A Reactive 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 reactive planning 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 [16] and UGV [15] projects. JPL's Robby used stereo vision [14] 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 [5]; the MIT rovers which rely on simple sensing modalities [1]; and several Japanese efforts[10]. Most of these perception systems use range images, from active ranging or passive stereo, and build a map of the terrain around or in front of the vehicle. The planning systems use the maps to generate safe trajectories. The approaches used in the existing planning systems range from purely reactive to fully proactive, depending on the type of maps. The main questions in building such systems are: What should be in the map, and when should the map be computed? On the one hand, the map could be a simple binary traversability map in which terrain regions are labelled as either traversable or untraversable. We call this first approach "early traversability evaluation" because the decision as to whether the terrain can be safely traversed or not is made early by the perception system and cannot be revisited later. On the other hand, the map could be a detailed elevation map of the terrain, including as much information as possible given a set of sensor data. In this second approach, all the sensing data is kept intact in the navigation system and can be accessed by any module. In this case, decisions about terrain traversability can be made in the higher level planning modules and can be revisited at any time.

In this paper, we argue that relatively simple methods of obstacle detection and local map building are sufficient for cross-country navigation. Furthermore, when used as input to a reflexive planner, the vehicle is capable of safely traveling 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. For these reasons, we propose an approach based on "early evaluation of traversability" in which the output of the perception system is a set of untraversable terrain regions used by a planning module to drive the vehicle. The system relies on "early evaluation" because the perception module classifies regions of the terrain as traversable or untraversable as soon as a new image is taken. This fast classification has the advantage of reducing the amount of data passed to the planner for path generation and reducing the amount of computation needed in later stages of planning. As we will show, early traversability evaluation allows for a more reactive approach to planning in which steering directions and speed updates are generated rapidly and in which the vehicle can respond to dangerous situations in a more robust and more timely manner. This approach is in contrast to other systems which build a detailed description of the terrain in the form, for example, of a high-resolution elevation map, and defer the traversability analysis to the planning module.

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 recently 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.

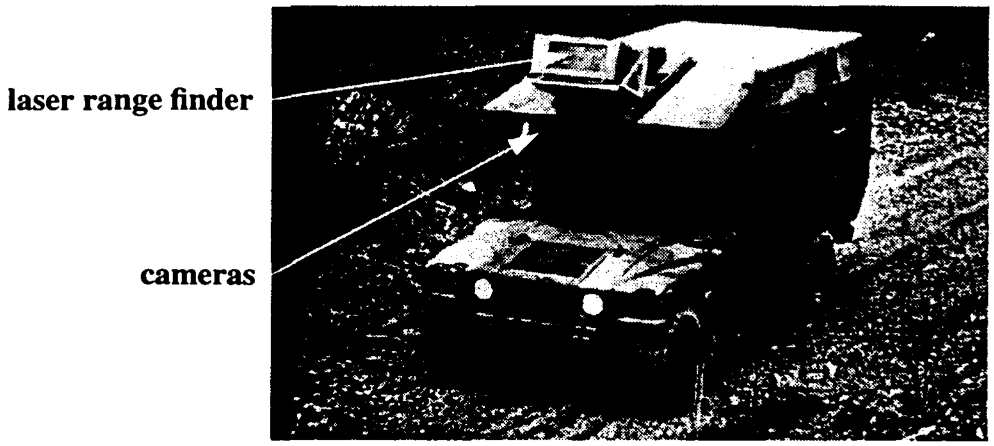


Figure 1: The testbed vehicle

Figure 2 shows a high level view of the architecture of the system. A perception module computes a list of regions that are untraversable 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 3 and 4, 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 5.) 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 EDDIE and TCX communication systems. (We refer the reader to [6] and [4] for a technical description of the communication systems.)

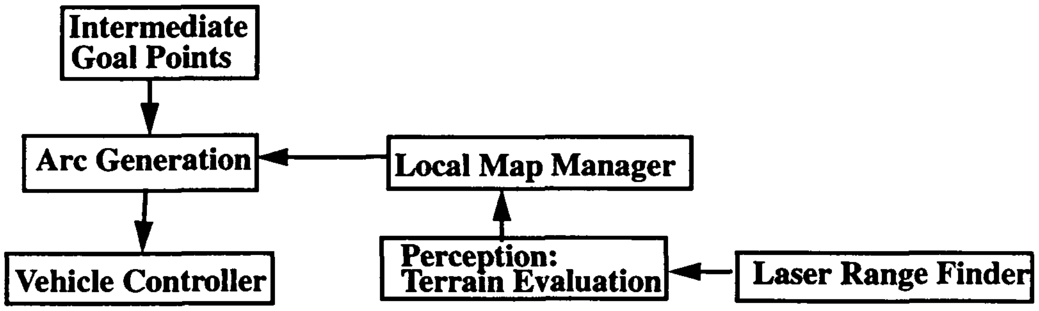
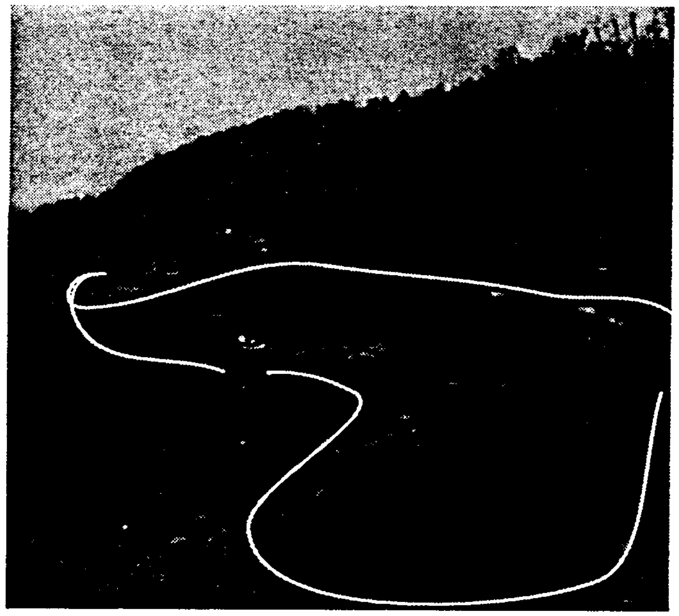


Figure 2: Architecture of the perception and navigation system

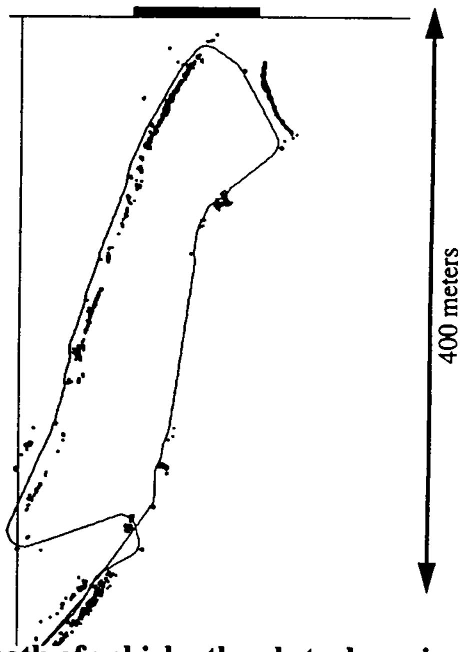
In order to be used in a real mission, this core system must be embedded in a larger navigation system so as to carry out a specific task. As we shall see in Section 5, the core system is embedded into a larger system by generating additional sources of steering commands. As will be explained in Section 5, 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 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 2 an example in which an additional module drives the vehicle through a set of intermediate goal points.

2 System Performance: An Example

Figure 3 and Figure 4 show a typical run of the perception and navigation system. Figure 3 (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 3 (b) shows the actual path recorded during the experiment projected on the average ground plane. In addition to the path, Figure 3 (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 4 (a) shows close-ups of three sections of the loop of Figure 4. The black lines show the approximate paths followed by the vehicle in these three sections. Figure 4 (b) shows the elevation map obtained by pasting together the images taken along the paths. In each figure, 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. The paths shown in Figure 4 (b) are the actual paths followed by the vehicle. It is important to note that these maps are included for display purposes only and that the combined elevation maps are not actually used in the system. Finally, Figure 4 (c) shows three views of 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 4 (a) and Figure 4 (b) by the white arrows. The trajectories are planned using this compact representation.

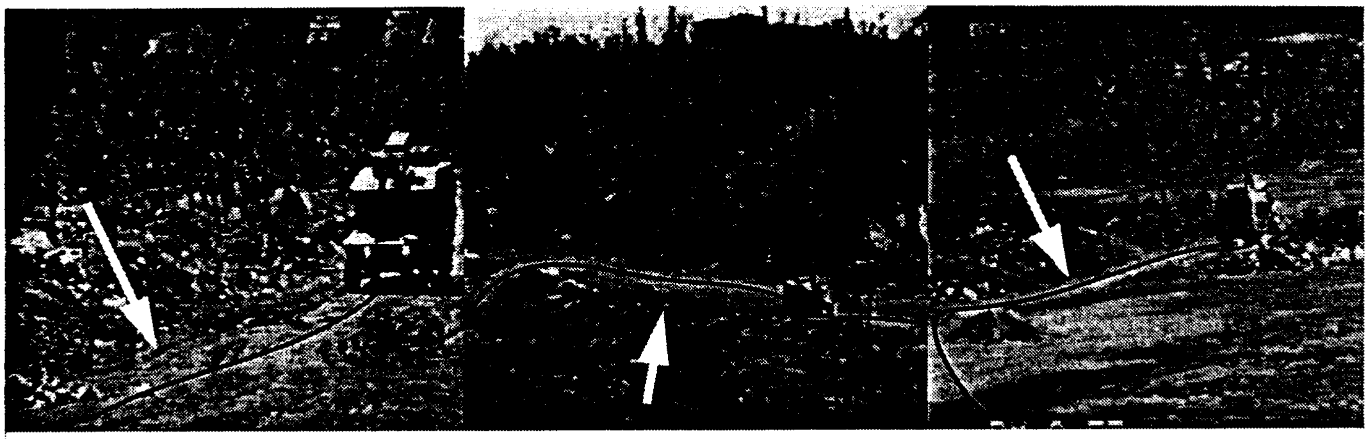


(a) Camera view of terrain with approximate path superimposed.

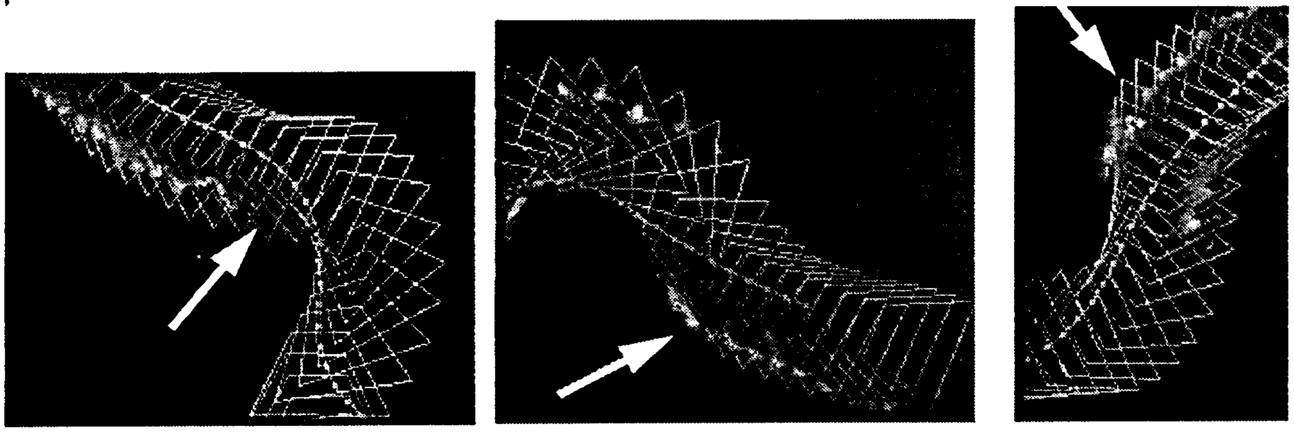


(b) Exact path of vehicle: the obstacle regions are shown as black dots; the intermediate goal points are shown as small circles.

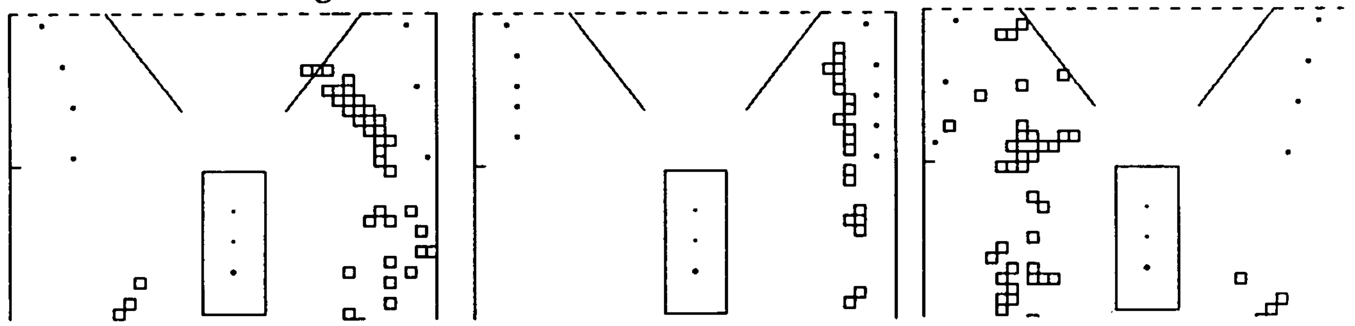
Figure 3: A loop through natural terrain



(a) Local path of vehicle in three sections of the loop of Figure 3. The arrows indicate the locations at which the local maps are displayed in (c) below.



(b) Display of the sensed terrain as elevation maps for the sections shown in (a). The polygons indicate the projection of the field of view of the sensor on the ground. The grey line shows the path followed by the vehicle in this section. The grey dots show the positions at which the images were taken. The arrows are placed at the same locations as in (a). This elevation is for display purposes; it is not used in the navigation.



(c) Display of the local traversability map at the locations marked by arrows in (a) and (b). Only the portion of the map in the immediate vicinity of the vehicle is displayed here. The vehicle is depicted by a rectangle. The untraversable regions are shown as squares.

Figure 4: Detailed views of three sections of the loop of Figure 3

3 Perception

The range image processing module takes a single image as input and outputs a list of regions which are untraversable (Figure 5). 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 grid resolution depends 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. 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.

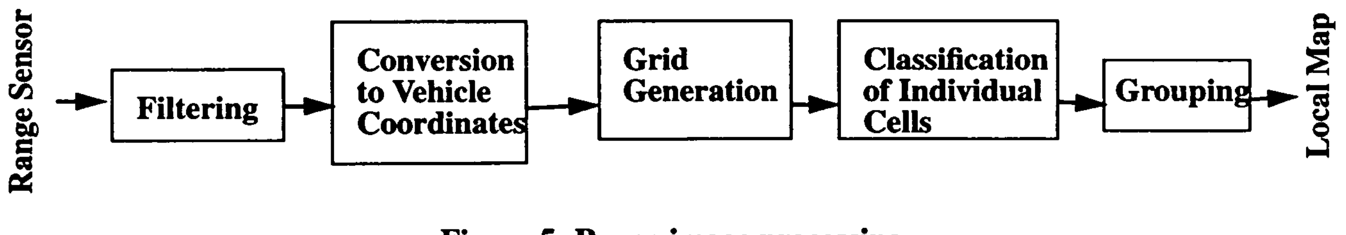
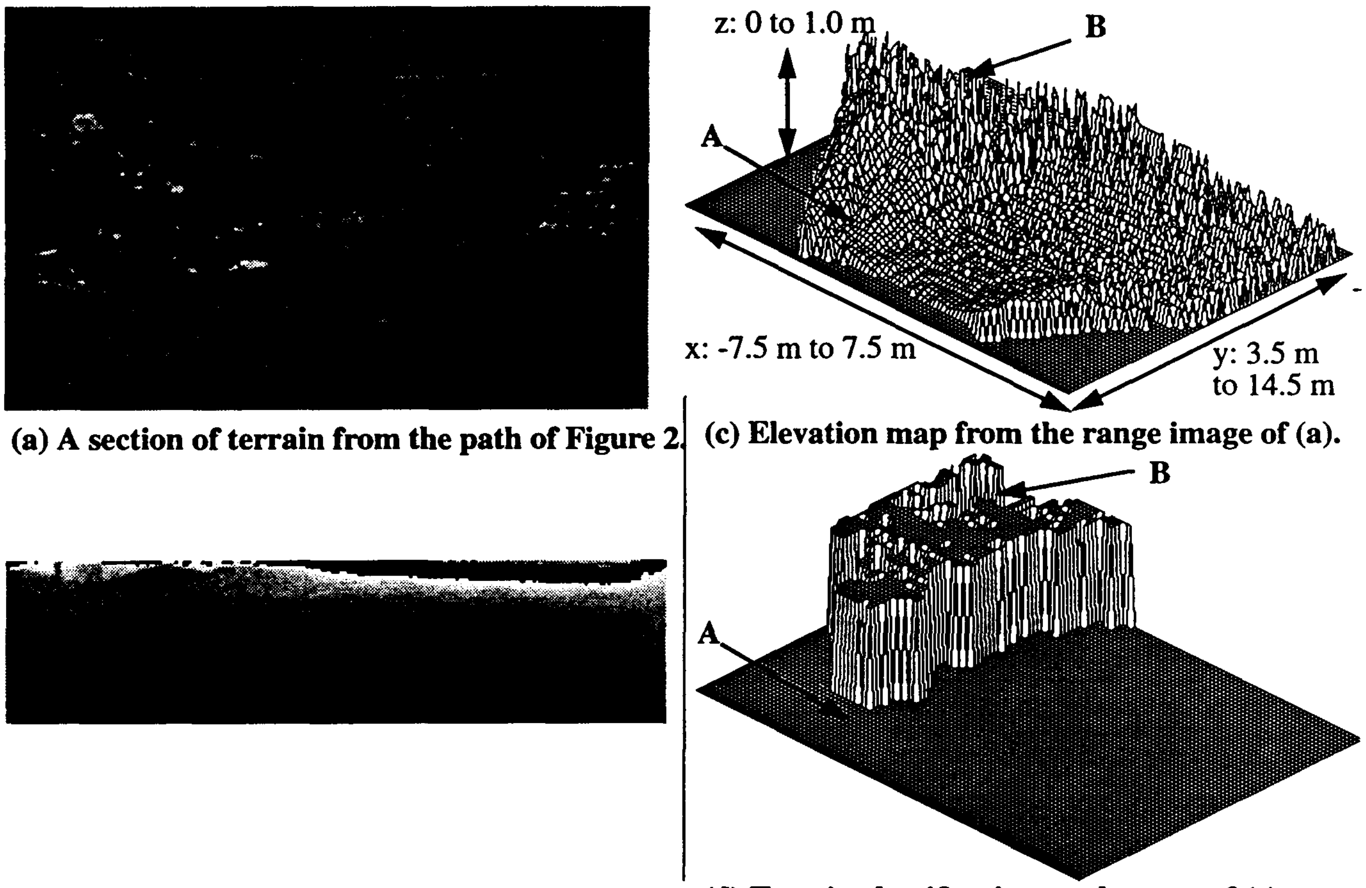

Range Sensor
Filtering
Conversion to Vehicle Coordinates
Grid Generation
Classification of Individual Cells
Grouping
Local Map

Figure 5: Range image processing

Figure 6 shows an example of terrain classification at one position along the path of Figure 3. A video image of the terrain is shown in Figure 6 (a); the corresponding range image is shown in Figure 6 (b). The scenes in the video and range images do not correspond exactly because the sensors are not in registration and because they have different fields of view. The video image is given only to show more clearly the actual terrain. 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 6 (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 stored at each element of the grid is the highest elevation of the set of data points within that cell. This particular choice of the maximum is for display purposes only since the perception module keeps track of the coordinates of all the data points within every cell. The scales are in meters in this display, the y coordinates ranging between 3.5 and 14.5 meters from the sensor and the maximum x coordinates being 7.5 meters on either side of the vehicle. 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. The display is generated by using the lowest point as the origin of the z values. The noisy appearance of the map is due to the sparsity of the data at long distances from the scanner. Specifically, the perception module does not perform any interpolation of the grid so that grid cells that are not filled with data remain unfilled. Although grid interpolation would make the map more visually appealing, it would add significant computation time but it would not add any information. In fact, interpolating may create additional problems by smoothing out real obstacles, or by introducing new obstacles which are merely artifacts of the interpolation algorithm. For more details, some of the undesirable effects of interpolation are discussed in [12] and the computational requirements of a robust interpolation algorithm are described in detail in [13]. Finally, Figure 6 (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. However, these regions of the terrain are correctly classified on the next image as the vehicle moves closer. Again, we argue that interpolating the map would create a dense map but would not classify the B cells consistently and robustly enough to justify the computational expense. In this example, the range image processing was able to detect the large untraversable region on the left of the image while classifying correctly the traversable regions, in particular, it correctly retained terrain region A because of its shallow slopes.

In practice, terrain features of size 30cm are detected at a range of ten meters from the sensor when vehicle speed is on average 2m/s. By comparison, the maximum range of the scanner is 18 meters with a distance between pixels of about one meter at that range. Figure 7 shows the detection distance as a function of height and width of an obstacle in front of the vehicle. The scales in height and detection distance are in meters. The detection distance is computed from the object to the front of the vehicle. These numbers were computed for a vertical object in the middle of the path of the vehicle. This curve was obtained by generating range images for width values 25cm and 50cm and height values 15cm, 25cm, 50cm, 75cm, and 1m, and for distances between object and sensor ranging from 5m to 15m in increments of 20cm. More than the limited range resolution, the reason for the limited detection range is mainly the angular resolution of the scanner which limits the number of pixels actually falling on the object. This explains why the graph of Figure 7 becomes flat as the height increases: It takes a large increase of height in order to increase the number of pixels on the object when the object is far from the sensor. Another illustration of the detection range is shown in Figure 8. Figure 8(a) shows the elevation profile computed from one column of the range image of a scene in which a small object is placed in front of the vehicle at a distance lower than the maximum detection range. The vertical axis is the elevation in centimeters and the horizontal axis is the distance from the sensor. Figure 8(b) shows the range image and elevation profile for the same object, but this time placed at the limit of the detection range. The origins used for plotting the elevation values in the two graphs are different, so only the relative values are meaningful. The variations in elevation in the graphs are due to the range resolution of the sensor, about 7cm. In this experiment, the object is about 20cm tall and the detection range is 12.5m. These graphs show that, from one column of the scene, at least five pixels are measured on the object when it is placed at close range, while only one pixel is measured on the object at the limit of the detectable range. Even though the range readings on the object are clearly accurate enough, the small number of readings due to the limited angular resolution is not sufficient to reliably classify the object as obstacle. The maximum detection range, together with sensor latency, are the main limitations of the system. They are the reasons why vehicle speed is limited to 3 m/s in the current implementation.



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

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

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

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

Figure 6: Example of terrain classification

The range image processing algorithm is general enough that, with minimal modifications, it can be used with other types of sensors to achieve higher performance. In particular, we have conducted preliminary experiments with the passive stereo system described in [20], and we are planning to experiment with a fast, single-line range sensor. These new sensor modalities will eliminate some of the limitations, in particular acquisition rate and maximum range.

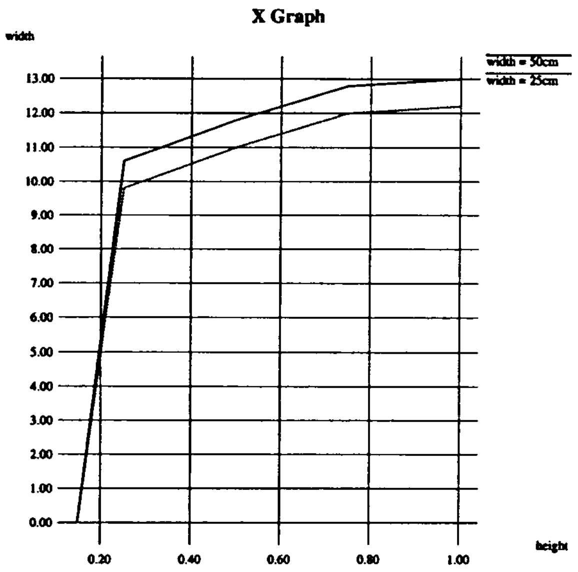
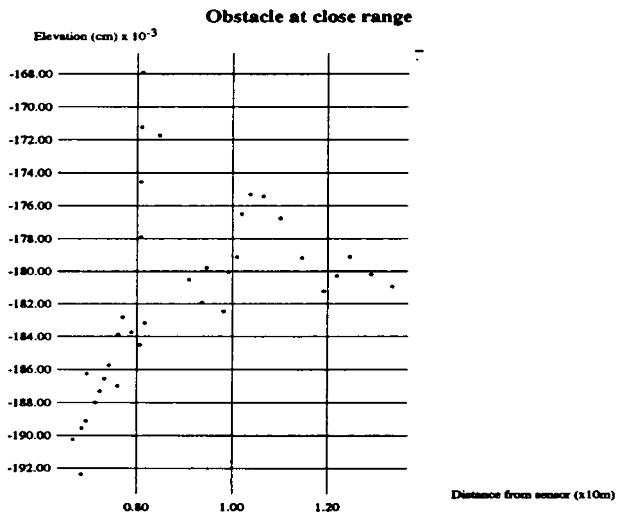
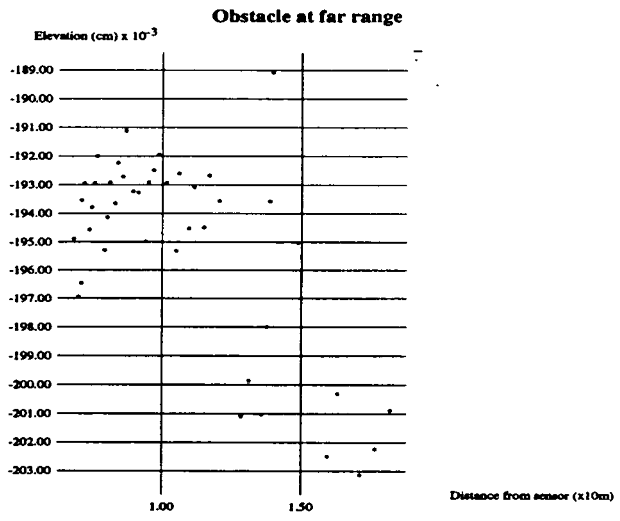


Figure 7: Detection range as function of height and width of object



(a) Elevation profile for an object placed within the detectable range.



(b) Elevation profile for an object placed at the limit the detectable range.

Figure 8: Range image and elevation profile on an object

Several problems can lead to occasional misclassification. 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. 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. Sparse vegetation also causes the detection of spurious obstacles. This is a more serious problem because spurious obstacles from sparse vegetation are detected only when the sensor is very close to the obstacle. As a result, not only is the system reporting a non-existent obstacle but also the vehicle is too close to the obstacle to execute any avoidance path. The reason why sparse areas of vegetation are detected only at very close range is that they appear in the range image as features that are too small to be detected at long range because of the size of the range pixel and the angular spacing between pixels. An additional sensor may be of some help if it is able to confirm the presence of an obstacle close to the vehicle. For example, we are considering the addition of sonars which can detect obstacles at short range but would not be fooled by vegetation. Another possibility is the use of an ad-hoc filter to eliminate features from the range image directly.

The range image processing algorithm has several important properties. First, it does not build a complete, high-resolution map of the terrain which would require interpolating between data points, an expensive operation. Instead, it evaluates only the terrain for which there is data. Second, the algorithm processes each image individually without explicitly merging terrain data from consecutive images. Instead, it relegates the task of maintaining a local map of untraversable regions to a separate local map module. The importance of this is that the local map module deals only with a few data items, the cells classified as untraversable, instead of with raw terrain data. As a result, maintaining the local map is simpler and more efficient. Because of these two features, range image processing is very fast, typically on the order of 200ms on a conventional Sparc II workstation. The main limitation is the 2 Hz acquisition rate of the sensor, not the processing time.

It is clear that the range image processing module may miss untraversable regions of the terrain because the terrain is evaluated only where data is present in the image and because the data may be too sparse to provide complete coverage of the terrain at long range. However, because of the processing speed, a region that is missed in a given image will become visible in subsequent images quickly enough for the vehicle to take appropriate action. Although this problem effectively reduces the maximum detection range of the perception system, we argue that the other possible solutions would reduce the maximum range even further and would introduce additional problems. The most obvious solution is to merge data from a few images before committing to a terrain classification. This solution effectively reduces the maximum detection range because the system has to wait until enough overlapping images are taken before a terrain region is evaluated. In addition, merging images is in itself a difficult problem because it requires precise knowledge of the transformation between images. In particular, even a small error in rotation angles between two images may introduce enough discrepancy between the corresponding elevation maps to create artificial obstacles at the interface between the two maps. (We refer the reader to [12] for a more quantitative description of this problem.) Therefore, unless the vehicle and position estimation systems are designed to produce accurate pose estimates, it is preferable to not merge images explicitly and to rely on fast processing to compensate for the sparsity of the data.

4 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. 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. The core of Ganesha is a single loop (Figure 9). 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. The cell attributes are used mostly for removing outliers in the map. 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.

In the rest of this section, we first discuss timing considerations in Ganesha and we describe the algorithm used for updating the local map. We then describe an important extension, that is, the explicit representation of the finite sensor field of view in Ganesha. We conclude this section with a discussion of the performance and key features of Ganesha.

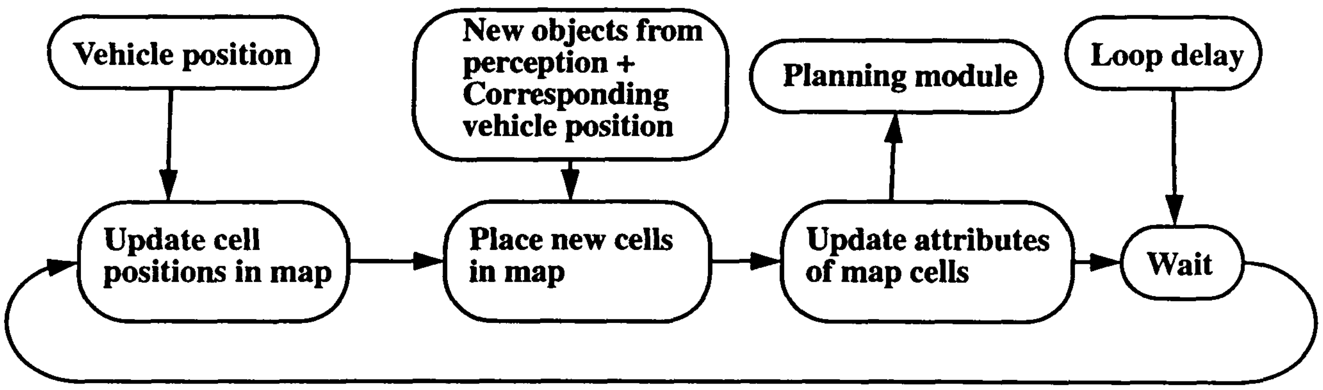

Vehicle position
New objects from perception + Corresponding vehicle position
Planning module
Loop delay
Update cell positions in map
Place new cells in map
Update attributes of map cells
Wait

Figure 9: Main loop in Ganesha

The loop delay used to determine the amount of time that Ganesha spends idle at the end of each loop is supplied by the user at the beginning of the mission. The selection of the loop delay affects system performance in a significant way. If the loop delay is too small, then overall system latency is reduced but Ganesha swamps the system both in terms of computation and communication resources. In particular the computation for one loop without any delay is only a few milliseconds which is too fast compared to the cycle times of the other modules. Conversely, a high value of loop delay frees up more resources for the other modules but introduces unacceptable latency. The selection of the loop delay is based on the target vehicle speed, and on the expected latency of the other modules. In the example system of Section 2, the loop delay was 100 ms.

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. The map at position (x_1, y_1, φ_1) is transformed to the map at position (x_2,y_2,φ_2) by first computing the position increment, δs, and the orientation increment, $\delta\varphi$:

$$\delta s = \sqrt{(x_1-x_2)^2 + (y_1-y_2)^2} \qquad \delta\varphi = \varphi_2-\varphi_1$$

Given a map cell at location $(x^m{}_1, y^m{}_1)$ in the first map, its coordinates $(x^m{}_2, y^m{}_2)$ in the transformed map are computed by using a polar transformation centered at the vehicle:

$$R = \sqrt{(x_1^m)^2 + (y_1^m)^2} \qquad \alpha = \operatorname{atan}\frac{y_1^m}{x_1^m}-\delta\varphi \qquad x_2^m = R\cos\alpha \qquad y_2^m = R\sin\alpha-\delta s$$

New objects detected by the sensor are added using a similar mechanism in which (x_1, y_1, φ_1) is the position of the vehicle at the time the objects were detected, and (x_2,y_2,φ_2) is the current position of the vehicle. This set of transformations is efficient and uses only relative estimates of vehicle displacement.

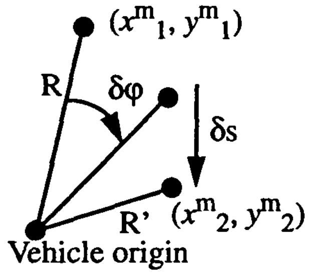


Figure 10: Updating the map in Ganesha

Because the map module deals only with a small number of terrain cells instead of a complete model, 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.

A different design of the local map module would be to maintain a much larger map with more information than just a list of terrain cells which would theoretically allow the navigation system to use data recorded from earlier images. There are two problems with this approach, however. First, the local map module is now forced to maintain a much larger amount of data, most of which is never used, and thus additional delays are introduced in the system. Second, errors in vehicle position accumulate to a point at which most of the map becomes useless. These two problems offset the occasional gain in additional information in the map.

In our design of the navigation system, the local map and planning modules do not have access to the original sensor data and therefore cannot correct possible errors in the output of the perception. In particular, a region which is mistakenly classified as traversable will never be reclassified because the local map module cannot go back to the original data to verify the status of the region. It is therefore important to use conservative values for the detection parameters in order to ensure that all the untraversable regions of the terrain are classified as such. The drawback of this approach is that the perception module may generate terrain regions which are incorrectly classified. For example, this may occur because of noise in the image or because of an erroneous reading of vehicle pose. Because the perception processes images individually without explicitly building maps, it cannot detect that this erroneous classification is inconsistent with previous observations. However, this problem is solved by the map maintainer which does maintain 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.

A severe limitation of any navigation system is the limited field of view of the sensor, in this case 80^o. 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. This is not desirable because the vehicle may run into a dangerous part of the terrain which was not previously observed. This is not a serious problem when Ganesha is used for on-road driving, for which the changes of direction are rarely abrupt enough to cause this problem to occur. In cross-country navigation, however, it becomes a serious problem because tight maneuvers are often necessary. For example, in the situation depicted in Figure 11, the vehicle would follow the arc marked (b) and would collide with the obstacle indicated by the white arrow because this obstacle has not been seen in any of the previous frame and because it is the best way of reaching the next goal point in the absence of any other information. A turn this sharp is not exceptional in cross-country navigation.

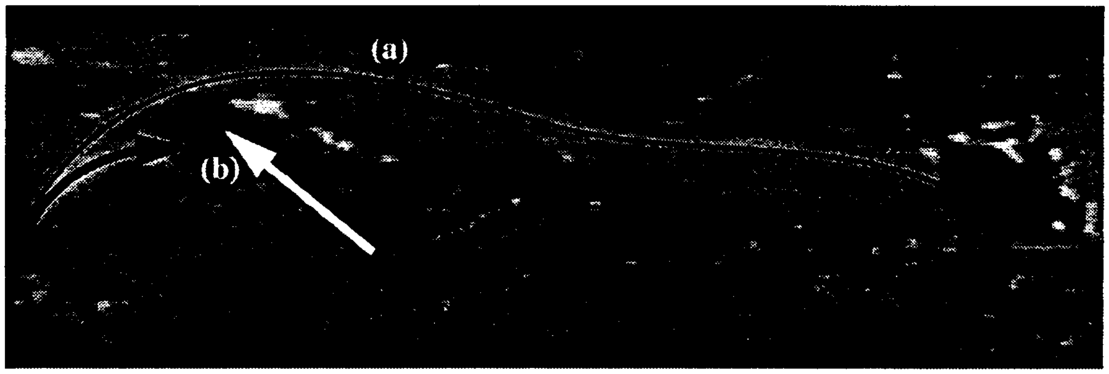


Figure 11: Field of view limitation in a real example; the vehicle could not execute the turn if the field of view were not taken into account

This situation occurs for small turning radii as shown in Figure 12. Here the field of view of the sensor is shown by the polygon **ABCD**. If the vehicle travels along a straight line, then the area covered by the sensor is the one bordered by lines **MC** and **NB**. Suppose now the vehicle starts turning sharply at point **P**, following the trajectory $\mathbf{PP_1P_2}$ as shown. Then the field of view of the sensor sweeps the annular area between $\mathbf{CC_1C_2}$ and $\mathbf{AA_1A_2}$. When the vehicle turns far enough then some parts of the vehicle may enter an unknown area that has not been covered by the sensor which is indicated by the shaded region in Figure 12. Also, images are taken at finite intervals which means that some parts of the terrain are not sampled that would be otherwise in the continuous case. For the following calculations we assume that images are taken continously and thus all possible areas of terrain are covered. The minimum possible turning radius that would keep the entire vehicle within the area swept by the sensor's FOV can be calculated from Figure 13. For simpler calculations, let the origin be at point **O**. The minimum possible turning radius is then given by the circle whose centre lies on the line passing through **PO** and which passes through point **O** and is tangent to the line **AB**. Let the coordinates of the tangent point **Q** be (x, y). Then (x, y) is at the intersection of line **AB**, of equation $y = mx + c$, with the circle of equation $(x - r)^2 + y^2 = r^2$. In these equations, r is the radius to be computed, c is the offset between sensor origin and vehicle origin, and m is the slope of the boundary line of the sensor field of view. Eliminating y between the two equations, we get a quadratic equation in x. Since the line is tangent to the circle, the discriminant of the equation must be zero, which yields the relation between the radius r of the circle traced by the right side of the vehicle and the field of view of the sensor:

$$r = mc + c\sqrt{m^2 + 1}$$

The minimum turning radius is $r_{\min} = r + w/2$. With the parameters of the ERIM scanner, the minimum possible turning radius is then $r_{\min} \sim 17.25$ m. In comparison, the minimum turning radius of the vehicle $R_{\min} = 7.5$ m. Therefore simply thresholding the turning radius by allowing only values that are greater than $r_{\min}$ would restrict vehicle terrainability too much.

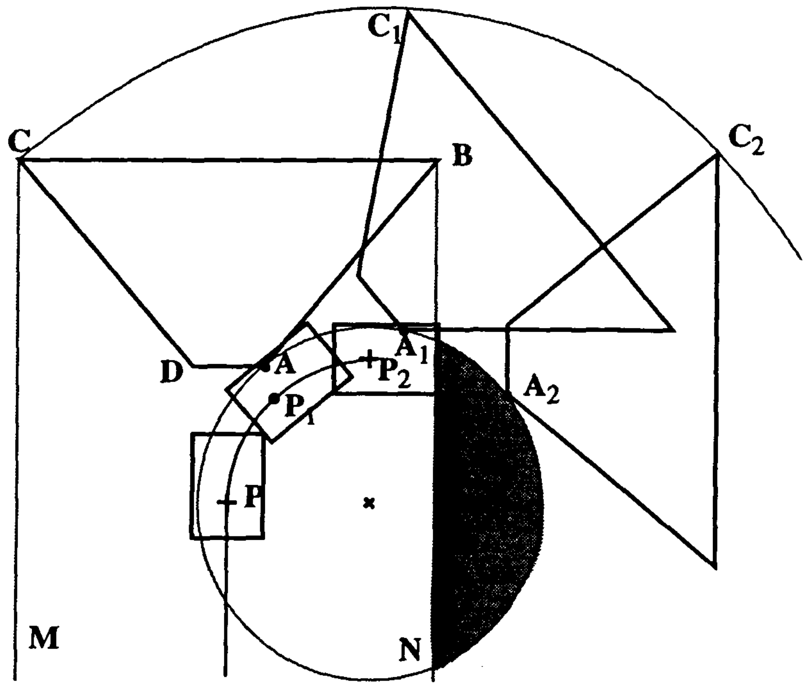


Figure 12: Geometry of the field of view limitation; the vehicle may drive on unmapped terrain because the field of view is too narrow compared to the turning radius

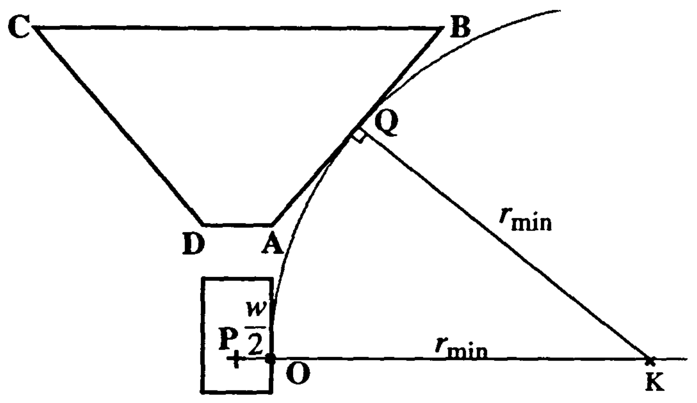


Figure 13: Minimum turning radius according to the FOV constraint

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 14. 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. These obstacles 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 14). Active FOV obstacles are treated just like any other regular obstacle. This mechanism ensures that the vehicle can still make sharp turns (Figure 14(a)), but will be prevented from steering into a path that falls into a locally unmapped area (Figure 14(b)). Here the vehicle is forced to move straight by the FOV obstacles whereas otherwise it would have kept on turning right. In Figure 14(c) the vehicle has passed the regular obstacle and continues to move normally. Vehicle speed is controlled as usual, i.e. the vehicle slows down when it gets close to an obstacle.

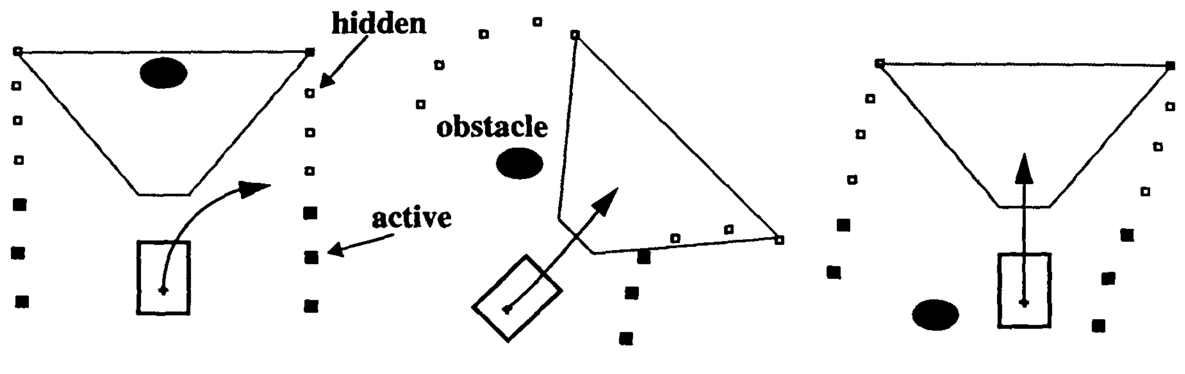
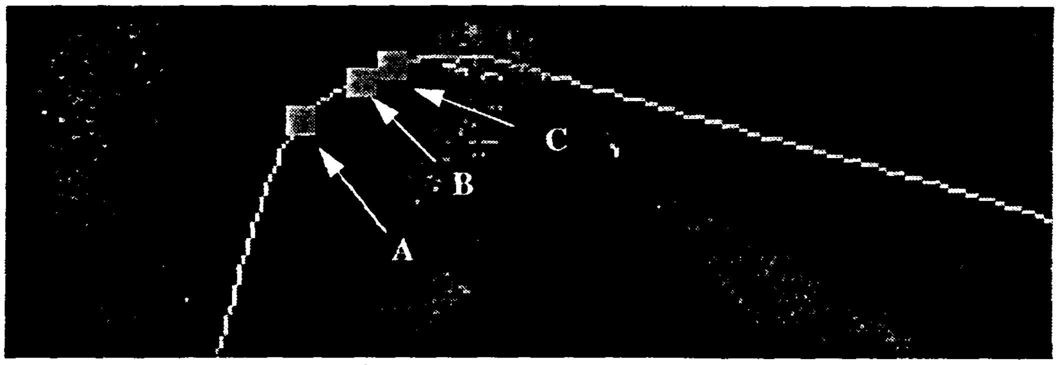
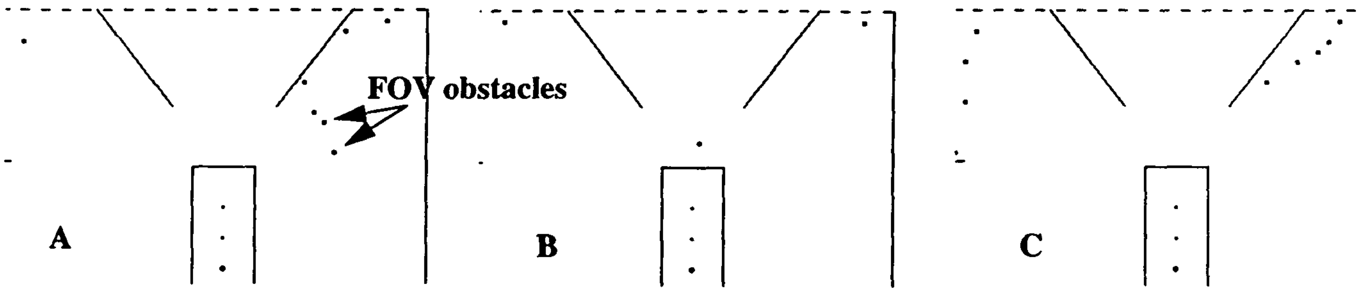


Figure 14: Use of FOV obstacles to limit turning radius

The net effect of the FOV mechanism is that the vehicle is prevented from steering into a path which is outside of the fields of view of the previous images. An important side-effect of the FOV mechanism is to slow down the vehicle when a turn of radius lower than the minimum turn radius is commanded, thus allowing for more images to be taken. In practice the vehicle slows down or even comes to a brief stop when the FOV obstacles come into the path. Figure 15 and Figure 16 illustrate the use of the FOV constraints during the turn shown in Figure 11. Figure 15(a) shows an overhead view of the path of the vehicle through the terrain during the turn. Figure 15(b) shows the local maps generated by Ganesha at the three points labelled in Figure 15(a). The local map show the FOV obstacles as small black dots. At point *A*, the vehicle is in the process of executing a sharp turn but the FOV obstacles do not yet affect the path. At point *B*, the active FOV obstacles start appearing in front of the vehicle and the turning radius and speed are adjusted. After changing the turn radius at *B*, the active FOV obstacles leave the current field of view and the vehicle resumes turning..



(a) Overhead view of the path and the terrain during a sharp turn.



(b) Ganesha local map at the locations marked in (a); only the FOV obstacles are shown.

Figure 15: Use of the FOV obstacles in Ganesha during the turn of Figure 11

Figure 16 shows the speed and curvature profiles at the turn shown in Figure 15. Figure 16(a) shows the variation of the path curvature over time. The effect of the FOV constraint is visible at point *B* where the curvature drops sharply to prevent the vehicle to run into an unmapped area. The curvature rises again at point *C* when the FOV obstacles leave the current field of view. Figure 16(b) shows the speed profile recorded during the same period of time. The speed drops sharply at A because of the presence of an FOV obstacle directly in front of the vehicle. In this case, the vehicle comes to a near stop during the turn which gives additional time to take an image of the unknown area. The graph shown in Figure 16(b) is quite noisy because the actual vehicle speed read from the controller is plotted instead of the commanded speed. The system behaves as would a human driver by slowing down as it executes the turn and by speeding up again once the terrain comes back into the field of view. 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. 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

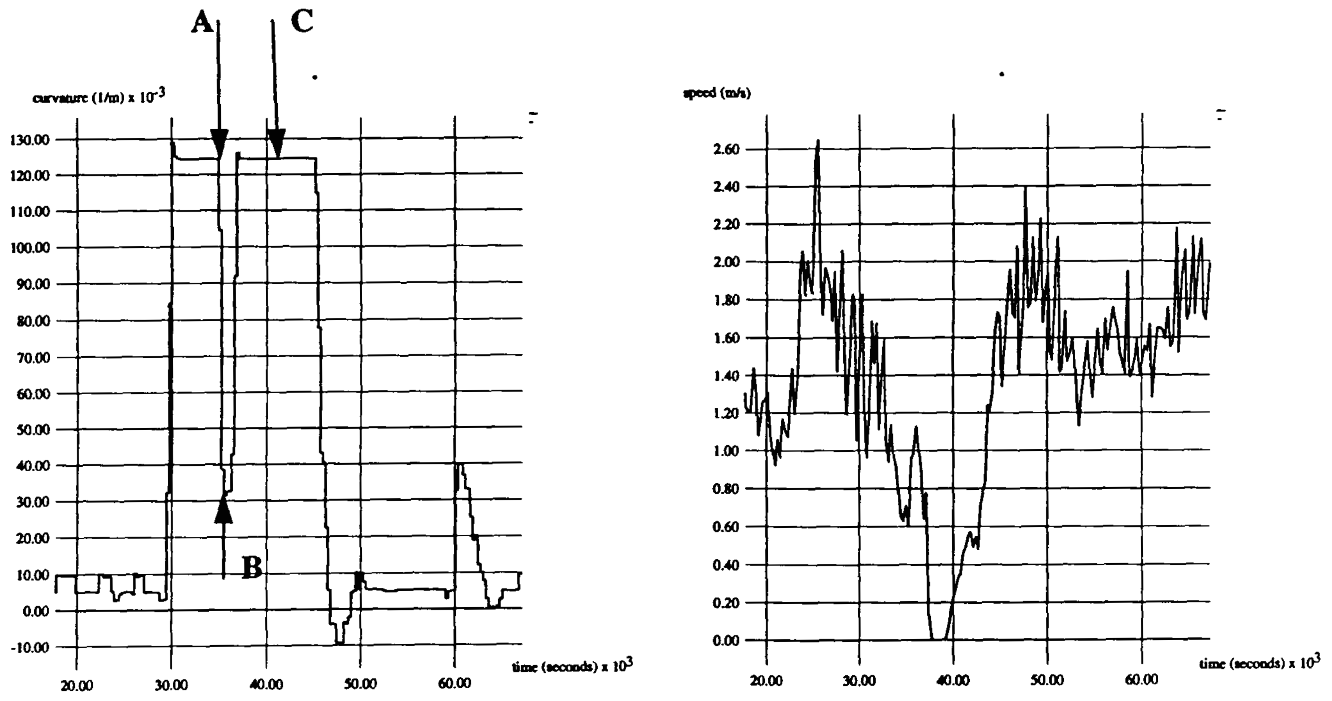


Figure 16: Speed and curvature profiles during the turn of Figure 15

5 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 with the framework of the Distributed Architecture for Mobile Navigation (DAMN). In this section, we will provide a brief overview of the DAMN system, and then explain how obstacle avoidance is combined with other tasks, such as road following or goal seeking, in order to provide vehicle control that achieves certain objectives while maintaining vehicle safety. We conclude the section by discussing some of the implementation issues involved in building the system described in Section 2.

DAMN is a behavior-based architecture similar in spirit to 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]. Figure 17 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. The behaviors that were used in the experiment of Section 2 are shown in solid lines; other possible behaviors are shown with shaded lines. We describe the use of DAMN in the context of the cross-country navigation system. (We refer the reader to [21] for a description of other systems built around DAMN in the context of road following.)

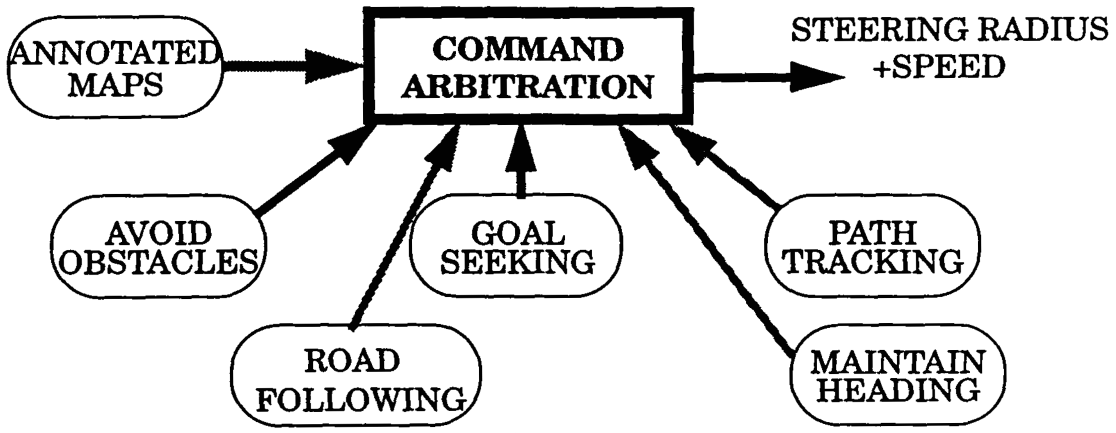


Figure 17: Behavior-based architecture; solid lines indicate the behaviors used in the cross-country system

It is incumbent upon a distributed architecture 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 [17]. An arbiter then performs *command fusion* to select the most appropriate action.

For example, vehicle commands such as turn radii are discretized into a fixed set of possible alternatives, as shown in Figure 18. For simplicity of illustration, only 5 options are shown, but an arbitrarily large set of command options may be used. Each behavior can then vote for or against each command option, with varying weights reflecting the relative priority of each behavior. Figure 18 illustrates the case where two behaviors are active, one responsible for obstacle avoidance and the other for goal seeking. 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.

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 is 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.

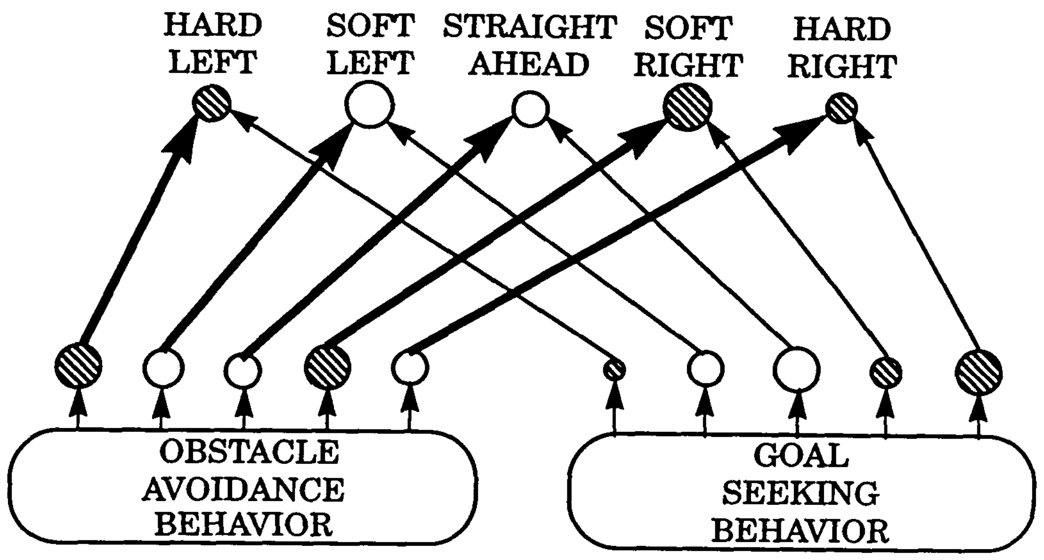


Figure 18: Command fusion in DAMN

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 turn choice with the maximum vote is found and is used as the command. The three cases are:
   - If there is a single turn choice 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 turn choices with the same maximum value, then the larger series of turn choices is used. For example, if the hard left, soft left, and hard right turn choices all have a value of 1.0, and the weighted sum for the other two choices is smaller, then the hard left-soft left combination would be used since there are two choices in a row with the maximum value, as opposed to just one for the hard right. If there are multiple series of turn choices with the same maximum value and of the same size, then one is chosen arbitrarily.

The emphasis in the research thus far has been in command fusion for the control of vehicle steering; the commanded speed is decided in a very simplistic fashion based upon the commanded turn radius. A maximum speed is set for the vehicle, and this value is simply multiplied by the normalized weighted sum for the chosen turn radius; the result is the speed command issued. For example, if the maximum speed is 5.0 m/s, then the vehicle will travel at that speed as long as all of the behaviors have voted fully for the chosen turn radius, i.e. the vehicle is headed directly to the goal and there are no obstacles obstructing its path. However, if the vehicle is deviating from the goal direction, the vote from the goal seeking behavior would be less than +1 and the resulting normalized summed weights would also be less than +1; therefore, the commanded vehicle speed would decrease. Likewise, if the vehicle is headed toward or close to an obstacle, the reduced vote from the obstacle avoidance behavior would result in a marked decrease in speed.

Current research involves the development of an entirely separate speed arbiter with its own set of associated behaviors. Thus, the turn behaviors could vote for the turn radii with varying strengths without regard for the effect it would have on speed. Because the choices of turn and speed commands are not completely independent and therefore must be coordinated, the speed behaviors would have as one of their inputs the command chosen by the turn arbiter, so that the choice of an appropriate speed can be influenced by the currently commanded turn radius. Other speed behaviors may instead use the actual estimated turn radius of the vehicle so that they would be operating in more of a closed loop fashion, albeit with greater delays.

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 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 19.

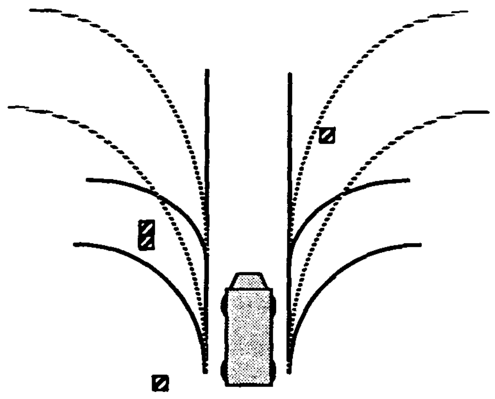


Figure 19: Arc evaluation in the obstacle avoidance behavior

If a trajectory is completely free of any neighboring obstacles (such as the hard right turn in Figure 19), 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 19) 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. In the context of cross-country navigation, the obstacles are the untraversable regions of the terrain computed by the range image processing. The obstacles are sent by Ganesha to the obstacle avoidance behavior at regular intervals in the form of lists of cells covering the untraversable regions.

If there is no obstacle along or near an arc for at least max_obstacle_distance meters, then the vote for that turn radius is +1. If an obstacle appears max_obstacle_distance meters away, then the vote becomes 0. If an obstacle on the path is min_obstacle_distance 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 along the arc at a distance greater than max_obstacle_distance, then set the vote to be 1.0
  - if an obstacle lies along the arc at a distance less than min_obstacle_distance, then set the vote to be -1.0
  - otherwise, set the vote to be: $-1.0 + \frac{\text{obstacle\_distance} - \text{min\_obstacle\_distance}}{\text{max\_obstacle\_distance} - \text{min\_obstacle\_distance}}$

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 near_miss_distance by which it misses is computed and 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 near_miss_factor * near_miss_distance, and take the smaller of that sum and 1.0

A typical set of values is: min_obstacle_distance = 5.0, max_obstacle_distance = 20.0, and near_miss_-factor = 0.5. With this set of values, if an arc were to come with a meter of an obstacle 20 meters distant, the vote for that turn choice would be 0.5. If an arc were to result in a collision with an obstacle 15 meters away, the vote would be -0.33. If the arc instead missed that same obstacle by a distance of 1.5 meters, the vote would be -0.33 + (0.5 * 1.5) = 0.42. These computations are performed for a given turn choice for each obstacle that has been detected, and the smallest value generated is the one used as a vote to the arbiter.

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 [22] 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. 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 [8].

DAMN is designed so that various behaviors can be easily added or removed from the system, depending on the current task at hand. Other behaviors that have been used include one that interfaces to the ALVINN road following system [18] and one that interfaces to the STRIPE teleoperation system [11]. When run in conjunction with the obstacle avoidance behavior described in this paper, the vehicle is capable of following roads or tracking user commands while concurrently insuring vehicle safety. 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.

In the experiment of Section 2, 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 of obstacles from Ganesha every 100ms, which is the loop delay in Ganesha. The data sent by Ganesha to the behavior is a list of obstacle locations in vehicle-centered coordinates. The obstacle avoidance behavior uses this data to evaluate the set of possible arcs, and sends its votes to the arbiter. The command arbitration module combines the votes from the two behaviors and issues a new driving command every 100ms. The set of possible arcs used for the experiments is represented by the 15 turn radii shown in Figure 20. This set of radii is sufficient for the type of environment in which the vehicle travels. 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 1.0 for the obstacle avoidance behavior and 0.25 for the seek goal behavior. After normalization, these are effectively 0.8 and 0.2, respectively. The absolute values are not important, only the relative weights. The obstacle weight is 4 times larger, since that is much more important than seeking the goal at any given moment.

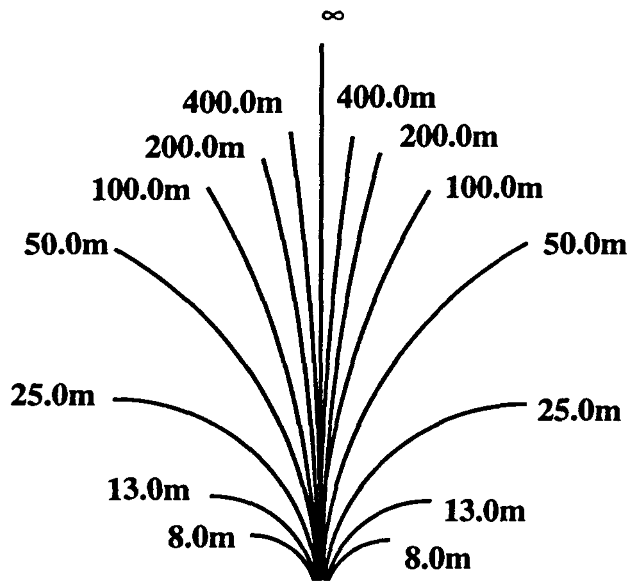


Figure 20: Set of possible turning radii in DAMN for the cross-country navigation system

Figure 21 and Figure 22 illustrate more precisely the way DAMN combines outputs from different behaviors. The lower part of Figure 21 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, which is receiving votes from two behaviors, the obstacle avoidance behavior described above, and a heading behavior which forces the vehicle to follow a constant heading. The upper part of Figure 21 shows the distribution of votes for each of the behaviors and for the arbiter. The horizontal axis is the time axis, the total length of the sequence being 30 seconds. For each module, the vertical axis is the set of turn choices from a hard left to a hard right turn. The votes were sampled every 200ms for display purposes. The maximum vehicle speed was 2m/s in this experiment. The set of fifteen turn radii shown in Figure 20 was used in this example. The votes are displayed as shaded pixels with the brightest pixels corresponding to the highest possible vote of 1.0 and the black pixels corresponding to the lowest possible vote of -1.0. In addition to the raw votes from the arbiter, the figure shows an enhanced version of the arbiter's distribution of votes. The enhanced version is obtained by normalizing the pixel values using the effective min and max values for every time slot instead of the extreme values, -1.0 and 1.0. The enhanced arbiter is included to facilitate the visual interpretation of the vote distribution but is not used for driving the vehicle. In addition to the vote distribution, Figure 21 also includes a graph of the commanded turn radius over time. The commanded turning radius is computed from the maximum of the vote distribution of the arbiter as described above.

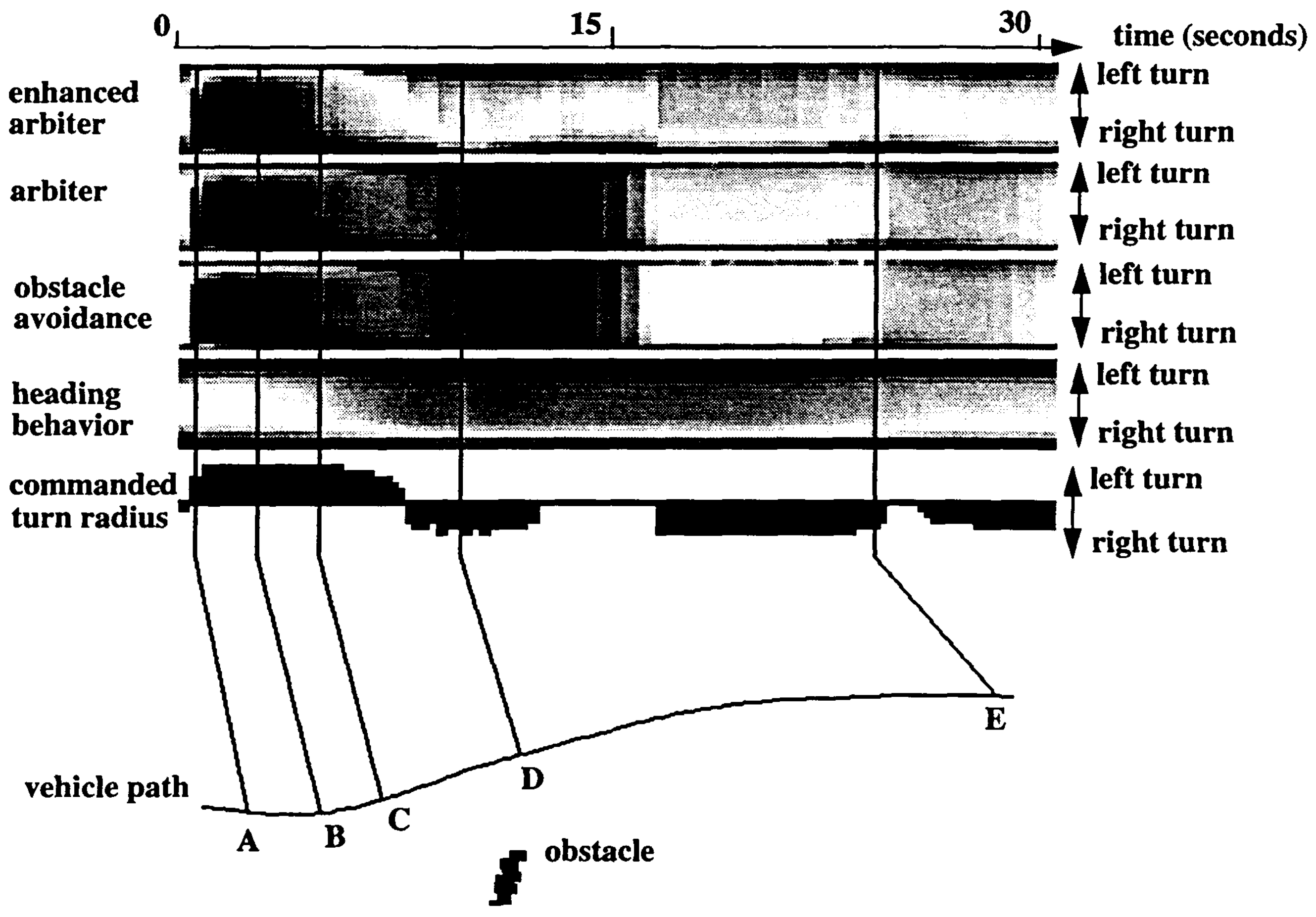


Figure 21: Distribution of votes over time in the arbiter and in the obstacle avoidance and heading behaviors; the distribution of votes at the locations marked by letters in this figure are discussed in detail in the text and are shown in Figure 22

Figure 22 shows the distribution of votes at five points of interest along the path, indicated by capital letters in Figure 21. 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 meters for hard left to +8 meters for hard right. The sequence of five points selected in Figure 21 is typical of a path around a single obstacle. 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 21, rather than abruptly switching modes from avoiding obstacles to following a heading, the transition proceeds smoothly and steadily as the situation gradually changes. Finally, at point *E*, the vehicle has completely cleared the obstacles and they are no longer in the local map, so that the votes from the obstacle avoidance behavior are mostly +1. The heading behavior now dominates completely and a right turn is commanded. Through the interaction of the avoid obstacles and heading behaviors, the vehicle was thus able to successfully circumvent an obstacle and regain its desired heading. This simple experiment shows that the arbiter combines the preferred commands from different in an efficient and natural way.

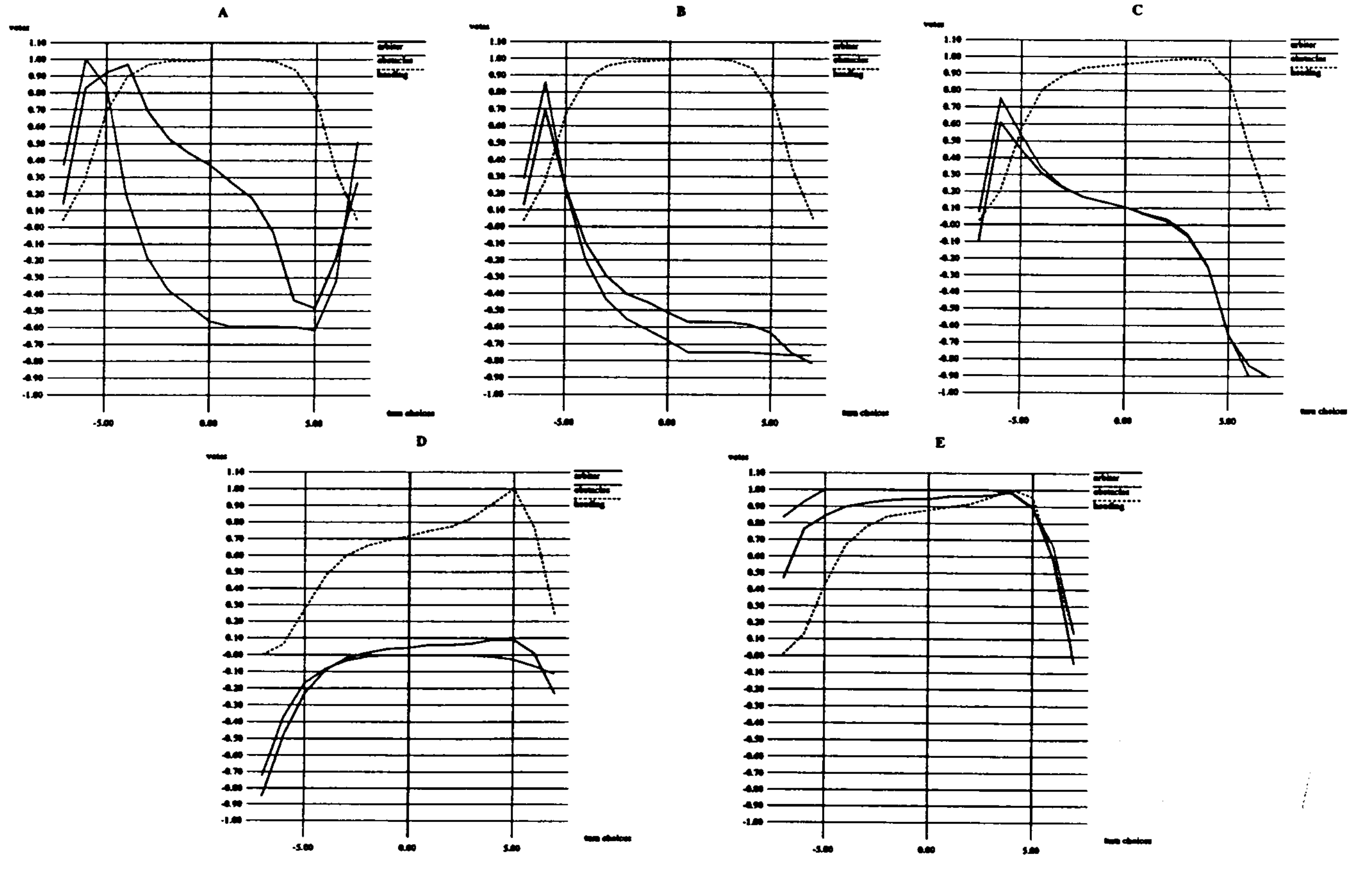


Figure 22: Distribution of votes in the arbiter, obstacle avoidance and heading behaviors at the points marked in Figure 21

The example of Figure 21 and the experiment of Section 2 show that a reactive architecture is capable of navigating cross-country terrain while pursuing mission objectives such as goal locations. The reactive nature of the core system is possible because the local map and perception modules simply output lists of binary obstacles. If instead the output of the perception system were a detailed geometric model of the terrain, then planning would require significantly more computation, thus precluding the use of a reactive approach. The current design of DAMN, with the arbiter taking inputs from a set of specialized behaviors, is flexible enough to a allow for a variety of other behaviors to use the core navigation system. For example, the goal seeking behavior could be replaced by a more sophisticated map navigation behavior if some prior knowledge of the environment is available.

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. In summary, early evaluation of terrain traversability allows us to achieve continuous motion at moderate speeds 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 cannot be corrected later in the system because only the perception module has access to the sensor data. This problem is eliminated 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. While appropriate in many instances, this approach is not appropriate for all vehicles. In particular, we have made the assumption that the vehicle can safely negotiate terrain variations which are detectable far enough in advance so that the vehicle is able to modify its path appropriately. For example, this vehicle at these speeds can tolerate terrain discontinuities of 20cm. With a range resolution of 7cm and an angular accuracy of 0.5^{o}, such a discontinuity can be detected in time to avoid it with an arc of radius greater than the minimum turning radius of 7.5 m, assuming a 2Hz image acquisition rate and an additional 0.5 seconds latency in the system. Sensor acquisition rate and resolution are the two numbers that set hard limits on the speed.

The main limitations of the system are the limited range and speed of the sensor, and the non-real-time nature of the system which is implemented on conventional Unix workstations using standard networking protocol. To address the first limitations, 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. Other limitations of the system are related to poor terrain reflectance properties and to vegetation. We feel that these problems can be addressed only by adding other specialized sensors.

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

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

[5]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.

[6]J. Gowdy. "EDDIE: User's Manual", Internal Report, The Robotics Institute, Carnegie Mellon, 1991.

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

[11]J.S. Kay, C.E. Thorpe. STRPE Supervised Telerobotics Using Inremental Polygonal Earth Geometry. In *Proc. Intelligent Autonomous Systems Conference*, 1993.

[12]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.

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

[14]L. H. Matthies. Stereo Vision for Planetary Rovers: Stochastic Modeling to Near Real-Time Implementation. *International Journal of Computer Vision*, 8:1, 1992.

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

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

[17]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.

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

[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]B. Ross. A Practical Stereo Vision System. In *Proc. Computer Vision and Pattern Recognition*, New York, June 1993.

[21] 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.

[22]R. Wallace, A. Stentz, C. Thorpe, H. Moravec, W. Whittaker, and T. Kanade. First Results in Robot Road-Following. In *Proc. IJCAI-85*, 1985.