Autonomous Geocaching: Navigation and Goal Finding in Outdoor Domains

James Neufeld
University of Alberta
114 St - 89 Ave
Edmonton, Alberta
neufeld@cs.ualberta.ca

Jason Roberts
University of Alberta
114 St - 89 Ave
Edmonton, Alberta
roberts@cs.ualberta.ca

Stephen Walsh
University of Alberta
114 St - 89 Ave
Edmonton, Alberta
walsh@cs.ualberta.ca

Michael Sokolsky
University of Alberta
114 St - 89 Ave
Edmonton, Alberta
sokolsky@cs.ualberta.ca

Adam Milstein
University of Waterloo
200 University Avenue West
Waterloo, Ontario
ahpmilst@cs.uwaterloo.ca

Michael Bowling
University of Alberta
114 St - 89 Ave
Edmonton, Alberta
bowling@cs.ualberta.ca

ABSTRACT

This paper describes an autonomous robot system designed to solve the challenging task of geocaching. Geocaching involves locating a goal object in an outdoor environment given only its rough GPS position. No additional information about the environment such as road maps, waypoints, or obstacle descriptions is provided, nor is there often a simple straight line path to the object. This is in contrast to much of the research in robot navigation which often focuses on common structural features, e.g., road following, curb avoidance, or indoor navigation. In addition, uncertainty in GPS positions requires a final local search of the target area after completing the challenging navigation problem. We describe a relatively simple robotic system for completing this task. This system addresses three main issues: building a map from raw sensor readings, navigating to the target region, and searching for the target object. We demonstrate the effectiveness of this system in a variety of complex outdoor environments and compare our system's performance to that of a human expert teleoperating the robot.

Categories and Subject Descriptors

J.7 [**Computers in Other Systems**]: Command and control

General Terms

Design, Experimentation

Keywords

Robotics, Outdoor navigation, Geocaching

1. INTRODUCTION

Geocaching is a human recreational activity in which participants use a handheld GPS to aid in finding an object hidden in an unknown environment given only the object's GPS coordinates. Typically the objects are placed in undeveloped areas with no obvious roads or paths. Natural obstacles along with the lack of any map of the terrain make it a challenging navigation problem even for humans. In addition, due to the error in GPS position estimates, a search is usually required to finally locate the object even after the GPS coordinate has been reached. This paper describes an autonomous robot system designed to solve the geocaching task. Some of the features of human geocaching will be relaxed: the target object is known in advance and can be identified when found, and the robot will start within a few kilometers of the goal. However, the core challenge — navigating in an unknown environment without a map to find a target object — remains unchanged.

Robot navigation in outdoor, and hence unstructured, environments is an exciting challenge for robotics. It has innumerable practical applications such as search and rescue, surveying, monitoring outdoor industrial or agricultural assets, and interstellar exploration. It also has proven to be considerably more difficult than the far more studied task of indoor navigation, where sensor range is not an issue and two-dimensional occupancy maps have been shown to be adequate. Even in outdoor settings most recent successes have focused on tasks such as road following where waypoints are provided and a road or path is known to exist between such waypoints. The more general problem of navigating without any map, waypoints, or known paths has, in comparison, received little attention. Robot geocaching is an ideal task for exploring this general navigation challenge, while also coupling it with another challenging problem of searching a local area for a target object.

Robot geocaching requires addressing three key issues: map building, navigation, and local search. Map building requires both identifying a good representation of the navigability of outdoor space, and extracting such a map from the robot's array of sensors as it moves about the environment. Navigation is required to identify an appropriate path to reach the target area, despite an inaccurate and incomplete map, as well as produce control commands to follow the path. Because GPS receivers have limited accuracy in estimating position, a local search is necessary when the

Cite as: Autonomous Geocaching: Navigation and Goal Finding in Outdoor Domains, J. Neufeld, J. Roberts, S. Walsh, A. Milstein, M. Sokolsky, M. Bowling, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Padgham, Parkes, Müller and Parsons (eds.), May, 12-16., 2008, Estoril, Portugal, pp. XXX-XXX.

Figure 1: Segway RMP outfitted for geocaching.

robot arrives within the range of the GPS goal. This paper describes a simple robot system that addresses these issues.

The paper is organized as follows. In Section 2 we describe the physical robot platform and the sensors that were used. Explanations of how the key challenges were addressed are presented in turn: map building in Section 3, navigation in Section 4, and the local search in Section 5. Results for the full system follow in Section 6 which includes evaluations for our system in two separate environments, a lightly wooded park area and the semi-urban setting of a university campus. We present both aggregate statistics on the robot's performance, sample paths, and a comparison with human operators shown the same sensory inputs as used by the robot. In Section 7 we relate this work to other examples of robot navigation in unstructured environments, and then finally conclude.

2. HARDWARE

The robotic system described in this paper is built on top of a Segway Robotics Mobility Platform (RMP): a 2-wheeled self-balancing robot. A picture of the robot can be seen in Figure 1.

2.1 Robot Platform

The Segway RMP is ideal for navigating in outdoor environments due to its high maneuverability, ruggedness, narrow chassis, large payload capacity, and water resistance. Our RMP has been out-fitted for autonomous navigation by adding all terrain tires, roll bars, two laptop computers, a combination GPS and inertial sensing unit, a color camera, and a laser rangefinder. The Segway RMP has a top speed of 3.6 m/s (13 km/h) but we limited the speed to 0.4 m/s which we determined as the maximum speed at which the system could detect and avoid obstacles given the sensor configuration described below. The RMP's internal batteries provide enough energy to last for approximately four hours of continuous navigation at our maximum speed. The two laptop computers mounted on the system each have extended batteries to provide approximately four hours of battery life. The first laptop is used only to communicate with the RMP and the sensors and contains a 800MHz Intel processor and 512MB RAM. The second laptop, containing a 2.0GHz dual core Intel processor and 1GB RAM, is used to interpret the sensor information and process the commands to direct the robot toward the goal.

2.2 Sensors

This robotic system is equipped with a SICK LMS 200 laser rangefinder used for measuring the terrain directly in front of the robot. This sensor is configured to return 401 distance measurements over a 100 degree sweep at 18.5Hz. The laser is rigidly mounted on the top of the robot and pitched forward 27 degrees. If the robot is on flat ground and perfectly upright, the laser returns horizontal scans of the terrain approximately 2 meters in front of the robot and 5 meters wide. Additionally, for goal identification, a color camera is mounted just above the laser sensor and pitched downward 30 degrees. This camera is configured to return images at a 320x240 resolution at 3.75 Hz.

In addition to these external sensors, the robot is equipped with a variety of proprioceptive sensors. The RMP has highly accurate wheel odometry encoders as well as gyroscopes that measure the pitch and roll of the platform. In addition we've added a Novatel ProPac G2 – Plus, which returns global position and heading information. This sensor combines a GPS receiver with an inertial measurement unit and is able to return orientation information accurate to 1/100th of a degree and positions accurate to 1 meter at 100Hz. Note that the goal object's location, though, is measured using a hand-held GPS and so is only accurate to 8–12 meters in the best conditions.

3. MAP BUILDING

In order to safely navigate to the goal location the robot requires a map that specifies the locations of unsafe terrain. Our map is constructed exclusively from measurements provided by the downward pointing laser. The map will necessarily be incomplete as the laser is only scanning 2 meters ahead, requiring us to integrate our sensor readings, and update our map, as we move in the environment. In order to integrate readings that were collected at different times we must know the robot's pose at the time each reading was recorded. In the robotics literature this is referred to as the localization problem. Given a set of laser distance measurements, from known poses, the system must integrate this information and identify sections of the terrain that pose a danger to the robot. This is referred to as the mapping problem with the additional element of traversability classification. We address each of these problems in turn.

3.1 Localization

To obtain an estimate of the robot's pose in the world reference frame we integrate information from our various sensors. Although we have an accurate GPS/IMU positioning system, it does not entirely solve our problem. This is because the measurements returned by the GPS/IMU sensor have a high relative error (error in the change in position over small periods of time). Relative error is critical because the

robot must be able detect very small obstacles by comparing laser readings taken in close succession. Odometry information, on the other hand, can provide very accurate relative positions, but its absolute error (error in the global position at any given time) accumulates as the robot moves. For a long distance navigation task this is a significant drawback as a robot localizing solely on odometry can end up looking for the goal object kilometers away from its actual location. In order to address the shortfalls of both of these sensors we combine the information received from each to maintain a robust position estimate.

The most significant source of pose estimation error when integrating odometry information comes incorrect heading (yaw) information. This error comes from two sources, first, wheels tend to slip much more when the robot is turning, and second, incorrect heading information has a compounding effect on all subsequent measurements. Fortunately, the GPS/IMU sensor returns yaw measurement with an absolute error less than 1/100th of a degree. We combine this yaw measurement with information from the odometry encoders, which provide an accurate measurement of the distance traveled, to update the current (x,y) position of the robot. Under normal operation we found that this position estimate was accurate to 1% of the distance traveled. In order to adjust for this accumulating position inaccuracy the position of the goal is updated periodically from its known GPS location and the current GPS position of the robot, which does not drift over time.

The other two rotational degrees of freedom, forward rotation (pitch) and the side rotation (roll), are provided directly by the RMP's internal gyroscopes. These values are crucial to the accuracy of the world map because the errors are magnified when laser information is converted into the world frame of reference. In fact, a 1°error in the pitch estimate can lead to a 4 cm error on the height estimate of the ground plane. And a 1°error in roll can lead to a 6 cm error in the ground plane estimate. Conveniently, the internal gyroscopes are essential if maintaining the balance of the RMP and are accurate to about 0.12°.

The only sensor measurement we have for the height of the robot (z) comes from the GPS/IMU sensor. However, this measurement has far too much relative error to be used in map building. Hence, we assume z to be zero at all times. We further discuss the problems that arise from this assumption in the evaluation section as well as discuss possible solutions in the conclusion.

3.2 Traversability

Once we have an accurate estimate for the position and orientation of our robot in the world, we can use it to integrate and interpret our sensor readings and identify unsafe areas of terrain. The laser rangefinder sensor sends out a series of laser pulses in a sweeping motion and returns the distance at which the beam is reflected back to the sensor. The first step taken in processing this information is to remove measurements caused by small airborne objects such as pollen, snowflakes, and dust. This is done by filtering out laser readings that differ by more than 30 cm from both of its neighbours in the same direction. The next step is to convert these laser readings into the world reference frame using the current pose estimate of the robot. The resulting 3D point cloud is combined with earlier scans to produce a somewhat sparse contour of the terrain. However, due to the computational limitations on board the robot, a 3D point cloud is not an ideal representation of the terrain. Instead, we summarize this information into a compact probabilistic representation. This representation is a discrete two-dimensional lattice of square cells, where each cell maintains a mean and variance over the heights of the laser points that fall within its boundaries. This representation reduces the overall computation and memory resources required to both remember past measurements and classify terrain.

Through a series of crash tests teleoperating the RMP, we determined that the robot is not physically able to drive over terrain if the load bearing surface changes in height by generally more than 8 cm over a 25 cm distance[1]. Therefore, to determine the traversability of a section of terrain we must obtain an estimate for the height of the load bearing surface. This turns out to be a very difficult problem because many terrain types, such as vegetation or snow, can occlude the true surface. While there has been some research on estimating the correct ground plane in vegetated terrain [10] these techniques are not robust enough to guarantee correct classifications. Unfortunately, this is a limitation of our sensor modality and we are only able to partially address this issue. This is done by estimating the height of the load bearing surface as equal to the mean for a cell in the probabilistic representation. Using this estimate terrain with a reasonable amount of variance, such as short grass, will generally appear as flat, and thus be classified as traversable.

The cell width parameter allows us to control how much smoothing we introduce into the terrain map. As the width increases the heights are averaged over a larger area and consequently the terrain will appear to be much smoother. As the width of the cells is reduced there is less of a smoothing effect which causes terrain variance and measurement errors become much more pronounced. The extremes in either case can lead to poor performance. If the cell width is too wide the classifier may not detect obstacles and if the width is too narrow the classifier may label safe terrain as untraversable. We experimented with this parameter in several different environments and found that a cell width of 12.5 cm struck a good balance between these two trade-offs. Additionally, we anticipated that this height estimate would incorrectly halve the height of solid objects such as curbs or walls. For this reason we automatically classified a cell as unsafe if its variance measure exceeded a set threshold of 10 cm.

Given this estimate for the height of the load bearing surface we classify each terrain cell by comparing its height (mean) to all of its neighbours that lie within the surrounding square 5x5 cell area. If the height difference between the cell and any of its neighbours exceeds a threshold, set at 8.3 cm, then both cells are marked unsafe. Figure 2 illustrates how this classification scheme works in a simple 2D example.

Improvements.

In order to reduce misclassifications due to measurement errors and terrain variance we do not include cells that have less than 5 laser measurements in the terrain classification step. Also, to avoid misclassifications that arise from this accumulated localization error we introduced two adaptations. The first adaptation is to immediately discard a cell's

[1]This is a guideline figure since there are a number of additional factors that affect traversability such as the current speed, payload, and heading of the robot.

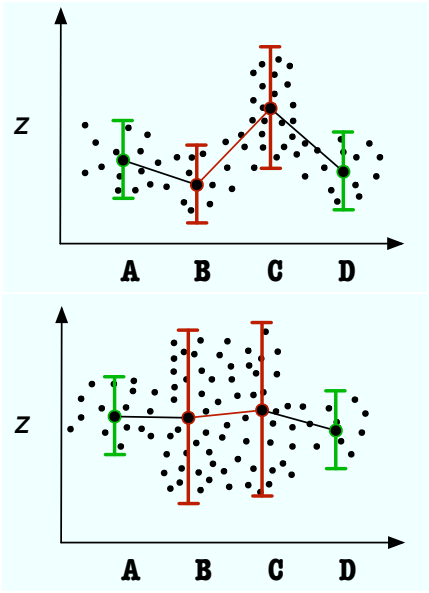

Figure 2: This figure shows two example point clouds, shown as the smaller dots, as a cross section. The large dots indicate the mean value of the point cloud and the bars represent the (scaled) variance. In the top figure cells B&C are not traversable because the difference between their means is above threshold. In the bottom figure cells B&C are not traversable because their variance exceeds threshold.

previous laser information older than 1.5 seconds upon receiving new laser data. The second adaptation is to ignore the height difference between cells if they have been updated more than 5 seconds apart. Also, in order to ensure that the robot will not incorrectly surround itself with obstacles we mark all of the grid cells that it traverses over as permanently safe. This is an important modification because often in rough terrain the robot will be forced to backtrack along the path it had previously traversed. In this instance a single misclassification can cause the robot to believe it is surrounded and that there is no safe path to goal.

4. NAVIGATION

The navigation system on this robot consists primarily of two independent systems: planning and control. The planning system produces a safe and efficient path to the goal while the control system follows that path as efficiently as possible.

4.1 Planning

The objective of the planning system is to produce a path that is optimal with respect to the robot's current map of the environment. This is a challenging problem because the system must produce long distance paths at a resolution high enough to allow for precise local navigation around and between obstacles. Additionally, the system must be able to efficiently alter the current path in order to adapt to the constant stream of new robot positions and map updates. This problem is further complicated by the limited computational resources available on board the robot. However, the robot moves only a small distance at a time and map updates only occur close to the robot. Fortunately, algorithms exist that have been specifically designed to exploit these two properties, one such an algorithm is D*-lite [4]. The D*-lite algorithm maintains a distance to goal for each vertex in the search graph and only makes local modifications as edge costs change. Because this system only updates cells relatively close to the robot there is often very little computation required to update the changed costs.

The search graph used for this planner is an 8-way connected graph that is the same resolution as the traversability map. Any cell that is marked as untraversable is simply removed from the search graph. Also, in order to account for the RMP being larger than a single 12.5 cm square cell we expand around obstacles by marking all the cells within a 3 cell radius of an untraversable cell as unsafe. This buffer is actually a full cell size larger than required so that the path is still safe even after it is smoothed as described in the next section. The D*-lite algorithm produces a path by essentially planning backwards from the goal in the same manner A* search would. Once the start location has been reached the closed list is saved and used as a cost map for producing a path. When the robot moves the algorithm will reuse the saved cost map or extend it out to the new start location if needed. Also, when new map updates occur they are added to the open list and their costs, as well as the costs of the cells they affect, are updated in a cascading fashion. Once the updates are finished a path is produced from the cost map with a simple depth first search, starting from the start state.

Improvements.

In order to produce a more direct path to the goal the depth first search breaks ties by choosing to expand the node that is closest to the straight line path between the start and goal. This alleviates some of the inefficiencies created by discretizing the terrain into an 8-way grid. The search was also adapted to deal with the goal location periodically changing due to accumulating localization error. To move the goal in the path planner we simply erased the entire map, moved the goal position, then re-added all of the obstacles, and planned a new path to the goal. This simple solution turned out to be surprisingly efficient because the large majority of the obstacles are not between the robot and the goal. Because D*-lite updates the cost map by planning backwards from the goal the changes introduced by these obstacles do not require computational resources.

4.2 Control

Once a safe path to the goal is returned by the planning system the control system must issue low level velocity commands to follow the path as efficiently as possible. Because the RMP is able to turn in place, we found that a simple point-based controller was an effective way to produce velocity commands. This controller receives its current location , orientation, and target location for input and produces two continuous velocity commands: forward velocity and angular velocity as output. This controller works by first supplying an angular velocity until it is almost facing the target location. As the robot begins to point the correct direction the controller will start supplying a forward velocity proportional to the distance it must travel. The embedded software on the RMP then attempts to meet these given velocities un-

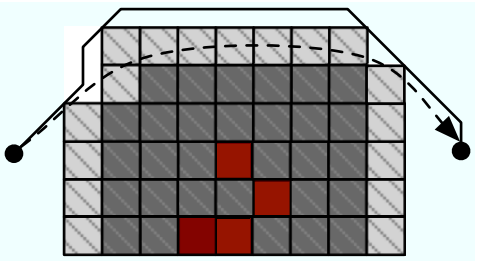

Figure 3: This figure shows the differences between the path produced by the 8-way D*-lite planner (solid line) and the actual path taken by the robot (dashed line). The solid dark cells have been marked unsafe by the classifier. The striped dark cells have more than one unsafe cell within a 3 cell radius. The line striped cells have only one unsafe cell within this radius.

der the additional constraint of keeping the robot balanced. Using this controller the robot can follow a path by continually navigating to the next point along the path. However, because the path is produced in a discrete 8-way grid it will have several sharp turns and following it so strictly will lead to slow choppy motion.

To create a smoother and more efficient path the control system chooses a navigation waypoint as far along the path as possible. The only restriction on this choice is that the robot must not drive over any unsafe terrain. To determine where to set the waypoint the control system starts at the point 1.5 m along the path and traces a ray back to the robot's current position. If the ray does not intersect any untraversable cells or any cells that have **more than one** unsafe cell in a 3 cell radius then the waypoint is safe. If the waypoint is not safe then the algorithm steps along the path toward the robot until it finds a point that is safe. Once a good waypoint is found it is sent to the point based controller and the robot will drive towards it. We deliberately made expanded region around obstacles larger than required so that this controller could produce a smoother path by cutting corners. Figure 3 demonstrates how this controller produces smoother paths by cutting corners.

4.3 Replanning

Since the robot is continually moving and receiving new information about its environment, its current path rapidly becomes outdated. Therefore, the robot must constantly make the decision of when to plan a new path. This is difficult because planning too frequently is not only computationally expensive but it can cause the robot to exhibit oscillating behavior. Replanning too infrequently is also problematic because the robot can collide with new obstacles in between planning stages. To make this decision the system keeps the current path until any one of the following conditions force a replan:

- The robot is closer to the waypoint than to the path's starting point (i.e., it is approximately half-way toward the waypoint.)
- More than a set amount of time has elapsed since the last replan.
- The line between the robot and waypoint becomes unsafe by the previous definition.

Under these conditions the robot will tend to stick with its current path and does not act indecisively. Also, this system ensures that the robot can navigate safely without having to waste system resources on unnecessary planning.

5. LOCAL GOAL SEARCH

The local goal search system takes over the planning as soon as the robot gets within a set range of the GPS goal. The objective of this final subcomponent is to search the local area around the GPS coordinate and find the goal object as quickly as possible. This involves deciding which part of the search area to scan next, safely navigating to the chosen area, marking off which sections have been searched, and identifying the goal object with the camera.

In order to find the goal object quickly it is important to search the areas where the goal object is most likely located. However, it is time efficient to search out areas that are close to the robot. In order to explicitly manage this trade-off, the system chooses which areas to search next according to a parameterized scoring function:

$$\text{score}(x, y) = \Pr(x, y) * \frac{C}{C + \text{dist}(x, y)^{\alpha}}, \qquad (1)$$

$\Pr(x, y)$ is the probability that cell (x, y) contains the goal object and is calculated using a Gaussian distribution centered at the GPS position. $\text{dist}(x, y)$ is the length of the shortest obstacle free path to the cell. A modified version of A* search is used to calculate this distance for every cell in the search space. This search reduces the amount of computation required by keeping the open and closed list from previous searches until the start location moves. The constant parameters C and α are used to trade off how much the system prefers closer points over farther ones. Also, because the camera is not able to see the ground close to the robot, cells that are within a 1.5 m radius of the robot are not considered as candidates.

Once a cell has been selected the system will attempt to scan the cell by navigating onto it. A path to the cell around obstacles is planned using A* and the same waypoint controller is used. The robot will continue to navigate toward the selected cell until any of the following conditions are met:

- The cell is seen by the camera.
- The cell is marked as an obstacle or unreachable.
- The robot has moved to within 1.2 meters of the cell.
- The goal has been found.

It is difficult to determine which parts of the terrain have been seen by the camera because the geometry of the terrain is unknown. Therefore, in order for a cell to be marked as seen it must first be scanned by the laser sensor. Once a cell has been scanned we can calculate whether or not it is in the camera's field of view. To make this calculation easier the camera is mounted on the robot so that the field of view lines up with the laser and we can mark cells as seen immediately after being scanned.

The camera based goal identifier uses a modified version of the CMVision [1] color threshold package to determine if

anything in the current field of view matches the distinct color and shape of the goal object. This identifier receives camera information at 3.75Hz and stops the local search program as soon as the goal is found.

6. EXPERIMENTAL ANALYSIS

This system was tested in two different environments each with their own unique obstacles. The first test site was a lightly wooded park near the University of Alberta campus. The park spans a distance of about 800 meters and consists of two large forested areas separated by a small access road. The obstacles in this site consist of picnic tables, large trees, small shrubs, fallen trees, barbecues, larger berms, and curbs. The second test site is a semi-urban area. Common obstacles in this site include trees, bicycle racks, garbage cans, benches, curbs, buildings, flower gardens, chain link fences, and hills.

6.1 Results Summary

The autonomous system was run on 11 test trials over the course of 5 days, 10 of these trials were run in the park test site and one 1.1 km test was run at the campus site. For each test trial the robot was placed at a random location in the environment and provided with only a GPS coordinate of the goal and a description of the goal object. The goal object in all tests was a 20 cm wide red disc placed within 8 meters of the specified GPS coordinate. The path, time taken, and number of interventions was recorded as the robot navigated. An intervention is defined as any time the tester took control of the robot to either move it out of a situation it could not safely escape from or to prevent it from hitting a dangerous obstacle which it did not detect.

The results for all 11 of these tests are recorded in Table 1. In all of the tests the robot was able to navigate to the target region successfully and in all but one test was able to locate the goal object. In the one failed test, the system mistakenly classified a patch of red leaves as the goal object, which highlights the need for more sophisticated object detection. The human operators had to intervene a total of three times over all of the test runs. In one case the RMP got caught on a tree while trying to recover balance in rough terrain. In the other two cases the RMP was unable to find a path to goal due to incorrect terrain classifications in hilly terrain. The system has a difficult time accurately modeling hilly terrain because of its inability to estimate the current height of the robot.

Over the course of testing, the autonomous robot was able to successfully navigate a cumulative distance of 4.7 km over unknown terrain while maintaining an average speed of 0.303 m/s. The system was able to maintain this high rate of speed due to its ability to plan paths in fractions of a second and identify obstacles early enough to navigate around them smoothly. The longest trial test we conducted was across the campus test site where the robot navigated continuously for a period of 66 minutes travelling a distance of 1127 meters. The path for this test, shown in Figure 4, shows the frequent backtracking the robot had to do in order to find its way around the large obstacles in the test site.

6.2 Comparison to Human Teleoperation

In addition to the autonomous test trials we also compared the performance of this system to that of a human expert teleoperating the robot. We ran a total of five tests with two different human controllers who had no prior knowledge of the environment. The human experts were shielded from the test environment and only provided with a 3D graphical interface of the robot's sensors. This interface provided the estimated position of the robot, the position of the goal, the point cloud from the laser scanner, as well as the obstacles detected by the terrain classifier. We elected to provide the human testers with the terrain classifications because a point cloud representation is at times difficult to interpret. In order to prevent the testers from navigating using the camera information the automatic goal detection subcomponent was used and the testers were simply told when the robot had found the goal. Additionally, the testers were encouraged to use the point cloud information whenever possible since the terrain classifier is not always correct.

Trials	11
Successes	10
Goal Finding Fails	1
Interventions	3
Cumulative Distance (m)	4700
Cumulative Time (min)	259
Average Speed (m/s)	0.302

Table 1: Results for the autonomous system on the trial tests

	Robot	Humans
Trials	5	5
Successes	5	5
Goal Finding Fails	0	0
Interventions	0	3
Cumulative Distance (m)	1517	1541
Cumulative Time (min)	76.9	83.2
Average Speed (m/s)	0.329	0.309

Table 2: Results for the human expert trials.

The results for both the humans and robot on the five test trials are shown in Table 2. The differences in the cumulative distance traveled for both the test groups was with an expected margin of error. However, the robot did navigate at a faster average speed than the human controllers. The human experts attributed this discrepancy mainly to time taken to try and infer the shape of the terrain from the point cloud data. The other major discrepancy between the two tests groups is the number of interventions. In all three of these cases the human controller incorrectly thought the terrain looked safe and overruled the slope classifier only to hit an obstacle. However, overruling the terrain classifier did reduce the traversal time for some of the tests. The paths shown in Figure 5 show how the human controller took a shorter path than the autonomous system. However, in this particular case the shortcut happened to take the human into much rougher terrain resulting in a lower average speed. Additionally, the human controllers did a much better job recognizing vegetation from the point cloud and thus not wasting time navigating around it.

7. RELATED WORK

Much of the work related to our system is in the area of outdoor navigation. See [8] for a succinct overview. Possibly the most well known system for outdoor navigation is the

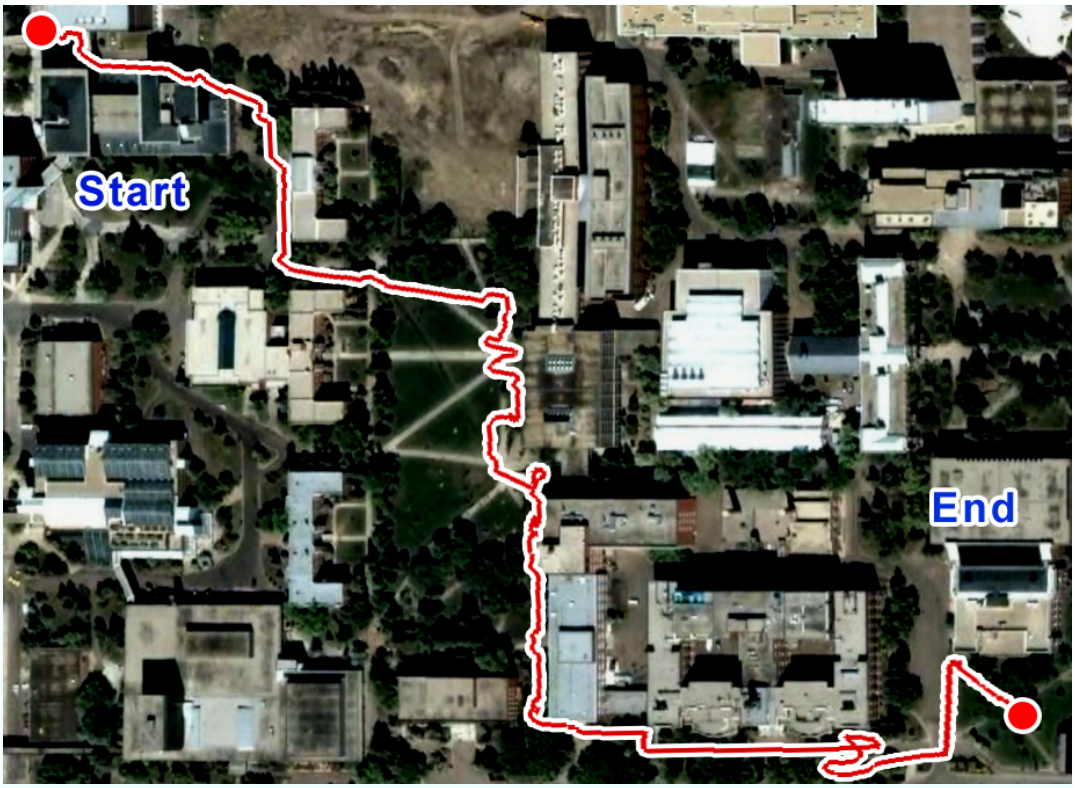

Figure 4: An example path followed by the robot in the university test site. The total distanced travelled was 1.1 km, and the run was completed in 66 minutes. Map image courtesy of Google Maps™.

Figure 5: Comparison of two paths in the park test site. The dark path was followed by a human expert teleoperating the robot. The lighter path was followed by the robot under autonomous control. Map image courtesy of Google Maps™.

CMU NAVLAB II autonomous vehicle [9]. This system has had demonstrated success navigating in outdoor terrain but did not incorporate a solution for searching for and recognizing a goal, although the authors do recognize the necessity of such a system. Several researchers have combined laser and image information together to build a complete system for navigating in outdoor terrain [6, 7, 3]. Each of these systems have had interesting contributions to the field of outdoor navigation but still have not touched on the subject of goal finding. There is also a significant body of research on object detection and classification with a laser sensor for outdoor environments. There are a number of highly applicable techniques for dealing with such data, including point based methods [6], vegetation detection and ground plane detection [10], structured obstacle detection [5], and machine learned classifiers [7].

The robotic vehicles participating in the 2005 DARPA Grand Challenge [2] all faced a number of similar challenges. This competition featured a number of robotic systems racing along a rough 132 miles desert road that featured a variety of dangerous obstacles. These vehicles had to solve some of the same problems as we did such as mapping and classifying terrain with a laser range finder and efficiently planning around obstacles. However, because the Grand Challenge race took place along a road, and GPS waypoints were provided, the main challenge was maintaining a high rate of speed. This is in contrast to the geocaching problem where the main challenge is searching for an efficient path to a long distance goal.

8. CONCLUSION

This paper described an autonomous system for robot geocaching. Geocaching is a challenging problem that requires navigating through an unknown environment without a map or obvious paths or roads. In addition, it requires an additional local search procedure for finding the target object after navigating to its rough position. We describe a simple system for solving the key three components to this task: map building, navigation, and local goal search. We then extensively evaluated the system in two different test environments facing a variety of challenging obstacles. The system exhibited a high rate of success, travelling a total of over four kilometers while requiring only three operator interventions. We also compared the performance to human expert operators making use of the same sensors in a teleoperated fashion. The autonomous system completed the same five test runs faster and with fewer interventions.

There are a number of future directions for improving the system. Two of the required interventions in our tests were due to hills in the environment. Hills create two problems for the current system. First, the fixed tilt of the laser means that it may not be able to scan the ground in front of it when facing downhill. Second, scanning the same terrain from different heights is problematic when we assume robot height remains constant. Active control over the laser pitch would help address the first issue. The robot's own grid height estimates could also be used to estimate the robot's height to address the second problem. Another useful addition would be pedestrian detection. Currently, when a pedestrian approached the robot in our tests, we paused it for safety reasons. Having an autonomous response to dynamically moving obstacles would allow the robot to be deployed in even heavily populated environments. Finally, top speed of the robot was limited because of the short range of our downward pointing laser. Using a longer range sensor, such as a camera, might allow the top speed to be increased in open areas. Vision-based outdoor navigation, though, is its own challenging research problem.

9. ACKNOWLEDGMENTS

We would like to thank the following people for their help: Marcus Trenton, Brad Joyce, Cam Upright, and the class of C608. We would also like to thank AICML, RLAI, NSERC, and iCore for funding this research.

10. REFERENCES

[1] J. Bruce, M. Veloso, and T. Balch. Fast and inexpensive color image segmentation for interactive robots. In *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2000.

[2] M. Buehler, K. Iagnemma, and S. Singh. *The 2005 DARPA Grand Challenge*. Springer, 2007.

[3] A. Kelly, O. Amidi, M. Happold, H. Herman, T. Pilarski, P. Rander, A. Stentz, N. Vallidis, and R. Warner. Toward reliable off road autonomous vehicles operating in challenging environments. *International Journal of Robotics Research*, 25(5–6):449 – 483, 2006.

[4] S. Koenig and M. Likhachev. Improved fast replanning for robot navigation in unknown terrain. Technical Report GIT-COGSCI-2002/3, Georgia Institute of Technology, 2002.

[5] S. Kolski, D. Ferguson, M. Bellino, and R. Siegwart. Autonomous driving in structured and unstructured environments. In *IEEE Intelligent Vehicles Symposium*, pages 558– 563, 2006.

[6] R. Manduchi, A. Castano, A. Talukder, and L. Matthies. Obstacle detection and terrain classification for autonomous off-road navigation. *Journal Autonomous Robots*, Volume 18(1):81–102, 2005.

[7] B. Sofman, E. Lin, J. Bagnell, N. Vandapel, and A. Stentz. Improving robot navigation through self-supervised online learning. In *Proceedings of Robotics: Science and Systems*, 2006.

[8] D. J. Spero. A review of outdoor robotics research. Technical Report MECSE-17-2004, Department of Electrical and Computer Systems Engineering, Monash University, Melbourne, Australia, 24 Nov. 2004.

[9] A. Stentz and M. Hebert. A complete navigation system for goal acquisition in unknown environments. In *Proceedings 1995 IEEE/RSJ International Conference On Intelligent Robotic Systems (IROS)*, pages 425–432, 1995.

[10] C. Wellington and A. Stentz. Learning predictions of the load-bearing surface for autonomous rough-terrain navigation in vegetation. In *Proceedings of the Int. Conf. on Field and Service Robotics (FSR 03)*, pages 49–54, July 2003.