Perception, Planning and Control for Walking on Rugged Terrain

Reid Simmons Eric Krotkov

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

The CMU Planetary Rover project is developing a six-legged walking robot capable of autonomously navigating, exploring, and acquiring samples in rugged, unknown environments. To gain experience with the problems involved in walking on rugged terrain, we built a full-scale prototype leg and mounted it on a carriage that rolls along overhead rails. This paper describes issues addressed in developing the software system to autonomously walk the leg through rugged terrain. In particular, we describe the insights gained into perceiving and modeling rugged terrain, controlling the legged mechanism, interacting with the ground, choosing safe yet effective footfalls, and planning efficient leg moves through space.

1 Introduction

The CMU Planetary Rover project is constructing the Ambler, a walking robot designed for planetary exploration [2]. The configuration is a six-legged vehicle with orthogonal legs and an overlapping gait [1]. These features are designed to maximize power usage and to simplify planning and control. To meet its mission goals, the Ambler must be able to autonomously traverse rugged and often uncertain terrain, while maintaining a stable platform for its sensors and scientific equipment.

A single leg of the Ambler was built and suspended from a carriage attached to overhead rails. We developed a distributed software system that integrated perception, planning, and real-time control to autonomously walk the mechanism through a variety of obstacle courses [6, 10]. The rationale was that ideas would be easier to develop using just a single leg, and that many of the concepts would transfer to the full six-legged walker.

This paper reports on our initial experiences using the single leg of the Ambler. It focuses on the special problems encountered in perception, control and planning for rough terrain walking. In particular, we discuss the problems of modeling 3D terrain, detecting and controlling forceful interaction with terrain, and planning steps that lead to a balance between efficiency, risk, and progress of the mechanism. Readers interested in more details of the single-leg walking system should consult [6, 10].

2 Single-Leg Testbed

A single leg of the Ambler (based on an early design [2]) was built to experiment with mechanism control and system integration before committing to the fabrication of a six-legged vehicle. The leg (Figure 1) has a working radius of approximately 2.5 meters and a vertical range of travel of about 1.5 meters. The dimensions were chosen to enable the Ambler to meet its design objectives of crossing one meter wide ditches and stepping over one meter high obstacles. The leg is supported by a carriage mechanism that is mounted on a pair of rails. The carriage can roll along the rails, providing one degree of translational freedom, and the leg can rotate freely under the carriage. The support system is designed to be statically and dynamically stable, and to allow the leg to walk in a manner sufficiently similar to the Ambler so that

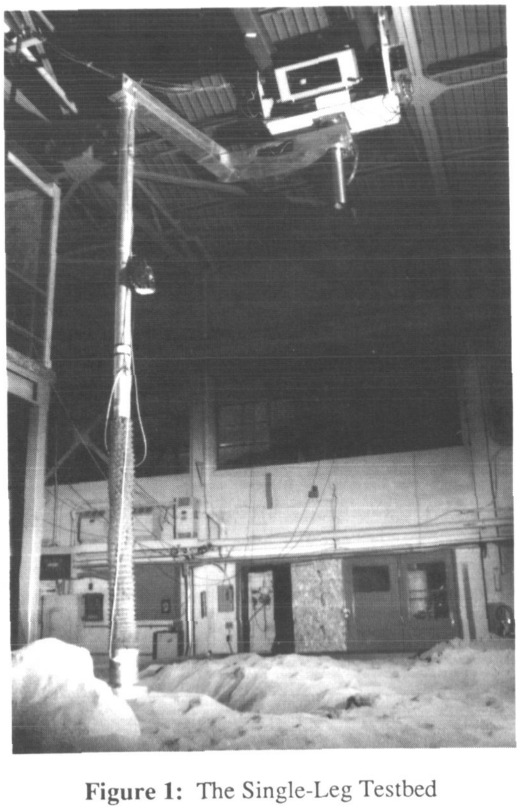


Figure 1: The Single-Leg Testbed

ideas generated could be easily transferred to the six-legged machine.

Sensors attached to the leg include a potentiometer to measure the position and velocity of the carriage along the rails, incremental and absolute encoders to measure leg positions, and two inclinometers to measure the rotation of the carriage. In addition, a six-axis force/torque sensor is attached to the bottom of the leg to measure the forces experienced by the mechanism as it moves.

A scanning laser rangefinder, manufactured by Erim, is fixed to the carriage (Figure 1). The scanner can acquire 64 by 256 pixel range and reflectance images in half a second. It digitizes to 8 bits with a range ambiguity interval of approximately 20 meters. This provides a range resolution of approximately 7.62cm. The measurements cover 80 degrees in the horizontal direction (azimuth) and 30 degrees in the vertical direction (elevation).

To provide for a variety of "Mars-like" terrains, we constructed an obstacle course below the rails measuring approximately 11 by 6 meters (Figure 2). The course is filled with over 40 tons of sand. Terrain features are introduced by resculpting the surface to form hills and trenches, and by placing objects on the sand. We have used styrofoam boulders, traffic cones, and large boxes to test the ability of the system to navigate over and around obstacles.

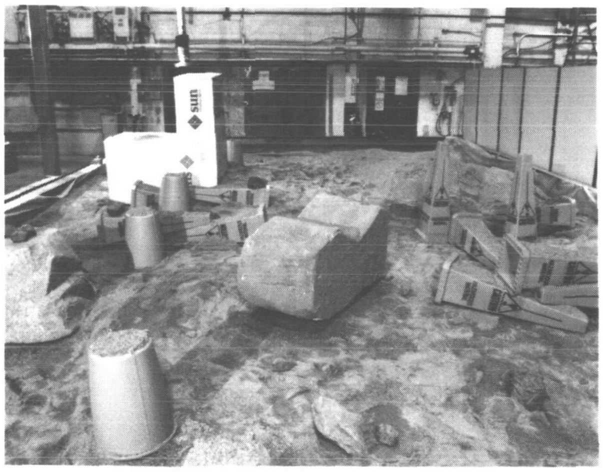


Figure 2: A Typical Arrangement of the Obstacle Course

3 Rough Terrain Walking

The single-leg walking system consists of five distributed modules (Figure 3) integrated by the centralized *Task Control Architecture (TCA)* [9, 11]. The modules communicate with one another (and with the TCA central control modules) by passing messages through the central control, which routes them to the appropriate modules and message handlers. TCA is basically a high-level robot operating system that provides utilities for building and coordinating mobile robot systems. The utilities are meant to bridge the gap between task-level planners and real-time control systems. In particular, TCA

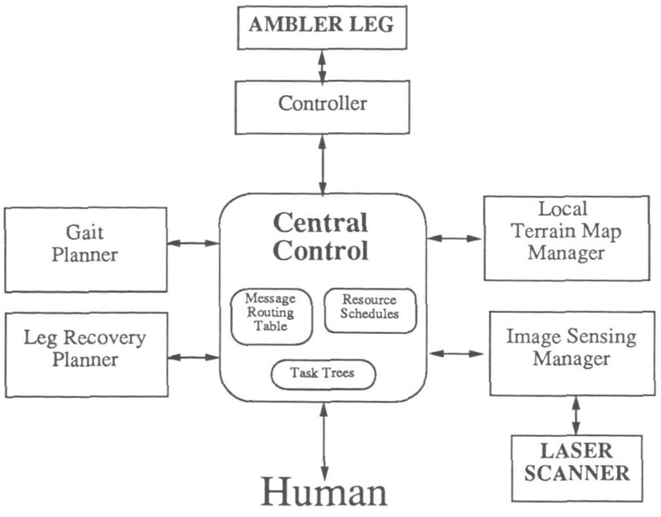


Figure 3: Modules for the Single-Leg Walking System

supports 1) distributed processing, 2) resource management, 3) hierarchical task decomposition, 4) temporal synchronization of tasks, 5) execution monitoring, and 6) error recovery.

The *Controller* module (Figure 3) handles all robot motions and responds to queries from other modules regarding leg position, carriage position and orientation, and force sensor readings. The Controller runs under the real-time vxWorks^{tm} operating system. The *Image Sensing Manager (ISM)* acquires scanner images from the Erim and determines the transformation from scanner to world coordinates. For debugging purposes, the ISM can also access images stored on disk. The *Local Terrain Map (LTM) Manager* processes scanner images to construct elevation maps of the terrain. The *Gait Planner* plans where to place the foot and how far to move the carriage in order to advance with minimal risk to the mechanism. The *Leg Recovery Planner (LRP)* determines a trajectory to the planned footfall location that is energy and time efficient and that avoids terrain collisions.

To walk the leg down the obstacle course, the user inputs a goal location along the rails. The walking system is totally autonomous from that point on. A message to plan (and execute) the walk is sent to the Gait Planner module. If the carriage position is close enough to the user-chosen goal, the Gait Planner signals success. Otherwise, it requests from the LTM Manager a terrain elevation map and a map that evaluates the potential support for the leg at various footfall locations. If the carriage position has changed from the last time a map request was issued, the LTM Manager requests a new scanner image from the ISM. In either case, the requested maps are constructed and sent back to the Gait Planner.

The Gait Planner combines constraints imposed by the terrain and footfall maps with geometric constraints on the leg's movement, and chooses the location that minimizes a weighted sum of the constraints. Based on the chosen footfall location, the Gait Planner chooses a body move that maximizes forward progress. The Gait Planner then sends the chosen footfall and body move to TCA, and then sends itself a message to plan the next step.

TCA forwards the footfall location to the LRP. The LRP uses a terrain map obtained from the LTM Manager to plan an obstacle-free trajectory. The trajectory is then forwarded through TCA to the Controller, which executes the trajectory and plants the foot at the desired location. After a successful leg move, TCA forwards to the Controller the body move generated by the Gait Planner. The Controller exerts enough force to compress the terrain, then relaxes to a force sufficient to provide traction. The horizontal (shoulder and elbow) joints are then actuated to drive the carriage forward. Finally, tension built up in the leg as a result of the body move is relieved, so that the leg does not slip when it is next lifted. At this point, the TCA forwards the message to the Gait Planner to plan out the next step.

Figure 4 presents a time breakdown by module for traversing a typical obstacle course. The system takes six steps in 13.5 minutes while covering about 8 meters (60cm/min). The darkly shaded areas of the chart represent times when a module is computing; lightly shaded areas are times when a module is awaiting a reply from another module. To reduce the chart's complexity, the 71 leg and body position queries to the Controller are not illustrated. In any event, they have a negligible effect on the timings since they are handled in less than 50msec each.

Figure 4 indicates that about 60% of the time is spent by the real-time Controller in moving the leg and carriage. Conversely the ISM, which spends only one half second for each of the seven images it acquires, is nearly always idle. Our measurements also show that the TCA central control module accounts for only about 3% of the total operating time. While, in theory, routing all messages through a central process could be a bottleneck, the evidence indicates that it is not a problem for this system.

We have used the walking system described above to navigate the single leg through a number of complex obstacle courses, such as illustrated in Figure 2. While not perfect (primarily due to sensor and mechanism inaccuracies), the system is generally successful at navigating the courses. The remainder of this paper describes perception, control and planning issues that we addressed in getting the system to walk on rugged terrain.

4 Issues in Perception

We use a scanning laser rangefinder because of the scanner's ability to directly recover the three-dimensional structure of the environment. Therefore, terrain maps can be constructed more rapidly and reliably than by passive vision techniques, such as binocular stereo or motion. In addition, using a laser scanner will enable the Ambler to walk at night. Although the scanner consumes more power than other imaging techniques, we believe its speed and accuracy more than offset this disadvantage.

Our primary terrain representation is an *elevation map*. An elevation map is a rectangular grid of real values, corresponding to the height of the terrain at a representative point within each grid cell (our current implementation uses the mid-point of the cell). Grid cells outside the scanner field of view are labeled *unknown*, and cells occluded by other objects are labeled as such, along with the maximum known elevation of the cell, given the available information. The map also contains an estimate of the uncertainty of the elevation value at each grid cell.

We chose to use elevation maps because 1) they provide a representation that is appropriate for a wide variety of tasks, 2) they can be constructed at multiple levels of resolution, 3) they are simple to manipulate, and 4) they can be accessed in a simple way (by a polygon that encloses the region of interest). A disadvantage is that our elevation maps record just a single value for each grid cell, hence overlapping objects (such as trees) cannot be represented. We do not, however, view this as a serious problem for navigating on Mars.

The LTM Manager uses the Locus Method to transform the raw range images into elevation and uncertainty maps [7, 5]. The Locus Method efficiently interpolates range data points to compute an evenly spaced grid of elevation points (Figure 5).

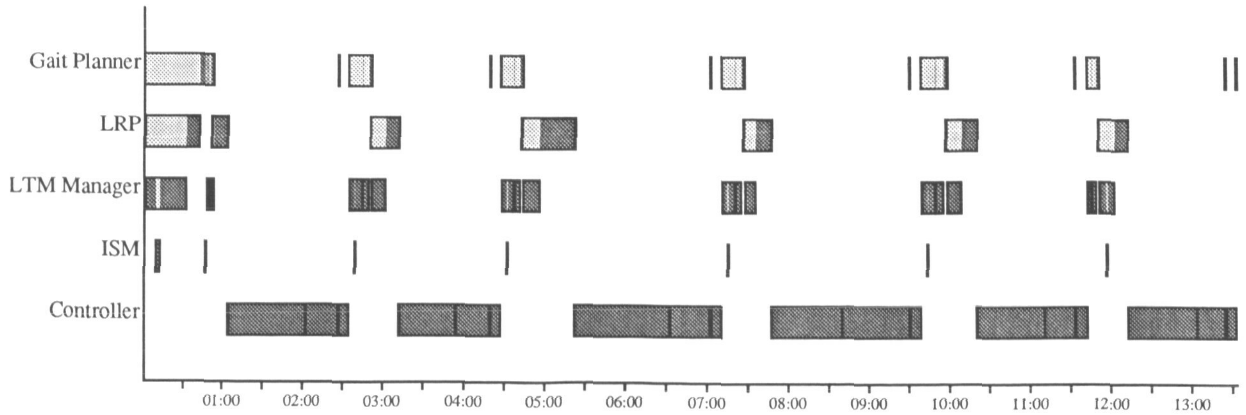


Figure 4: Timing Chart for a Typical Run

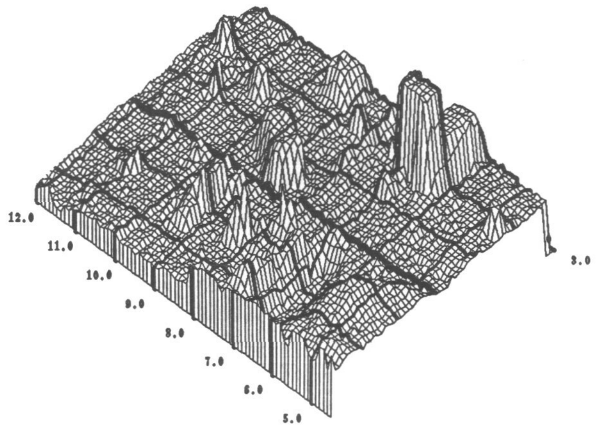


Figure 5: Elevation Map of Obstacle Course in Figure 2

The maps created from the most recent images are then merged with the current elevation maps using maximum likelihood estimation techniques. The merging operation is necessary because maps created from a single image do not, in general, have a wide enough field of view to support the necessary planning tasks. In planning a leg trajectory, for example, the LRP must take into account obstacles below and behind the vehicle. Because the scanner looks forward, the map constructed from the most recent range image cannot possibly cover this area, and so the planner needs a map constructed from a number of past images.

The portions of elevation maps requested by the planners are computed on demand, but are cached so that future queries that request the same (or overlapping) regions do not have to recalculate the values. Along with caching maps, we need a means of uncaching as the maps become larger than available local memory. The LTM Manager maintains a 20 by 20 meter window centered around the vehicle, outside of which grid cells are paged out, or clipped. While this method of computing on demand and caching is quite efficient, we are looking at pre-computing some maps concurrently with the planning and execution of walking commands.

5 Issues in Mechanism Control

The major issue we addressed in controlling the mechanism was the forceful interaction of the leg with the terrain. This impacted the leg and body move procedures, and also error detection and recovery performed by the real-time control system.

Moving the leg through free space posed few problems. The leg is moved through a series of user-supplied way-points, which are given in joint space. The Controller calculates the amount of time required for the slowest joint to move between successive way-points and then scales the speeds of the other joints so that all joints arrive at each way-point simultaneously. To smooth the motion, the way-points are linked with constant velocity segments connected together by constant acceleration segments [4].

For contacting the terrain, the motion command specifies that the last way-point is to be made in *transition mode*. In transition mode, the force/torque sensor is monitored and the motion is stopped if a specified (user-settable) force is achieved before the actual way-point is reached. If the way-point is reached first, a failure message is issued to TCA.

One problem encountered early in our experiments was the tendency of the leg to hit terrain features, even though obstacle-free paths were supposedly being followed. This was traced to inaccuracies in our kinematic model of the leg: we had initially assumed a rigid body, but the length of the leg and its method of connection to the rails led to a large amount of compliance in the mechanism. We partially solved this problem by measuring the deflections in the leg and updating the kinematic routines using a simple deflection model fit to the data. This improved the accuracy of the leg moves, as measured in Cartesian space, from about 20cm down to about 5cm.

More troublesome was the body move procedure. Our initial implementation commanded the position of the horizontal joints to follow a linear trajectory. This procedure proved to be very inaccurate due to the compliance of the mechanism, friction between the carriage and rails, and compliance of the terrain. We often witnessed errors of more than 40cm over a (commanded) one meter body move.

Our remedy was to use a velocity, rather than position, control procedure. To move the body, the force on the leg is first increased to 800 pounds, to compress the underlying terrain. The force is then relieved to 500 pounds, which provides sufficient tractive force. The shoulder and elbow joints are then commanded to achieve given velocities. First, the Cartesian velocity of the carriage is computed as a clipped, linear function of the error between the present carriage position (as read from the potentiometer) and the commanded goal position. This velocity is then converted into joint velocities using an inverse Jacobian function. The body move control loop is operated at a frequency of about 60 Hz, which differs sufficiently from the natural frequency of the system so that resonance does not occur.

This velocity-controlled body move procedure is accurate to within 5cm. The algorithm was subjected to extensive testing to gain confidence in its performance. Over 1000 moves were performed with the leg starting at various X, Y locations relative to the carriage. The resultant data not only confirmed the general accuracy of the body move procedure, but also provided a ''cost map'' for the Gait Planner to indicate how far the carriage can reliably advance from different footfall locations (see Section 6).

During a body move, the compliance of the mechanism causes overshoot of the expected positions of the joints, assuming a rigid kinematic description of the leg. This overshoot takes the form of stored strain energy which causes the foot to drag across the terrain when the leg is next lifted. To prevent this, the tension is relieved by adjusting the final joint angles to correspond with the expected Cartesian position of the leg.

The control system also contains several procedures for detecting and reacting to errors. The joint limit sensors and the motion control cards are continually monitored for possible failures. During the body move control loop, the system monitors the forces exerted on the foot. The leg is stopped if the force drops off rapidly, indicating that the foot may have broken free. As described above, the force sensor is also monitored during transition mode to detect when the terrain is contacted.

When errors are detected, the Controller halts any ongoing leg motions and informs TCA, which passes the failure message on to the appropriate exception handler. In addition, the control software permits recovery from hardware errors without restarting the entire walking system. Such errors include tripping limit switches, amplifier faults, servo errors, excessive force readings, and ''kill'' messages from users.

6 Issues in Planning

Planning problems for single-leg walking include deciding where to plant the leg, how to move it through space, and how far to move the carriage at each step. Our approach utilizes constraints imposed by the robot's design to plan movements that are efficient, reliable, and provide a good rate of progress for the mechanism.

The Gait Planner plans footfalls by combining various geometric and terrain constraints. For each constraint, a *cost map* is created that indicates the goodness of the constraint within each grid cell (Figure 6). The cost maps are combined using a weighted sum, and the grid cell with the lowest cost is chosen as the footfall location. The Gait Planner then chooses a body move that is the minimum of 1) the best possible advance from the chosen location, and 2) a user-defined threshold (we typically constrain the body advance to 1.5m to get a reasonable number of footfalls over the length of our testbed).

The constraints used by the Gait Planner were derived from both analysis and experimental evidence. The geometric constraints include 1) the mechanical limits of the leg, 2) how far the carriage can travel from a given footfall location, which is based on empirical values derived from testing the controller's body move algorithm (Section 5), and 3) the visibility of the leg in the scanner field of view, to avoid occluding terrain. Terrain constraints include 4) the terrain elevation, since the leg cannot reach areas that are too high or too low, 5) an evaluation of the flatness of the terrain around each grid cell [3], since relatively flat terrain is preferable both for stability and for providing traction in moving the body, and 6) the closeness of the footfall location to adjacent obstacles, in order to compensate for inaccuracies in the mechanism control and scanner resolution.

In combining the cost maps, constraints 1 and 4 above are used as binary constraints: if the location is not reachable, it is eliminated from consideration, no matter what the other values are. The remaining two terrain constraints are given high weights relative to the remaining two geometric constraints. This reflects our concern for the safety of the machine over its progress.

Advantages of this constraint-based approach are that 1) the planner does not have to commit *a priori* to which constraint is most important, and 2) it is easy to add new constraints as relevant ones are identified [12]. Although this approach evaluates a large number of grid cells, in practice the gait planning is fast relative to other computations.

Once the Gait Planner decides where to put the foot, the LRP determines the trajectory that will get the leg to that position without hitting any obstacles. The LRP uses the novel *Envelope Trajectory Finding Algorithm (ETFA)* to find time and energy efficient moves through 3D space, while searching only a 2D grid. The LRP starts by creating a configuration search space for the elbow and shoulder joints [8], dividing the

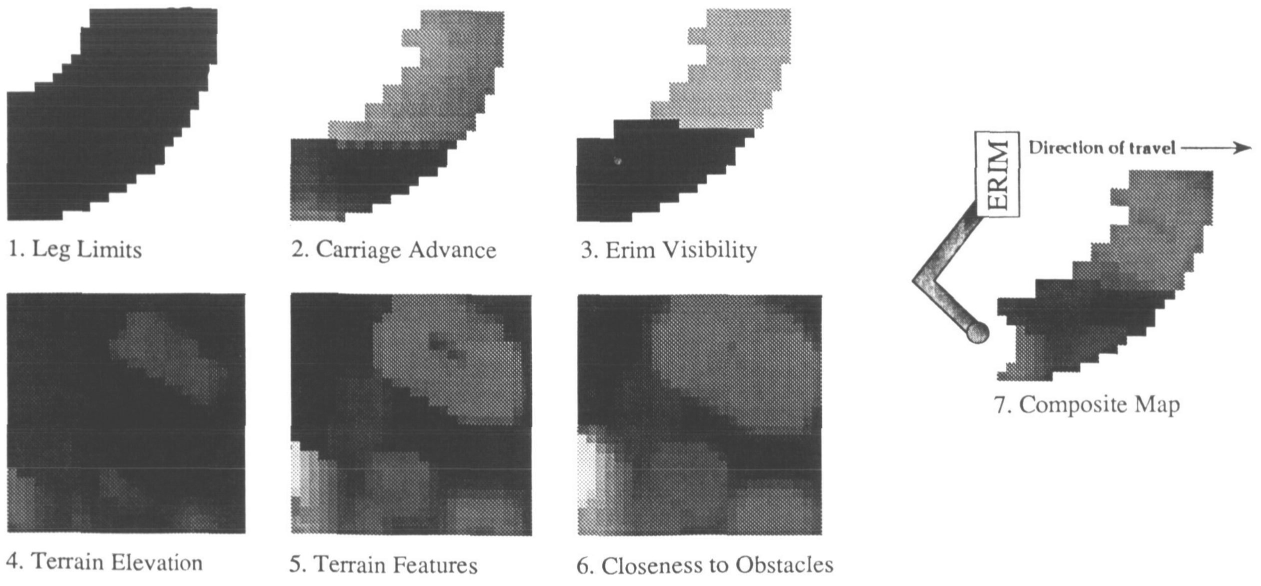
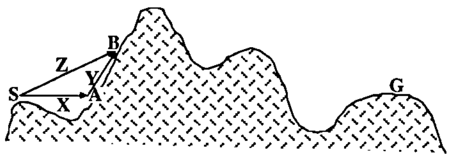


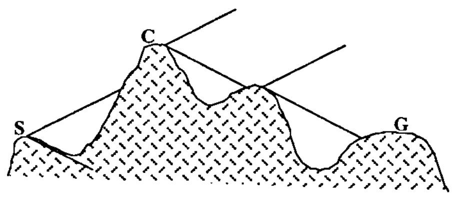
Figure 6: Constraint Cost Maps for Choosing a Good Footfall
(darker shades indicate better footfall locations)

space into a discrete grid approximately 0.1 radian wide. The LRP fills the grid with obstacles, growing the terrain features and other legs (for the six-legged case) by the radius of the foot plus an uncertainty factor.

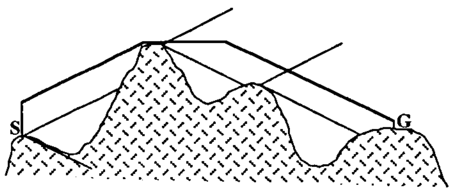
To search the space, the ETFA needs to estimate the energy and time needed to travel between grid cells. The energy consumed is estimated simply as the sum of the energy needed to move the elbow and shoulder joints to a cell, plus the energy needed to raise the leg above the terrain elevation at the cell. While this assumes that the power consumption in each joint is independent, it is a reasonable approximation given the slow speeds of our mechanism.



a. Different Trajectories



b. Trajectory Envelopes



c. Final Path

Figure 7: The Envelope Trajectory Finding Algorithm

It is more difficult to estimate the time needed to get to a cell, as Figure 7a illustrates. If we just add up the times to get to each individual cell, path **X** is the quickest way to get to point **A**. To get a little further to point **B**, however, path **Z** is faster than **X** followed by **Y**, since in path **Y** the horizontal joints must stop and wait for the vertical lift, while in path **Z**, the leg is lifting while it is moving horizontally.

In essence, we need to keep track of all possible paths that the leg can take in reaching a particular grid cell. This is what the "envelope" part of the ETFA is about. The algorithm keeps track of the maximum and minimum heights that the leg can reach in any particular cell, assuming that the leg lifts/lowers at full speed while moving horizontally (Figure 7b). Thus, the leg can reach anywhere within the envelope in the same amount of time. Only if the terrain is above the top of an envelope (e.g., point **C**) does the leg have to stop moving horizontally and lift.

The ETFA finds the minimum-cost trajectory using A* search and a weighted sum of the energy and time metrics described above. At the end of the search, the planner determines an actual trajectory through the envelope space by choosing vertical moves that minimize the risk to the machine while maintaining the optimality of the path found. In particular, this means performing all purely vertical lifts at the start and delaying all purely vertical descents until the end of the move (Figure 7c).

In actual use, the Gait Planner performs very well, typically choosing safe footfalls that skirt obstacles, while enabling the carriage to be moved at, or near, its maximum advance. The LRP typically chooses trajectories that hug the ground when the terrain is relatively flat. For obstacle-filled terrain, the LRP typically chooses to go around, rather than over, large obstacles, since the vertical joint of the leg is much slower than the two horizontal joints.

7 Conclusions

To date, the leg has autonomously traversed several hundred meters through various obstacle courses. The effort has taught us much about perception, locomotion, and planning for rugged-terrain walking, lessons that apply to the full six-legged Ambler.

Perhaps the most important result is that our experience with the single-leg testbed has led to some significant changes in the configuration of the Ambler, especially with regard to compliance. The single leg was too flexible to permit the type of accurate control needed to negotiate very rugged terrain. The legs of the new Ambler design are extremely rigid [1]. Our experience to date with the full Ambler indicates that we can do leg and body moves to within a centimeter of commanded positions. In any event, we believe our experience with the single-leg testbed will enable us to handle any residual compliance.

As for the software system, the Task Control Architecture has been ported to the Ambler without any modifications. The LTM Manager and ISM needed only minor modifications to handle the new Ambler geometry. The Erim scanner itself, however, was found to have insufficient resolution and accuracy for our purposes. While this did not prevent successful walking, it did limit the roughness of the terrains that the system could traverse. For the six-legged Ambler we have procured a scanner, manufactured by Perceptron, that overcomes most of these problems.

One surprise in the endeavor was the fine balance between geometric and terrain constraints for gait planning. Much of our effort in getting the leg to negotiate terrain was in fine-tuning the weighting function that combined constraints. Our current methodology is empirical: trying the system on a variety of terrains and tweaking the weights to reflect the results of the experiments. To make the process of choosing weights less *ad hoc*, we are considering the use of adaptive algorithms that autonomously adjust the constraint weights based on the difference between the planned moves and actual outcomes. Another problem was that the footfall evaluation constraints used did not always yield what we subjectively believed to be the best footfall location. We are currently investigating a more feature-based approach to provide better evaluations. In general, gait and footfall planning are areas of on-going research and will undoubtably consume much of our effort in getting the Ambler to walk on rugged terrain [13]. We believe, however, that the constraint-based structure of the Gait Planner will enable us to experiment with various constraints and weighting schemes without much alteration to the basic planning algorithm.

Error detection and recovery is an important area that, to date, has received only modest attention by our group. The real-time Controller continually monitors its sensors and electronics to detect anomalies, and halts the mechanism when they occur. It then passes error information through TCA for action by higher-level exception handlers. Currently, the exception handlers halt the system if the error was caused by a hardware fault (e.g., a bad amplifier), and replan the last step if the error was caused by a bad footfall (e.g., the foot slips while doing a body move). Much more work remains, however, in detecting additional errors (such as colliding with obstacles while moving through space), automated diagnosis of errors, and intelligent error recovery.

The major impetus for the single-leg walking program was to gain experience for six-legged walking. To that extent, the project was quite successful. We have gained much insight into perceiving and modeling rugged terrain, controlling the legged mechanism, interacting with the ground, choosing safe yet effective footfalls, and planning efficient leg moves through space. The task ahead is to apply our experiences and successes to an autonomous walking system for the full six-legged Ambler.

Acknowlegements

Many members of the Planetary Rover project contributed to the single-leg walking system. Significant contributions in the integration and testing of the system were provided by P. Balakumar, L. Chrisman, C. Fedor, R. Hoffman, M. Hebert, G. Roston, and D. Wettergreen. This research is supported by NASA under Contract NAGW-1175.

References

1. Bares, J., Whittaker, W. Walking Robot with a Circulating Gait. Proc. of IEEE International Workshop on Intelligent Robots and Systems, Tsuchiura, Japan, July, 1990.

2. Bares, J., et al. Ambler: An Autonomous Rover for Planetary Exploration. IEEE Computer, Vol. 22, No. 6, 1989.

3. Caillas, C.,Hebert, M.,Krotkov, E., Kweon, I.S., and Kanade, T. Methods for Identifying Footfall Positions for a Legged Robot. Proc. IEEE International Workshop on Intelligent Robots and Systems, Tsukuba, Japan, September, 1989, pp. 244-250.

4. Craig, J.. *Introduction to Robotics, Mechanics and Control.* Addison-Wesley Publishing Company, 1986.

5. Krotkov, E.,Caillas, C., Hebert, M., Kweon, I.S., and Kanade, T. First Results in Terrain Mapping for a Roving Planetary Explorer. Proc. NASA Conf. on Space Telerobotics, Jet Propulsion Laboratory, Pasadena, CA, January, 1989.

6. Krotkov E., Simmons, R., Thorpe, C. Single-Leg Walking with Integrated Perception, Planning, and Control. Proc. of IEEE International Workshop on Intelligent Robots and Systems, Tsuchiura, Japan, July, 1990.

7. Kweon, I. S. and Hebert, M. and Kanade, T. Perception for Rugged Terrain. Proc. SPIE Mobile Robots III Conf., Cambridge, Massachusetts, November, 1988.

8. Lozano-Perez, T. Spatial Planning: A Configuration Space Approach. IEEE Transactions on Computers, C-32:108-120, 1983.

9. Simmons, R., Mitchell, T. A Task Control Architecture for Autonomous Robots. Proceedings of Space Operations and Autonomous Robotics Conference, Houston, TX, July, 1989.

10. Simmons, R., Krotkov, E., Roston, G. Integrated System for Single Leg Walking. Tech. Rept. CMU-RI-90, Robotics Institute, Carnegie Mellon University, July, 1990.

11. Simmons, R., Lin, L.J., Fedor, C. Autonomous Task Control for Mobile Robots. Proc. of IEEE Symposium on Intelligent Control, Philadelphia, PA, September, 1990.

12. Stentz, A. Multiresolution Constraint Modeling for Mobile Robot Planning. Proc. SPIE Symposium on Advances in Intelligent Robotics Systems, November, 1989.

13. Wettergreen, D., Thomas, H., and Thorpe, C. Planning Strategies for the Ambler Walking Robot. Proc. IEEE International Conference on Systems Engineering, August, 1990.