows that the cells with insufficient data are processed on time provided that vehicle speed is properly tuned to the sensing rate.

Cells with insufficient data due to occlusions occur because of the presence of obstacles. The arc generation module steers the vehicle away from the obstacles, and, just as before, the cells in an occluded area are scanned before the vehicle reaches this area. Based on this analysis and the fact that maintaining the traversable cells in the local would degrade performance appreciably, we decided not to represent explicitly the traversable regions.

2.3. Performance and limitations

The range image processing module is efficient because it does not build a dense, high-resolution map of the terrain, and because each image is processed individually without any terrain matching and merging. Specifically, the range processing algorithms run at 200 ms/image on Sparc II workstations.

Although the image acquisition time (500 ms) is effectively the main limitation because it is over twice the processing time, we have put the emphasis on the efficiency of the processing for two reasons. First, faster processing translates to lower latency between the time an object appears in the field of view of the range image and the time it is placed in the local map, irrespective of the image acquisition time. Second, the slow image acquisition is specific to the scanner that was available to us for the experiments reported in this paper. We believe that efficient processing is needed for the next generation of faster sensors with which we are experimenting.

In practice, terrain features of size 30cm are detected at a range of ten meters when vehicle speed is 2 m/s. By comparison, the maximum range of the scanner in its first ambiguity interval is 18 m with a separation of one meter between pixels at that range. More than the range resolution, the reason for the limited detection range is mainly the angular resolution of the scanner which limits the number of pixels measured on a given object.

Several problems can lead to occasional misclassification of the terrain. The first problem is the presence of terrain regions with poor reflectance characteristics, such as water. In practice, such points can be removed from the image during the initial filtering phase. However, the missing data creates large gaps in the map in which the terrain cannot be classified. This problem can really be solved only with the help of additional sensors suitable for terrain typing.

The second problem is the presence of vegetation in typical natural outdoor environments. This problem can manifest itself in several ways. Dense vegetation appears as an obstacle in the range image, causing the vehicle to come to a stop even when the underlying terrain is traversable. Sparse vegetation also causes the detection of spurious obstacles at short range from the sensor. Obviously there is no solution to this problem using the laser range finder alone. Additional sensing, such as millimeter wave radar, or a different type of range sensing, such as passive stereo may help.

3 .Local Map Management

The purpose of the local map module is to maintain a list of the untraversable cells in a region around the vehicle. In the current system, the local map module is a general purpose module called Ganesha [9]. Ganesha uses a 2-D grid-based representation of the local map. In this system, the active map extends from 0 to 20 meters in front of the vehicle and 10 meters on both sides. Each grid cell has a resolution of $0.4 \times 0.4\ m^2$ which was found to provide sufficient accuracy for navigation functions as it is small with respect to vehicle size and large with respect to sensor resolution. This module is general purpose in that it can take input from an arbitrary number of sensor modules and it does not use any explicit knowledge of the algorithms used in the sensor processing modules.

3.1. Overview

The core of Ganesha is a single loop shown in Figure 4. At the beginning of the loop, the current position of the vehicle is read and the coordinates of all the cells in the map with respect to the vehicle are recomputed. Cells that fall outside the bounds of the active region are discarded from the map. The next step in the loop is to get obstacle cells from the perception modules, and then to place them in the local map using the position of the vehicle at the time the sensor data was processed. The sensing position has to be used in this step because of the latency between the time a new image is taken, and the time the corresponding cells are received by the map module, typically on the order of 300ms.

After new cells are placed in the map, internal cell attributes are updated. Cell attributes include the number of times a cell has been previously observed and a flag that indicates whether it is inside the current field of view of the sensor. Finally, Ganesha sends the list of current obstacle cells in its map to the planning system. At the end of each loop, Ganesha waits before starting a new iteration in order to keep a constant loop cycle time.

3.2. Map scrolling

All the coordinates are maintained in Ganesha relative to the vehicle. In particular, vehicle position and orientation are kept constant in the map. At every iteration, Ganesha reads from the positioning system a position (x,y) and a heading φ. Assuming that the local map is currently expressed with respect to vehicle pose $P_1 = (x_1, y_1, \varphi_1)$ and the new vehicle pose read from the controller is $P_2 = (x_2,y_2,\varphi_2)$, Ganesha first computes the relative transformation $P = P_2 P_1^{-1}$ and then transform every cell (x^m_1, y^m_1) of the current map to its new location with respect to the vehicle pose P_2: $(x^m_2, y^m_2) = P\ (x^m_1, y^m_1)$.

It is important to note that Ganesha deals only with *relative* transformations from one vehicle position to another. Therefore, Ganesha relies on the relative accuracy of the vehicle positioning system, not on the absolute accuracy of the global positions. Assuming that the maximum error on the positioning system of k% of distance travelled under normal conditions.Assuming Ganesha maintains a L meter deep map, the maximum error in the position of a map object occurs when an object that was added when it was at a distance *L* from the vehicle is now about to disappear from the map after the vehicle has travelled a distance *L*. In that case, the error in the position of the cell is at most *kL* because it is the relative error between two vehicle positions separated by *L*. In our case, $k < 1\%$ and $L = 20$m, and the error is therefore 20cm which is below the 40cm resolution of the Ganesha grid. The estimate of *k* is based on experimental work with the vehicle controller as reported in [20]. Based on these numbers, the accuracy of the positioning system is sufficient to provide enough relative accuracy for the Ganesha map to be updated correctly.

The fact that the relative accuracy of the positioning system is sufficient for this resolution of the Ganesha grid and for our vehicle does not preclude the use of other sources of position information. For example, we could imagine using 3-D landmark or feature tracking in the range images in order to refine the position estimates. However, this type of visual positioning should be separate a process of which Ganesha uses the output in the form of periodical position updates. In particular, vehicle position from visual registration should not be part of this system.

Because the map module deals only with a small number of terrain cells instead of a complete model and the number of obstacle cells is small compared to the number of traversable cells, the map update is fast. In practice, the update rate can be as fast as 50 ms on a SparcII workstation. Because of the fast update rate, this approach is very effective in maintaining an up-to-date local map at all times. One last advantage of Ganesha's design is that the module does not need to know the details of the sensing part of the system because it uses only information from early terrain classification. In fact, the only sensor-specific information known to the map module is the sensor field of view which is used to check for consistency of terrain cells between images as described below.

3.3. Error correction

The perception module may produce occasional false positive detection because of noise in the image, for example, which would affect the path of the vehicle. This problem is solved by the map maintainer which maintains a history of the observations. Specifically, an untraversable map cell which is not consistent across images is discarded from the local map if it is not reported by the perception module as untraversable in the next overlapping images. Because the terrain classification is fast compared to the speed of the vehicle, many overlapping images are taken during a relatively short interval of distance travelled. As a result, an erroneous cell is deleted before the vehicle starts altering its path significantly to avoid it.

DECAY ALGORITHM

3.4. Field of view management

A severe limitation of any navigation system is the limited field of view of the sensor, in this case 80°. The problem with the limited field of view is that if the vehicle turns too sharply it may drive into a region that falls outside of the region covered by the previous images. Specifically, the minimum possible turning radius that would keep the entire vehicle within the area swept by the sensor's FOV is given by:

$$r_{\min} = w/2 + mc + c\sqrt{m^2 + 1}$$

With the current parameters of the current laser range finder, the minimum possible turning radius is then $r_{\min}$ ~ 17.25 m.

In order to address the field of view constraint, we introduced a new mechanism in Ganesha to explicitly represent fields of view. The idea is to add artificial obstacles in the map, the FOV obstacles, which represent the bounds on the sensor field of view at every recorded position as shown in Figure 5. The FOV obstacles are generated at the two outermost boundary points of the field of view polygon. Since adding FOV obstacles every time a new position is read might be too expensive and is not necessary, they are added at one meter intervals.

The FOV cells are first hidden which means that they do not affect the behavior of the vehicle under normal operating conditions. A hidden FOV obstacle occupies a map cell and is transformed with vehicle motion just like the regular obstacle cells, but is not sent to the planner and thus not considered an obstacle by the avoidance behavior. A hidden obstacle is converted to an active obstacle when it comes close to the vehicle. Currently this is done when the FOV obstacle is less than one meter from the front of the vehicle (Figure 5). Active FOV obstacles are treated just like any other regular obstacle. This mechanism ensures that the vehicle can still make sharp turns (Figure 5(a)), but will be prevented from steering into a path that falls into a locally unmapped area (Figure 5(b)).

Using FOV obstacles is a simple and effective way of dealing with the field of view constraint. It has the advantage that it has no effect on the behavior of the vehicle in normal operation and the increase in computation time in Ganesha is negligible as only a few artificial obstacles are used. Another advantage is that the FOV objects mechanism is completely transparent to the rest of the system. In particular, the planning system does not have any knowledge of the way the field of view is handled in Ganesha. As a result, different sensors with different fields of view may be used without any modification of the planner.

Because FOV obstacles increase the amount of computation in Ganesha, one could be tempted to maintain a representation of the traversable area rather than the untraversable area. In reality, this approach would degrade performance: Because the FOV obstacles are inserted every meter, there are on the order of $2L$ FOV cells in the Ganesha at any given time, where L = 20m is the depth of the grid. At the same time, there are on the order L^2/c^2 empty cells in the grid, where c = 40cm is the resolution of the grid. Therefore, even in cluttered terrain, it would always considerably less efficient to transform the traversable cells instead of the combination of untraversable cells and the FOV cells. The situation is worse in the case of a flat terrain in which the map update would be $L/2c^2$ = 66 times slower.

4 Planning

Once obstacles have been detected by the terrain evaluation modules and the local map has been updated by Ganesha, the next step is to use this map to generate commands that steer the vehicle around these obstacles. This is done within the framework of the Distributed Architecture for Mobile Navigation (DAMN).

DAMN is a behavior-based architecture similar in some regards to reactive systems such as the Subsumption Architecture [2]. In contrast to more traditional centralized AI planners that build a 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 [19]. 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 do maintain internal representations of the world (e.g. Ganesha's local map). Brooks has argued that "the world is its own best model" [3], 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 are visible to the vehicle sensors at all times. This adds unnecessary constraints and reduces the flexibility of the system.

Figure 6 shows the organization of the DAMN system in which individual behaviors such as road following or obstacle avoidance send preferred steering directions to the command arbitration module which combines these inputs into a single steering direction and speed command. We describe the use of DAMN in the context of the cross-country navigation system only and we refer the reader to [20] for a description of other systems built around DAMN in the context of road following.

4.1. The arbiter

The role of the architecture is to decide which behaviors should be controlling the vehicle at any given time. In the Subsumption Architecture, 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. 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 [18]. An arbiter then performs *command fusion* to select the most appropriate action.

More precisely, each behavior generates a vote between -1 and +1 for every possible steering command. A vote of -1 indicates that the behavior recommends that the vehicle should not execute the command, because the vehicle would encounter an obstacle, for example. A vote of +1 indicates that the behavior has no objection to that steering command, because there is no obstacle nearby. The votes generated by the behavior are only recommendations to the arbiter. The arbiter computes for each steering command a linear combination of the votes from all the behaviors. The coefficients of the sum reflect the relative priorities of the behaviors. The steering command with the highest vote is send to the vehicle controller.

In the current implementation, the set of steering commands is the set of 15 arcs shown in Figure 7. Although other combinations of arcs can be used, we limit ourselves to the configuration which we used in the cross-country navigation experiments at the time of this writing.

The arbiter computes the turn command to be sent to the controller as follows:

1. Compute the weighted sum of the votes received from each active behavior (each behavior has an associated weight between 0 and 1)

2. Normalize the votes by dividing by the sum of the weights for all active behaviors; thus the normalized weighted sums lie between -1 and +1, as each behavior is constrained to vote within that range.

3. The arc with the maximum vote is found and is used as the command. The three cases are:

- If there is a single arc with the maximum value, then its value is used.
- If there is a series of consecutive turn choices with the maximum value, then the average of the curvatures is used (curvature is the inverse of turn radius).
- If there exist multiple non-consecutive arcs with the same maximum value, then the larger series of arcs is used. If there are multiple series of arcs with the same maximum value and of the same size, then one is chosen arbitrarily.

In addition to steering, speed is also controlled by the arbiter. The commanded speed is decided based on the commanded turn radius. A maximum speed is set for the vehicle, and this value is simply multiplied by the normalized weighted sum of the votes for the chosen turn radius; the result is the speed command issued.

The voting and arbitration scheme described above bears some obvious similarities to Fuzzy Logic systems [22] [10]. Fuzzy Logic is typically used within the framework of rule-based systems, as in [12], but behavior-based systems are generally procedural in nature and do not readily lend themselves to the use of if-then rules. In [21], an architecture is described which proposes to extend DAMN by using Fuzzy Logic, but in doing so it restricts each behavior to having a uniquely determined desired steering direction which is then voted for using a fuzzy membership set. The scheme described here is more general in that it allows for an arbitrary distribution of votes. For example, the obstacle avoidance behavior described below independently evaluates each of the proposed steering directions, whereas forcing it to choose, albeit fuzzily, a single heading would necessarily restrict the overall decision-making process. One advantage of using fuzzy sets in lieu of the discrete sets used in DAMN is that the output is continuous, thus yielding smoother control. Current work on DAMN will provide command interpolation, which is analogous to defuzzification, but using assumptions on the nature of the votes received that are more reasonable for this domain.

4.2. The obstacle behavior

Within the framework of DAMN, behaviors that provide the task-specific knowledge for controlling the vehicle must be defined. 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.

The most important behavior for 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 arcs.

If a trajectory is completely free of any neighboring obstacles, 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. 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, although the evaluation is not as unfavorable as for those trajectories leading to a direct collision.

More precisely, the obstacle avoidance behavior uses the following algorithm: If there is no obstacle along or near an arc for at least $L_{\max}$ meters, then the vote for that turn radius is +1. If an obstacle appears

$L_{\max}$ meters away, then the vote becomes 0. If an obstacle on the path is $L_{\min}$ meters away or less, then the vote is -1. For obstacles of an intermediate distance, the value is linearly interpolated between 0 and -1. The voting algorithm in the obstacle avoidance behavior is summarized below:

- for each turn radius choice:
    - if no obstacle lies along the arc (or near it, see below), then set the vote to be 1.0
    - if an obstacle lies at a distance greater than $L_{\max}$, then set the vote to be 1.0
    - if an obstacle lies along the arc at a distance less than $L_{\min}$, then set the vote to be -1.0
    - otherwise, set the vote to be: $-1.0 + (L_{obs} - L_{min}) / (L_{max} - L_{min})$, where L_{obs} is the distance between the obstacle and the current vehicle position.

This algorithm will vote against any turn radii that lead to a direct collision with a detected obstacle. However, it is also desirable to avoid coming unnecessarily close to an obstacle. For this reason, if an arc misses an obstacle, then the distance L_{miss} by which it misses is used to set the vote as follows:

- compute the vote as if a collision were imminent, using the obstacle distance in the algorithm described above
- add to this value the quantity $k_{\text{miss}} \cdot L_{\text{miss}}$, and take the smaller of that sum and 1.0

The set of values used in the experiments reported in this paper is: $L_{\min} = 5.0$, $L_{\max} = 20.0$, and $k_{\text{miss}} = 0.5$.

4.3. Additional behaviors

Once the ability of the vehicle to avoid collisions is ensured, it is desirable to provide the vehicle with the ability to reach certain destinations; the *Goal Seeking* behavior provides this capability. This fairly simple behavior uses pure pursuit to direct the vehicle toward a series of user-specified goal points. The outcome of the pure pursuit algorithm is a desired turn radius; this 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.

Various other auxiliary behaviors that do not achieve a particular task but issue votes for secondary considerations may also be used. 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.

5 Experimental Results

In order to illustrate and evaluate the performance of the system, we conducted a series of runs of the navigation system on our testbed vehicle in natural terrain. We now discuss those experiments in the next Sections. Starting with a description of a one kilometer run, we then analyze the operation of the arbiter and of the behaviors on data collected in a real run and we conclude with an discussion of the limitations of the system and of its failure modes.

5.1. System example

Figure 8 and Figure 9 show a typical run of the perception and navigation system. Figure 8 (a) shows the environment in which this experiment takes place. The terrain includes hills, rocks, and ditches. The white line superimposed on the image of the terrain shows the approximate path of the vehicle through this environment. The path was drawn manually for illustrative purpose. Figure 8 (b) shows the actual path recorded during the experiment projected on the average ground plane. In addition to the path, Figure 8 (b) shows the obstacle regions as black dots and the intermediate goal points as small circles.

In this example, the vehicle completed a one kilometer loop without manual intervention at an average speed of 2 m/s. The input to the system was a set of 10 waypoints separated by about one hundred meters on average. Except for the waypoints, the system does not have any previous knowledge of the terrain. Local navigation is performed by computing steering directions based on the locations of untraversable regions in the terrain found in the range images. An estimated 800 images were processed during this particular run.

Figure 9 (a) shows a close-up view of one section of the loop of Figure 9. The black lines show the approximate paths followed by the vehicle in this section. Figure 9 (b) shows the elevation map obtained by pasting together the images taken along the path. The grey polygons are the projections of the fields of view on the ground, the curved grey line is the path of the vehicle on the ground, and the grey dots indicate locations at which images were taken. In this case, the images are separated by approximately two meters.

Finally, Figure 9 (c) shows the local map which is maintained at all time around the vehicle. The squares corresponds to 40x40 cm patches of terrain classified as untraversable regions or obstacles. These local maps are computed from the positions shown in Figure 9 (a) and Figure 9 (b) by the white arrows.

In this experiment, the core system is configured with two behaviors: the obstacle avoidance behavior and the goal seeking behavior. The obstacle avoidance behavior receives a new description of the local map from Ganesha every 100ms. The arbiter combines the votes from the two behaviors and issues a new driving command every 100ms. The goal points are on average 100 meters apart and the goal seeking behavior switches goals whenever it comes within eight meters of the current target goal point. The weights are 0.8 for the obstacle avoidance behavior and 0.2 for the seek goal behavior.

5.2. Arbiter performance

In order to analyze the performance of the arbiter more closely, we recorded additional data on a simple path around a single obstacle. The result of this experiment is shown in Figure 10 and Figure 11.

The lower part of Figure 10 shows the path of the vehicle, shown as a black line, around a set of obstacle points, shown as black dots. In this example, the vehicle was controlled by the steering commands issued by the arbiter. In this experiment, the arbiter was receiving votes from two behaviors, the obstacle avoidance behavior described above, and a heading behavior which forces the vehicle to follow a constant heading.

Figure 11 shows the distribution of votes at five points of interest along the path, indicated by capital letters in Figure 10. Each graph depicts the votes issued for each turn choice by the obstacle avoidance and heading behaviors, as well as the weighted sum of these votes as computed by the arbiter. The horizontal axis of each graph shows the possible turn radius choices encoded from -8 m to +8 m.

At point *A*, the obstacle is first reported and the obstacle avoidance behavior generates high votes for the left turns to go around the obstacle and inhibits the right turns with negative votes, as shown by the solid line in the graph. At the same time, the heading behavior's vote distribution is relatively flat around the straight direction since the vehicle is currently headed in the desired direction; this is shown by the dashed line in the graph. Because of the small relative weight of the heading behavior, the combined vote distribution in the arbiter, shown as a thicker solid line in the graph, is dominated by the votes received from the obstacle avoidance behavior; a left turn is therefore commanded.

At point *B*, the obstacle is still close to the vehicle and the votes distributions are similar to the ones at *A*, thus maintaining the vehicle to the left of the obstacle. At point *C*, the obstacle avoidance behavior is still voting in favor of a sharp left turn, but the votes for the softer left turns is now not as low as it was at *A* or *B*, since the vehicle is now clearing the obstacles. At the same time, the heading behavior is starting to shift its votes towards turning right in order to bring the vehicle back to the target heading. The summed votes are still at a maximum for a left turn because of the greater weight of the obstacle avoidance behavior's votes, and so the arbiter continues to steer the vehicle well clear of the obstacles.

By the time the vehicle has reached point *D*, it is just passing the obstacles, so that the obstacle avoidance behavior is now only disallowing extreme right or left turns because of the field of view constraints. The heading behavior is now able to have more influence over the chosen turn direction, and a right turn is now executed so that the desired vehicle heading is restored. Note that between points *C* and *D*, there is a transition period where the turn radius chosen by the arbiter is neither the hard left favored by the obstacle avoidance behavior nor the medium right favored by the heading behavior. Instead, as turns other than hard left become acceptable to the obstacle avoidance behavior, and as the distribution of votes from the heading behavior shift further to the right, the vehicle is commanded to make ever softer left turns, and eventually turns to the right.

As can be seen in the graph of the commanded turn radius in Figure 10, the arbiter exhibits the desired behavior. Namely, rather than abruptly switching modes from avoiding obstacles to following a heading, the arbiter generates a smooth and steady transition as the situation gradually changes.

5.3. Limitations and failure modes

A fundamental limitation of the system is its inability of dealing with dead-ends such as a closed corridor with a depth greater than the field of view of the sensor. Because the system deals only with a local representation of the terrain, it is incapable of dealing with this type of situations of a more global nature. In particular, in the experiment of 5.2 it is clear that the vehicle can easily run into a dead-end situation by using the simple goal-seeking behavior, which is why multiple intermediate goal points are used rather than a signle global goal. If fact, this is by far the most common failure mode of the system. This limitation can be addressed by adding to the system another behavior which uses knowledge about the global representation of the terrain in order to plan appropriate vehicle paths. In this approach, the perception, local map, and obstacle avoidance modules are responsible for local navigation, while the other modules are responsible for global navigation.

As mentioned earlier, the speed of the vehicle is in large part limited by the maximum range of the sensor and by the image acquisition rate which introduces a minimum 500ms latency between the time an object enters the field of view of the sensor and the time the corresponding image is fully acquired. The effective steering radius of the vehicle is limited by the 80^o horizontal field of view of the sensor. The detection range is limited by the angular resolution of the scanner. These three limitations can be addressed only by using other sensors. We investigating the use of fast single-scan laser range finders and of passive stereo.

Because the system is implemented as a distributed system communicating through the standard Unix network facilities, there are no guarantees on the delays involved in sending messages between the modules. In practice, these delays are compounded into a significant latency between the time an object appears in the field of view of the sensor and the time a steering command is actually executed by the controller. Although the latency varies, we have observed latency as high as one or two seconds. This problem is being addressed by combining the modules in a way that is more suitable to predictable scheduling, and by porting the system to a real-time operating system.

6 Conclusion

We have presented a navigation system based on early evaluation of terrain traversability. We have demonstrated this system in a one kilometer traverse of unmapped cross-country terrain. This experiment demonstrates: the robustness of the approach; its ability to accommodate additional driving behaviors in addition to the basic obstacle avoidance behaviors, such as driving toward intermediate goal points; and its ability to compensate for constraints imposed by the sensing system, such as the limited field of view of the sensor. This is achieved by: reducing the amount of computation required by the perception system; simplifying local map management and path planning; hiding the details of sensing from all the modules except perception; and avoiding the problems caused by merging multiple terrain maps using inaccurate position estimates.

The drawback of this approach is that an error in the perception system may propagate unchallenged through the system because only the perception module has access to the sensor data. This problem is addressed by using a fast reactive path planner and a simple perception algorithm with fast cycle time relative to vehicle speed, both of which allow the system to correct quickly for occasional perception errors, and by incorporating a history mechanism in the local map manager which eliminates false positive detection.

The main limitations of the system are the limited range and speed of the sensor, the non-real-time nature of the system which is implemented on conventional Unix workstations using standard networking protocol, and the poor performance of perception on certain types of environments, such as vegetation. To address the first limitation, we have conducted preliminary experiments with a passive stereo system, and we are planning to experiment with a fast, single-line range sensor. These new sensor modalities will improve both acquisition rate and maximum range. To address the second limitation, we are planning on porting the critical parts of the system to a real time operating system and to combine parts of the system, such as perception and local map management into a single module with better control over scheduling. We feel that the third class of problems, due to poor reflectivity of surfaces and vegetation, can be addressed only by adding other specialized sensors.

In addition to research on the cross-country navigation system proper, we are continuing the development of the DAMN architecture. In particular, we are developing new approaches to speed which involve a separate speed arbiter and we are investigating ways for the arbiter to interpolate between the reference set of arcs based on the vote distributions from the behaviors. These two improvements will lead to smoother control in both speed and steering.

Acknowledgments

The UGV project is 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. Julio Rosenblatt is supported by a Hughes Research Fellowship. The authors wish to thank Jay Gowdy, Bill Ross, Jim Moody, Jim Frazier, George Mueller, and Mike Blackwell for their help in the cross-country navigation experiments. Many thanks to Dave Payton for the advice and support he has provided over the years in the development of the planning system.

References

[1]R. Brooks and A. Flynn. Fast, Cheap, and Out of Control: A Robot Invasion of the Solar System. *J. British Interplanetary Society*, 42(10):478-485, 1989.

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

[3]R. Brooks. Intelligence Without Reason. Proc. IJCAI'93. 1993.

[4]B. Brummit, R. Coulter, A. Stentz. Dynamic Trajectory Planning for a Cross-Country Navigator. In *Proc. of the SPIE Conference on Mobile Robots*, 1992.

[5]C. Fedor. TCX, Task Communications User's Manual. Internal Report, The Robotics Institute, Carnegie Mellon, 1993.

[6]G. Giralt and L. Boissier. The French Planetary Rover VAP: Concept and Current Developments. In *Proc. IEEE Intl. Workshop on Intelligent Robots and Systems*, pp. 1391-1398, Raleigh, 1992.

[7]M. Hebert, E. Krotkov. 3D Measurements from Imaging Laser Radars. *Image and Vision Computing 10(3)*, April 1992.

[8]Keirsey, D.M., Payton, D.W. and Rosenblatt, J.K., "Autonomous Navigation in Cross-Country Terrain," in Image Understanding Workshop, Cambridge, MA, April, 1988.

[9]D. Langer and C. Thorpe. Sonar based Outdoor Vehicle Navigation and Collision Avoidance. In *Proc. IROS '92*. 1992.

[10]C. Lee, 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

[11]T. Iwata and I. Nakatani. Overviews of the Japanese Activities on Planetary Rovers. In *Proc. 43rd Congress Intl. Astronautical Federation*, Washington, D.C., 1992.

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

[13]A. Kelly, T. Stentz, M. Hebert. Terrain Map Building for Fast Navigat ion on Rugged Outdoor Terrain. In *Proc. of the SPIE Conference on Mobile Robots*, 1992.

[14]I.S. Kweon. *Modeling Rugged Terrain by Mobile Robots with Multiple Sensors*. Ph.D. thesis, Robotics Institute, Carnegie Mellon University, January, 1991.

[15]L. H. Matthies. Stereo Vision for Planetary Rovers: Stochastic Modeling to Near Real-Time Imple-

mentation. *International Journal of Computer Vision*, 8:1, 1992.

[16]E. Mettala. Reconnaissance, Surveillance and Target Acquisition Research for the Unmanned Ground Vehicle Program. In *Proc. Image Understanding Workshop*, Washington, D.C., 1993.

[17]K. Olin, D.Y. Tseng. “Autonomous Cross-Country Navigation”. *IEEE Expert*, 6(4), August 1991.

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

[19]J.K. Rosenblatt and D.W. Payton. 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.

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

[21]J. Yen, N. Pfluger, A Fuzzy Logic Based Robot Navigation System, AAAI Fall Symposium, 1992.

[22]Zadeh, Lotfi A., “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

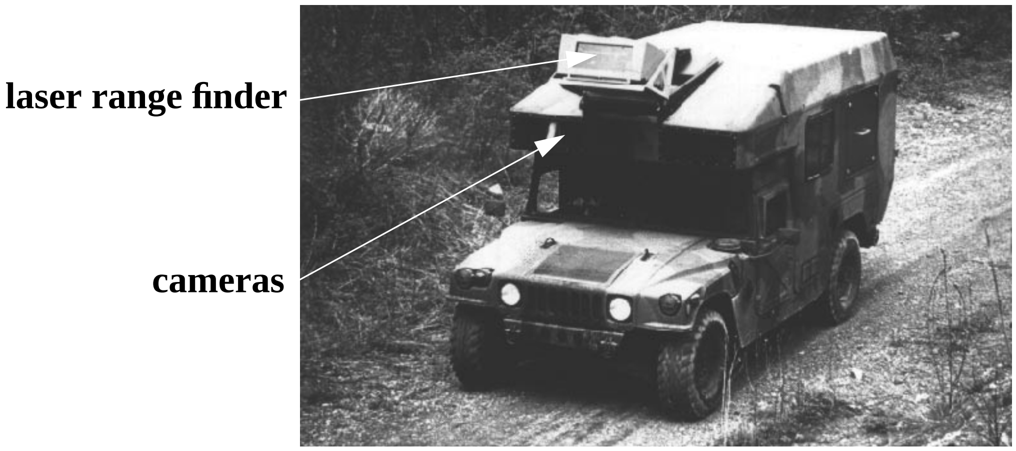


Figure 1: The testbed vehicle

.

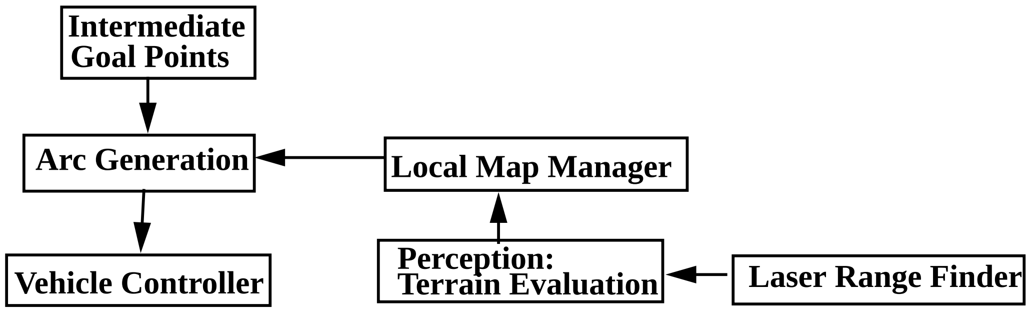
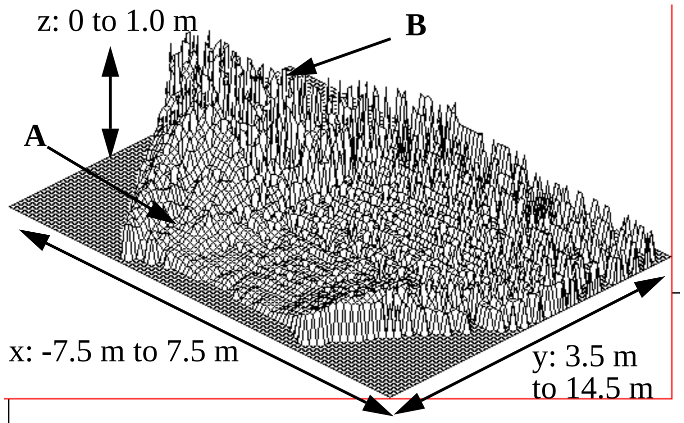
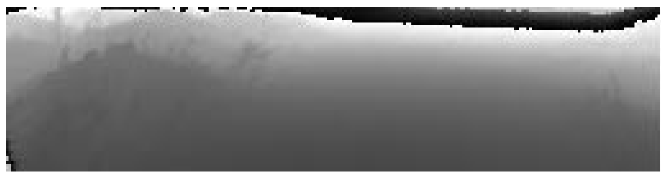


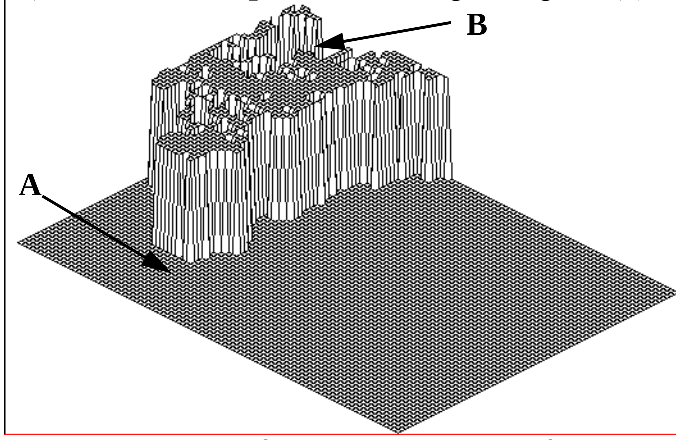
Figure 2: Architecture of the perception and navigation system

(a) A section of terrain from the path of Figure 2.



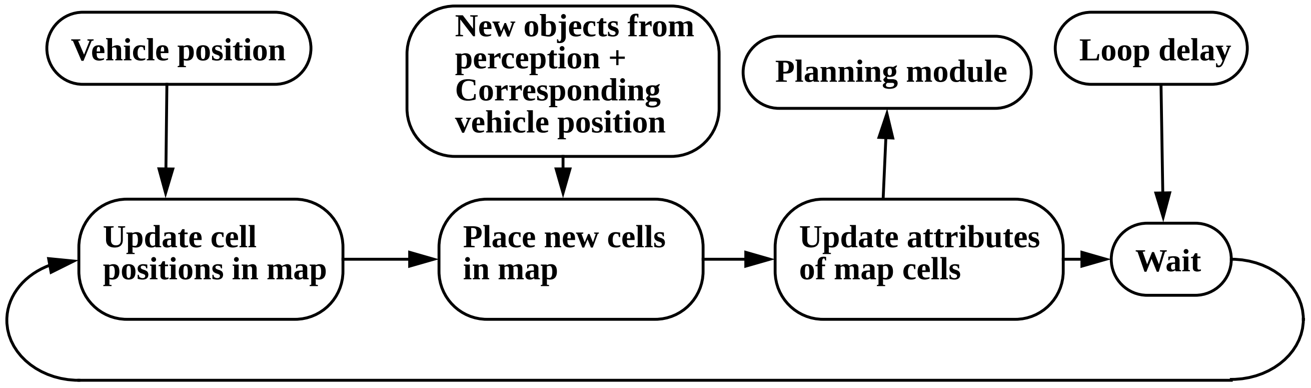
(c) Elevation map from the range image of (a).


(b) Range image of the terrain shown in (a).



(d) Terrain classification on the map of (c); non-zero values indicate obstacle regions.

Figure 3: Example of terrain classification


Vehicle position
New objects from perception + Corresponding vehicle position
Planning module
Loop delay
Update cell positions in map
Place new cells in map
Update attributes of map cells
Wait
Figure 4: Main loop in Ganesha

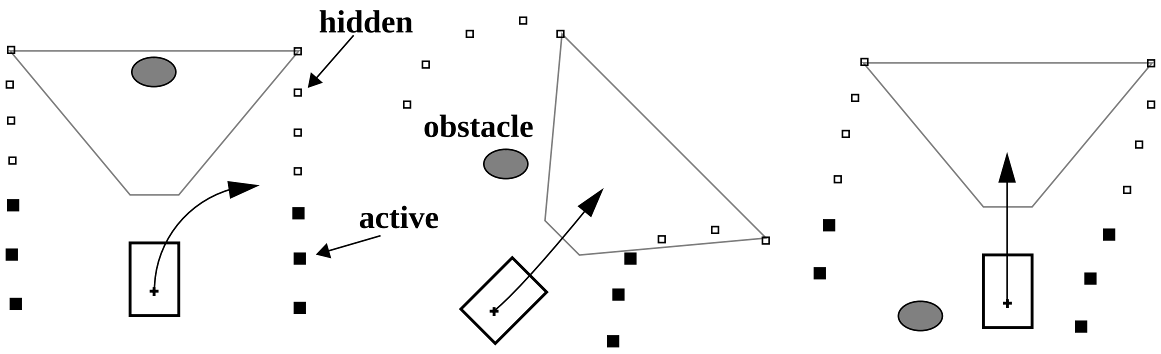


Figure 5: Use of FOV obstacles to limit turning radius

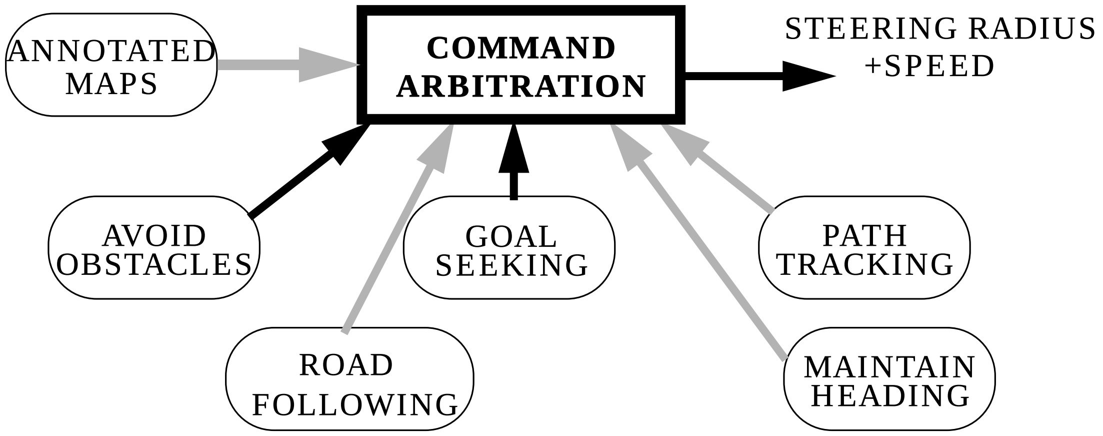


Figure 6: Behavior-based architecture; solid lines indicate the behaviors used in the cross-country system

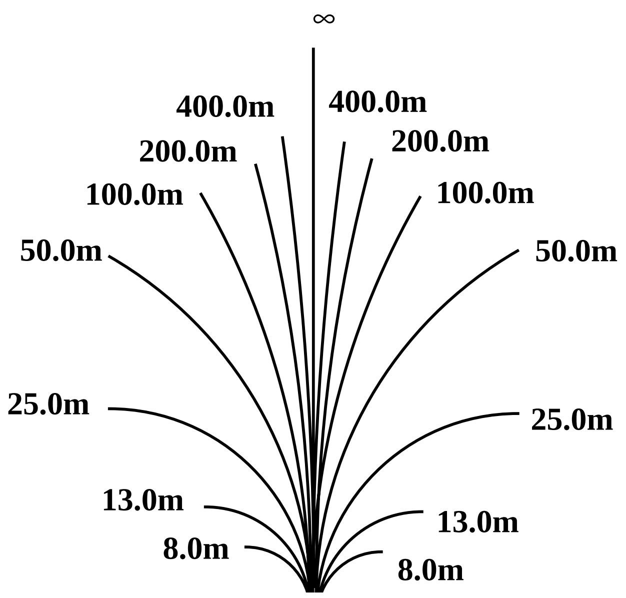
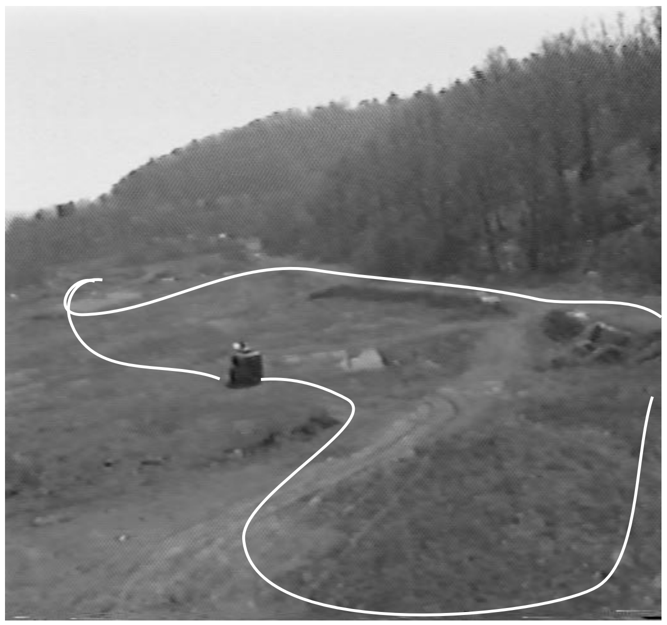
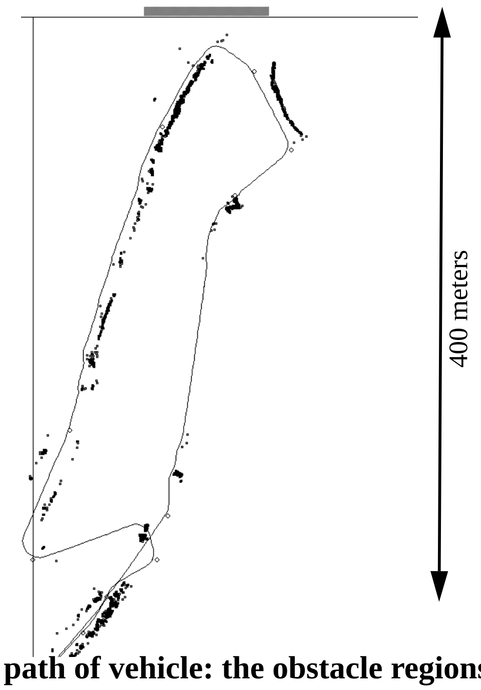


Figure 7: Set of possible turning radii in DAMN for the cross-country navigation system



(a) Camera view of terrain with approximate path superimposed.



(b) Exact path of vehicle: the obstacle regions are shown as black dots; the intermediate goal points are shown as small circles.

Figure 8: A loop through natural terrain

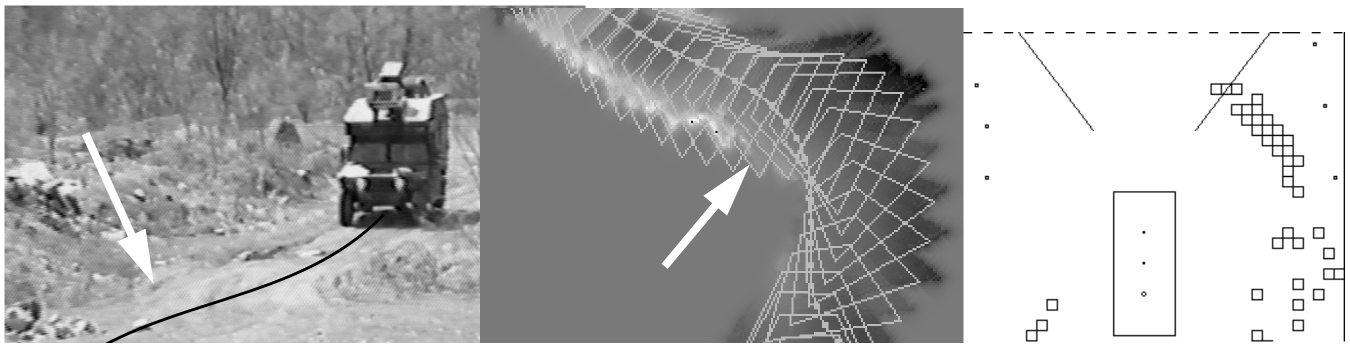


Figure 9: Detailed view of one section of the loop of Figure 8; (a) Approximate local path of the vehicle; (b) Overhead view of actual path of vehicle overlaid on terrain map; (c) Local map around the vehicle at the location indicated by the arrows in (a) and (b).

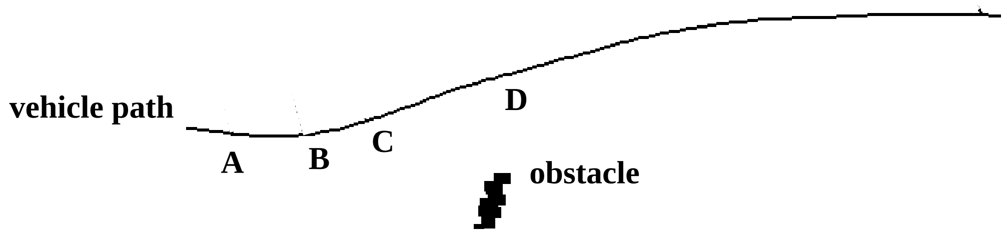


Figure 10: Distribution of votes over time in the arbiter and in the obstacle avoidance and heading behaviors; the distribution of votes are shown in Figure 11

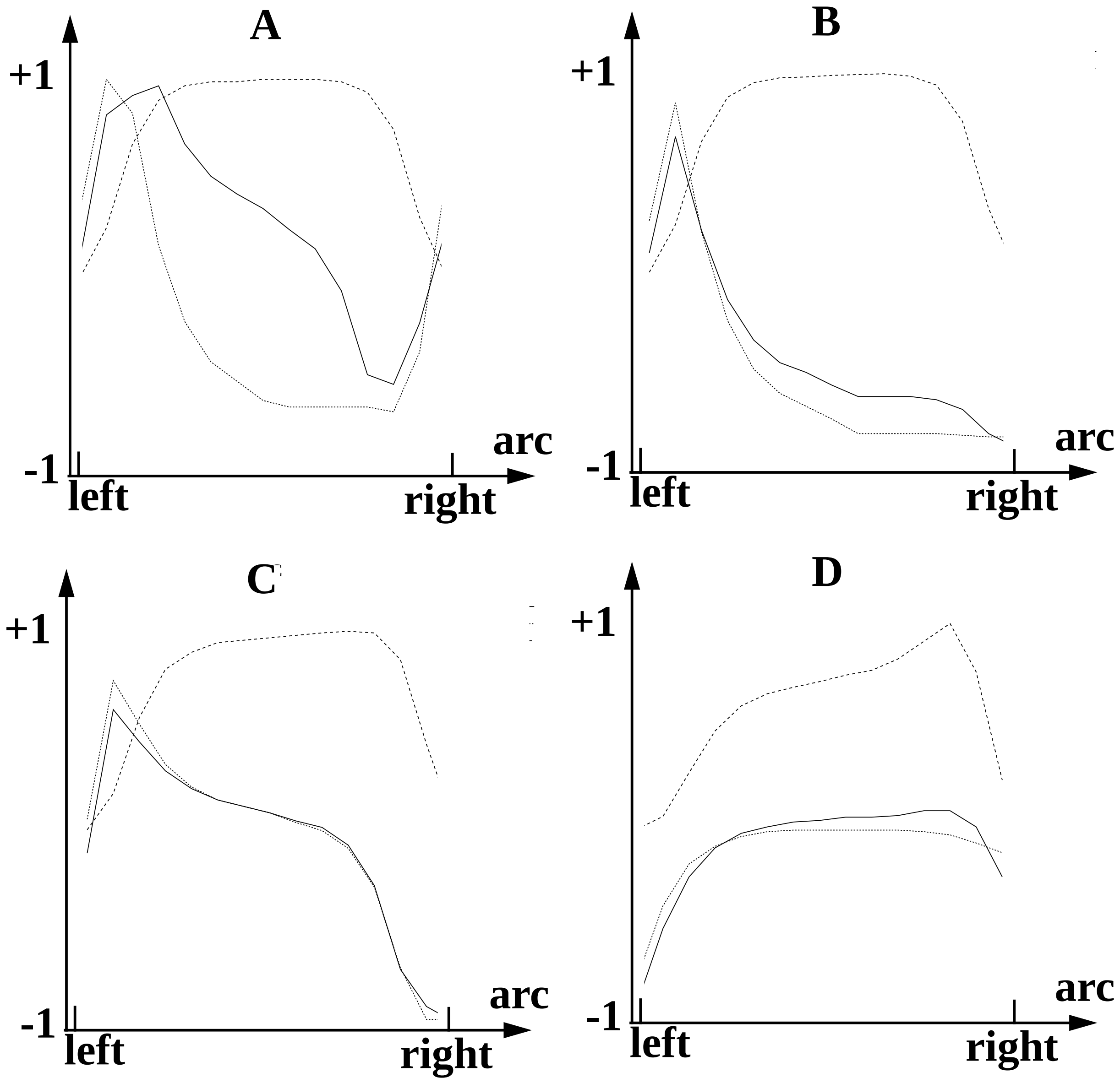


Figure 11: Distribution of votes in the arbiter, obstacle avoidance and heading behaviors at the points marked in Figure 10