Proceedings of the 1991 IEEE
International Conference on Robotics and Automation
Sacramento, California - April 1991

An Integrated Walking System for the Ambler Planetary Rover

Reid Simmons and Eric Krotkov
School of Computer Science, Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

The CMU Planetary Rover project is developing the Ambler, a six-legged robot designed for planetary exploration. We have developed reliable and efficient control, perception, and planning algorithms suitable for navigating rugged terrain. The components have been integrated into a system that autonomously walks the Ambler along routes and over obstacles.

1 Introduction

To autonomously explore other planets, a robot must have mobility, reliability, and a high degree of competence in sensing, planning, and executing motions. The CMU Planetary Rover project is developing a highly mobile and energy efficient walking robot, called the Ambler (Figure 1), which is designed for autonomous navigation in unknown, rugged terrain.

The six-legged Ambler features orthogonal legs and a circulating gait [1]. The orthogonal (RPP) legs decouple horizontal and vertical motions. This minimizes power consumption and simplifies planning and control. The Ambler is able to negotiate meter-sized boulders and ditches, while on a 30^o slope. The legs are arranged in two stacks on central shafts. The stacked legs and central body cavity enable the Ambler to exhibit a unique *circulating gait*, where trailing legs recover through the body to become leading legs. With this circulating gait, the Ambler uses approximately one third the footsteps of a typical follow-the-leader gait.

Walking robots have several advantages over rolling vehicles for navigating terrain. For one, walking vehicles need only discrete contact points, rather than contiguous paths of support which may not exist in rugged environments. Walking robots can also be less prone to tipover, since they can adjust leg lengths to maintain a level attitude, and adjust leg extensions to position the body's center of gravity.

Previous research in walking robots has concentrated

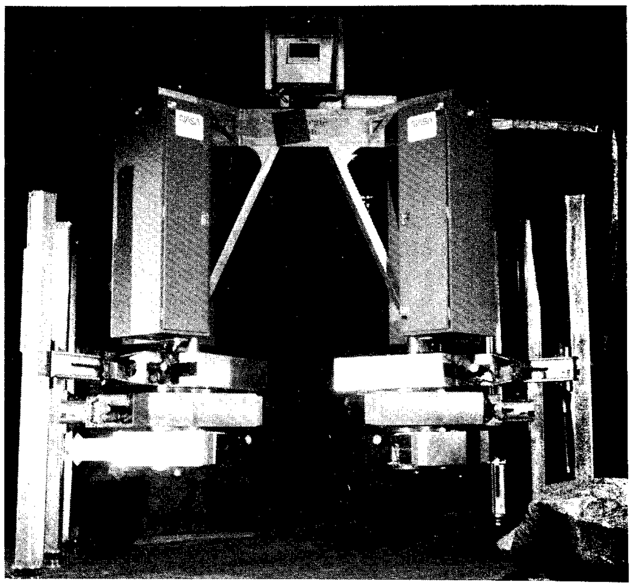
Figure 1: The Ambler Robot

on design and real-time control, both for statically and dynamically stable mechanisms [3, 4, 10, 13]. These robots tend to be either teleoperated or operate under supervisory control. In particular, issues of perception and planning for autonomous walking has received relatively little attention.

We have developed a software system for autonomous navigation of the Ambler that combines the component technologies of control, perception, and planning. Much of our methodology, and several algorithms, derive from our experience developing a walking system for a prototype single leg of the Ambler [7].

The control software coordinates the Ambler's eighteen joints, using force sensors, inclinometers, and kinematic knowledge to detect dangerous configurations. The perception subsystem efficiently constructs elevation maps from laser range images. It also automatically calibrates the laser scanner by sighting targets on the legs. The planning algorithms combine kinematic and terrain constraints to choose leg and body moves

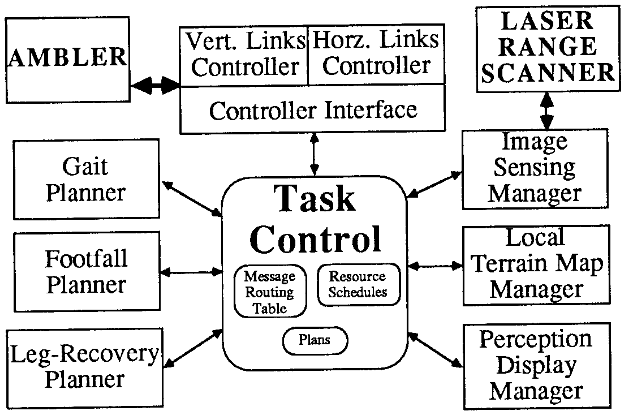

Figure 2: Modules for the Six-Leg Walking System

that attempt to maximize forward progress while minimizing risk to the machine.

The components are integrated using the Task Control Architecture (TCA) [12], which provides interprocess communication, task synchronization, and resource management. The integrated system (Figure 2) has autonomously navigated the Ambler over a variety of routes and across undulating terrain that includes boulders, ramps, and trenches. Given this early success, we are actively working to increase the reliability, competence, and speed of the walking system.

This paper presents an overview of the integrated walking system. The following three sections briefly describe the individual components of control, perception and planning, respectively. Section 5 describes our framework for integrating the components, and presents results to date in autonomous walking.

2 Real-Time Control

A critical component of the walking system is the real-time Controller, which must exhibit high performance and reliability. To achieve the necessary performance, the actuators are controlled using nine Creonics motion control boards, and the Controller itself runs on three CPU boards in a real-time, multi-tasking environment (Figure 2). One CPU is dedicated to trajectory generation, maintaining the dead-reckoned position, and interfacing to the TCA. The other two CPUs execute horizontal and vertical trajectories, sending updates to the Creonics boards every 10msecs. The CPUs communicate with the motion control boards and other electronics via a VME backplane, and communicate with one another through shared memory using semaphores.

Leg motion is specified by a sequence of joint position waypoints. The Controller interpolates the waypoints to produce a trajectory that is linear in joint space, starting and stopping all joint motions simultaneously. To contact the ground, the Controller monitors six-axis force sensors in the feet, halting motion when a threshold is exceeded.

Body motion is specified by waypoints that define 3D positions and rotations in a global coordinate frame. The body is propelled by locking the vertical joints and commanding linear Cartesian-space trajectories for the horizontal joints, then lifting the body, if necessary, to achieve the commanded height. While currently all twelve horizontal joints are actuated, we are using a dynamics model [9] and experiments with the mechanism itself to quantity the tradeoffs between propelling with various subsets of the joints. While fewer joints reduce actuator conflict, each propelling joint needs greater tractive forces, and passive joints must be backdriven.

The Controller also maintains the dead-reckoned position of the Ambler. At power-up, the Controller is supplied with an initial global position and orientation. From the forward kinematics, the Controller computes the positions of all feet in a world reference frame. Following each body move, the Ambler's location is estimated by finding the rotation matrix and translation vector that, with the least squared error, transform the positions of the feet in the vehicle frame to their stored positions in the world frame. This method of dead-reckoning is fairly accurate (typically within 1-2cm), especially since the Ambler is very rigid and control is quite precise.

The Controller includes many safeguards to protect the Ambler mechanism. Each motor has a fail-safe load holding brake that is activated when power is lost. We have developed a safety circuit that monitors the health of the electronics and shuts down motions whenever a failure is detected or a limit switch is passed.

A key safeguarding concept is redundancy in sensors and computation. For example, to augment the safety circuit, the software will not generate trajectories that move legs past joint limits. The Controller both checks whether requested leg motions will collide with other legs or the body, and monitors the force sensors for unexpected collisions with the terrain or the mechanism itself. Similarly, while the Controller precludes movement outside the polygon defined by the supporting legs, it also uses inclinometers to detect excessive tilt.

For autonomous walking, the Controller is integrated with the other software components (Section 5). In addition, a menu-based mode of interaction is provided that enables users to manually control the Ambler, set parameters such as controller gains, and view status for debugging.

3 Perception

Our imaging sensor is a scanning laser rangefinder, which measures both reflectance and range. We use a laser scanner because it directly recovers the environment's three-dimensional structure. Therefore, it supplies 3D data more rapidly and reliably than passive vision techniques such as binocular stereo and motion.

Mounted on top of the Ambler's body is a scanner manufactured by Perceptron (Figure 1) [8]. The Perceptron acquires data in 256x256 pixel images at a rate of 2Hz. It digitizes to 12 bits over approximately 40m, which provides a range resolution of 0.98cm. The measurements cover 60^o in azimuth and 60^o in elevation, just enough of a field of view to image where recovering legs are typically placed.

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 the scanner field of view, and processes the reflectance image to locate the leg. The leg positions in the images and those positions returned by the Controller are used to compute the transformation that minimizes the measurement errors [5].

The functionality of the perception subsystem is divided into three software modules that acquire images, construct maps, and display terrain data (Figure 2). The Image Sensing Manager (ISM) initializes the Ambler's laser scanner and acquires frames of range data. The ISM can acquire 1) real images from the Perceptron itself, 2) virtual images stored on disk, and 3) synthetic images computed by our 3D graphics simulator [14]. Each image is tagged with a transformation relating the scanner frame and world coordinate frames, composed of transformations derived from the calibration procedure and the Ambler's dead-reckoned position.

The Local Terrain Map (LTM) Manager constructs and maintains elevation maps. Elevation maps are used as the primary terrain representation because they are well-suited for our navigation planning algorithms and for representing the types of rugged terrain the Ambler is meant to traverse. In addition, they can be accessed in a simple way, by specifying the boundary of a region of interest.

The LTM Manager uses the *Locus Method* to construct local elevation maps from single frames of range data (Figure 3). The Locus Method is an efficient algorithm for transforming and interpolating range data from the sensor frame into Cartesian coordinates [2]. To construct maps useful for planning, the LTM Manager also masks occluding legs out of the range image, marks occluded and unknown regions of the map, and filters the map to reduce noise [6]. The LTM Manager creates composite terrain maps by combining the maps constructed by the Locus Method, updating points where they overlap with the maximum likelihood estimate of the elevation [6].

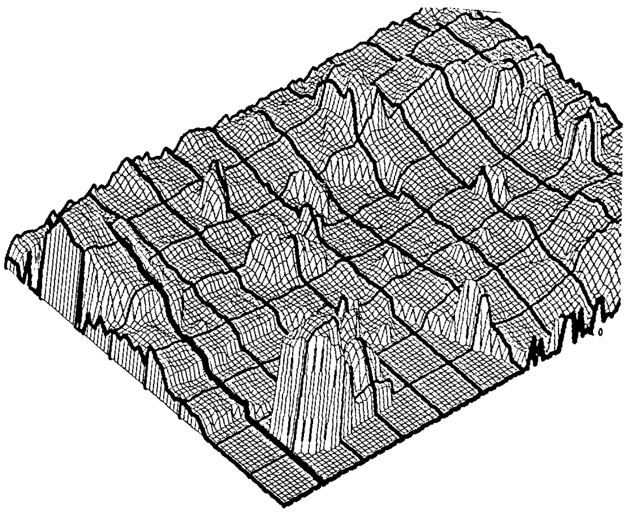
Figure 3: Local Terrain Map

The planning modules interface with the LTM Manager by requesting portions of the elevation map within a specified polygonal region and at a specified map resolution. The LTM Manager returns a rectangular grid that is big enough to contain the polygon, labeling cells outside the polygon or outside the scanner field of view as *unknown*. Cells occluded by other objects are labeled as such, along with the maximum possible elevations derived from knowledge of the viewing geometry. While the requested maps are computed on demand, they are cached so that future requests for the same (or overlapping) regions do not have to be recalculated.

The Perception Display Manager (PDM) interactively displays elevation maps within a user-specified region, which has proven vital in debugging both perception and planning software. Creating an independent module to manage map display was done to decouple display from map construction. Thus, the user can display maps when it is convenient or desirable. Also, it is easier to extend a well-modularized graphics interface. For example, to aid in correlating planned and sensed footfall locations, the PDM has been extended to indicate requested 3D positions on the map.

4 Planning

To take advantage of the Ambler's high mobility, and to cope with a wide range of terrain, we are developing a flexible planning algorithm that produces circulating gaits that are conservative with respect to Ambler stability. The Gait Planner produces leg and body moves to follow arcs of any radius [15]. In particular, large radius arcs yield nearly straight-line motion, and zero radius arcs produce point turns. The planner produces a set of feasible leg and body moves by combining kinematic and pragmatic constraints on motion. This constraint-based approach often reduces search, since no *a priori* commitment is made to which constraints are the most important. Also, it is fairly easy to incorporate new constraints as needed.

The kinematic constraints include limits on leg movements and the need to avoid collisions with other legs. The constraint on body motion is computed by intersecting the reachable space for the legs on each stack (Figure 4). For safety, we impose the constraint that the Ambler's center of gravity must always remain within the *conservative support polygon (CSP)*. The CSP is the intersection of the support polygons obtained (conceptually) by removing the support from each leg in turn (Figure 5). By staying within the CSP, the Ambler will remain stable even if any single leg fails.

Once the set of geometrically desirable footfalls has been determined, the Gait Planner chooses one footfall (and associated body move) based on terrain characteristics. The basic terrain constraints are stability and traction. While these cannot be measured directly from elevation maps, we can extract features from the maps that are good indicators, such as terrain slope, roughness, and curvature.

The Footfall Planner combines these features using an evaluation function whose coefficients are learned by a neural net. The net is trained using human-supplied preferences on footfall locations (e.g., "good", "very bad"). For each pair of locations where the evaluation function indicates a contrary preference, an error term is created proportional to the difference between the two values. Back-propagation is used to update the weights in the net based on the error term. To force the learned function to distribute in the range [0,1], the training data is periodically seeded with perfectly good and perfectly bad footfalls (i.e., all features either 0 or 1).

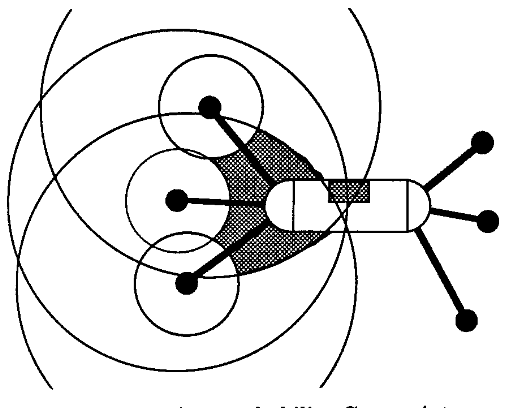
Figure 4: Body Reachability Constraint

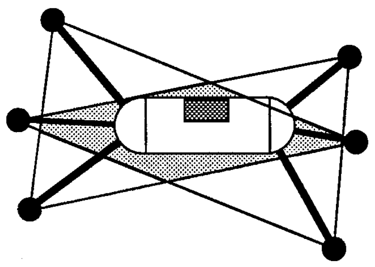
Figure 5: Conservative Support Polygon

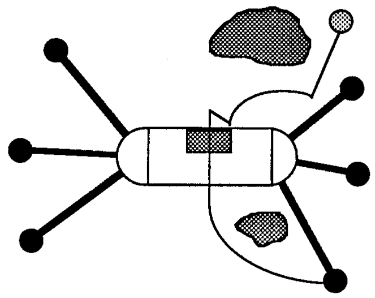
Figure 6: Horizontal Leg Recovery Trajectory

This type of training will prevent the Ambler from stepping in bad locations. In addition, we intend to incorporate feedback from the Ambler's dead-reckoning and force sensors to enable the system to incrementally refine the footfall evaluation function as it walks.

Once a footfall location is chosen, the Leg-Recovery Planner (LRP) determines an efficient trajectory of the leg through space. The LRP first produces a trajectory for the horizontal joints, using several simple strategies. One strategy is to move directly to the goal location. Another, used to avoid tall obstacles, is to first pull the leg in close to the body, swing the rotary joint, then extend the leg. Yet a third strategy produces circulating gaits by using the above strategies to solve two subgoals: 1) get to the entrance of the body cavity, and 2) get from the other side of the body to the goal location (Figure 6). After producing a horizontal trajectory, the LRP adds vertical motions that ramp the foot over the terrain while minimizing travel time.

5 Integration

The perception, planning, and real-time control components described above are integrated using the *Task Control Architecture (TCA)*, which provides facilities for coordinating distributed robot systems [11, 12]. In particular, TCA supports interprocess message passing, hierarchical task decomposition, resource management, and execution monitoring and error recovery.

The Ambler walking system consists of six modules, plus the TCA central control (Figure 2). The planning, perception, and TCA modules are run on three Sun workstations, with the real-time Controller operating on an additional three processor boards. The basic system architecture, including the TCA and interfaces between components, was adopted from our prototype single leg walking system [7].

Autonomous walking begins when the Gait Planner receives a message to traverse an arc. The planner uses the kinematic and terrain constraints to choose a footfall and body move. TCA routes the footfall request to the Leg-Recovery Planner which produces a leg trajectory. TCA queues the body move command until the planned leg move has been executed by the Controller. The Gait Planner is then recursively invoked to plan the next step, until the goal location is reached. In addition, the planners request elevation maps from the Local Terrain Map Manager, and all modules request dead-reckoned positions from the Controller.

An advantage of the message passing protocol of TCA is that it mandates the specification of interfaces between modules. Besides minimizing module interactions, well-defined interfaces facilitate the development of "plug-compatible" modules. For example, to aid in developing the planners, we implemented 2D and 3D graphics simulators that have the same interface as the Controller module [14].

We have embarked on a comprehensive experimental program to test the walking system. The program includes both simulations and experiments with the actual mechanism. Experiments with the Ambler itself are conducted within a large indoor area that is covered with sand and includes a wooden ramp, hillocks, trenches, and boulders (Figure 7).

Our experimental methodology is to incrementally add modules to the system in order to tackle increasingly difficult terrain. We began on flat, non-compliant floor, manually operating the Ambler through the Controller module. During these runs minimal slippage (less than 1cm) was observed during body moves.

Next, the Gait Planner commanded the Controller (via TCA) to walk along a variety of arcs, including straight-line paths and point turns. This tested the interfaces between modules and demonstrated the Gait Planner's ability to produce acceptable footfalls.

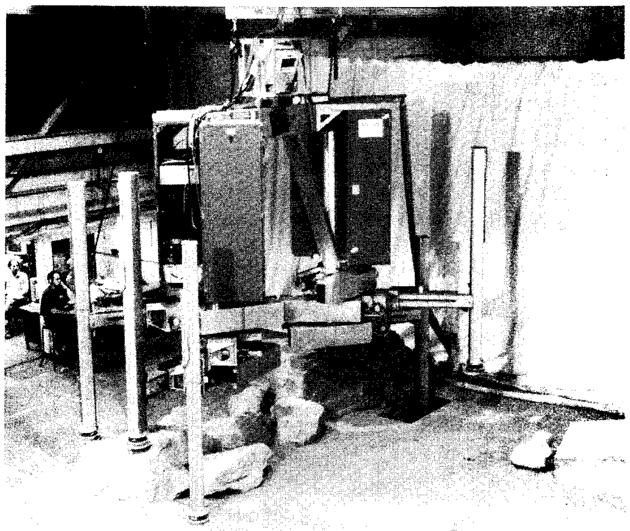
Figure 7: The Ambler Obstacle Course

At this point, the perception modules were added. The scanner calibration and map building software were rigorously tested and evaluated. Although we experienced some problems with our Perceptron scanner, the software was eventually able to produce fairly accurate and reliable maps (Figure 3).

Given perception, the addition of the Leg-Recovery Planner enabled the Ambler to autonomously traverse its obstacle course (Figure 7). The system uses the methods described in [11] to concurrently plan and execute steps, and to monitor for plan failures. Operating concurrently, the system has an average walking speed of 35cm/min, of which 75% of the time is spent by the Controller in actually moving the mechanism, with the planners and perception subsystems each active about 50% of the time. In total, the Ambler has walked autonomously well over a kilometer.

Future prospects include complete integration of Footfall planning, and increasing the performance and self-reliance of the walking system. By adding further concurrency to the system, particularly to the Controller, we expect to double the walking speed and achieve continuous motion. During testing, the Ambler is normally attached to a crane sling to catch it in case of mechanical or algorithmic failure. Although on occasion the Ambler has walked "slingless", we want to add more hardware and software checks before removing the sling entirely. In addition, we intend to move the electronic and computing equipment to on-board racks this year.

6 Conclusions

The task of autonomous navigation in rugged terrain requires reliable and efficient control, perception, and planning algorithms. This paper has presented the components used to autonomously control the Ambler, a six-legged walking robot. Advances have been made in real-time control of legged robots, mapping 3D rugged terrain from range data, and planning safe and efficient leg and body moves for the Ambler mechanism.

We also described how the individual components are integrated into a complete robotic system. The Task Control Architecture is used to connect distributed components and coordinate their activities.

Much of our integration effort was spent in developing tools for displaying and recording the workings of the perception, planning and control algorithms as aids for debugging. In addition, methods for simulating the hardware components (both the real-time control and image acquisition) proved to be advantageous in testing individual components prior to integration.

The experimental results to date in autonomous walking are encouraging. Our development program is ongoing, with the ultimate aim of autonomous, long-term walking in outdoors, natural terrain.

Acknowledgements

Many members of the Planetary Rover project are directly responsible for the Ambler integrated walking system. In particular, we acknowledge the efforts of Brian Albrecht, John Bares, Lonnie Chrisman, Christopher Fedor, Regis Hoffman, Goang-Tay Hsu, Terry Lim, Henning Pangels, Gerry Roston, Jay West, and David Wettergreen. This research is sponsored by NASA under contract NAGW-1175.

References

[1] J. Bares and W. Whittaker. Walking robot with a circulating gait. In *Proc. IEEE International Workshop on Intelligent Robots and Systems*, Tsuchiura, Japan, July 1990.

[2] M. Hebert, T. Kanade, and I. Kweon. 3-d vision techniques for autonomous vehicles. Technical Report CMU-RI-88-12, The Robotics Institute, Carnegie Mellon University, 1988.

[3] S. Hirose. A study of design and control of a quadruped walking vehicle. In *International Journal of Robotics Research*, Summer 1984.

[4] Y. Ishino, T. Naruse, T. Sawano, and N.Honma. Walking robot for underwater construction. In *Proc. Internaltional Conference of Advanced Robotics*, pages 107–114, 1983.

[5] E. Krotkov. Laser Rangefinder Calibration for a Walking Robot. In *Proc. IEEE International Conference on Robotics and Automation*, Sacremento, CA, April 1991.

[6] E. Krotkov, C. Caillas, M. Hebert, I. Kweon, and T. Kanade. First results in terrain mapping for a roving planetary explorer. In *Proc. NASA Conf. on Space Telerobotics*, Jet Propulsion Laboratory, Pasadena, CA, January 1989.

[7] 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*, Tsuchiura, Japan, July 1990.

[8] I. Kweon, R. Hoffman, and E. Krotkov. Experimental Characterization of the Perceptron Laser Rangefinder. Technical Report CMU-RI-TR-91-1, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, January 1991.

[9] D. Manko and W. Whittaker. Inverse dynamic models used for force control of compliant closed-chain mechanisms. In *Proc. ASME Design and Automation Conf.*, pages 61–66, Montreal, Canada, August 1989.

[10] M. Raibert. *Legged Robots That Balance.* MIT Press, 1986.

[11] R. Simmons. Concurrent planning and execution for a walking robot. In *Proc. IEEE International Conference on Robotics and Automation*, Sacremento, CA, April 1991.

[12] R. Simmons, L. J. Lin, and C. Fedor. Autonomous task control for mobile robots. In *Proc. IEEE Symposium on Intelligent Control*, Philadelphia, PA, September 1990.

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

[14] H. Thomas, D. Wettergreen, and C. Thorpe. Simulation of the ambler environment. In *Proc. Modeling and Simulation Conference*, Pittsburgh, PA, May 1990.

[15] D. Wettergreen, H. Thomas, and C. Thorpe. Planning strategies for the ambler walking robot. In *Proc. IEEE International Conference on Systems Engineering*, August 1990.