PATTERNED SEARCH PLANNING AND TESTING FOR THE ROBOTIC ANTARCTIC METEORITE SEARCH

Kimberly Shillcutt (Carnegie Mellon University)
Dimitrios Apostolopoulos (Carnegie Mellon University)
William Whittaker (Carnegie Mellon University)
Field Robotics Center
Carnegie Mellon University
Pittsburgh, PA 15213
kimberly@ri.cmu.edu
412-268-7086

Abstract

The goal of the Robotic Antarctic Meteorite Search is to enable discovery of meteorites in Antarctica by a mobile robot. The extreme environment makes it one of the best places to find meteorites, but one of the worst places for humans to work. The meteorite-finding robot will traverse an ice field in a pattern designed to cover the area completely, stopping to investigate potential meteorites with an array of sensors. High level autonomy is needed for this project in many areas: scientific sensing, scientific analysis, navigational sensing, navigational planning, and mission planning. Navigational planning for this project primarily involves generating *coverage patterns* for traversing the ice fields as completely as possible. Multiple types of patterns are considered, and some of their benefits and problems are discussed. Several of the coverage patterns have been tested in a gravel slag heap in Pittsburgh and on the ice in Antarctica, and the results of these tests are described. The affect of different environments and robot locomotion configurations on control issues, such as the implementation of the pure pursuit algorithm, and maintaining and regaining a path, are discussed.

Project Goal

Antarctica is one of the most remote locations on Earth. Its cold and pristine environment makes it one of the best places to find meteorites, with little melting or surface erosion to hide meteorite falls. The extreme conditions also make it difficult for humans to work there, but a robot designed to explore this area could provide great scientific returns. Studies of meteorites provide information about other bodies in the solar system, leading to a better understanding of the entire system's formation. Exploring and mapping the Antarctic area also provides information about geologic and other processes active there. For example, stranding surfaces, where numerous meteorites can often be found, result from the combination of upward thrusting ice flows and the sublimation of surface ice, along with constant winds which expose the ice and prevent new snow accumulation, revealing previously buried objects.

In the same manner as human meteorite searchers (see Figure 1), the meteorite-finding robot would traverse a designated area in a pattern designed to completely cover the entire location, using a long range sensor to pick out target objects with a high probability of being meteorites. When such a target object is found, the robot would maneuver closer to the object and use additional sensors to classify it as a meteorite or specific type of terrestrial rock, and then return to and continue the coverage pattern.

Figure 1: Human Team Systematic Searching of an Ice Field

To enable a robotic search for meteorites, the Nomad robot has been winterized for Antarctic conditions. Nomad is a four-wheeled robot designed by researchers at Carnegie Mellon University as a prototype planetary explorer. In the summer of 1997, Nomad and its autonomous navigation system was tested extensively in Chile's Atacama Desert. With NASA funding, Nomad was then retrofitted for the cold and sent to the Patriot Hills area of Antarctica in the late fall of 1998. There, the robot, its navigation system, several scientific sensors, and additional autonomy software modules were tested over a month-long period. Included in the tested autonomy modules was the navigation planner, which produces commands for following coverage patterns, using differential GPS and dead reckoning for localization. The remainder of this paper reviews the autonomy system designed for the search for meteorites, details the navigation planner, and describes the field testing results from both Pittsburgh and Antarctica.

Overview of Nomad's Autonomy System

The two primary robotic tasks for this meteorite search project are navigating, either to designated points or along a coverage pattern, and taking measurements with various sensors. These two tasks are represented by two software modules, the *navigation planner* and the *sensor manager.* In brief, the navigation planner is given a path or pattern to follow, and then computes the required steering direction based on the robot's current position. The steering direction can be sent to a simulator, producing a simulation of the entire path or pattern to use in planning. Alternately, to physically enact the path or pattern, the steering direction is sent to the navigation system, which directs the command to an arbitration module. This module takes input not only from the navigation planner, but also from an obstacle avoidance module using navigational sensors. If the steering direction desired by the planner is not vetoed by the obstacle avoidance module, the steering command is given to the real-time system and the robot hardware. A block diagram showing this autonomy system structure is in Figure 2. [1]

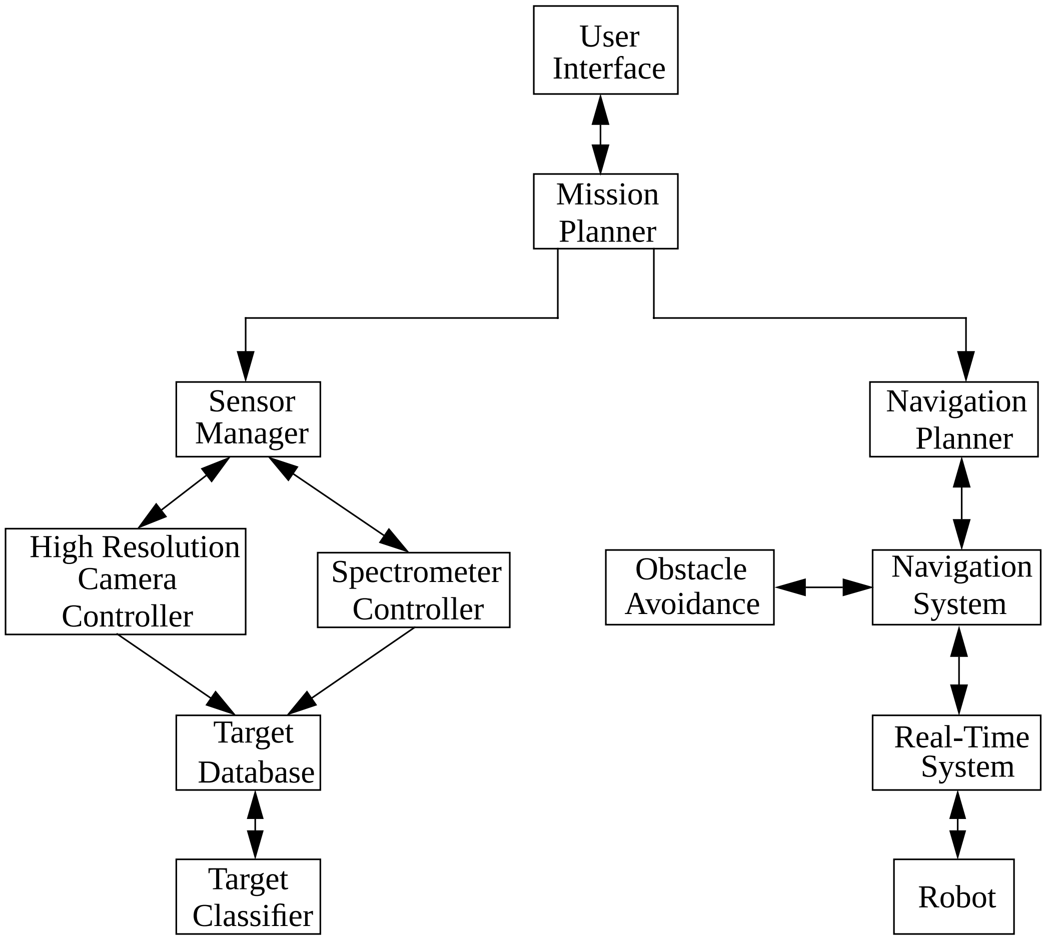


Figure 2: Autonomy System Block Diagram

The second software module, the sensor manager, takes commands to use a sensor on a target object, and then routes the command to the proper sensor controller. If multiple steps are involved in taking a measurement, such as first performing a calibration, the sensor manager handles these details. The manager also reports on the success or failure of a measurement. The sensor controllers interact with the sensor hardware, such as a high resolution camera with a pan-tilt head or a spectrometer. The controllers also save the measurement data and notify the database that a new measurement has been taken. The database is then accessed by the classifier, which processes the sensor measurements and uses them in a Bayes network to determine whether a target object is a meteorite or a terrestrial rock.

Overseeing the navigation planner and the sensor manager is the mission planner. This planner takes information about the costs and benefits of performing different tasks, relative to the robot's resources and goals, and decides which task to perform. It then sends the chosen command to the appropriate module. In determining the costs and benefits, the planner may request a path or pattern simulation, information about the time and power costs of using the sensors, or information from the classifier about the need for gathering more data on a particular target object. Currently, the mission planner operates in an alternate mode, which routes commands from a user interface directly to the navigation planner or sensor manager, replacing the autonomous planning aspects with human decisions.

Navigation Planner

The navigation planner produces plans for three types of driving tasks: driving to a waypoint, driving in a coverage pattern, and maneuvering into a given orientation and location. The first two types have been implemented so far. To travel to a waypoint, the x and y coordinates of the waypoint relative to the robot's current position are needed as input. If the waypoint is behind the robot, indicated by a negative y coordinate, the robot is told to turn as sharply as possible, either clockwise or counterclockwise depending on the x coordinate of the waypoint. If the waypoint is in front of the robot, two cases are possible. In one case, the x coordinate is small enough, and the robot drives straight forward. In the other case, the robot turns toward the waypoint as it drives until the x coordinate becomes small enough.

Future plans for the waypoint task include incorporating an obstacle map and using a common optimizing planner such as the A* algorithm to compute more complex paths in the presence of obstacles. The A* algorithm can use a grid-based map with obstacles marked, and finds the least costly path in the shortest amount of time. [5] At buffer distance is also part of the waypoint input, describing how close the robot needs to get to the waypoint in order to consider the goal reached. Due to the lag in obtaining the robot's new coordinates from the differential GPS and in stopping the robot, this buffer needs to be some value other than zero. Values close to a meter are generally sufficient for Nomad. For simulation and planning purposes, the robot's speed and turning radius can be used to predict the next position of the robot after some discrete time period has passed. Currently, the position is calculated and reported every second.

The goal for coverage pattern planning is to cover an area as completely as possible, while also considering any other constraints that may be present, such as power usage and obstacles. Two types of coverage patterns have been implemented so far. One is the common straight rows, or back and forth pattern. The second is a spiral pattern, with the robot starting in the middle following a circular pattern and increasing the circle's radius every half of a circle. Another possible type of pattern is a sun-following pattern. For a polar location such as Antarctica, in the summer time the sun is always visible, circling around the horizon at a low elevation. A robot with solar panels on its sides can maintain an optimal orientation with respect to the sun by turning just enough to keep up with the sun's rotation. One way of doing this and still covering an area is to follow a roughly straight row, but with a slight curve to it to maintain the proper orientation. After the robot turns around at the end of the row, the robot can continue turning slightly such that the solar panel on the opposite side of the robot maintains the best orientation. Figure 3 shows the simulated groundtrack of a robot following such as pattern. There is overlapping of the area covered with this pattern, as well as missed area near the middle, but this pattern enables a near optimal amount of power to be generated by the robot's solar panels. Conversely, the straight rows and spiral patterns have minimal overlap and complete coverage, but do not produce as much power, which is a vital concern for a solar powered robot. [6]

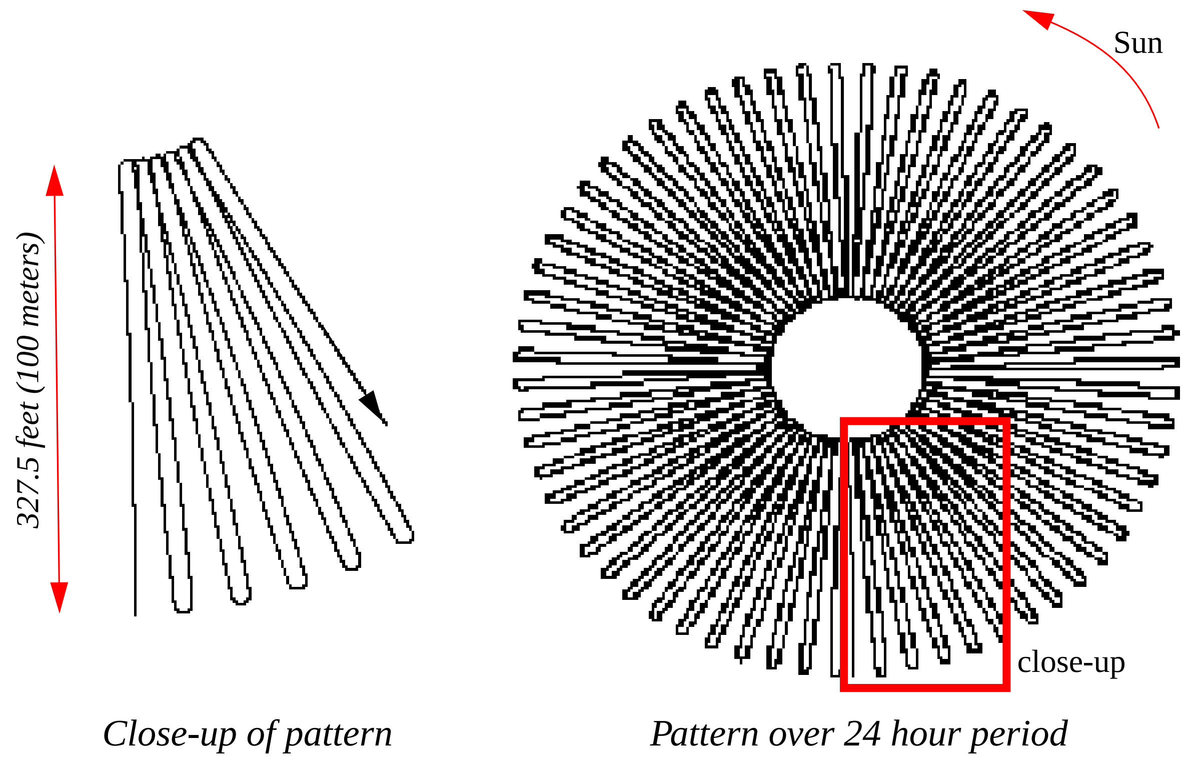


Figure 3: Simulated Example of a Coverage Pattern: Sun-Following

The information needed to initiate a coverage pattern depends on the type of pattern. All patterns require the robot's starting point and the width between successive rows or spirals. This width is generally based on the field of view of the sensor being used for the search. For the straight rows pattern, the length of the rows to travel and the total width of the pattern is needed. Alternatively, a polygonal area to search can be given, and the navigation planner can modify the pattern to fit the area. For the spiral pattern, the maximum radius to spiral out to, and the direction of travel, clockwise or counterclockwise, needs to be given.

The navigation planner maintains state information for the pattern being enacted, including the pattern parameters described above, the current direction of travel, and the current segment of the pattern, such as row or end turn. This information is used to determine the closest point in the pattern to the robot's current location, since deviations may occur due to obstacles or inaccuracies in following the path. [3] With this state information stored, a pattern can be interrupted temporarily by commands to drive to a waypoint, with a later command then returning the robot to the coverage pattern at the point where it first departed. The patterns can be simulated in the same manner as waypoint paths are simulated, with the robot's position calculated and reported at discrete time intervals.

In order for the robot to follow the patterns or paths accurately, even in the absence of obstacles, a path following adjustment algorithm is required due to terrain variations and time lag of the robot in following commands, as well as mechanical imperfections. In addition, the navigation system only allows a discrete number of turning arcs to be commanded, so if the exact curvature required for a pattern is not available, the slight difference between commanded and desired turning radii will cause deviations. To remedy these problems, a *pure pursuit algorithm* was implemented, where an intermediate waypoint on the path is selected and the robot is commanded to drive toward that waypoint. [2] The intermediate waypoint changes continually, and is based on the robot's current position and a defined lookahead distance.

When the robot is near the path, the waypoint selected is the point on the path a lookahead distance away from the current location, in the direction the robot is traveling. When the robot is farther off the path than a lookahead distance, the waypoint selected is the nearest path point to the robot's current location.

Large lookahead distances result in a gradual and smooth regaining of the path, but one which may take a considerable amount of time. Short lookahead distances regain the path quicker, but may result in oscillations about the path. Selecting the best lookahead distance depends on the reaction time and maneuverability of the robot, as well as the curvature of the path. For paths with sharp turns or frequent changes in curvature, such as the inner portions of a spiral or the end turns of a straight rows pattern, shorter lookahead distances are needed to ensure that the robot can regain the path quickly enough. For straighter paths, a longer lookahead distance can be used to prevent oscillatory behavior.

The coverage patterns implemented in the navigation planner are separated into discrete path segments, each with its own lookahead distance. For the straight rows pattern, each row and each half-circle turn at the ends are individual path segments. For the spiral pattern, each half circle is a path segment. Discrete changes in curvature occur when switching from one path segment to another, but the robot cannot instantaneously switch to a new curvature. To ensure a smoother transition, the pure pursuit algorithm is modified slightly. As the robot nears the end of a path segment, the intermediate waypoint chosen needs to be a point on the next path segment. The point at which the waypoint switches from the current to the next path segment is not when the robot reaches the new segment, but at some earlier point. This earlier point is when the robot is a lookahead distance away from the endpoint.

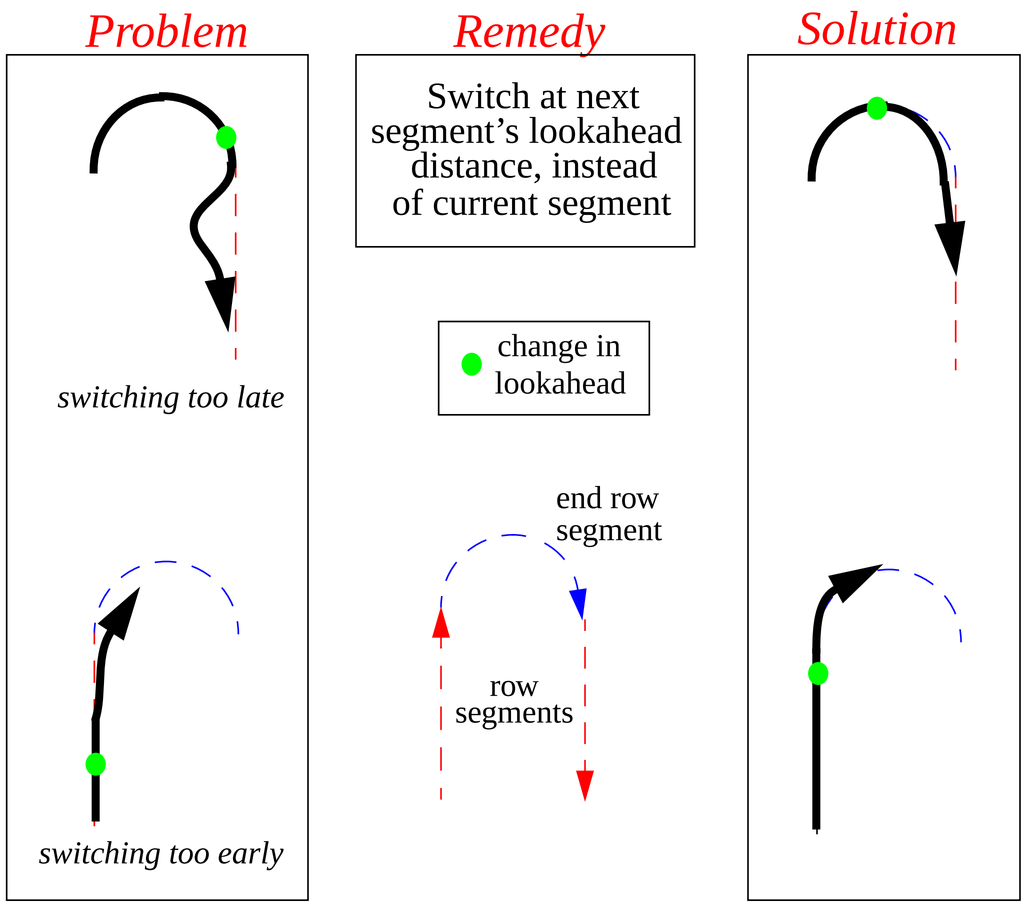


Figure 4: Switching Between Path Segments in the Straight Rows Coverage Pattern

However, different path segments have different lookaheads, so a choice must be made as to which lookahead to use. The distance from the robot to the current path segment's endpoint is compared to the lookahead distance of the next path segment, instead of the current path segment. When the distance to the endpoint is equal to or less than the next lookahead distance, the intermediate waypoint switches to being selected using the next path segment and its lookahead distance instead of that of the current path segment. The result is that when switching to a straighter path segment, with a longer lookahead, the robot starts straightening out sooner than it would otherwise. When switching to a more curved path segment, with a shorter lookahead, the robot moves into the curve later, preventing the robot from entering the curve too soon and becoming misaligned with the path. Prior to switching to the new path segment, the robot just selects a waypoint on the extension of the straighter path segment, as if it were continuing along that segment past the endpoint. Figure 4 demonstrates the deviations from the desired path, which is shown by the dashed line, when the intermediate waypoint is selected using the wrong lookahead distance.

Field Testing

A. Slag Heaps

A slag heap in Pittsburgh, used to test robots for many years (see Figure 5), provided a relatively flat gravel surface large enough to perform trial runs of the straight rows and spiral coverage patterns. Data from four different days of testing in August and September of 1998 are described here. Images of the groundtracks were produced using Matlab to display the position data from Nomad's differential GPS as the robot performed the patterns.

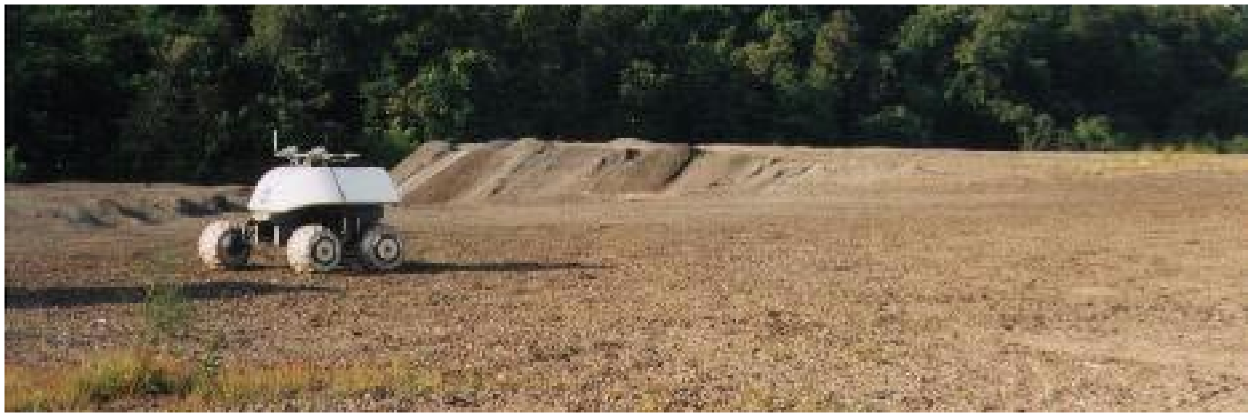


Figure 5: Nomad at the Slag Heaps

Before implementing the pure pursuit algorithm, an alternate method of path following was used, where the path segment endpoints were chosen as intermediate waypoints and the robot commanded to drive toward them. When the robot came within a close enough distance to these waypoints, driving along the next path segment was initiated. This method, effectively a pure pursuit method with a continually decreasing lookahead distance, resulted in two main problems due to physical limitations of the robot. The first problem was oscillation along the path caused by an improper lookahead distance, where the robot continually overshot the path when trying to regain the proper position. This oscillation was compounded by wheel slippage on the terrain and mechanical inaccuracies in the locomotion system. The second problem was large correcting turns near the endpoints, sometimes causing the robot to completely miss finding the endpoint. This second problem was due to the fact that the robot used angular deviation from the waypoint to calculate the amount of correction needed. As the robot drew near the endpoint, the lookahead distance grew increasingly short, and small linear deviations from the path caused large angular deviations, resulting in sharp turn correction commands. When the robot could not turn sharp enough to reach the waypoint, the robot continued circling around the waypoint, never getting close enough to trigger the change to the next path segment. Figure 6 shows actual groundtracks of the robot in several trials at the slag heaps, demonstrating these problems.

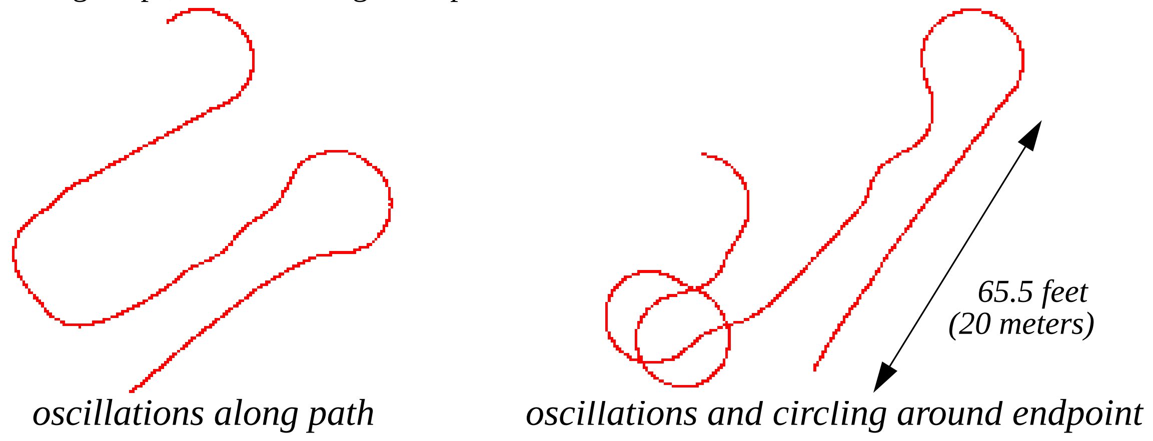


Figure 6: Problems Enacting Straight Rows Pattern

After implementing the modified pure pursuit method for path following, the straight rows pattern was followed more accurately. Using this method, the robot's position was compared with its desired position every second, resulting in fast corrections to any path deviations. The selection of the lookahead distance still needed to be coordinated with the robot's speed and maneuverability to prevent oscillations. For the trial shown in Figure 7, the lookahead distance for the straight path segment was 16.4 ft (5.0 m), and for the circular row end was 6.5 ft (2.0 m). The rows themselves were 26.2 ft (8.0 m) apart and 65.5 ft (20.0 m) long. The minimum turning radius of the robot was set to be 8.2 ft (2.5 m), and the robot traveled at 5.9 in/s (0.15 m/s). This speed allowed the robot enough time to make the appropriate changes in its turning angle, while faster speeds used earlier resulted in more oscillations. Another parameter used was the correction gain, which is concerned with how strong a correction is made relative to the amount of angular deviation. A too large gain results in overshooting the desired path, while a too small gain results in a too slow correction.

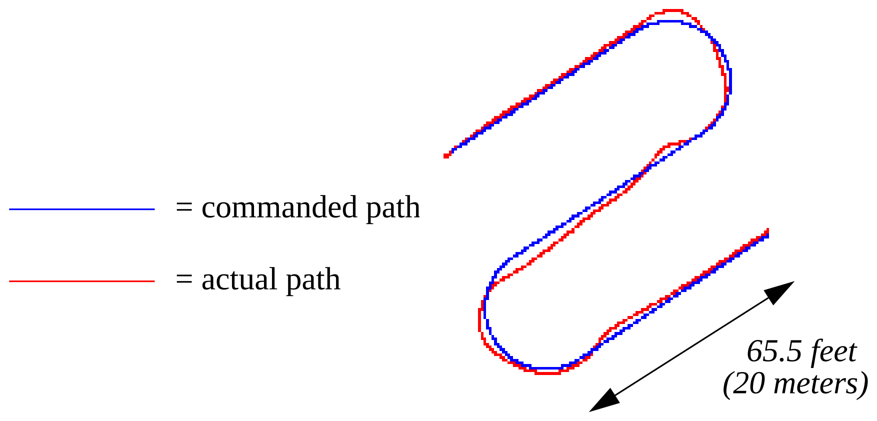


Figure 7: Straight Rows Pattern with Pure Pursuit

For the spiral coverage pattern, initially no path following or feedback method was implemented, but commands were sent assuming that the robot was on course. The turning radius was increased when the robot's orientation had rotated 180 degrees, which frequently resulted in too soon or too late radii changes. In addition, the commanded turning radii were not always accepted by the navigation system, which instead just picked the closest radius available from its set of allowed radii. Groundtracks of test trials with these problems are shown in Figure 8.

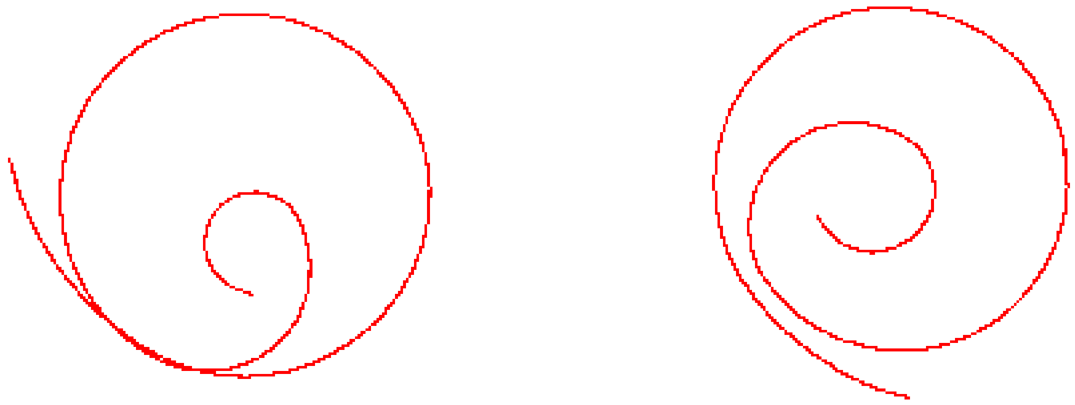
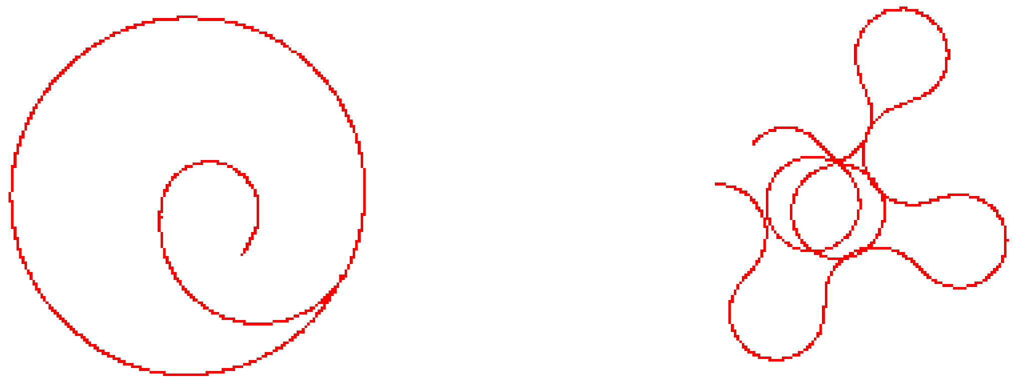


Figure 8: Imprecise Radii Selection for Spiral Pattern

The next improvement selected waypoints every half circle, but this resulted in the robot occasionally not coming close enough to the waypoint to recognize it had traveled half a circle, and therefore not increasing the turning radius at all (see Figure 9a). When the pure pursuit algorithm was implemented, the selection of lookahead distances also caused problems, as seen in Figure 9b, where the oscillations were so great that a spiral pattern could not be maintained.



(a) Missed Waypoint *(b) Oscillations*

Figure 9: More Problems with Spiral Pattern

The final tests at the slag heap show the proper selection of parameters. In the patterns shown in Figure 10, the lookahead distance was set to 3.3 ft (1.0 m) at the start of the pattern, and increased by 3.3 ft (1.0 m) every half circle. The speed was 5.9 in/s (0.15 m/s), just as in the straight rows pattern.

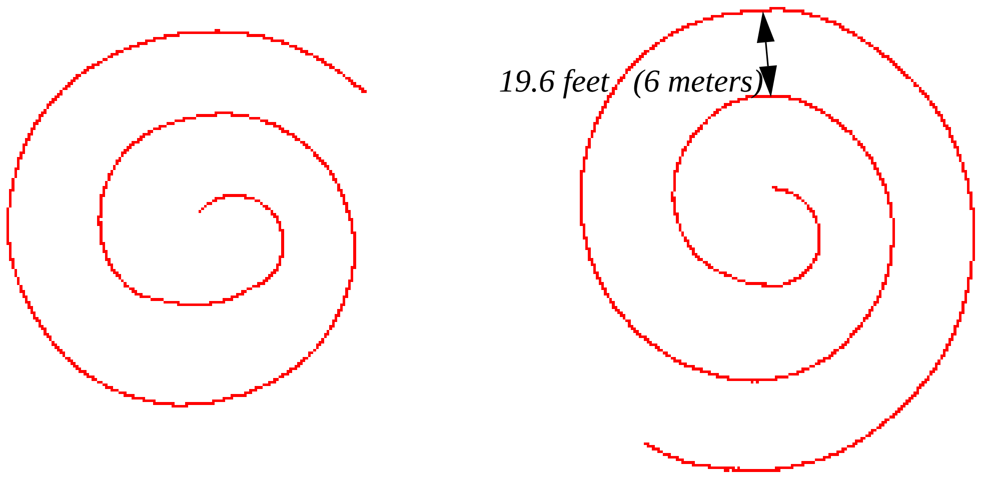


Figure 10: Good Spiral Patterns

B. Antarctica

The straight rows and spiral coverage patterns were tested on blue ice fields in the Patriot Hills area of Antarctica in November of 1998 (see Figure 11). Several differences in the robot configuration and the environment were present, leading to the possibility of required changes in the path following parameters. One difference was Nomad's wheels. For the Pittsburgh testing, Nomad's wheels had aluminum shells with nylon cleats. For Antarctica, these were replaced with modified conventional rubber snow tires with studs. In addition, at the time of the pattern testing, the robot's generator had broken, and a replacement generator was attached externally and pulled along on a sled behind the robot.

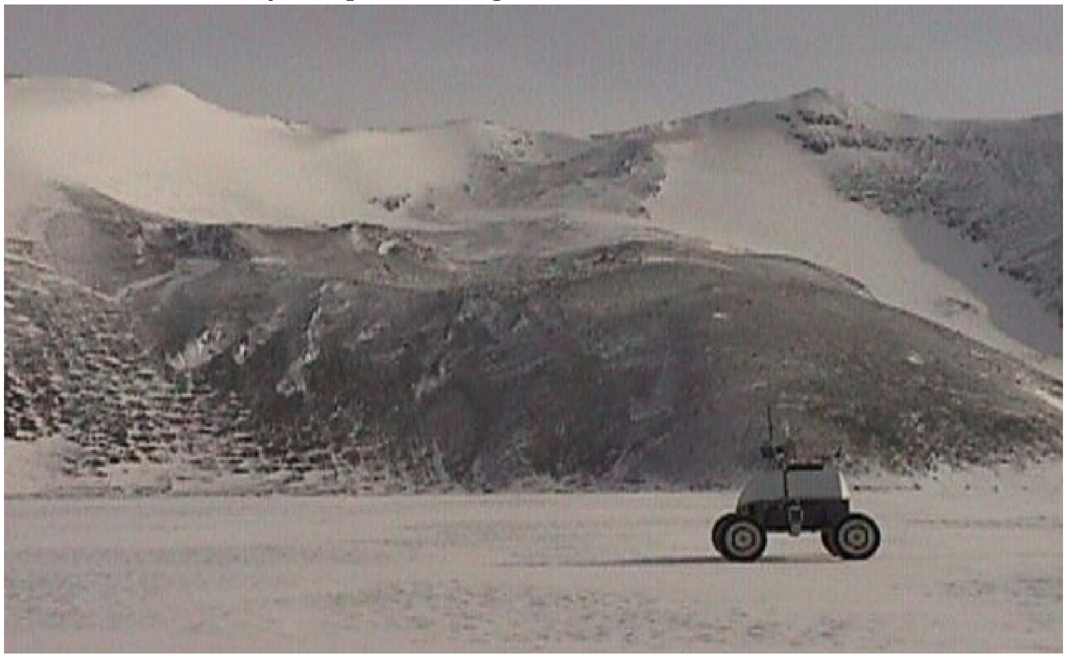
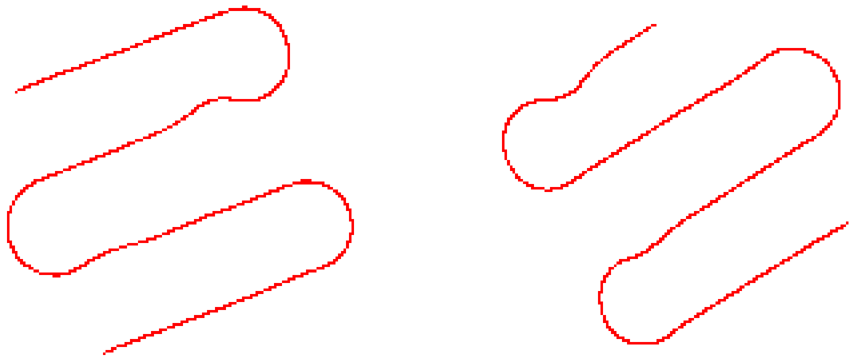
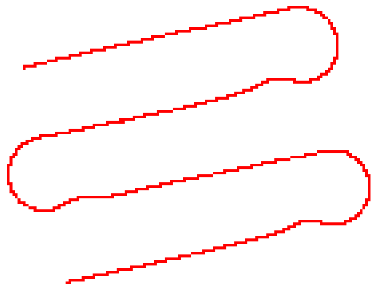


Figure 11: Nomad Traversing Patriot Hills Ice Field

The other differences were environmental. The extreme cold weather had the potential to affect the maneuverability and flexibility of Nomad’s steering system. The terrain differences, from loose gravel to hard blue ice, caused changes in traction. However, the path following parameters were kept the same for the first trials in Antarctica, and the resulting patterns were close enough to the desired patterns that no parameter changes were made.

Six of the seven trials performed well, with the seventh failing due to Nomad’s wheels being turned in an undesirable configuration at the start of the pattern. In all of the trials, Nomad’s minimum turning radius was set at 13.1 ft (4.0 m), and the robot traveled at 5.9 in/s (0.15 m/s).Two of the straight rows patterns had rows 26.2 ft (8.0 m) apart and 65.5 ft (20.0 m) long (Figure 12a). A third had rows 26.2 ft (8.0 m) apart and 98.3 ft (30.0 m) long (Figure 12b). The fourth straight rows pattern had rows 39.3 ft (12.0 m) apart and 98.3 ft (30.0 m) long, and included a deviation to a designated waypoint in the middle of the second row before continuing the pattern (Figure 12c). The spiral patterns performed are shown in Figure 13.


(a) 65.5 ft (20 m) rows, 26.2 ft (8 m) width


(b) 98.3 ft (30 m) rows, 26.2 ft (8 m) width

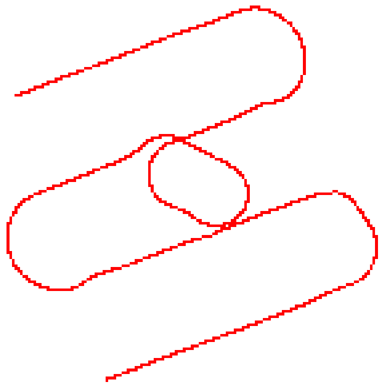

(c) 98.3 ft (30 m) rows, 39.3 ft (12 m) width pattern deviation

Figure 12: Straight Rows Patterns in Antarctica

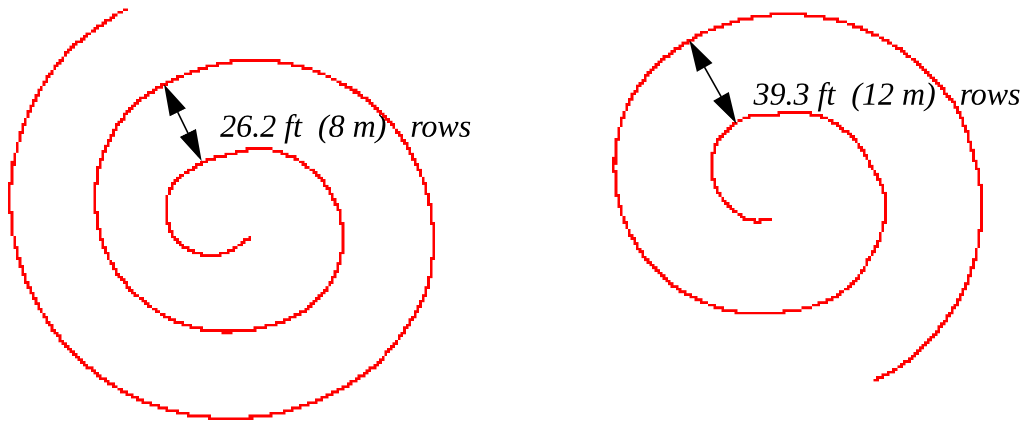


Figure 13: Spiral Patterns in Antarctica

Discussion

The fact that the path following parameters did not need to be altered in Antarctica from the values used in Pittsburgh suggests two hypotheses. One, any differences in traction, cold weather performance and robot configuration counteracted each other. However, the changes from these factors would have to be nearly exactly opposite to cancel each other out, and preliminary tests with the new rubber tires at the slag heaps provided no indication that these tires caused much difference in path following capabilities. The second hypothesis is then more likely, which is that the path following method is robust enough to accommodate different environmental conditions once the proper parameters have been found for the particular robot's turning rates and speed. [4]

Changes in the travel speed did affect the path following performance considerably. Using the same lookahead distances for the straight rows pattern, but two different speeds, 5.9 in/s (0.15 m/s) and 9.8 in/s (0.25 m/s), resulted in considerable oscillation and path deviation for the faster speed, but a near perfect pattern for the slower speed. Future work to be done includes developing a metric for computing the amount of path deviation over an entire pattern, as well as a method of relating travel speed to lookahead distances. Hardware and software both affect coverage pattern planning as they interact with each other. Software parameters must be tuned to the specific hardware in use, dictating the capabilities of the robot in carrying out desired plans.

Incorporating a map to be used with the coverage patterns will enable the patterns to be altered to accommodate large obstacles and irregularities in the area to be covered. The obstacle avoidance module would then only be used as a backup for detecting and avoiding objects not in the map. Other work is the implementation of additional patterns, such as the sun-following pattern, and the evaluation of power consumption required for the different patterns. Consideration of the shape of the area to be covered, the goals of the robot, and the power requirements will lead to an autonomous method for selecting the best pattern to perform. Incorporating this capability into the rest of the autonomy system will enable an efficient search for meteorites to be performed.

Acknowledgments

This project was supported by NASA through grant NAG5-7707. We would like to thank Dave Lavery, Solar System Exploration Program Executive and Telerobotics Technology Program Manager at NASA Headquarters. Also we would like to thank the National Science Foundation, the Chilean Air Force, the Chilean Antarctic Institute, and the 1998 RAMS expedition team members: William Cassidy of University of Pittsburgh, Pascal Lee of NASA Ames, Nicolas Vandapel of LAAS-CNRS, and Matthew Deans, Alex Foessel, Stewart Moorehead, Michael Parris, Liam Pedersen, and Mark Sibenac of Carnegie Mellon University.

Bibliography

1 Alami, R., R. Chatila, S. Fleury, M. Ghallab, F. Ingrand, "An Architecture for Autonomy," *International Journal of Robotics Research*, Vol. 17, No. 4, April 1998, pp. 315-337.

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

3 Chatila, Raja, Simeon Lacroix, Thierry Simeon and Matthieu Herrb, "Planetary Exploration by a Mobile Robot: Mission Teleprogramming and Autonomous Navigation," *Autonomous Robots*, Vol. 2, 1995, pp. 333-344.

4 Ollero, Anibal and Guillermo Heredia, "Stability Analysis of Mobile Robot Path Tracking," *Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 3, August 5-9, 1995, pp. 461-466.

5 Pearl, J., *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesley, Reading, MA, 1984.

6 Shillcutt, Kimberly and William Whittaker, "Modular Optimization for Robotic Explorers," *Integrated Planning for Autonomous Agent Architectures, AAAI Fall Symposium*, Orlando, FL, October 23-25, 1998.