Journal of Computer Vision, Volume 8, Number 1, 1992, pp. 71-91.

putes drivable paths more accurately, and hence, should produce improved performance on rougher terrain.

The limitations of the controller and the sensor prevent the system from achieving higher speeds. When these limitations are removed, higher speeds should become achievable. Other future work will include active sensor pointing, incorporation of terrain characteristics into velocity determination, and inclusion of a more complete model of the vehicle's capabilities in order to permit more comprehensive planning.

7. ACKNOWLEDGEMENTS

As well as the authors, the software development team has included Alonzo Kelly, Omead Amidi, Mike Blackwell, William Burky, Jay Gowdy, George Mueller, Annibal Ollero, Dong Hun Shin and Jay West. The authors thank Ben Motazed and the NAVLAB II design and retrofit team for producing a capable ACCN testbed.

This research was sponsored by DARPA, under contracts "Perception for Outdoor Navigation" (contract number DACA76-89-C-0014, monitored by the US Army Topographic Engineering Center) and "Unmanned Ground Vehicle System" (contract number DAAE07-90-C-R059, monitored by TACOM).

8. REFERENCES

1. Amidi, O. "Integrated Mobile Robot Control", Robotics Institute Technical Report CMU-RI-TR-90-17, Carnegie Mellon University, 1990

2. Chang, T.S., Qui, K. and Nitao, J. J. "An Obstacle Avoidance Algorithm for an Autonomous Land Vehicle", Proceedings of the 1986 SPIE Conference on Mobile Robots, pp. 117-123.

3. Daily, M. et al. "Autonomous Cross Country Navigation with the ALV", Proceedings of the 1988 IEEE International Conference on Robotics and Automation, pp. 718-726.

4. Dickmanns, E. D. "Dynamic Computer Vision for Mobile Robot Control", Proceedings of the 19th International Symposium and Exposition on Robots, pp. 314-27.

5. Feng, D. "Satisficing Feedback Strategies for Local Navigation of Autonomous Mobile Robots", Ph.D. Dissertation at CMU, May, 1989.

6. Keirsey, D., Payton, D., Rosenblatt, J. "Autonomous Navigation in Cross-Country Terrain", proceedings of Image Understanding Workshop, 1988.

7. Kelly, A., Stentz, A. and Hebert, M. "Terrain Map Building for Fast Navigation on Rough Terrain" Proceedings of the 1992 SPIE Conference on Mobile Robots (to appear).

8. Lozano-Perez, T. and Wesley, M.A. "An Algorithm for Planning Collision Free Paths Among Polyhedral Obstacles". Communications of the ACM, Vol. 22, Num. 10, October 1979, pp. 560-570.

9. Moravec, H. P. "The Stanford Cart and the CMU Rover", Proceedings of the IEEE, Vol. 71, Num 7, July 1983, pp. 872-884.

10. Olin, K. and Tseng, D. "Autonomous Cross Country Navigation", IEEE Expert, August 1991, pp. 16-30.

11. Shin, D., Singh, S. and Shi, W. "A Partitioned Control Scheme for Mobile Robot Path Planning", Proceedings of the IEEE Conference on Systems Engineering, August, 1991.

12. Thompson, A. "The Navigation System of the JPL Robot", IJCAI, 1977, pp. 749-757.

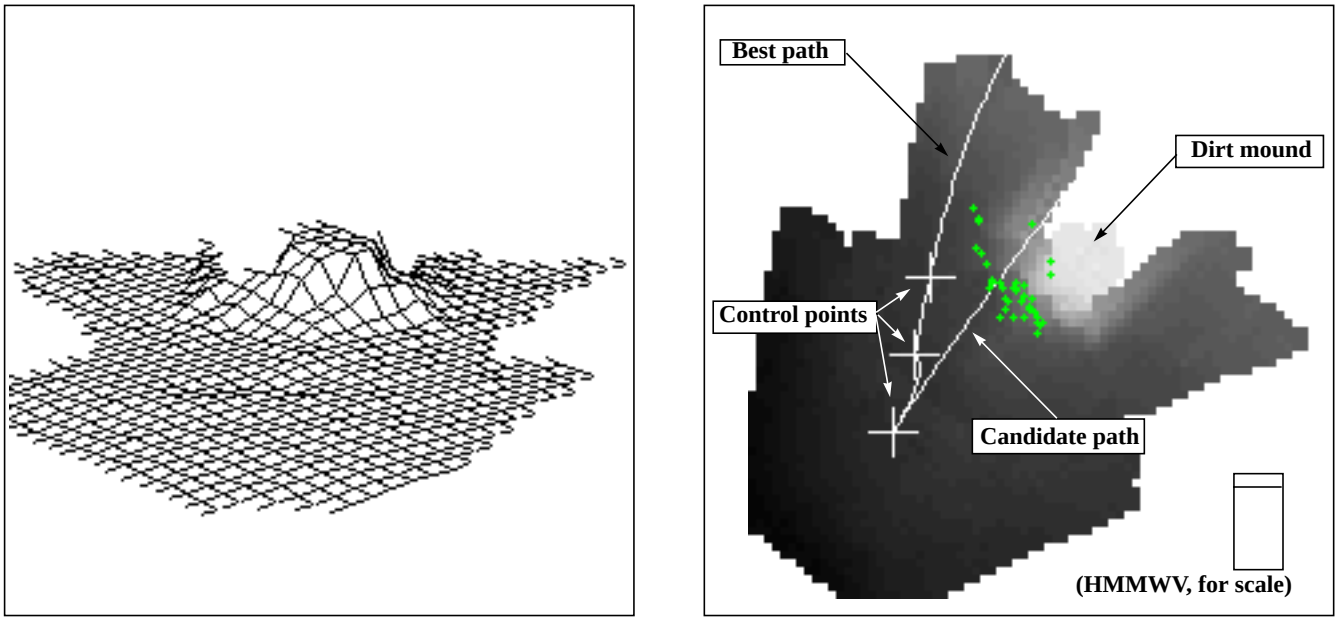
13. Matthies, L. "Stereo Vision for Planetary Rovers: Stochastic Modeling to Near Real-Time Implementation", International

Figure 10 shows a map and a path found around a clearly inadmissable region. Regions of lower intensity correspond to regions of higher elevation. The planning image shows and actual run with an initial path through the obstacle (a mound of debris) and the final path found around the obstacle. Each cross corresponds to a control point in the planned path. The last control point is outside of the region displayed.

Inability to drive the vehicle backward prevented the testing of the velocity optimizing algorithm. Further controller-related difficulties in bringing the vehicle to a rapid stop prevented testing of the guaranteed stop in case of planning impasse. Current controller related problems with speed regulation prevented thorough testing of the dynamics models during actual runs. However, these models have been tested with success in simulation.

Most natural obstacles found in the testing environment required no more than 4 control points to define a path around them. The limited ability to follow features, such as cliffs, was also observed.

Over 22km of natural terrain was traversed autonomously during the latest series of test runs.



Wire-frame Elevation Map

Intensity-encoded Elevation Map

Figure 10. Example of a Obstacle Avoidance on Rough Terrain

6. CONCLUSIONS

A system for Autonomous Cross-Country Navigation has been developed and demonstrated successfully at speeds up to 4.0 m/s. Furthermore, distances as great as 6.7km have been traversed without stopping. Robust obstacle avoidance has been observed at 2m/s.This planning system supports this performance on natural terrain, integrates a recursive generation planning paradigm with simulated traversal of paths, and includes vehicle dynamics as part of the planning process.

The major difficulties encountered were limited sensor horizon and limited response time of the vehicle. A longer sensor horizon would permit more time for planing, and would allow the system to make direction changes around obstacles more quickly. Better vehicle response would allow the system to successfully follow these planned paths. It should be noted that the vehicle is physically capable of following the planned paths, but that the controller lacks the needed response characteristics.

Future work, outlined below, will allow the system to travel along more tightly constrained paths, at higher speeds. Planned sensing modifications will permit the longer sensor horizon needed to achieve these goals. The ability to drive in reverse will be added, permitting clean recovery from a failure to find a sufficient path in time. An improved planning algorithm is being designed and implemented, using a similar algorithm which runs faster, considers more completely the vehicle yaw, and com-

puted velocities ensures that at least one performance limit is met, but that none are exceeded.

For an alternate interpretation of this method, consider the path in its entirety. The path has *n* points across a patch of terrain, and it is required for safety reasons that the vehicle be stopped at the end of this terrain. So, the maximum permissible speed is calculated for point *n-1* that will allow the vehicle to stop by point *n*, while avoiding dynamic instabilities. Reverse state propagation then uses the performance limits to compute the maximum permissible speed at point *n-2*, and so on.

4.3. Performance Optimization and Velocity Constraints

The dynamically determined maximum velocity profile is filtered in order to smooth the ride, and also to optimize performance. Ride smoothing in the first part of the path (driving segment) is accomplished by applying a small acceleration, thus ramping up to the maximum dynamically allowable velocity. In the second part (stopping segment), a deceleration is applied, keeping the velocity profile below the profile determined by the dynamics. For performance optimization, the algorithm measures the idle computing time in our planning cycle, and can determine a vehicle velocity which will reduce this time to a minimum, maximizing vehicle speed. See Figure 9.

Temporal planing, like spatial planning, can fail. If the maximum permissible velocity of the first way point on the path is less than the matching point on the previous path, the system cannot guarantee vehicle safety on the next path, so the current path is driven to completion. Again, failure does not compromise vehicle safety.

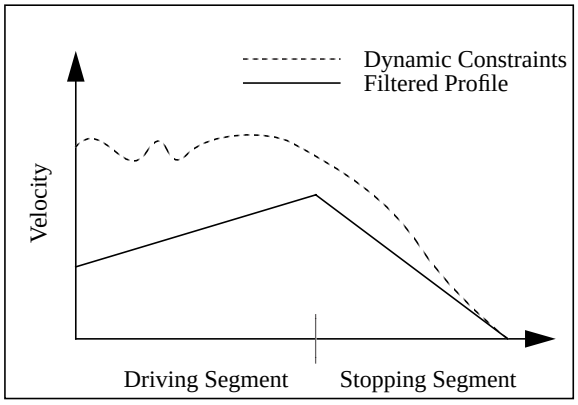


Figure 9Velocity Profiles

5. RESULTS

Tests of the vehicle were performed in a large open area, characterized by gently rolling terrain, sparse low vegetation, rock outcroppings, and water puddles. The system was implemented using a pure pursuit path tracker[1].

Obstacle avoidance was successful at speeds up to 2.0 m/s. Obstacles were detected, and paths planned in the allotted

time constraint at speeds in excess of 4.25 m/s, as predicted for more benign terrain. However, the ability of the vehicle's tracker and controller to accurately follow these paths prevented the vehicle from actually driving around obstacles while maintaining this speed. By all indications, an improved controller will enable us to achieve our theoretical maximum speed. Runs at 4.0m/s in excess of 0.3km in length were successfully undertaken.

However, the most successful runs involved driving in a large (90-100m diameter) circle for a total of 5.1 km while avoid obstacles located on the circle. Several runs of 1-3km were undertaken with obstacle avoidance. The longest run was 6.7km, without obstacle avoidance, though checking for obstacles was performed. These runs demonstrated a high level of reliability in the entire system, including perception, planning, tracking and control. All of these runs were done at speeds averaging 1.8m/s. For comparison, the ALV completed runs of 0.5km at speeds around 0.9m/s during a series of tests in December, 1987[6].

$$\dot{s} = \sqrt{Ra_x} \tag{6}$$

where R is the instantaneous radius of curvature at a point and a_x is the chosen lateral acceleration limit.

• Parallel Acceleration: Acceleration along the direction of motion (y-axis) induced by braking can cause skidding.

$$a_y = \mu g_z \tag{7}$$

Integrating the speed and parallel acceleration from the initial speed and position to the final speed and position:

$$a_y = \frac{d}{dt}\dot{s} = \frac{dy}{dt}\left(\frac{d}{dy}\dot{s}\right) = \dot{s}\frac{d}{dy}(\dot{s}) = \mu g \tag{8}$$

$$\int_{\dot{s}_i}^{\dot{s}f} \dot{s}\,d\dot{s} = \int_{yi}^{yf} (\mu g)\,dy \tag{9}$$

$$\dot{s}_f = \sqrt{2\,(\mu g)\,\Delta y + \dot{s}_i^2} \tag{10}$$

• Vertical acceleration: Excess acceleration vertically (z-axis) induced by changes in terrain elevation can cause loss of traction.

The standard equation for parabolic motion:

$$z = \frac{a_z \Delta t^2}{2} + v_0 \Delta t + z_0 \tag{11}$$

where z is terrain elevation, v_0 is initial velocity, and z_0 is initial terrain elevation.

The implication is that the terrain is permitted to “accelerate” at the performance limit. This acceleration is the second derivative of terrain elevation. Currently, the terrain accelerations are limited to 1g downward, and are not limited upward.

However, (11) still contains a Δt . Clearly, $\Delta t = \dot{s}\Delta y$. Substituting, and solving yields:

$$\dot{s} = \frac{-\frac{v_0}{a_z} \pm \left(\sqrt{\left(\frac{v_0}{a_z}\right)^2 + \Delta z}\right)}{\Delta y} \tag{12}$$

4.2. Reverse State Propagation.

Each of the constraint equations ((6),(10), and (12)) relates $\dot{s}$ (the forward velocity) to one of the orthogonal performance limits (a_x, a_y, a_z). However, these constraint equations require knowledge of the velocity at a previous point to compute the velocity limit for the current point. For example, in Equation (10), the final velocity depends on the initial velocity. Thus, by beginning with a known state, and computing maximum velocities at each point sequentially, a maximum velocity profile can be determined. By the definition of the stopping segment, the state of the last point of the path is known: the vehicle must be stopped at this point. This provides an initial velocity for the reverse state propagation. Propagating these velocities backwards along the path for each way point, and taking the minimum of the three computed velocities produces a velocity profile that ensures that the vehicle remains within the performance limits at each point. Choosing the minimum of the three com-

In this way, an inadmissible point is moved around the obstacle, and becomes a control point which indicates the probable edge of an inadmissable region. Because the control point searching is done by continuously moving the control point toward the edges of the known terrain during each search step, termination is guaranteed, as the point cannot be moved past the edge of the known map.

There is one exception case for this algorithm. If an inadmissable point is found that is near an old control point, then the old control point becomes the candidate control point (and searching continues from this point), thus preventing the clustering of control points in a small area.

3.4. Path Length Considerations

Unfortunately, there is a significant complication to this algorithm. The edges of the elevation map are not known *a priori*, as they depend on the shape of the terrain. Edges are manifested as regions of kinematic inadmissibility because of unknown terrain. These edges cannot be found quickly by searching, because large specular regions and other complex terrain may also return this condition (unknown terrain) at a location far from the true edge of the map. For these reasons, it is necessary to fold some concept of path length into control path selection. Furthermore, significantly sloped terrain can cause the distal bound to be significantly closer or farther away, invalidating the use of a simple distance threshold for determining the edge of the map.

Our solution is to have an acceptable path length function which decreases with total search time. Path length is defined as distance between the first control point and the first inadmissable point on the generated path. As consecutive control paths are considered, the minimum acceptable length before an inadmissible point may be encountered decreases. Eventually, a threshold is crossed, and a path is accepted with length less than the full distance. As a final implementation note, the longest control path found during the generate and test algorithm is saved. If the minimum length drops low enough, this path will become acceptable.

3.5. Spatial Search Failure

There are two types of search failure. First, a traversable path may not be calculated before the vehicle drives onto the stopping segment of the current path. In this case, the vehicle stops, drives backward to the last point in the intended region, and then drives over the next path. Second, the planner may not be able to find a path through the region. In this case, the current system will terminate the run after the vehicle comes to a stop on the current path.

4. TEMPORAL TRAJECTORY GENERATION

The planned path is a spatial path; it designates where the vehicle is to drive, but does not constrain the speed at which the vehicle is to traverse the path. It is clear that when driving across rugged terrain, one cannot pick an arbitrarily high speed; it becomes necessary to assign a maximum speed to each way point along the path. Temporal planning is the process of calculating these maximum speeds based on a set of dynamic constraints and then further limiting those velocities by considering computation time, desired performance, and the previous vehicle state.

4.1. Dynamic Constraints

The velocity profile generator calculates a set of maximum speeds along this path that ensures that the vehicle will not exceed a set of three constant orthogonal accelerations called performance limits. These limits are maximum allowed acceleration components and are chosen by the user to include the vehicle's stability as well as any other acceleration limitations that the user wishes to consider. In the following formulations, $\dot{s}$ denotes the vehicle's speed and a_i denotes the i^{th} component of the vehicle's acceleration as expressed in the vehicle's coordinate system. Three equations for $\dot{s}$ each a function of one of the three limiting accelerations, are now derived using a particle model of the vehicle.

- Lateral Acceleration: Acceleration laterally (x-axis) induced by centripetal acceleration can cause lateral slippage.

$$a_x = \frac{\dot{s}^2}{R} \tag{5}$$

map. This method is similar to that described by Daily[3].

3.3. Control Point Selection

If an inadmissable way point is found while traversing a local path in simulation, it is assumed that an inadmissible region exists in configuration space "in front" of the inadmissible point. This inadmissible point is added as a candidate control point to the list of control points (the control path) used to generate the path. See Figure 5. The above assumption is made because inadmissible regions in configuration space produced by natural objects tend to occur in groups; therefore, when an inadmissible point is found, it is reasonable to assume there are others "nearby."

The candidate control point is moved around the edge of the obstacle in successive sideways steps, on each step trying to proceed as far forward as possible. These steps are tested in the order shown in Figure 6. Lower numbered directions are chosen preferentially. In the example shown, Direction 6 would be chosen.

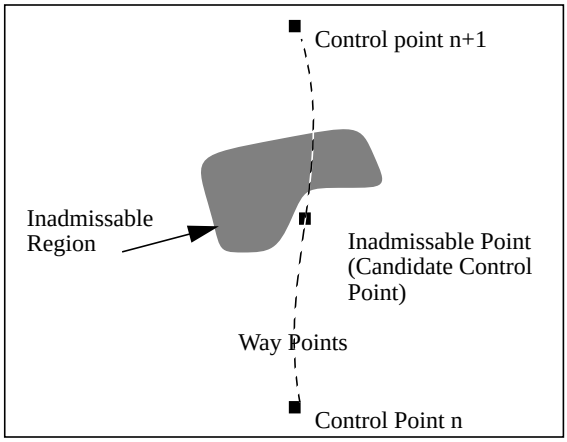


Figure 5. Inadmissable Point Identification

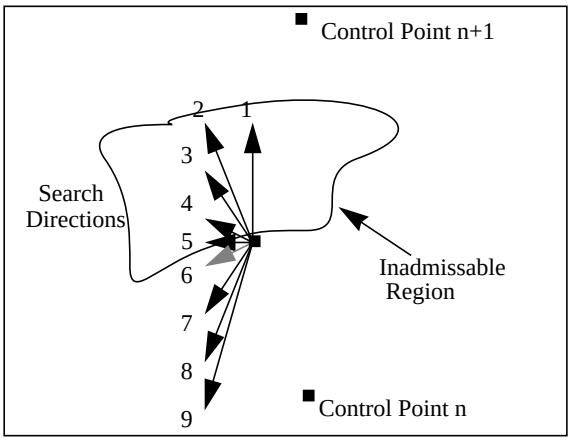


Figure 6. Edge Tracing Procedure

When a step straight forward (Direction 1) is admissable, it is assumed that the side edge of the obstacle has been found. To compensate for tracking errors, an extra step to the side is taken. For computational efficiency, several steps are taken along directions the final path is likely to go through, and these are checked for admissability. See Figure 7. This procedure is applied recursively between any two control points when an inadmissable way point is found. In this example (Figure 8), two possible paths are found.

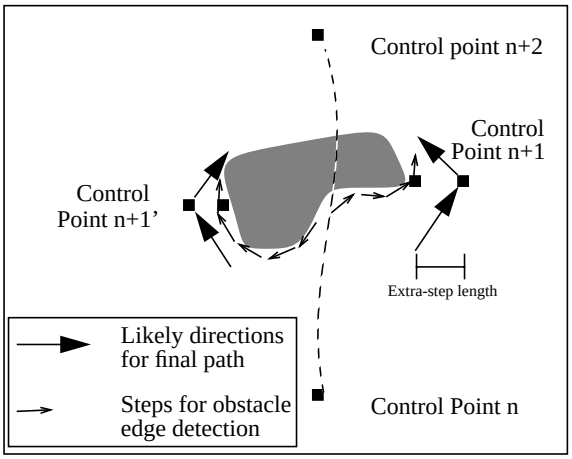


Figure 7. Edge Detection/Area Verification

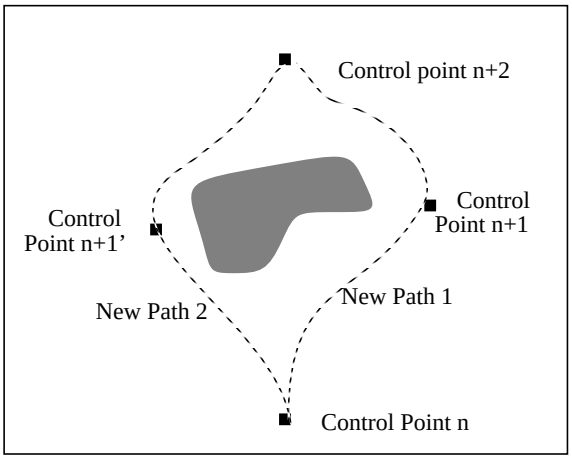


Figure 8. Final Paths with Obstacle Avoided

The system begins initially with a long *global path* describing a desired path between start and goal. During every sense-plan-drive cycle, the planner tries to move the vehicle along and/or closer to the global path by choosing *control points.* Control points are defined as the ends of the local path, or a point which may lead the vehicle around an obstacle. A set of control points which will be used to generate a local path is called a *control path*. Initially, each planning cycle for the next path begins with a control path having two control points: a point from the current path for continuity, and a point a fixed distance ahead to move the vehicle toward the global path. When a control path is selected for testing, a spatial curve is fit to the control points, and closely-spaced *way points* are generated along the entire length of the curve. Figure 4 illustrates these concepts. In this figure, Control Point 1 and Control Point 2 make up the first control path.

Curve generation is accomplished through a finite differencing approach, limiting the curvature and rate of change of curvature to those physically realizable by the vehicle. Clothoid curves fit to control points would also be effective, but the required computation exceeds the time permitted in this application.

The way points are traversed in simulation along the path in the direction the vehicle drives. When a way point is tested in simulation and found to be unsafe, up to two new control paths may be generated. Each new control path contains an additional control point found by the planner that is intended to modify the path so as to direct the vehicle around either side of the obstacle encountered. These new control paths are added to a list of control paths. The planner then selects a new control path from this list based on the estimated cost of traversing the path. (The cost is a function of the complexity and length of the path.) This process repeats until a safe path of sufficient length is found. In theory, this algorithm should be capable of generating traversable trajectories through arbitrarily narrow admissable regions. In practice it is possible, given the known limitations in sensing resolution and vehicle control, to plan through a corridor as wide as the vehicle plus maximum map and path tracking uncertainties.

In the sections below, the test applied to each way point to determine safety, or admissibility, in a kinematic/static sense is discussed. Details of the algorithms for identifying control points that ensure safe passage around obstacles are also given

3.2. Kinematic Admissibility

Kinematic admissability implies that the geometry of the path during vehicular motion will permit traversal. Traversal can be impeded by an obstacle, such as a tree or a rock, or by vehicle geometry, such as a point below the minimum turning radius. Kinematic admissibility is the criteria for determining obstacles in the environment. Kinematic Admissability is defined as follows:

A way point is *kinematically admissable* if there is no geometric cause to impede the proposed motion of the body and if the vehicle and its environment cannot collide.

This definition is embodied in a set of four constraints, each of which is evaluated at a given way point to determine kinematic admissability. The way point is admissable only if it satisfies all four constraints:

- Minimum turning radius; the curvature of the path at the way point cannot exceed bounds given by the vehicle's minimum turning radius.
- Locomotion support; the motion of the vehicle must not be geometrically impeded by the presence of insurmountable obstacles in front of the wheels.
- Body collision; the vehicle cannot collide with its surrounding terrain.
- Unknown terrain; the vehicle must not be situated over unknown terrain (off of the map.)

Vehicle statics are a part of this kinematic admissability test. It is intuitive and straightforward to define static admissability as follows: a way point is *statically admissable* if the vehicle is in a state of static equilibrium at that point. The planner is conservative because the vehicle is required to be statically stable at every point along the path, even though there may be some executable paths that do not meet this requirement.

The notion of kinematic admissibility is applied to a path by simulating the vehicle at each point along a path on the elevation

for one or more the following reasons:

- Polygonal obstacle assumption is not appropriate for rough terrain, since the motion of an obstacle can include any pose (such a steep terrain) that is unsafe. This set of poses can be computationally prohibitive to precompute in it's entirety and may not assume a polygonal shape.
- A limited sensor horizon precludes knowledge of the entire search space before planning.
- The intrinsic limitations of the vehicle's ability to follow the given path are not considered (e.g. turning radius).
- Complete configuration space generation consumes too much time to permit continuous motion at desired speeds.

Work by Olin[10] & Daily[3], relaxes the flat world, polygonal obstacle assumption and introduces the notion of simulated traversal of paths, taking into consideration kinematic constraints at discrete points along these simulated planned paths. Their system, however, limits the space of potential paths by only considering a fixed set of paths across the sensed area. This methodology does not allow small deviations around obstacles, and prevents more than a single change of direction within a planning cycle. This planner might find a region uncrossable, simply because its static space of predetermined paths is insufficient to include a safe path. Furthermore, this system does not consider dynamics, as it operated in a velocity region where dynamic effects can be neglected.

Thompson[12] considered the need for a robust searching algorithm for vehicle navigation within a generate and test scheme. Previously, others [Lozano-Perez[8]] have generated the entire space of segments first, then searched a connected graph for a complete path. Thompson's algorithm generates segments only as the search reaches them, and thus avoids generating many unneeded segments. However, he limits his notion of unsafe regions to polygonal areas which is not sufficient for our purposes, as discussed above. Feng[5] recognized the need for planning algorithms to handle vehicle dynamics in changing environments. However, his approach is limited to polygonal obstacles. Computational limitations at higher vehicle speeds prevents *a priori* generation of unsafe regions and the grouping of these regions into polygons

The trajectory planning software on the Navlab II considers both kinematic and dynamic constraints on the vehicle, plans paths through tight spaces, copes with a changing sensor horizon, and accounts for the intrinsic ability of the vehicle to track a given path. The planner performs spatial searching based on kinematic constraints and then applies dynamic constraints to compute the temporal portion of the path, that is, the speed at each spatial point along the planned path. This section discusses the aspects of spacial planning while the following section (Section 4.) is concerned with temporal planning.

3.1. Spatial Trajectory Recursive Generation

To plan paths in natural terrain, a recursive generation paradigm is used to choose and evaluate successive paths until a safe path which traverses the known terrain is found. This method avoids the problem of preprocessing the terrain to find all unsafe regions. The generate and test method also permits the planner to compute any path needed without limiting the search to a precomputed set. This algorithm, referring to the example in Figure 3, plans BE while the vehicle drives AB.

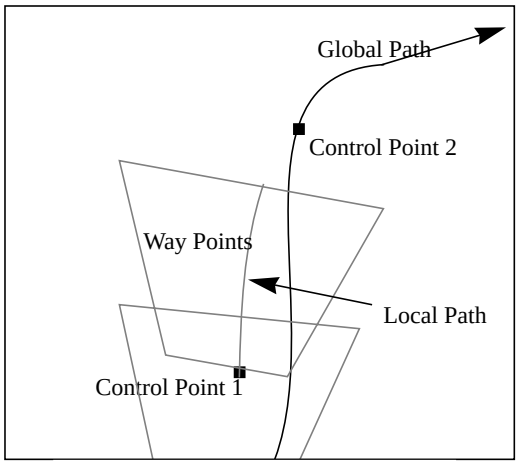


Figure 4. Global, Control, and Way Points

Under normal operation, the vehicle will never drive along a stopping segment, but rather will drive along one driving segment after another. If an impasse, such as a software failure or an unavoidable obstacle, prevents the next path from being planned during the drive along AB, the vehicle will travel along the current path on BC, coming to a stop at point C. The stopping segment of every path is tagged with decreasing velocities to stop the vehicle by the end of this segment. An example of speed profiles of stopping and driving segments is also shown in Figure 3.

Thus, in the example given, if planning during the drive on AB fails, the vehicle drives on BC and stops, never having ventured off of known terrain. If planning succeeds, the vehicle drives on AB of the current path and then on BD of the next path. This planning cycle repeats, planning a path across the map generated from an image taken at B while driving BD and so on. In the case of an impasse, the execution of a stopping segment across known terrain ensures vehicle safety, an essential feature.

The maximum speed (v) of the vehicle can be computed by considering various physical constraints. Sensing and planning requires an execution time of t_p = 1.0 seconds. The maximum deceleration of the vehicle is a = -0.2g. The length of the ERIM image is x_{AC} = (19-7)m = 12m. Assuming the vehicle travels at constant velocity across AB, and then at maximum deceleration across BC, the distance AB is given by:

$$x_{AB} = vt_p \tag{1}$$

The distance travelled during deceleration is given by:

$$x_{BC} = -\frac{(v)^2}{2a} \tag{2}$$

From Figure 3, $x_{AB} + x_{BC} = x_{AC}$, and solving for v yields:

$$v \le at_p + \sqrt{(at_p)^2 - 2ax_{AC}} \tag{3}$$

However, due to the electronic latency
of the ERIM, images previously assumed to have been taken at the beginning of the path may actually have been taken up to t_s = 0.5 seconds earlier. Furthermore, since the ERIM takes 0.5 seconds to scan from top to bottom, the resulting image is reduced in depth by the distance the vehicle covers during that half second. Thus, the effective map width becomes x_{AC}'= $x_{AC} - t_s v - 0.5v$, so the revised equation for v (rearranging terms) is,

$$v \le a\left(t_p + t_s + \frac{1}{2}\right) + \sqrt{\left(a\left(t_p + t_s + \frac{1}{2}\right)\right)^2 - 2ax_{AC}} \tag{4}$$

Substituting the given values for a, t_p, t_s and x_{AC} gives v_{max}=3.98 m/s.

For different geometric configurations, including significantly sloped terrain, these geometries change, and therefore the maximum velocity varies. Also, the density of unsafe regions effects planning time. In benign terrain, higher speeds are possible.

This algorithm provides several guarantees. First, the vehicle can operate in an uncertain environment since each extension to the current path provides new sensor information. This allows the planned path to change radically when unknown obstacles are introduced into the terrain map. Second, through the inclusion of a stopping segment, the system ensures vehicle safety in case of a planning failure, which is essential for high speed driving in rough terrain.

3. SPATIAL TRAJECTORY GENERATION

Path planning is defined as finding a path through space from a start configuration to a goal configuration while avoiding unsafe configurations. Much of the previous work in this area[8], addresses path generation for a polygonal body through a space containing polygonal obstacles. This approach assumes that all obstacles in the traversable space between start and goal are known. This approach, and other similar configuration-space type approaches[2], do not satisfy the requirements for ACCN,

2.2. Sense-Plan-Drive Cycle

In order to traverse unknown terrain, the navigation software must first sense the terrain in front of the vehicle. It then plans a trajectory across this terrain map and drives along the path. This process cycles as the vehicle drives. The sensing consists of taking the latest available range image from the ERIM, and transforming the data into a $2\frac{1}{2}$ D discretized terrain map. This operation requires approximately 0.5 seconds. Kelly[7] discusses this perception system in detail.

The planner follows a recursive generation paradigm. In the spacial planning phase, a path is generated and then tested through simulated terrain traversal for safety. If it fails, more paths are generated and tested, each further modified to avoid obstacles on the known terrain, until a safe path is found. This portion of the planning considers only the kinematics constraints on the vehicle. In the temporal planning phase, the speed of the vehicle at each point along the path is determined, turning the spatial path into a spatio-temporal trajectory. This phase considers the dynamic constraints on the vehicle. The separation of dynamics and kinematics makes the planning problem tractable. Computation time for planning is about 0.5 seconds.

The trajectory is passed to the path tracker, which tracks the path by issuing steering and velocity commands to the vehicle controller. These components are illustrate in the software system design (Figure 2). Previous work by Singh[11] and Amidi[1] addresses issues involving the tracker and controller.

2.3. Planning Configuration

The region of known terrain in front of the vehicle is shown in Figure 2. The goal of the planner is to extend the known safe path for the vehicle by planning a path across recently sensed terrain. Figure 3 illustrates the geometric constraints on the planning problem. (Note the vertical offset between map boundaries and between the current and next path is for illustrative purposes only.) The goal of the planner is to plan the *Next Path* while the *Current Path* is traversed. Each next path corresponds to a path across a map generated from an image taken at the first point on the current path.

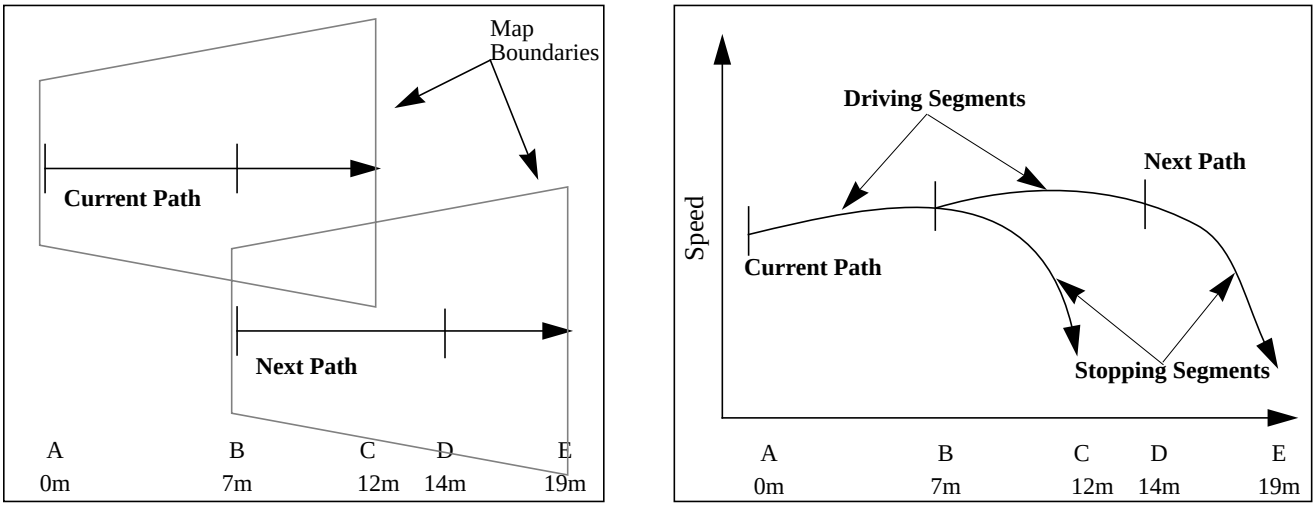


Figure 3. Geometry for Planning

Assume the vehicle is driving on the given current path at point A and a range image is taken and processed. Point B corresponds to the proximal bound of the terrain sensed in this image and point E corresponds to the distal bound. Likewise, Points A and C are the proximal and distal bounds of the map segment previously used to plan the current path, AC. While the vehicle continues driving from A along path AB, the system plans path BE across the newly sensed terrain. To ensure continuity between the current and next path, point B is constrained to have the same position, heading and curvature on both paths.

The planned path (BE) has two parts, a *driving segment* and a *stopping segment*. Segments BC and DE correspond to the stopping segments of the current and next paths respectively and are long enough to permit safe stopping from the vehicle's projected speed. Similarly, AB and BD correspond to the driving segments.

As a result, sophisticated planning is needed, and the system must cycle quickly to cope with limited sensor horizons. On a smooth gently sloping streambed with sparse obstacles, the JPL rover has travelled 100 meters in a start/stop mode.[13] On rougher terrain, the ALV has reached speeds of 3.5 km/hr over distances of several hundred meters.[3,6,10] The speed of the ALV is low enough that the stopping distance is negligible, diminishing concerns for vehicle safety. Furthermore, this slow speed eliminates the need to model dynamics in the planning process. This paper presents a planning system which addresses the five problems introduced above, thus permitting ACCN in natural terrain at speeds higher than that of previous systems.

This paper is divided into sections concerning an overview, spacial planning, temporal planning, results and conclusions. The overview of the system (Section 2.) describes the hardware and software architecture, and also includes an analysis of the geometry intrinsic to this planning problem. The spatial planning section (Section 3.) first presents a summary of the planning algorithm, and then details the algorithm used to plan spatial paths around obstacles. A discussion of what is meant by "obstacle" is also included. The temporal planning section (Section 4.) describes the methodology used to assign vehicle speed to planned paths. Finally, results of recent runs and conclusions are presented (Sections 5 & 6.)

2. SYSTEM OVERVIEW

2.1. Vehicle and Sensor

The testbed for the planner is the Navlab II, a computer-controlled High Mobility Multi-purpose Wheeled Vehicle (HMMWV.) The equipment on the Navlab II can be divided into three categories: computation, perception, and actuation. For computation, there are three Sparcstation 2's connected by Ethernet. For perception, an ERIM (Environmental Research Institute of Michigan) two-axis scanning laser range finder is used. Position estimation is accomplished through odometry dead reckoning and through inclinometers. Actuation consists of controls on the throttle, steering and brakes, with feedback from encoders on the transmission. The vehicle is self-contained; there are no provisions for off-vehicle communications or power supply while in the field.

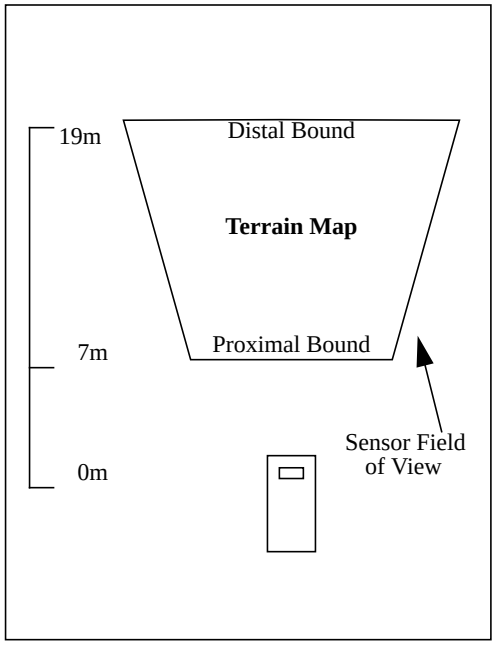


Figure 1. Sensor-View/Vehicle Relationship

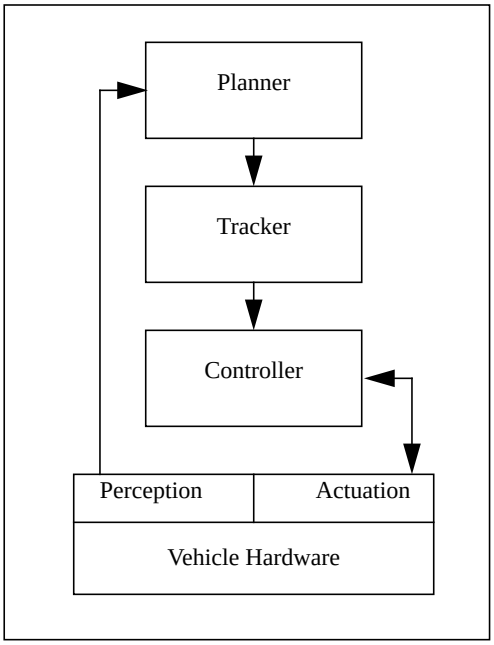


Figure 2. Software-Hardware Interaction

The ERIM sensor returns a 2-d range image of terrain in front of the vehicle. The sensor is positioned about 3 meters off the ground, and 1.0 meters inset from the front of the vehicle. On flat terrain, the proximal bound of the sensor viewframe projected onto the ground is approximately 7m from the sensor, while the distal bound is near 19m (Figure 1). This distance is affected by the shape of the terrain and by the 19.5m current usable range of the sensor itself. The sensor is used to build terrain elevation maps of cartesian grid cells of size 0.34m on a side and is capable of providing two range images per second.

Dynamic trajectory planning for a cross-country navigator

Barry L. Brumitt, R. Craig Coulter, Anthony Stentz

Robotics Institute, Carnegie Mellon University
Pittsburgh, Pa. 15213

ABSTRACT

Autonomous Cross-Country Navigation requires planning algorithms which supports rapid traversal of challenging terrain while maintaining vehicle safety. The central theme of the work is the implementation of a planning system which solves this problem. The planning system uses a recursive trajectory generation algorithm, which generates spatial trajectories and then heuristically modifies them to achieve safe paths around obstacles. Velocities along the spatial trajectory are then set to insure a dynamically stable traversal. Ongoing results are presented from the system as implemented on the NAVLAB II, an autonomous off-road vehicle at Carnegie Mellon University. The planning system was successful in achieving 5.1km autonomous test runs with obstacle avoidance on rugged natural terrain at speeds averaging 1.8 m/s. Runs of up to 0.3km at 4.5m/s were acheived when only checking for obstacles

1. INTRODUCTION

Autonomous Cross-Country Navigation (ACCN) can be applied to tasks such as military reconnaissance and logistics, medical search and rescue, hazardous waste site study and characterization, and to industrial applications, including automated excavation, construction and mining. The need to safely travel at a reasonable velocity on natural terrain with unknown obstacles is common to all these endeavors.

This paper addresses the problem of trajectory planning to support the fastest possible continuous autonomous navigation on rough terrain. This task poses new issues not encountered in slower navigation systems on flat terrain.

- Uncertain environment: sensing must be done simultaneously with driving; a precomputed path is impossible.
- Vehicle safety: if computing fails, or a safe trajectory cannot be found, the vehicle must be brought to a stop to avoid collision.
- Computation time: as the vehicle speed increases, planning must be performed more quickly.
- Complex terrain: the geometry of the terrain is sufficiently complex that the assumption of flat terrain with sparse obstacles does not suffice.
- Dynamics: the vehicle's speed is large enough that dynamics as well as kinematics must be considered in vehicle models.

Early work in outdoor navigation includes the Stanford Cart[9]. This system drove in a stop-and-go manner, at a slow speed and modelled terrain as flat with sparse obstacles.

Higher speeds have been acheived in systems operating in more structured environments. Road following systems assume benign and predictable terrain. Obstacles are assumed to be infrequent, so that bringing the vehicle to a stop rather than driving around them does not significantly degrade performance. This assumption negates the need for complex path planning. In this way, speeds of up to 100 km/hr. have been achieved in road following[4] using cameras, and speeds up to 30km/hr. have been achieved on smooth terrain with sparse obstacles using a single scanline lidar[11]. While these systems consider problems of dynamics and vehicle safety, they do not consider the need for planning continuous paths around obstacles in natural terrain.

When attempting to navigate over unstructured natural terrain, no assumptions about the shape of the terrain ahead can be made. Obstacles include both discrete objects, but more generally, include unsafe configurations of the vehicle on the terrain.