A Robotic Mobile Sensor Network for Achieving Scientific Measurements in Challenging Environments

Stephen Williams, Antidio Viguria and Ayanna M. Howard
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332
swilliams8@gatech.edu, antidio@gatech.edu, ayanna.howard@ece.gatech.edu

***Abstract*— Recently, it has been discovered that the giant ice sheets covering Greenland and Antarctica have been shrinking at an accelerated rate. While it is believed that these regions hold important information related to global climate change, there is still insufficient data to be able to accurately predict the future behavior of those ice sheets. Satellites have been able to map the ice sheet elevations with increasing accuracy, but data about general weather conditions (i.e. wind speed, barometric pressure, etc.) must be measured at the surface. A mobile, reconfigurable sensor network would allow the collection of this data without the expense or danger of human presence. For this to be a viable solution though, a method must be developed to allow multiple robotic scientific explorers to navigate these arctic environments. Specific technological achievements that must be addressed for deployment of this surface-based mobile science network include estimating terrain characteristics of the arctic environment, incorporating these characteristics into a robot navigation scheme, and using this scheme to deploy multiple robotic scientific explorers to specific science sites of interest. In this paper, we discuss an infrastructure that addresses these issues in order to enable successful deployment of these robotic scientific explorers.**

I. INTRODUCTION

Although weather data from glacial regions is considered important and valuable, this data is difficult to obtain. Currently, human expeditions must be sent to collect this data, which is costly, time consuming, and dangerous. Yet, this approach yields data about a very limited area, and covers only a short duration in time. To help alleviate this issue, a set of fixed weather stations have been installed, known as the Greenland Climate Network. While these weather stations provide a continuous data feed, with only 18 such stations covering an area of over 650,000 sq. mi, the spatial resolution is still very course.

In contrast, a group of autonomous mobile weather stations could be deployed in these regions. This would allow scientists to gather arbitrarily dense weather data about the area of their choosing, all while staying safely within the arctic outpost. In order to achieve such goals, several technological achievements must first be addressed. Namely, a mobile platform capable of traversing arctic terrain must be developed, a means of assessing environmental hazards must be added to the navigation system, an intuitive interface must be design to allow scientists to command the rovers' position and formation, and the rovers must be able to automatically reassign tasks between themselves in the event of failure.

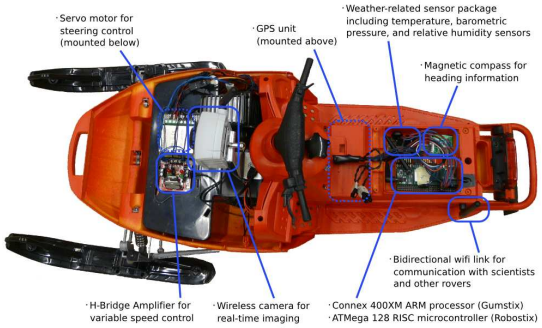
II. ROBOT PLATFORM

Despite being covered by snow, arctic regions present a large assortment of terrain challenges. Large quantities of fresh surface snow can be present during certain times of the year. This fresh snow is soft, creating a potential sinking hazard for wheeled vehicles. Over time the winds harden the snow surface making it more amenable to locomotion. However, these same winds also sculpt the snow into dune-like structures that can be as large as one meter, again impeding motion. Tracked vehicles have been developed to overcome the specific challenges of arctic travel. The most famous of these devices is the snowmobile, but other variations exist ranging in size from small single person vehicles to bus-sized multi-passenger coaches. These platforms have been successful in military, commercial, and science applications since their development in the 1940s.

For these reasons a snowmobile chassis was selected as the base for the "Arctic Crawler" prototype mobile sensor. A set of three prototype robotic rovers were constructed in our lab in anticipation of field testing. The rovers are based on an RC snowmobile chassis and have been retrofitted with a Connex 400XM processor from Gumstix. This motherboard contains a 400MHz ARM processor, wireless 802.11g ethernet, and bluetooth capabilities. Additionally, a Robostix board was added, which includes an Atmel ATMega 128 RISC microcontroller, providing both SPI and I2C serial ports, general purpose IO pins, PWM outputs, and an ADC unit. The original steering mechanism was replaced by a servo motor to provide proportional steering control, while an H-bridge amplifier provides modulated voltage to the DC drive motor for variable speed control. Both motors are controlled by the Gumstix processor using the PWM outputs.

For navigation, a GPS unit connects to the embedded processor via the bluetooth interface, while a magnetic compass provides heading information via the I2C serial bus. Sensor data and internal state information are exchanged between scientists and other rovers over the bidirectional wifi link. Additionally, a 0.3 Megapixel wireless camera on-board each Arctic Crawler provides real-time images.

To simulate the science objectives of the mobile sensor network, a weather-oriented sensor suite was added to each rover. Ultimately this science package will include



(a)

(b)

(c)

Fig. 1. (a) A diagram illustrating the internal electronic components of the Arctic Crawler rovers. Images (b) and (c) show the fully assembled rovers in a lab setting and at a test site near Cleveland, Ohio respectively.

an anemometer and a solar radiation sensor, among others. However, the size and expense of these types of sensors were not a good fit with the small footprint of the prototype platform. Instead, a set of solid-state sensors were selected that could measure meaningful weather related data and still fit within the confines of the rover's chassis. The final instument suite includes sensors to measure temperature, barometric pressure, and relative humidity. Figure 1 shows a diagram of the internal robot components, as well as images of the fully assembled rovers.

III. NAVIGATION SCHEME

In each robot, a path planning algorithm, an obstacle avoidance routine, a slope assessment algorithm, and a task execution unit have all been integrated into a single behavior-based architecture. Navigation is implemented using the DAMN architecture [11] to combine the competeing outputs of each behavior module. This architecture was designed to combine different behaviors for mobile robots in unknown and dynamic environments. Within the DAMN architecture, each behavior votes for a set of possible actuator values satisfying its objectives. Then, an arbiter combines those votes and generates actions which reflect the behavior objectives and priorities.

The path planning unit implements two of the most popular algorithms: A^* [8] and RRTs (Rapidly-exploring Random Trees) [6]. These allow the system to integrate map-based information in the navigation scheme. The first algorithm is based on a heuristic estimator to find the optimal solution faster than a general search algorithm, such as breadth-first or depth-first searches. Even so, for robotic applications, the A^* algorithm still requires a significant amount of processing power, specially for large state spaces with constraints. RRTs is another search algorithm which exhibits a random nature and non-deterministic solution. However, the RRT algorithm is significantly faster than A^*. This algorithm works like a search tree that starts from an initial state and is expanded by performing incremental motions towards the direction of random points. A Pure Pursuit algorithm [9] has been used to follow the path generated by the planner. The Pure Pursuit algorithm geometrically determines the curvature that will drive the vehicle to a chosen path point defined as one lookahead distance from the current position of the robot.

Despite the robustness provided by the snowmobile chassis, the threat of roll-over is still a major concern. To minimize the liklihood of roll-over, a fuzzy logic slope assessment scheme has been developed to keep the rover on level terrain [10]. The behavior makes use of a slope estimation technique using only a single camera described in [12]. An example of the slope estimates produced is shown in Figure 2. First, the slope estimates are converted into fuzzy linguistic sets. For the slope estimates, the following five sets are used to classify each input: *Positive Steep*, *Positive Sloped*, *Flat*, *Negative Sloped*, and *Negative Steep*. A fuzzy rule base was generated in terms of *IF-THEN* statements to control the robot's direction. A human expert was used to generate a set of simple rules with the intention of keeping the robot driving on level ground. Due to symmetry, the rules that are used to turn right mirror the rules to turn left. A total of seven rules, and their corresponding mirrors, were implemented as part of the fuzzy rule base. This control scheme is able to easily capture inherently nonlinear heuristic knowledge, providing a flexible, easily extendable architecture for designing navigational control laws [5].

IV. SCIENTIST INTERFACE

Once the rovers can successfully navigate within the arctic environment, a means of sending command positions must be

Fig. 2. A example of the slope estimation technique applied to an image of a Colorado glacier.

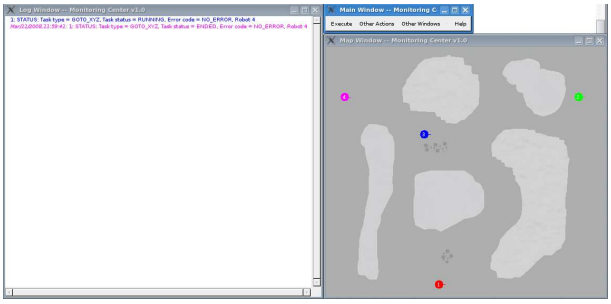


Fig. 3. On the left, the log window with all the information related to one of the tasks. On the right, an aerial view of the terrain with the current positions of the robots.

created. A simple graphical interface has been developed that allows scientists to specify the desired sensor measurement locations. The main window of this interface is an aerial view of the terrain, as shown in Figure 3, where scientists can see the current location of the robots and the specified positions. A log window allows all the different actions taken by the robots to be viewed. For example, the current state of each task and to which robot it is assigned may be easily assessed via the log window. Also, this information is saved with an associated timestamp for each action for later review.

A menu window is available to configure and select the different options of the GUI. One such option is the source of destination information for the team of robots. This can be specified graphically using the map of the environment or with a plain-text mission file (see Figure 4), which is useful for offline mission planning. The tasks can be sent directly to a specific robot or a distributed algorithm can take care of the task allocation (see Section V). Future work includes the possiblity to show, in real time, the weather related data taken by each of the robots.

V. MULTI-ROBOT COORDINATION

After the desired sensor locations have been established, the specified positions must be allocated to the different robots. A market-based algorithm [1] has been used to solve the coordination problem. Market-based algorithms work as a regular auction where two roles are played dynamically by robots: auctioneers and bidders. The auctioneer is the agent

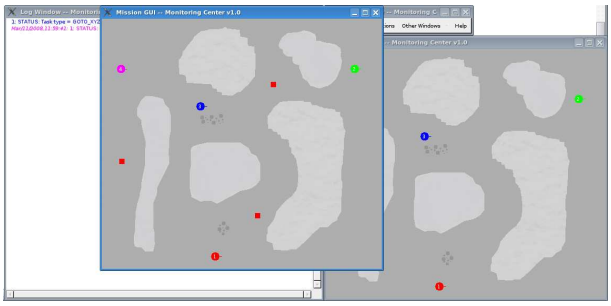


Fig. 4. Graphical interface to specify the locations to be sent to the team of robots. Red squares represent the locations from which the robots will take environmental measurements.

in charge of announcing the tasks and selecting the best bid from bidders. In our case the bid is a quantity that reflects how much it will cost the robot to go to a certain waypoint, such as the euclidean distance or the traversability index [3].

The coordination algorithm has to solve two main problems:

- If I won more than one task, how do I determine which one to keep?
- How do I calculate the bid for a certain task?

First, we try to choose, in an intelligent way, the task that must be kept when a robot wins more than one task. This is accomplished using additional knowledge available to the system. The task with the highest difference between the distance to the robot and the mean of its distance to all the robots will be selected. In other words, if there are N tasks T and robots R and robot R_k has won tasks T_i and T_j, then robot R_k will keep task T_i if and only if:

$$\sum_{l=1}^{N} \frac{D(R_l, T_i)}{N} - D(R_k, T_i) > \sum_{l=1}^{N} \frac{D(R_l, T_j)}{N} - D(R_k, T_j), \quad (1)$$

where $D(R_a, T_b)$ is the distance between robot R_a and task T_b. The rationale is to have the robot choose the task that is best for the team, not just for itself. Thus, robots are more likely to win tasks that have a high cost for the rest of the team, but a relatively low cost for itself.

The question that arises now is how to calculate the mean of the distances for a certain task. During the normal operation of the algorithm, the auctioneer receives bids from all functioning robots in order to allocate the task to the best robot. At this moment, the auctioneer knows all the distances between every robot and the current task. Thus, the mean is calculated by the auctioneer and transmitted to the robot within the task allocation message. The associated mean is stored with the task information and remembered by the robot. The robot is able to utilize this information to select which job to resell in the event of multiple successful bids.

Additionally, task costs are calculated in a non-standard way. Cost functions usually measure the distance between the robot and the task. However, in this algorithm, the cost function is the difference between the distance of the robot and the task minus the mean of the distances between the

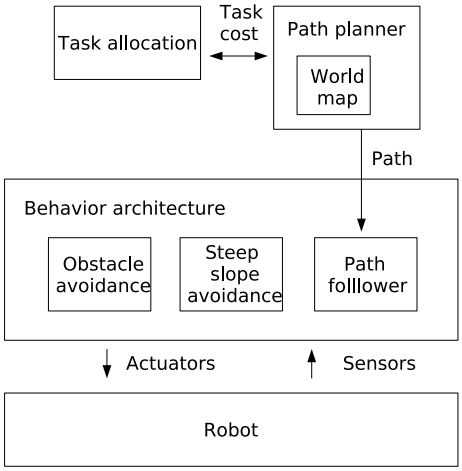


Fig. 5. Scheme that shows the integration of a task allocation algorithm in a complete system ready to be used in a real world application. The path planning algorithm is used to calculate the cost of the tasks and as an input for the path follower algorithm which is combined with obstacle and steep slope avoidance using the DAMN architecture.

robot and all the tasks, i.e.,

$$C(R_i, T_j) = D(R_i, T_j) - \sum_{k=1}^{N} \frac{D(R_i, T_k)}{N}, \qquad (2)$$

where $C(R_i, T_j)$ is the cost function for robot R_i and task T_j and $D(R_a, T_b)$ is the distance between robot R_a and task T_b. The idea is to decrease the cost of tasks that are far away from most of the robots. So, if a robot is close to a task and the rest are far away, the cost will be decreased. Also, if a robot is far away from a task and the rest are close to it, the cost of the task will be kept almost the same. More information about this algorithm and other market-based algorithms applied to the task allocation problem can be found in [4].

On the other hand, market-based algorithms are independent from the number of robots. Therefore, if one robot fails, tasks will be allocated to the remaining robots automatically. Also, these algorithms are well-suited for applications when a high level of fault tolerance is needed. If one robot cannot execute a task, it can be reallocated to another robot starting a new auction.

VI. INTEGRATION OF THE TASK ALLOCATION ALGORITHM WITHIN A ROBOTIC SYSTEM

We have integrated our task allocation algorithm, using a multi-robot architecture based on modules [7], within a complete robotic system ready to be used in real world applications. As can be seen in Figure 5, in each robot the task allocation algorithm has been integrated with a path planner algorithm and the execution of the tasks are within a behavior architecture that combines the path following algorithm with obstacle and steep slope avoidance.

When the task allocation algorithm has to calculate the cost for a specific task, it sends the location data to the path planning module. Next, the path planning calculate the path using the information from an internal wold model and send it back the task allocation module as a list of points. We have used a 2D grid as the world model where each grid is considered as navigable or non-navigable. Different types of non-navigable terrain has been considered for an arctic scenario (see Figure 6(b)).

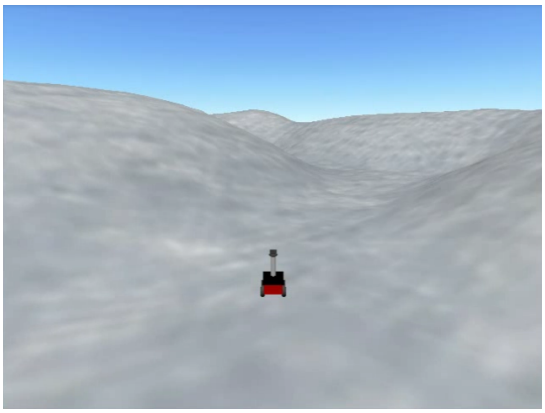
During the negotiation process, it is possible that the task allocation algorithm has to calculate several times the cost for the same task. For this reason, everytime the path planning module calculates the path for a task, it will save the path and its cost. In this way, we reduce the computation power and the calculation time for future requests. Each task is identified by a unique sequence number created by the scientist interface.

After all tasks have been allocated, each robot starts the execution of its own. The path planning module sends the path to the path follower module which is combined with an obstacle and steep slope avoidance behaviors. All three behaviors are combined using a DAMN architecture, as was commented in Section III.

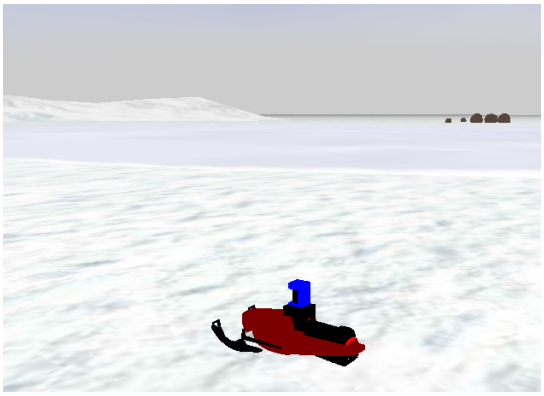
VII. SIMULATION ENVIRONMENT

Expeditions to the arctic are time consuming and expensive. Consequently, having a realistic simulation environment in which to test various navigational algorithms is beneficial. The Player/Stage/Gazebo [2] software suite provides an open-source physics-based 3-D simulation environment which can be easily tailored to different environments. To create an arctic world within the confines of the Gazebo simulation system a grayscale heightmap was created that exhibits the gentle, rolling elevation changes common in arctic areas. This creates the physical terrain for the simulated world, but does not accurately reflect the visual quality of natural environments. Texture files may be applied to produce more realistic surfaces. These texture files are generally small texture patches that are tiled repeatedly across the surface. While this method is not resource intensive, it produces tiling artifacts which are not realistic or visually pleasing. To overcome this issue, a single texture file was created to span the entire terrain. This texture file was hand-drawn to fit the elevation map, with coloring and texturing inspiration taken from images of real arctic scenes. While this is a labor-intensive approach, the resulting simulation is visually similar to arctic photographs. As additional validation, the visual slope estimation algorithms developed to work on actual arctic images also perform well within this simulation environment. Shown in Figure 6 are images taken from the Gazebo simulation of arctic environments.

Finally, the simulation environment has been used to test the navigation scheme and the combination of the different behaviors involved in the execution of the tasks. Therefore, all the different aspects of the application, task allocation and execution, have been integrated and tested using the simulation environment. The step from simulation to real robots has been simplified due to the fact that our robots implement a Player server, as the low level controller, which



(a)



(b)

Fig. 6. Snapshots of the Gazebo simulation of arctic terrain. Image (a) shows some gentle, rolling hills common in snow-covered terrain, while (b) shows three different kinds of non-navigable terrain: hills with high slope on the left, rocks on the right, and ice layer in the middle.

has the same interface than the Player/Gazebo simulation environment.

VIII. CONCLUSIONS

Valuable scientific data exists in the harsh arctic regions of Greenland and Antarctica. To mitigate the cost, effort, and danger of human presence in these regions, mobile sensor networks should be considered as a viable, alternative data collection methodology. However, before autonomous arctic sensor webs can become a reality, certain technological deficiencies must be addressed. Presented here were possible solutions for robust arctic locomotion, as well as a low-cost terrain assessment method to aid navigational processes. A distributed, fault-tolerent task allocation algorithm was described that eliminates the tedious process of assigning individual goal positions to a large number of mobile sensor nodes. Additionally, this algorithm allows the robot team to quickly adapt to the dynamic environment. Finally, an intuitive interface for selecting and modifying goal positions and robot formations was introduced. These advancements serve not only to increase the availability of valuable ice sheet surface measurements, but also to reduce the human work load associated with collecting this data and coordinating a large number of independent agents.

IX. ACKNOWLEDGMENTS

This work supports research in Reconfigurable Sensor Networks for the National Aeronautics and Space Administration, Earth Science Technology Office. Thanks to our collaborators Dr. Derrick Lampkin, Pennsylvania State University, for providing the scientific motivation for this research and Dr. Magnus Egerstedt, Georgia Institute of Technology, for providing his experience in multi-agent formations.

REFERENCES

[1] M.B. Dias, R. Zlot, N. Karla, and A. Stentz. Market-based multirobot coordination: a survey and analysis. *Proceedings of the IEEE*, 94(7):1257–1270, 2006.

[2] B. Gerkey, R. T. Vaughan, and A. Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th International Conference on Advanced Robotics (ICAR 2003)*, pages 317–323, Coimbra, Portugal, 2003.

[3] A. Howard, H. Seraji, and B. Werger. Global and regional path planners for integrated planning and navigation. *Journal of Robotic Systems*, 22(12):767–778, December 2005.

[4] A. Howard and A. Viguria. Controlled reconfiguration of robotic mobile sensor networks using distributed allocation formalisms. In *NASA Science Technology Conference (NSTC 2007)*, Maryland, USA, 2007.

[5] Ayanna Howard and Homayoun Seraji. Vision-based terrain characterization and traversability assessment. *Journal of Robotic Systems*, 18:577–587, Oct 2001.

[6] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, 2001.

[7] I. Maza, A. Viguria, and A. Ollero. Networked aerial-ground robot system with distributed task allocation for disaster management. In *Proceedings of the IEEE International Workshop on Safety, Security and Rescue Robotics*, 2006.

[8] N. Nilson. *Problem-Solving Methods in Artificial Intelligence*. McGraw-Hill, 1971.

[9] A. Ollero and O. Amidi. Predictive path tracking of mobile robots. In *Proceedings of 5th International Conference on Advanced Robotics, Robots in Unstructured Environments (ICAR '91)*, volume 2, pages 1081–1086, 1991.

[10] Kevin M. Passino and Stephen Yurkovich. *Fuzzy Control*. Addison-Wesley, 1998.

[11] J. K. Rosenblatt. DAMN: a distributed architecture for mobile navigation. *Journal of Experimentation and Theoretical Artificial Intelligence*, 9(2):339–360, 1997.

[12] S. Williams and A. Howard. A single camera terrain slope estimation technique for natural arctic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Pasadena, USA, 2008.