Perception, Planning, and Control for Autonomous Walking

with the Ambler Planetary Rover

Eric Krotkov   Reid Simmons

Robotics Institute

Carnegie Mellon University

Pittsburgh, PA 15213

January 30, 1995

Abstract

Producing a robotic system that can operate autonomously on other planets is a challenging task: the robot must be extremely self-reliant; it must be able to operate on a limited power budget;

and it must be able to traverse a wide range of rugged terrain features. This article describes an integrated software system that combines perception, planning, real-time control, and task-level control to navigate autonomously the Ambler, a six-legged rover. The overall approach to walking is highly deliberative: where actions are goal-directed, steps are planned out in detail using explicit models of the terrain and the vehicle, and moves are checked for safety prior to execution. We believe that this approach is well-suited to meeting the challenges of reliability, efficiency, and terrainability faced by autonomous planetary rovers. The Ambler system has been extensively tested, both indoors and outdoors, and the results confirm our expectations regarding the degree of system capability and performance afforded by our approach.

Contents

List of Figures

List of Tables

1 Introduction

To operate in remote, natural environments, a mobile robot system needs to be highly reliable, capable and efficient. These requirements extend to both the hardware and software aspects of the robotic system. This paper presents results of our investigation of perception, planning, real-time and task-level control for the Ambler, a six-legged walking robot. The result is a robot system that can autonomously traverse rugged terrain for extended periods of time.

The Ambler (Figure 1) is built for mobility and efficiency. A related paper [2] describes in detail the configuration and physical form of the robot. The Ambler's novel features include stacked, orthogonal legs (as in Figure 1) and the ability to walk with a *circulating gait*. The three degree-of-freedom orthogonal legs decouple horizontal and vertical motions, leading to simplified planning and control. Further, the design increases mobility by allowing legs to be placed adjacent to steep terrain features, such as sheer faces of rocks.

The stacked arrangement of legs enables the Ambler to exhibit a unique circulating gait. To move forward, one rear leg circulates through the body cavity, moving ahead of the other two legs on its stack (Figure 10. This process repeats as the robot walks to its destination. The circulating gait reduces the number of footfalls needed to cover the same distance as a comparable follow-the-leader walker, increasing efficiency and reducing planning and perception needs.

The Ambler is totally self-contained, equipped with on-board computing and power. Computation is provided by two 68020 boards for real-time control, and one SPARC workstation each for planning and perception. Power is provided by a combination of batteries and a propane generator.

Sensors on the Ambler include both relative and absolute encoders, inclinometers to measure tilt, six-axis force/torque sensors on the feet, a scanning laser rangefinder for mapping terrain, and black and white camera to aid in position estimation. The robot communicates with an off-board graphical user interface via a radio Ethernet link.

Figure 1: The Ambler

Objective

The chief objective of this work is to enable robotic exploration of remote, natural areas. We consider exploration missions with two defining characteristics:

- **Remote area** By "remote" we mean conditions under which teleoperation is either not feasible (typically because of delay) or not economical (typically because of the high cost of ground support).

- **Natural environment** The forces of nature produce irregular shapes and materials, causing natural terrain to exhibit roughness over a wide range of scales [6]. There are no prepared surfaces in nature.

To make our objective more concrete, we choose the planet Mars as an instance of an area that is remote, natural, and unexplored. Others have detailed the expected benefits of observing geophysical, meteorological, and biological conditions on Mars, and have assessed the risks and costs of sending human crews as highly dangerous and very expensive [34]. Robotic exploration of Mars provides an alternative which eliminates risk to human lives and promises to substantially reduce mission cost.

These objectives are also present in many environments on Earth. For example, the ideas discussed in this paper may apply equally well to other remote areas such seabeds, volcanos, and subsurface mines. Further, creating a robot with the capability to explore Mars is a motivation, not an objective. As a consequence, the research does not require flight-qualified components for computing, data storage, or communications, nor does it require formulation for extremes of gravity, vacuum, or other conditions.

There are three specific challenges addressed by the Ambler project which follow from the stated objectives: self-reliant control; limited power; and rugged terrain traversal. These challenges effect each part of the integrated walking system—perception, planning, real-time control, and task-level control—and, in Section 3, we describe how each of the systems faces the three challenges.

- **Self-reliant control** The remoteness of the area to be explored effects the feasibility and cost of different control regimes, ranging from pure teleoperation (reliance on operator) to pure rover autonomy (reliance on self). Regimes from the teleoperation end of the spectrum are less feasible, because round-trip signal delays preclude the rover from responding quickly to hazardous conditions (e.g., sinking into quicksand), and less economical, because it requires costly ground support. The challenge is self-reliant control that enables timely rover responses, requires little human interaction, and enables operation for extended periods and distances. The longer the duration and wider the coverage, the greater the likelihood of

Figure 2: Martian surface

meaningful return from exploration missions.

- **Limited power** The remoteness of the area requires the rover to supply its own power, which in turn dictates power budgets that are meager by laboratory standards. Limited power places a premium on efficiency of operation and penalizes unproductive behavior such as bumping into obstacles (which transfers energy to the terrain) and wandering repeatedly in the same small subregion (which serves no exploration purpose).

- **Rugged terrain traversal** The surface of Mars viewed by the Viking landers is rugged, uneven, and rocky (Figure 2). This terrain challenges most existing mobile robot mechanisms [2]. Even with a suitable form of locomotion, the terrain poses significant challenges to stability control, since the surface may shift, fracture, or deform. The irregular terrain also challenges established machine perception techniques, because it does not satisfy standard constraints on shape (e.g., symmetry) or surface properties (e.g., smoothness), nor does it admit controlled lighting or fixtured objects, and it challenges planning techniques, since the irregularly shaped robot must move in three dimensions while avoiding terrain collisions and self-contact.

We determined that a legged robot would best be able to meet these challenges. In particular, walking has advantages over rolling in mobility and efficiency. Walkers have relatively high ground clearance, and they contact the terrain at only a few discrete points. This allows walkers to surmount higher obstacles and traverse more rugged terrains. Similarly, unlike rolling and crawling mechanisms, walkers are able to select favorable footholds, avoiding transferring energy to terrain by unsupportive contact. Finally, the use of legs isolates the walker's body from the underlying terrain, enabling sensors and scientific equipment to travel along smooth and predictable trajectories.

The results of this project are primarily in the design of a capable legged robot [2], its control, both at the real-time and task levels, 3D perception for rugged, unstructured terrain, and planning at the gait, footfall, and leg-recovery levels. All this provided the basis for autonomously operating the Ambler in fairly rough terrain. However, this project did not address issues of extreme long-term reliability, planetary exploration, or global navigation. We are currently in the midst of a successor project that addresses these important issues.

Approach

There are two distinguishing features to our approach to enabling high-performance autonomous walking:

1. **Autonomy** Our approach is to develop planning algorithms to evaluate conservatively each Ambler motion, checking them several times for feasibility and stability. While executing the plan, if robot sensors indicate signs of trouble, then reactive behaviors stabilize the robot. This combination of approaches enables the Ambler to operate for extended periods.

2. **Explicit models** To facilitate the deliberation necessary to plan good moves, we need to model both the environment and the capabilities and limitations of the robot. In particular, the Ambler walking system explicitly models the terrain geometry, mechanism kinematics, vehicle stability margins, and sensor uncertainty. These models play critical roles in meeting the challenges described above.

   - Models of terrain geometry enable planners to select favorable footholds, thus meeting the challenge of rugged terrain traversal and capitalizing on the high-mobility characteristics of walking robots.
   - Models of terrain geometry permit planners to select leg and body motions minimizing terrain contact, thus meeting the challenge of limited power.
   - Models of mechanism kinematics allow controllers to accurately execute the plans, thus addressing the challenges of rugged terrain traversal and limited power.
   - Models of stability margins allow planners and controllers to ensure the safety of different motions, thus meeting the challenge of extended operation under self-reliant control.

   While the complexity of the models is high compared to purely reactive approaches, the complexity produces significant benefits. In particular, using models to plan and analyze moves contributes to the rover's safety and power efficiency, which are primary concerns for planetary missions.

This paper proceeds as follows: Related research is reviewed in the next section, and Section 3 presents the integrated walking system. Section 4 documents experimental walking results, and the paper concludes with a summary, an evaluation, and a look towards the future.

2 Related Research

Researchers have advanced a spectrum of mobile robot concepts suited for exploring remote, natural environments. In this section we review relevant mobile robots that have been built and operated.

In general, the variety of related research in this area has (i) produced mechanisms with high mobility but without high autonomy, (ii) researched autonomous units but without mission capabilities, and (iii) developed autonomous systems with mission capabilities but only in structured environments. To our knowledge, there are no robot systems other than the Ambler that combine high mobility, high autonomy, reliability and mission capabilities for use in unstructured, remote, natural environments.

Sustained effort by researchers at NASA's Jet Propulsion Laboratory (JPL) has developed a family of rovers which includes the Surveyor Lunar Rover Vehicle, Robby, Rocky, Go-For, and the Mars Pathfinder microrover. Using these wheeled mechanisms as testbeds, JPL researchers have developed control architectures, such as Computer-Aided Remote Driving [43], behavior control [13, 12], Semi-Autonomous Navigation (SAN) [44], and mission-specific hybrids [39], which share mission control between human operators and robots. Using these control architectures, various vehicles have successfully demonstrated navigation in desert settings. Still, the degree of autonomy is limited even with SAN, which requires more computation and look-ahead planning than the other control architectures.

Researchers at LAAS have developed a complete system that provides for mission planning and supervision from a ground station, and for on-board mission interpretation and execution [8]. They have demonstrated this system with a six-wheeled rover called ADAM in an experimental testbed called EDEN. The demonstration involved execution of a "GO-TO(Landmark)" task. Although the degree of autonomy is high, the system has yet to be tested in extreme terrain for long durations.

Researchers at MIT have argued that small rovers (1 to 2 kg) are suitable for planetary exploration [4] and lunar base construction [5]. They have developed the Attila family of small six-legged walkers, and are striving to make the robots totally autonomous units using the subsumption architecture and "behavior languages" [1]. Although the degree of autonomy inherent in their approach is high, the potential for efficiency appears to be low. For example, local behaviors may cause the rover to take suboptimal actions, causing the rover to transfer substantial energy to the terrain by repeatedly bumping into obstacles. Worse, unanticipated interactions between behaviors may cause the rover to take actions that are hazardous to its well-being. Researchers advocating behavior-based rovers point out that it is feasible to send many such rovers, and thus it is not important if some do not survive [4]. However, it is entirely possible that each rover will exhibit the same behavioral problems, causing all to fail.

Other researchers have built and operated prototype planetary rovers, including the Quadra-

Rhomb [16], the Marsokhod [21], and the Walking Beam [10]. To date, these research efforts have concentrated primarily on mechanical design and real-time control, and have not yet addressed or achieved autonomous performance.

Mobile roboticists have developed a number of integrated systems that exhibit (or potentially exhibit) relatively high degrees of autonomy. Examples of indoor robots with autonomous capabilities include Hilare [9], the longest-running mobile robot project, and many others. Examples of outdoor robots with autonomous capabilities include the Autonomous Land Vehicle [41], the Navlab [40], and VaMoRs [11]. These robots typically operate in environments relatively rich in structure (e.g., well-defined hallway corridors, or roadways with lane markings) compared to planetary surfaces.

Researchers have developed a variety of walking mechanisms with the high mobility required by planetary rovers. These walkers include the Adaptive Suspension Vehicle [38], Titan I through Titan VI [15, 17], the Odex [33], the Recus [20], the Aquabot [20], and hopping machines [18, 30]. Many of these efforts emphasize electro-mechanical design and performance. Thus, these robots tend to be either teleoperated or operate under supervisory control, so the degree of autonomy achieved is low.

3 Integrated Walking System

This section describes each of the modules illustrated in Figure 3: the perception modules which acquire images and construct terrain elevation maps; the planning modules which plan leg and

Figure 3: Modules in the integrated walking system

Figure 4: Graphical user interface

body movements; the real-time module which controls the robot and monitors its internal sensors; the task-control module which coordinates components and monitors progress; and the graphical user interface which enables paths to be entered.

3.1 Example of Operation

To initiate walking, a human operator inputs a series of arcs (that is, arcs of a circle) via a graphical user interface (Figure 4). The user indicates desired waypoints and the orientation to be achieved at each, and the user interface then computes a series of arcs to achieve those position and orientation constraints.

Each arc is forwarded in turn to the gait planning module, which plans a series of leg and body moves. The gait planner first determines the leg that maximizes subsequent body motion. It finds a region of geometrically feasible footfalls based on the limits of the leg's motion, the need to avoid colliding with the current leading leg, the desire to avoid placing the leg in the path of the body, and the need to maintain support in subsequent moves [42].

Next, the terrain within the footfall region is analyzed to estimate the goodness of the footfalls. The footfall planner sends a request to the perception subsystem for a terrain elevation map within the boundaries of the footfall region. The local terrain mapping module sends a request to the

Figure 5: Conservative support polygon

image queue module requesting subimages that contain laser range data relevant to constructing the desired terrain map. The mapping module then combines the images to construct a terrain elevation map, and returns it to the footfall planning module. The footfall planner uses a neural network to evaluate the terrain map, estimating the goodness of each foot-sized patch in terms of stability and traction. The gait planner then chooses the best footfall location within the specified region.

After setting a leg move, the gait planner finds the maximum body move that follows along the arc, one which does not violate limits on leg motion, and keeps the Ambler's center of gravity within the *conservative support polygon (CSP)* (Figure 5). This guarantees that the Ambler won't tip over even if any single leg fails. The gait planner then forwards the chosen leg move to the leg-recovery planner, which plans a fast trajectory that avoids terrain collisions. The chosen leg and body moves are then forwarded to the real-time controller module. Concurrently, if the end of the arc has not been reached, the gait planner starts planning the next pair of leg and body moves.

The real-time controller generates leg move trajectories that are linear in joint space, coordinating the leg joints so that all motions start and end simultaneously. The controller module checks the leg move for feasibility, refusing to perform the move if it would exceed kinematic or geometric limits. During execution, the controller monitors the force sensors in the feet and the inclinometers. If an unexpected terrain collision or tilt occurs, all motions are halted, the brakes are engaged, and an exception is raised to the task-level controller. Many of these exceptions are handled autonomously (see Section 3.4.2) and walking continues once the steps are replanned.

In a similar manner, the real-time controller executes planned body moves. It computes, for each leg, the trajectory of horizontal joint motions that will simultaneously translate and rotate the body to achieve the commanded position. The twelve horizontal joints are synchronized to achieve smooth acceleration and deceleration of the body. Once the body move is completed, the task-level controller notifies the perception subsystem to acquire a new laser range image. This enables pre-processing of the image to occur concurrently with planning and control tasks. Typically, by the time the leg and body moves have been executed, the planners are ready with a new set of moves. In this way, the Ambler achieves nearly continuous motion.

3.2 Perception

From the challenges identified in Section 1 follows a need for a perception system that provides:

- **Self-reliant control** The rover must avoid stepping in unsafe areas that could cause it to lose traction or become unstable and tip over.
- **Limited power** The rover maximizes its power efficiency by planning motions minimizing terrain contact.
- **Rugged terrain traversal** The rover must select favorable footholds in rugged terrain based on quantitative information about terrain properties such as elevation, slope, and roughness.

Figure 6: Reflectance (left) and range images

The scene consists of a sand base with boulders 1-2 m tall (foreground) and a wooden ramp.

To address these challenges, we have developed perception algorithms and systems that build maps for the purpose of locomotion. The primary terrain representation is an elevation map, in which each (x, y) grid point is labeled with the maximum terrain elevation at that point. Elevation maps are well-suited for representing natural, rugged terrain, since they make few assumptions about the shape of the terrain. In addition, subsections of the map can be accessed by other system modules simply by specifying the polygonal boundary of a region. These characteristics match the needs of the Ambler's planning algorithms (see Section 3.3).

3.2.1 Sensing and Calibration

The primary sensor is a scanning laser rangefinder, which measures both reflectance and range (Figure 6). The sensor views terrain 1 to 8 meters ahead of the Ambler. We use a laser scanner because it directly recovers the environment's three-dimensional structure, supplying 3-D data more rapidly and reliably than passive vision techniques such as structure-from-motion and binocular stereo (although, recent advances [29, 27] make stereo far more feasible for planetary rovers than it was in 1987, when the decision was made to use a laser rangefinder).

The sensor (a Perceptron scanner) acquires data in 256×256 pixel images at a rate of 2 Hz. It digitizes to 12 bits over approximately 40 m, providing a theoretical range resolution of approximately 1 cm. The measurements cover 60° in azimuth and 60° in elevation.

Figure 7: Perception system modules

To ensure accurate sensor data, we have developed an automatic calibration procedure that identifies the transformation between the Ambler-centered reference frame and the sensor-centered frame. The procedure moves the leg to various positions within view of the scanner, and processes the reflectance image to locate the leg. The leg positions in the images and those positions computed by the Controller from the vehicle kinematics are used to compute the transformation which minimizes the distance between pairs of corresponding points referred to the Ambler-centered frame. The technique achieves an accuracy of 2–7 cm with a precision no lower than 2–5 cm [22].

3.2.2 Perception Subsystem

The perception system consists of three major modules: one to acquire images; one preprocess and store images; and one to construct terrain maps (Figure 7). Approximately ten other modules perform data analysis and display tasks, each executing concurrently. There are many different system configurations that meet the perception needs identified at the beginning of Section 3.2. We selected this configuration because it is modular, extends easily to more and different sensors, decouples imaging and mapping, and manages large quantities of data.

Image Acquisition

The Perception Imaging Manager (PIM) acquires frames of range and reflectance data, selecting either 1) real images from the rangefinder, 2) virtual images stored previously, or 3) synthetic

Figure 8: Preprocessed range image

(a) Range discontinuities. (b) Valid pixels shown white, corrupt pixels shown black.

images computed by a 3-D graphics simulator. The PIM tags each image pair with the body pose (position and orientation), the six leg positions, the transformation relating the scanner frame and world coordinate frames (composed from transformations derived from the calibration procedure and the Ambler's dead-reckoned pose), and other useful pieces of information.

Image Preprocessing, Storage, and Access

The PIM transmits the tagged images to the Image Queue Manager (IQM). The IQM preprocesses, stores, and provides access to the images.

The preprocessing steps seek to filter out spurious data, and to detect range discontinuities. Figure 8 illustrates the results of preprocessing the scene shown in Figure 6.

- Identify range discontinuities using the Canny operator [7]. The extracted edges are used in subsequent mapping algorithms to prevent interpolation between distinct objects in the scene.

- Mark pixels suspected to be corrupted by limitations of the particular sensor used, including internal reflections at specific locations, and a “vignette” effect (caused by the exit window) appearing at the bottom corners of the image [14].

- Mark pixels belonging to the image of the robot's legs. First, predict where in the image the leg(s) appear, and what the range values should be. Then search the predicted regions, marking those pixels with range values consistent with the leg location.

- Identify pixels corresponding to scene points that are not connected to the ground plane. These "floaters" can occur under several circumstances documented elsewhere [14], and summarized here:

  1. If the scene point lies at a distance ρ beyond the ambiguity interval of the sensor (i.e., farther than 40 m), then the range reported by the sensor will be ρ mod 40.
  2. If the scene point belongs to an object that poorly reflects the infrared radiation, then the range reported by the sensor will be significantly greater than the true range.
  3. If the scene point belongs to an object oriented so that the angle of incidence is small (far from perpendicular), then the range reported by the sensor will be significantly greater than the true range.

  In each case, the incorrectly reported range of the scene point causes the computed elevation of the point to be substantially different from the ground elevation. Consequently, the reconstructed scene point appears to float above or below the terrain.

  We use standard image processing techniques to extract range-connected regions. We eliminate from further consideration small regions (including regions of size one, namely, single disconnected pixels).

After preprocessing, the IQM stores the processed images in a fixed-length queue, removing older images as the queue fills up. For most of the experiments, the queue size is 30, since that has empirically sufficed for the most challenging case: point turns in rugged terrain. The image queue data structure facilitates access to ordered sequences of images. Sequences are needed because the single forward-looking sensor on the Ambler cannot image obstacles either below or behind the vehicle, and must rely on previous images to accomplish tasks such as planning the trajectory of a recovering leg.

The IQM is selective because transferring images is expensive: the size of the images is large (256K per pair, plus variable-size auxiliary structures such as edge images), and many images may be required (with typical queries for $1m^2$ areas, 5 are required for a straight-line trajectory on flat terrain, and 25 are required for a point turn over obstacles). Given a polygonal region-of-interest, the IQM selects only rectangular subimages that, after projection onto a ground plane, enclose or intersect the given region-of-interest. In typical operation, this reduces data transfer by 75-90 percent compared to the policy of transferring the entire image whenever it contains some part of the region-of-interest.

Terrain Mapping

Terrain elevation maps are constructed and maintained by the Local Terrain Mapping (LTM) module. Other modules request elevation maps, specifying (i) a polygonal region, (ii) a map resolution, and (iii) an arbitrary reference frame to be used as the map origin. Typical reference frames are the current body frame, the world frame, and a future body frame (to allow for advance planning). To construct the maps, the LTM first queries the IQM for a sequence of subimages potentially containing information about the polygonal region. The LTM then uses the subimages, starting with the most recent, to compute the elevation of each grid point in the polygonal region. The LTM computes and stores the terrain elevation with the Locus Method, an efficient algorithm for transforming and interpolating range data from the sensor frame into Cartesian coordinates.

To summarize the Locus Method, let ρ be the sensed range, and let ϕ and θ be the vertical and horizontal scanning angles of the laser beam direction corresponding to row and column position in the image. The problem of finding the elevation z of a point (x, y) is equivalent to computing the intersection of the surface observed by the sensor with the vertical line passing through (x, y). The basic idea of the locus algorithm is to convert the latter formulation into a problem in image space. A vertical line is a locus (curve) in (ϕ, θ, ρ) space. Similarly, the range image can be viewed as a surface $\rho = I(\phi, \theta)$. The problem then is to find the intersection, if it exists, between a curve parameterized by ϕ and a discrete surface. Since the surface is known only from a sample of data, the intersection cannot be computed analytically. Instead, we must search along the curve for the intersection point (we refer interested readers to [24] for details of the search).

The LTM assigns one of three labels to each point in a rectangular grid enclosing the specified polygon:

- *unknown*, for those cells outside the polygon, outside the field of view of all images, or containing pixels marked as invalid by the preprocessing steps.

Figure 9: Elevation map

Elevation map computed from range images like those in Figures 6 and 8. The units are meters.

- *occluded*, for those cells occluded by other objects. An (x, y) location in the map is in an occluded area if its locus intersects the image at a pixel that lies on an occluding edge. We implement this idea by first detecting edges in the range image using the Canny operator. Then, when we apply the Locus Method and observe that the locus of a given location intersects the image at an edge pixel, we mark that location as lying in an occluded area, and compute an upper bound on the elevation of the occluded terrain.

- *known*, for all other cells.

Figure 9 illustrates the elevation map computed from the range image of Figure 8. This map contains points labeled *known* and *occluded*.

3.2.3 Performance

Table 1 records perception system statistics gathered from several representative walking trials. The large volume of images and maps processed indicates the robustness of the system.

We found from these trials that taking a single step requires approximately 20 seconds of perception system processing, which involves examining between 5 and 28 range subimages in order to compute elevation values for some 1700 points, on average. Breaking the timing down further, we find that for SPARC 2 workstations, image acquisition takes about 1 second, image

Trial	Distance (m)	Images Acquired	Subimages Transferred	Maps Built	Map Points
Outdoor	46	151	8,974	626	375,003
Indoor	68	185	10,015	640	267,104
Indoor	107	397	20,286	1,306	597,078
Outdoor	527	1219	101,129	4,716	2,578,102

Table 1: Perception system statistics in four separate trials

"Distance" represents the distance traveled by the robot during the trial. The results vary because the terrain and software modules vary between trials. For example, in the 46 m run in hilly terrain, a module planned vertical body motions, computing more map points than in the 68 m run in non-hilly terrain, during which that module did not execute.

pre-processing and storage requires about 10 seconds, and construction of elevation maps takes about 5 milli-seconds per point.

One measure of map accuracy is the difference between the computed elevation and the elevation determined by encoders on the Ambler's legs. According to this measure, we find that the map accuracy varies from step to step and from trial to trial, but never strays far from the scanner precision of 10–15 cm. This accuracy is sufficient for the Ambler, whose feet are 30 cm in diameter.

As the Ambler walks, images and maps consume more and more memory. As a countermeasure, the perception system utilizes memory management routines that detect memory saturation, reclaim storage for new use, and delete less recent data. In one typical trial, after acquiring 100 images, memory usage levels peaked at 1.6 Mb for PIM, 2.8 Mb for LTM, and 30.5 Mb for IQM (with queue size 30). After this, the perception system maintained essentially constant memory utilization, thus demonstrating the effectiveness of memory management.

In summary, the results show that the perception system constructs quantitative models of terrain geometry that enable power-efficient locomotion in rugged terrain.

3.3 Planning

We have developed automated planning algorithms for the Ambler that address the three main challenges identified in Section 1, as they relate to the planning system.

- **Self-reliant control** Given a path to follow, the rover must autonomously plan leg and body moves that are safe and effective (i.e, maintain good progress along the path).

- **Limited power** The planner should choose moves that minimize energy expenditure. This includes minimizing leg and body travel, and avoiding unnecessary terrain collisions.

- **Rugged terrain** The algorithms must enable the rover to climb hills, traverse rock-strewn terrain, and to cross meter-tall and meter-wide obstacles (such as boulders and trenches).

All three challenges indicate the utility of a deliberative approach to planning leg and body moves. In particular, for the type of remote, rugged terrain walking that we address, the effects of ill-considered decisions can be catastrophic. Thus, to provide for safe, power-efficient movements, the planning algorithms utilize detailed models of the vehicle kinematics, the terrain, and vehicle/terrain interaction.

One potential problem with deliberative planning of moves is the computation involved in searching for acceptable moves. This is exacerbated in the Ambler, since a large number of moves are possible from a given pose. The planning algorithms attempt to minimize the amount of search needed by adding constraints to the rover motion, such as always using a circulating gait.

To date, our research in autonomous planning has concentrated on path following: the rover is constrained to walk along an arc of a circle: if the arc has large radius, the path is nearly straight; if the arc has zero radius, the rover turns in place. We have developed a hierarchy of three planning algorithms—gait, footfall and leg recovery—to handle path following.

Figure 10: Circulating gait

3.3.1 Gait Planning

Gait planning chooses the sequence of leg and body moves to advance the rover along a given path. Typically, walking robots use a fixed gait pattern, such as tripod or wave gaits [3, 38]. However, due to the mobility of the Ambler and the extreme terrain it has to traverse, we developed a planner that uses a *free* gait, one in which the choice of which leg to move is made step-by-step, depending on current kinematic and terrain constraints.[1] The planner also capitalizes on the unique features of the Ambler to implement a *circulating* gait, one in which the rear-most legs move through the body cavity to become the new front-most legs (Figure 10) [2, 42].

The input to the gait planner is a radius and angle that describes an arc to traverse (the angle can be positive or negative, to enable the rover to walk forwards or backwards). The basic walking cycle is to alternate moving the body so that it remains oriented tangent to the arc and then moving the leg that most constrains subsequent body motion. For arc radii less than the tread width of the Ambler (about 2.5 m), the gait planner first circulates the rear leg on one stack to the front, and then circulates the front leg on the other stack to the rear. In this way, the Ambler can turn in place by having the legs on one stack move retrograde with respect to the legs on the other stack.

The gait planner utilizes two constraints to determine the extent of the body move. The first constraint is based on the kinematic limits of the legs. Given that the feet are fixed, the

[1] We have also developed a planner for a *crab* gait, one that enables the Ambler to walk from side to side.

Figure 11: Stack reachability constraint

planner determines how far each stack can move along the arc before reaching a kinematic limit (Figure 11a). The second constraint is based on stability. To be statically stable, the rover's center of mass must lie within the support polygon formed by the convex hull of the feet. For added safety, we insist that the Ambler always lie within the *conservative support polygon (CSP)*, which is defined as the intersection of all five-legged support polygons (Figure 5) [25]. Thus, if the Ambler remains within the CSP, any one leg can fail and the vehicle will still remain statically stable. The gait planner finds where the body travel intersects the leading edge of the CSP, and uses this as an additional constraint on body motion. The minimum motion determined by the kinematic and support constraints is then chosen as the next body move. This planning suffices for movement in the horizontal plane.

To plan vertical movements, the planner obtains a terrain map from the perception system of a region in the direction of travel. It finds the maximum elevation in the region and, if needed, commands a vertical move (lift or lower) to keep the Ambler within a fixed window above the terrain (typically, 50-90 cm). This decoupling of horizontal and vertical body motion planning is possible because the Ambler always walks with a level body motion (parallel to gravity) and its legs decouple horizontal and vertical motion. Thus, changing the body elevation has no effect on the kinematic limits of leg travel or on the static stability of the rover.

After commanding body motion, the planner then selects the leg move that will afford the greatest subsequent body progress. This is done by determining the maximum body movement,

Figure 12: Constraints on geometrically feasible footfalls

assuming the leg is not fixed to the ground. In most situations, the planner considers the rear-most legs on each stack; however, for tight turns (arc radii less than the distance between the body and stack centers, or about two meters), one of the stacks moves in retrograde motion (Figure 11b), so that the planner must alternate moving the legs on one stack from rear to front, and the legs on the other stack from front to rear. There are a number of geometric constraints that come into play in determining where to place the feet. The kinematic limits of the leg define a torus outside which the leg cannot reach. In addition, the leg must be placed so that it does not collide with the current front-most leg, nor should it be placed within the current path of the body (Figure 12). Finally, there are constraints needed to ensure the feasibility of subsequent movements: the leg must be placed far enough forward so that a subsequent leg move on the opposite stack keeps the body within the CSP, and the leg must not be placed so far forward that subsequent leg moves on the same stack become kinematically infeasible.

These kinematic constraints define an area within which all footfalls are geometrically feasible. If the area for a leg is insignificant, the planner tries moving the leg on the other stack. If there are no feasible moves on either stack, the planner attempts to *shuffle* the legs, and tries to plan from there. To shuffle legs, the planner moves legs forward, backward, and side-to-side (without circulating through the body) so that the legs are placed close to a standard configuration. Although in principle up to five legs might have to be moved, in practice we have observed that shuffling usually involves moving one or two of the front or middle legs. Shuffling of the legs is needed, on average, every 30 steps or so.

3.3.2 Footfall Planning

Given the region of geometrically feasible footfalls determined by the gait planner, the task of the footfall planner is to find the best terrain patch within that region, in terms of its ability to provide stability and traction.

One problem is the difficulty of operationalizing those criteria. While flat, smooth areas are obviously ideal, and steep, rough areas tend to provide the worst footing, it is unclear how to quantify the areas in between. We tried several methods of evaluating footfall quality, including hand-coded rules and non-linear regression techniques. We eventually settled on a neural network learning scheme, which was found to produce consistently the best results [19].

The purpose of the neural network learning scheme is to estimate footfall quality of the terrain based on features such as slope, roughness, and curvature. The neural net learns an evaluation function which combines these features. It is trained off-line using human-supplied indications of footfall quality (e.g., “good” or “very bad”) in a variety of terrain. The training data are the pairs of locations where the numerical evaluation produced, using the current set of weights, differs from the human preferences. The difference in the evaluations is used as an error term to update the network weights using backpropagation [32].

This method of using a neural network to combine precomputed features (rather than operating directly on the raw data) has proven quite effective, both in learning speed and on-line operation. In particular, the features provide sufficient structure to make the learning tractable, and the features are fairly easy to compute from the terrain elevation maps produced by the perception system. Initially, we considered refining the net weights on the fly, as the Ambler gained experience walking, but the initial function learned proved adequate enough, and so the adaptive approach was not actively pursued. On-line learning techniques remain a possibility, however, especially if the rover would be exposed to novel environmental conditions (such as might occur on Mars).

3.3.3 Leg Recovery Planning

The task of the leg recovery planner is to plan energy- and time-efficient trajectories for the legs. This involves minimizing joint movement (especially the slow vertical actuators) and maximizing the amount of time all three leg joints are actuating simultaneously. The planner must also ensure that the legs do not collide with the terrain, or any part of the rover.

In general, this is a complex three-dimensional search problem. An early version of the leg recovery planner did in fact search a 3D space to find optimal moves [23]. This proved to be very expensive, both in terms of planning time and the demands it placed on the perception system. The current leg recovery planner minimizes search by decoupling the planning of horizontal and vertical motions.

The planner first finds a trajectory in the horizontal plane that minimizes (horizontal) joint travel while avoiding leg/leg and leg/body collisions. The planner uses an explicit kinematic and geometric model of the Ambler, as well as information on maximum joint velocities. The algorithm handles both circulating and non-circulating recoveries (the latter are used primarily when shuffling legs).

Once a horizontal trajectory is determined, the planner projects an “envelope” of vertical motions along the trajectory. The envelope assumes that the leg raises/lowers at full speed while traveling horizontally. Purely vertical moves are needed only when the envelope is projected to intersect the terrain, or if needed to clear the top of the Ambler’s body arch. Although this algorithm is not necessarily optimal (a longer horizontal trajectory might result in significantly less vertical travel), in practice the moves produced by the leg recovery planner are very good.

3.3.4 Remarks

While the circulating gait and independently controllable legs give the Ambler advantages in efficiency and mobility, planning for the Ambler is significantly more complex than for other walkers, due to its many degrees of freedom and relatively narrow front-to-back stability margin [2]. On the other hand, in several situations we have been able to take advantage of the decoupling of horizontal and vertical motions afforded by the orthogonal legs to simplify planning algorithms and to reduce search. Overall, the planning system is very robust and reliable, enabling Ambler to autonomously walk along varied paths and through varied terrains (see Section 4).

The Ambler’s deliberative mode of walking contrasts sharply with the reactive approaches taken by most other researchers [3, 28, 30]. The main advantage of the deliberative approach is that by carefully analyzing kinematic and terrain constraints, the Ambler can produce more reliable and efficient moves. Reliability is increased by decreasing the chance of tipover or placing feet in unstable locations. Efficiency is increased by minimizing actuator motion and minimizing the number of terrain collisions.

The main disadvantage of the deliberative approach is that it is computationally intensive: the Ambler planners take 30-40 seconds per step (of which about 75% of that time is spent by the perception system creating terrain maps for the various planners). This is acceptable for the Ambler due to the slow top speed of the mechanism: the planning is still about 50% faster than the cycle time of body/leg moves. By planning one step while executing the previous step (see Section 3.4.2), the Ambler planning system can provide a continual stream of moves to the real-time control system, demonstrating that computation does not create a bottleneck for the Ambler. With a faster, mechanically simpler rover, we would probably change our planning strategy, relying more on reactive (although not totally reflexive) control.

3.4 Control

The Ambler system uses two types of control: real-time and task-level control. Real-time control is used to interface with the rover: driving servo loops, reading internal sensors, and maintaining the dead-reckoned position. Task-level control is concerned with integrating perception, planning, and real-time control into a concurrent, distributed system. The task-level control manages the system's resources, schedules and synchronizes tasks, and provides for execution monitoring and error recovery.

3.4.1 Real-Time Control

The design of the Ambler's real-time control system directly addresses the three challenges identified in Section 1:

- **Self-reliant control** The rover must autonomously react to and safeguard against unanticipated electro-mechanical conditions such as amplifier faults, and unexpected environmental situations, such as soil failure and excessive tilt.

- **Limited power** The rover must faithfully execute the leg and body trajectories planned to avoid obstacles to avoid wasteful transfer of energy to the terrain.

- **Rugged terrain traversal** The rover must move its legs with sufficient precision to be able to step in tightly constrained spaces (e.g., between two boulders).

The Controller runs on two 68030 CPU boards in a real-time, multi-tasking environment. One CPU generates trajectories, maintains the dead-reckoned position, and communicates with other modules. The other executes horizontal and vertical body trajectories at 10 milli-second intervals.

Motion Control and Safeguarding

The Controller's main task is to move the legs to a given position and to move the body to a given pose. Given a commanded body position, the Controller decomposes the motion control task into two parts, a planar translation and/or rotation followed by a vertical lift or lower. For the planar motion, the Controller computes the horizontal joint motions that simultaneously translate and rotate the body to achieve the commanded horizontal position. It executes a horizontal body move by first locking the vertical joints, and then servoing the twelve horizontal joints to achieve smooth acceleration and deceleration of the body. For the vertical motion, the Controller follows the same scheme, but servos only the six vertical axis joints.

Given a path for the leg to travel, denoted by a series of waypoints, the Controller generates a sequence of joint positions beginning with the current leg position and ending with the final waypoint position. It then interpolates those joint positions, producing a trajectory that is linear in joint space. In executing this trajectory, the Controller coordinates the three leg joints so that all motions start and end simultaneously, as with the body move. During execution, the force sensor is set to trigger an interrupt if either normal or lateral forces exceed threshold values, signifying terrain contact. The Controller treats the interrupt either as the successful completion of a leg move or as an unexpected collision, depending on where in the trajectory it occurred.

The Controller includes many safeguards to protect the Ambler. Before committing to execute a move, the Controller performs a kinematic and geometric analysis of the move to check that the move will not exceed joint limits, that leg motions will not collide with other parts of the mechanism (in particular, that the leg will pass safely through the center of the body while executing a circulating gait), and that the vehicle will remain statically stable at all times during the move. If any unexpected situations occur during execution, such as premature ground contact, the Controller halts all motion and engages the brakes. In addition, the Controller monitors inclinometers, and automatically levels the robot when the body tilt exceeds a threshold value. This also helps safeguard against failure of the force sensors — if a force sensor fails to register ground contact and the leg is driven into the ground, at some point the tilt monitor will register and stop motions before the Ambler tips itself over. For further safeguarding, a hardware safety circuit sets the brakes and shuts down all motions if it detects anomalies such as servo motion or amplifier faults.

Dead Reckoning

Many exploration missions require or benefit from estimates of robot pose and course (sequence of poses). For example, without course estimation it would be impossible to produce a map of an area wider than the sensor field of view, or to achieve such objectives as searching exhaustively for ice in a specified region. For such missions, dead reckoning is a widely used method that identifies the pose of a vehicle by integrating its motion. We have developed a dead reckoning algorithm that increases the robot's capability to address the challenge of self-reliant control.

To formulate the problem, we define a stationary world reference frame $\mathcal{W}$, and a body frame $\mathcal{B}$ attached to the Ambler body, which moves with respect to $\mathcal{W}$. Let $\mathbf{x}_{w,i}$ denote the position of foot i in $\mathcal{W}$. Let $\mathbf{x}_{b,i}$ denote the position of foot i in $\mathcal{B}$; these positions can be computed using the mechanism kinematics.

We assume that the Ambler can be treated as a rigid body, which means that the feet do not move unless commanded to do so. Then the dead reckoning problem amounts to finding the rigid body transformation (the rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{t}$) such that

$$\mathbf{x}_{w,j} = \mathbf{R}\mathbf{x}_{b,j} + \mathbf{t} \; . \tag{1}$$

We seek a solution that minimizes the squared error

$$\sum_{j=1}^{6} w_j \mathbf{e}_j^T \mathbf{e}_j \ , \tag{2}$$

where $\mathbf{e}_j = \mathbf{x}_{w,j} - \mathbf{R}\mathbf{x}_{b,j} - \mathbf{t}$, and the w_j are weights on the observed positions.

We implemented a solution to this minimization problem in three steps [31]:

1. Identify feet that have slipped, and prevent them from being used in further computation by setting w to zero. Foot slippage is detected using the constraint that the distances between feet should not be affected by a body move: by noting the distances before and after a move, it can be determined which feet might have slipped relative to one another.

2. Solve for the rigid body transformation that best transforms foot positions in $\mathcal{B}$ to $\mathcal{W}$, using a technique, developed by Matthies [26], based on singular value decomposition.

3. Update foot positions, if necessary, based on the new body pose and the current leg joint angles. Ideally, no update should be required, but in practice, the least-squares solution may cause discrepancies between the current and stored foot positions.

To test the implementation of dead reckoning, we conducted trials in which we commanded the Ambler to advance a specified distance. After execution of the command, we computed the dead-reckoned pose and measured the *ground truth* pose by using a surveying device. For each move we recorded the dead-reckoned and ground truth poses.

Figure 13 illustrates the results of one trial with 12 body advances covering six meters along a straight line. The figure plots the planar error $\sqrt{\delta_x^2 + \delta_y^2}$, where the δ terms represent the component-

Figure 13: Dead-reckoned position error

The points represent the difference between the dead-reckoned and ground truth body positions in the plane of travel. With each advance, the error increases by about two percent of the distance traveled.

wise differences between dead-reckoned and ground truth poses. This data indicates that the dead reckoning error u is two percent of the distance traveled. Given the high precision of the prismatic and rotary joints, and the high encoder resolution, this accuracy is lower than expected.

Further analysis of the data show that systematic error dominates random error: dead reckoning "thinks" that the body travels a smaller distance than the surveying device measured. We believe the source of the systematic error to be an incomplete kinematic model of the Ambler, one that does not adequately address factors such as small differences between leg dimensions, or structural deflection. To reduce the systematic error, we could adjust the dead-reckoned planar pose by adding $(u \cos\theta, u \sin\theta)$, where θ is the heading. With this correction, we anticipate accuracy better than one percent.

3.4.2 Task-Level Control

Task-level control involves the integration and coordination of perception, planning and real-time control to produce a complete robotic system. The Ambler's use of task-level control effects two of the three challenges described in Section 1.

- Self-reliant control. The rover must be able to detect and recover from a wide range of exceptional conditions, ranging from hardware faults to situations which the normal planning

Figure 14: Task tree for Ambler walk planning

software cannot handle.

- Limited power. Efficient use of the rover's resources minimizes overall power consumption. This includes reducing the idle time, since there is a fixed overhead incurred just standing still, which argues for use of a concurrent, distributed software architecture.

The Ambler subsystems are integrated using the *Task Control Architecture (TCA)*. The architecture provides a language of communication and control constructs for coordinating distributed mobile robot systems, and a library of utilities to facilitate development of concurrent robotic systems [35, 37]. The communication constructs support coarse-grained message passing between processes, with TCA handling the tasks of encoding and decoding message data and routing messages and responses to the appropriate processes. The TCA control constructs support hierarchical planning, task and resource scheduling, monitors, and exception handling. The control constructs are designed around the concept of hierarchical *task trees,* which serve as the TCA representation for plans (see Figure 14).

Concurrency Besides encoding hierarchical goal/subgoal relationships, TCA task trees encode temporal constraints between subtasks. The temporal constraints enable the specification of partial orderings between subtasks. For example, the task tree in Figure 14 indicates that to walk along an arc (following the algorithm described in Section 3.1), the rover must take a sequence of steps. Each step consists of recovering the leg, moving the body, and recursively taking a sequence of steps

Figure 15: Timings of Ambler modules during concurrent walking cycle

(until the desired goal position is reached). The "sequential achievement" temporal constraints indicate that subtasks must be executed in order; in particular, all command messages (the leaf nodes in one subtree) must be handled before any of the commands of the subsequent subtask can begin.

A "delay planning" constraint indicates that a goal message cannot be handled (i.e., decomposed into subtasks) until the previous task has been achieved. Thus, due to accumulated dead reckoning error we specify that the plan for walking along one arc should not be planned out until the previous arc has been traversed. On the other hand, the absence of a "delay planning" constraint between the "moveBody" and "takeSteps" subtasks indicates that the system can plan out one step while executing the previous one. By having planning and execution occur concurrently, the Ambler system can walk nearly continuously, doubling the average walking speed compared with the original sequential sense/plan/act system [37]. In a similar manner, we use TCA control constructs to specify when laser scans should be initiated and to indicate that processing of range images can occur concurrently with planning and execution.

Figure 15 shows a Gantt chart of the resource utilization of the various modules. Utilization of the planning and perception modules increased by about 33% over the original sequential system, while the real-time controller was busy 95% of the time. The average walking cycle dropped from 80 seconds to about 60 seconds. This reduces power consumption per distance traveled, since many components (e.g, computers and amplifiers) draw significant current at all times.

Monitoring and Error Recovery To be self-reliant, an autonomous mobile robot system needs to react to environmental changes in an appropriate manner. In particular, it must determine when its present course of action is failing, and try alternative plans. The Task Control Architecture facilitates building such self-reliant systems through its monitoring and exception handling constructs.

The Ambler system utilizes several strategies to provide self-reliance: use of redundant monitoring and sensing; use of a hierarchy of exception handlers; and incrementally increasing reliability by layering monitors and exception handlers.

Redundancy is important in constructing reliable systems — single points of failure should be avoided if at all possible. This is manifested in the Ambler system partly through the use of redundant hardware and software safety checks during critical operations, and partly through the use of redundant sensing and monitoring. For example, the planning code uses a model of the Ambler to plan moves that fall within the kinematic limits of the mechanism. The real-time controller uses a similar geometric model to assure the validity of moves prior to executing them. Finally, hardware limit switches will be triggered in cases where the software fails to limit the moves. Similarly, the real-time controller monitors the force sensors in the feet to detect ground contact. Occasionally the sensors fail, causing the leg to jack up the body as it drives into the ground. This situation is detected by monitoring the inclinometers: if more than one degree of tilt is observed, all motion is halted and the body is automatically releveled.

An organizing principle used in the Ambler software system is to keep planning and exception handling functions separate, using separate modules for each, but with the exception handlers

Figure 16: Task tree with monitors and exception handlers

organized into a hierarchy similar to the planning hierarchy. These principles of separating and hierarchically structuring exception handlers are directly supported by TCA. TCA provides control constructs to associate monitors and exception handlers with nodes in the task tree. When a module raises an exception, or a monitor detects a particular condition, TCA searches up the task tree to dispatch the closest handler. If that handler cannot deal with the error, the search continues up the tree. In this way, exception handling is kept context-dependent and handlers are tried in order of decreasing specificity. This usually results in the lower levels handling the majority of exceptions, with minimal impact on the overall walking plan.

Figure 16 shows the positions in the task tree of some of the monitors and exception handlers used in the Ambler system. Before every move, a stability monitor checks whether the move keeps the Ambler's center of gravity within the support polygon of the legs, aborting the walking cycle if the move might cause the Ambler to tip over. After every footfall, a monitor analyzes the force/torque measurements to determine if the foot is in an unstable configuration (i.e., if the point of ground contact is very off-center). If so, the leg is moved 10 cm at a time away from the point of contact until a stable footfall is found.

During leg motion, the controller monitors the force sensor and stops motion if contact is detected. An exception handler moves the foot up and away from the point of contact and then commands the controller to finish the leg move. The controller also detects hardware faults (e.g., amplifier faults, servo faults, limit switches). For faults that might be intermittent, the Ambler system retires the move, giving up (and cancelling all active tasks) if the fault continues after several attempts.

Finally, there are several exception handlers to deal with failures of the planners to find suitable plans. If the gait planner fails, the first exception handler tries to re-plan with the latest robot pose (since the system plans in advance, sometimes the planned and actual poses differ). If this fails, the system *shuffles* the legs, putting them into a standard configuration, after which the normal planning cycle takes over. If that fails, the walking tasks are terminated, and the user must input a new path to traverse.

These monitors and exception handlers (and there are several others, as described in [36]), together with those implemented in the real-time controller (described in Section 3.4.1) provide the software system with enough reliability to walk many steps without human intervention. In particular, the Ambler has walked continually over 1000 steps (over half a kilometer). Exceptional situations are detected on the average of once every 10–20 steps, depending on the ruggedness of the terrain and the curvature of the path. Only on rare occasions is control passed back to the human operator—most situations are handled by the controllers. Even on those rare occasions, however, the Ambler system safely halts motions and maintains itself in a stable configuration while awaiting further input from humans. This is exactly the type of behavior required of a spatially and temporally remote vehicle, such as one operating on the Martian surface.

Remarks One of the more difficult tasks in developing concurrent, distributed mobile robot systems is dealing with interactions: interactions between the vehicle and the environment, and interactions among subsystems. These include problems of handling changes and uncertainty in the environment, resource contention between modules, race conditions, etc. The Task Control Architecture provides support for controlling interactions, such as managing physical and computational resources, synchronizing subtasks, and monitoring and handling exceptional situations.

Besides providing a set of commonly needed control constructs, the system development methodology supported by TCA facilitates dealing with interactions. First, all control decisions are encoded explicitly in the module software and maintained explicitly by TCA. This provides for greater predictability and understandability of what the system will do in various situations: exactly how subtasks are temporally related, under which contexts monitors and exception handlers are active, etc. Second, TCA enables new functionality to be added incrementally, without modifying existing code. For example, with the Ambler we first developed a sequential sense/plan/act system (which was easier to test and debug) and then added concurrency by removing temporal constraints [37]. Similarly, we added new monitors and exception handlers incrementally as our experimentation with Ambler uncovered limitations in the way the robot reacted to unexpected situations [36].

All this flexibility comes at a price, however, since TCA must maintain the task trees and track all message traffic in order to assure that the desired control decisions are carried out. While there is the potential for a centralized system bottleneck, in practice we have not found this to be a problem. For the Ambler system, the TCA central control module is active only about 3% of the time. We view this as a small price to pay for the flexibility and ease of use afforded by using the architecture.

4 Autonomous Walking Results

The Ambler autonomous walking system has been tested extensively over the past several years, in both indoor and outdoor trials. This section presents results from a representative sample of the many trials, including indoor navigation over rugged terrain and outdoor, long-distance, long-duration, autonomous walking.

A typical trial begins by executing perception, planning, and control processes on the on-board computers. For convenience, the standard outputs of these processes are displayed on windows on off-board workstations. A human operator enters a path as a sequence of arcs, and issues the command for the Ambler to start walking. The Ambler then operates autonomously, planning and executing every footfall, leg move, body move, leveling maneuver, and any other action or reaction needed.

To date, the Ambler has walked autonomously a total of over 4 km, much of it over rugged, difficult terrain. Table 2 reports statistics of several walking trials, which were selected to indicate the progressive escalation of challenges and capability. In the table, the column labeled "planar travel" indicates the horizontal distance traveled by the Ambler body, but does not include vertical body motions. The "leg travel" column reports the sum of the planar distances traveled by the legs, measured from pick-up to set-down. The total leg distance can be greater than six times the body travel during trials in which the Ambler is executing many point turns, since in that maneuver

Date	Terrain	Body Moves	Planar Travel (m)	Body Rotation (rad)	Leg Travel (m)	Total Time (hr)
1990	Indoor obstacle course	27	11	2	—	0.7
1991	Outdoor obstacle course	100	25	2	—	3.5
1992	Indoor obstacle course	397	107	55	901	7.0
1992	Outdoor field	151	46	—	—	—
1992	Outdoor field	1219	527	20	3330	20.8

Table 2: Statistics from selected walking trials

The dash indicates that this data was not recorded.

some legs travel in retrograde motion (see Section 3.3.1). Most of the leg travel occurs during circulation. Walking time includes the time required for movement, sensing, mapping, planning, and processing; it excludes idle time due, for instance, to debugging. Average walking speed over all types of terrain is 35 cm/minute. Walking speed for straight-line, benign terrain is about 40 cm/minute.

4.1 Indoor Trials

For indoor trials, the Ambler operated on obstacle courses fashioned from 40 tons of sand, 20 tons of boulders, a 30° wooden ramp, and various other objects (Figure 17). The courses typically

Figure 17: Ambler traversing indoor obstacle course

Ambler on sandy terrain with meter-tall boulders (under legs and body), ditches (the center leg on the far stack is standing in one), and ramp (lower right).

Figure 18: Top view of figure-eight circuit

Boulders are drawn grey. Units are meters.

included rolling, sandy terrain with several boulders one meter or taller, approximately ten boulders one-half meter tall, a ditch, and a ramp. The largest of the boulders was 1.5 m tall, 4 m long, and 2 m wide. With such materials, the obstacle courses pose significant barriers to locomotion. We know of no other robot that could surmount all of the obstacles in the course.

Traversing a variety of these obstacle courses, the Ambler has demonstrated long-term autonomous walking. In one trial, the Ambler took 397 steps and traveled about 107 meters following a figure-eight pattern, each circuit of which covers about 35 m and 550 degrees of turn (Figure 18). In the figure, the thin lines indicate the course followed by the center of the Ambler's body for one complete circuit. The black triangle indicates one randomly selected body pose obtained during the trial. Other trials typically resulted in somewhat different courses, due to changes in the terrain and exact starting location of the robot.

Completing more than two circuits during the trial, the Ambler turned nearly 9 complete revolutions, thus devoting a significant fraction of the trial to turning rather than to advancing. The two circuits were different in the sense that the robot did not take the same step twice.

Figure 19: Ambler walking 527 m in field

4.2 Outdoor Trials

A number of trials were performed outdoors. These differed from the indoor trials primarily in that we were able to test the ability of the Ambler to traverse significant slopes (tens of degrees over multiple vehicle lengths) and to operate autonomously for long distances (up to half a kilometer at a time).

For one set of outdoor trials, the Ambler operated in a parking lot strewn with wooden obstacles such as boxes, pyramids, and ramps. In one of these trials, the Ambler took 100 steps along a gently curving arc, traveling about 25 m over a variety of obstacles. In another trial, the Ambler walked in the parking lot at night, without lights, since the laser rangefinder does not require ambient illumination. In fact, we observed the range images to be noisier during the day, because the signal-to-noise ratio is higher without ambient illumination.

For another set of outdoor trials, the Ambler operated on rolling, grassy terrain (Figure 19). The site was a field located within a hilly, wooded area at an industrial park. Although this site did not contain obstacles like those encountered in the indoor trials, the site posed its own challenges: steeper slopes and side-slopes, as well as soft ground.

In the longest outdoor trial, the Ambler took 1219 steps, traveling 527 m horizontally and 25 m vertically. The robot followed a meandering course, first climbing a hill and descending it, then roughly following an iso-elevation contour for 250 m, executing a point turn of roughly π radians, climbing a bit to a higher elevation, and then following an iso-elevation contour back to the starting region. While this course is simpler than the indoor figure-eight pattern (Figure 18), because it involves less turning and no boulders, the outdoor trial is notable for both the length and duration (over 20 hours) of autonomous operation. In addition, the Ambler traversed rather steep slopes: it ascended and descended a 30 percent grade, and traversed side-slopes of up to 15 percent grades.

4.3 Remarks

Average walking speed over these trials, including all computation, is 35 cm/min (typical body moves travel about 50 cm). Moving the mechanism is the main limitation to the speed. During operation, the real-time controller is active about 80% of the time, while the planners and perception subsystems are each active about 50% of the time, and the centralized task-level controller is active only about 3% of the time (the total is greater than 100% because operations occur concurrently).

The results of the indoor and outdoor walking trials show that the approach we adopted to control the Ambler meets the challenges identified in Section 1:

- **Self-reliant control** The approach meets this challenge, since the integrated walking system exhibits a high level of autonomy in planning and executing motions (e.g., traversing a specified path), reacting to unsafe conditions (e.g., leveling), learning good footfalls on the basis of past experience, and operating for extended periods. Still, there was some manual intervention in the form of restarting processes and occasionally over-riding a default value.

- **Limited power** The approach meets this challenge by reducing unwanted terrain contact to a negligible (but non-zero) level. Thus, the system transfers far less energy to the terrain than would a purely reactive or purely randomized robot.

- **Rugged terrain traversal** The approach enables locomotion over challenging obstacle courses, soft ground, and significant slopes and side-slopes.

5 Conclusions

This article has presented a robotic system to control the six-legged Ambler planetary rover. The approach presented here integrates perception, planning, real-time control and task-level control to navigate the Ambler, reliably and autonomously, over long distances and rugged terrain. These are capabilities that are essential for operating in remote, natural environments, such as on Mars.

- **Self-reliant control** The rover must depend on its own resources for planning, taking actions, and maintaining its safety and integrity.

- **Limited power** The rover must carry its own power source, forcing it to make energy efficient moves.

- **Rugged terrain traversal** The rover must be capable of traversing a wide variety of terrain, to get to points of geologic interest.

These challenges have influenced our approach to the problem. In particular, the resultant system is quite deliberative, where actions are highly goal-directed, steps are planned out in advance using explicit models of the terrain and the vehicle's kinematics and geometry, and moves are checked multiple times by different modules for safety prior to execution. We believe that this approach to walking is well-suited to meeting the three challenges stated above. For reasons of reliability, we want the robot to avoid making potentially hazardous moves; for reasons of efficiency, we want the robot to take steps that minimize transfer of energy to the environment and maximize progress along the prescribed path; for reasons of rugged terrain traversal, we desire that the robot accurately perceive its environment and take it into account when walking.

This approach carries a price, however: it is both computationally and perceptually intensive. For the Ambler, this is not a problem. Since the mechanism is inherently quite slow (for reasons of power efficiency [2]), the system has sufficient time to perceive and plan while the controller concurrently executes steps. For faster robot mechanisms, there is a tradeoff to be made. In general, we would advocate our deliberative approach to walking for applications in which reliability and safety are more important than speed, and/or if the mechanism will typically be operating in rugged or hazardous environments.

The research reported here is continuing. We are striving to increase the range and reliability of autonomous walking systems by including more monitoring and error recovery strategies in both the low-level controller and higher-level planners. We are also extending system autonomy to higher levels of cognition, including route planning and global navigation, and are trying to better characterize our perception, planning and task-level control algorithms. The aim is to produce a walking system capable of multi-kilometer, multi-day, autonomous operation in rugged, natural environments, whether on Earth or on other planets.

Acknowledgements

Many members of the Planetary Rover project have contributed to the results reported here. In particular, we acknowledge the efforts of Red Whittaker, John Bares, Brian Albrecht, Christopher Fedor, Regis Hoffman, Henning Pangels, Gerry Roston, and David Wettergreen. This research is sponsored by NASA under grant NAGW-1175.

References

[1] C. Angle and R. Brooks. Small Planetary Rovers. In *Proc. IEEE Intl. Workshop on Intelligent Robots and Systems*, pages 383–388, Tsuchiura, Japan, July 1990.

[2] J. Bares and W. Whittaker. Configuration of Autonomous Walkers for Extreme Terrain. *Intl. J. Robotics Research*, 13(1), February 1994.

[3] R. Brooks. A robot that walks: Emergent behavior from a carefully evolved network. *Neural Computation*, 1(2):253–262, Summer 1989.

[4] 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.

[5] R. Brooks, P. Maes, M. Mataric, and G. More. Lunar Base Construction Robots. In *Proc. IEEE Intl. Workshop on Intelligent Robots and Systems*, pages 389–392, Tsuchiura, Japan, July 1990.

[6] P. Burrough. Fractal Dimensions of Landscapes and Other Environmental Data. *Nature*, 294:240–242, 1981.

[7] J. F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[8] R. Chatila, R. Alami, S. Lacroix, J. Perret, and C. Proust. Planet Exploration by Robots: From Mission Planning to Autonomous Navigation. In *Proc. Intl. Conf. Advanced Robotics*, pages 91–96, Tokyo, Japan, November 1993.

[9] R. Chatila and J.-P. Laumond. Position Referencing and Consistent World Modeling for Mobile Robots. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 138–145, St. Louis, Missouri, 1985.

[10] W. Chun, S. Price, and A. Spiessbach. Design and Construction of a Quarter Scale Model of the Walking Beam. Martin Marietta Space Systems, 1989.

[11] E. Dickmanns and B. Mysliwetz. Recursive 3-D Road and Relative Ego-State Recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 14(2), February 1992.

[12] E. Gat. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *Proc. National Conference on Artificial Intelligence*, pages 809–815, San Jose, CA, July 1992.

[13] E. Gat, R. Desai, R. Ivlev, J. Loch, and D. Miller. Behavior control for robotic exploration of planetary surfaces. *IEEE Transactions on Robotics and Automation*, 10(4):490–503, Aug. 1994.

[14] M. Hebert and E. Krotkov. 3-D Measurements From Imaging Laser Radars: How Good Are They? *Intl. Journal of Image and Vision Computing*, 10(3):170–178, April 1992.

[15] S. Hirose. A Study of Design and Control of a Quadruped Walking Vehicle. *Intl. Journal of Robotics Research*, 3(2):113–133, Summer 1984.

[16] S. Hirose and N. Ootsukasa. Design and Development of a Quadra-Rhomb Rover for Mars Exploration. In *Proc. Intl. Conf. Advanced Robotics*, pages 109–114, Tokyo, Japan, November 1993.

[17] S. Hirose, K. Yoneda, and K. Arai. Design of Prismatic Quadruped Walking Vehicle TITAN VI. In *Proc. Intl. Conf. Advanced Robotics*, pages 723–728, 1991.

[18] J. Hodgins and M. Raibert. Adjusting Step Length for Rough Terrain Locomotion. *IEEE Trans. Robotics and Automation*, 7(3):289–298, June 1991.

[19] G. T. Hsu and R. Simmons. Learning footfall evaluation for a walking robot. In *Proceedings of Machine Learning Workshop*, Chicago, IL, June 1991.

[20] Y. Ishino, T. Naruse, T. Sawano, and N. Honma. Walking Robot for Underwater Construction. In *Proc. Intl. Conf. Advanced Robotics*, pages 107–114, 1983.

[21] A. Kemurdjian, V. Gromov, V. Mishkinyuk, V. Kucherenko, and P. Sologub. Small Marsokhod Configuration. In *Proc. IEEE Intl. Conf. Robotics and Automation*, pages 165–168, Nice, France, May 1992.

[22] E. Krotkov and R. Hoffman. Terrain Mapping for a Walking Planetary Rover. *IEEE Trans. Robotics and Automation*, 10(6):728–739, December 1994.

[23] E. Krotkov, R. Simmons, and C. Thorpe. Single-leg walking with integrated perception, planning, and control. In *Proc. IEEE International Workshop on Intelligent Robots and Systems*, pages 97–102, Tsuchiura, Japan, July 1990.

[24] I. Kweon and T. Kanade. High-Resolution Terrain Map from Multiple Sensor Data. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 14(2):278–292, February 1992.

[25] S. Mahalingam and W. Whittaker. Terrain adaptive gaits for walkers with completely overlapping leg workspaces. In *Proc. Robots 13*, pages 1–14, Gaithersburg, MD, May 1989.

[26] L. Matthies. *Dynamic Stereo Vision*. PhD thesis, Carnegie Mellon University, October 1989. Available as Technical Report CMU-CS-89-195.

[27] L. Matthies. Stereo Vision for Planetary Rovers: Stochastic Modeling to Near Real-Time Implementation. *Intl. J. Computer VIsion*, 8(1):71–91, July 1992.

[28] F. Pfeiffer, H. J. Weidermann, and P. Danowski. Dynamics of the walking stick insect. *IEEE Control Systems*, 11(2):9–13, February 1991.

[29] C. Proy, B. Hotz, O. Faugeras, P. Gernesson, and M. Berhod. Onboard Vision System for a Mobile Planetary Exploration Robot. In *Proc. Workshop on Computer Vision for Space Applications*, pages 2–12, Antibes, France, September 1993.

[30] M. Raibert. *Legged Robots That Balance*. MIT Press, Cambridge, Massachusetts, 1986.

[31] G. Roston and E. Krotkov. Dead Reckoning Navigation for a Six-Legged Walking Robot. In *Proc. IEEE Intl. Workshop on Intelligent Robots and Systems*, pages 607–612, Raleigh, North Carlolina, July 1992.

[32] D. Rumelhart, G. Hinton, and R. Williams. Learning internal representations by error propagation. In D. Rumelhart and J. McClelland, editors, *Parallel Distributed Processing, Volume 1: Foundations*, pages 318–362. MIT Press, 1988.

[33] Russell. Odex I: The First Functionoid. *Robotics Age*, 5(5):12–18, September/October 1983.

[34] U. S. Congress Office of Technology Assessment. *Exploring the Moon and Mars: Choices for the Nation*. U.S. Government Printing Office, Washington, DC, July 1991. OTA-ISC-502.

[35] R. Simmons. An architecture for coordinating planning, sensing, and action. In *Proceedings of DARPA Workshop on Innovative Approaches to Planning, Scheduling and Control*, pages 292–297, San Diego, CA, November 1990.

[36] R. Simmons. Monitoring and error recovery for autonomous walking. In *Proc. IEEE International Workshop on Intelligent Robots and Systems*, pages 1407–1412, July 1992.

[37] R. Simmons. Structured control for autonomous robots. *IEEE Transactions on Robotics and Automation*, 10(1):34–43, Feb. 1994.

[38] S. Song and K. Waldron. *Machines that Walk: The Adaptive Suspension Vehicle*. MIT Press, Cambridge, Massachusetts, 1989.

[39] H. Stone. Design and Control of the MESUR/Pathfinder Microrover. In *Proc. Intl. Conf. Advanced Robotics*, pages 263–270, Tokyo, Japan, November 1993.

[40] C. Thorpe, editor. *Vision and Navigation: the Carnegie Mellon Navlab*. Kluwer, 1990.

[41] M. Turk, D. Morgenthaler, K. Gremban, and M. Marra. VITS- A Vision System for Autonomous Land Vehicle Navigation. *PAMI*, 10(3), May 1988.

[42] D. Wettergreen, H. Thomas, and C. Thorpe. Planning Strategies for the Ambler Walking Robot. In *Proc. IEEE Intl. Conf. Systems Engineering*, August 1990.

[43] B. Wilcox and D. Gennery. A Mars Rover for the 1990's. *Journal of the British Interplanetary Society*, 40:484–488, 1987.

[44] B. Wilcox, L. Matthies, D. Gennery, B. Cooper, T. Nguyen, T. Litwin, A. Mishkin, and H. Stone. Robotic Vehicles for Planetary Exploration. In *Proc. IEEE Intl. Conf. Robotics and Automation*, pages 175–180, Nice, France, May 1992.