EVOLUTION OF AUTONOMOUS SELF-RIGHTING BEHAVIORS FOR ARTICULATED NANOROVERS

Edward Tunstel

Jet Propulsion Laboratory, California Institute of Technology,
4800 Oak Grove Dr., MS 107-102, Pasadena, CA 91109 USA
phone: (818) 393-2666, fax: (818) 354-8172, e-mail:tunstel@robotics.jpl.nasa.gov

ABSTRACT

Miniature rovers with articulated mobility mechanisms are being developed for planetary surface exploration on Mars and small solar system bodies. These vehicles are designed to be capable of autonomous recovery from overturning during surface operations. This paper describes a proposed computational means of developing motion behaviors that achieve the autonomous recovery function. Its aim is to reduce the effort involved in developing self-righting control behaviors. The approach is based on the integration of evolutionary computing with a dynamics simulation environment for evolving and evaluating motion behaviors. The automated behavior design approach is outlined and its underlying genetic programming infrastructure is described.

1 INTRODUCTION

Recent advances in micro-technology and mobile robotics have enabled the development of scientifically capable rovers of mass on the order of tens or hundreds of grams. Development of such *nanorovers* will permit mobility-based science surveys on planetary surfaces with a small fraction of the science payload expected for currently planned, and future, rover missions. Nanorovers have been proposed as possible payloads on landers used for missions to Mars, small bodies, or the moons of gas giant planets [1]. They could be used as individual units or cooperative teams to survey areas around a lander, or even to conduct long-range exploration involving measurement of surface mineralogic and morphologic properties. Research efforts are underway to develop nanorovers that include mobility, computation, power, and communications in a package of several hundred grams in mass [1]. Thus far, a functional nanorover prototype has been developed that is capable of autonomous mobility, science data gathering, and transmission of telemetry to an operator control station [2]. A flight version of the rover is currently under development as a technology experiment on an asteroid sample return mission called MUSES-C. The MUSES-C flight mission is being implemented by Japan's Institute of Space and Astronautical Science (ISAS) and NASA [3]. In addition to the flight development effort, the nanorover concept and design are being refined through ongoing technology research efforts. The aim is to develop miniature, but scientifically capable, rovers that could easily fit within the projected mass/volume reserves of future missions to Mars and small planetary bodies.

The current nanorover prototype features a novel wheeled mobility mechanism that allows it to execute motions beyond conventional rolling and turning. Its articulated mechanism of wheels on posable-struts can be thought of as a hybrid wheeled-legged mobility system. With this design, the rover is capable of operating with its chassis upside down, recovering from accidental overturning, and even hopping in very small gravity fields. Herein, we focus on the important mobility control feature of autonomous self-righting and present an approach to automatic discovery of associated motion control behaviors. We use the term *self-righting* to refer to the act of maneuvering the rover's articulated mobility mechanism to effect recovery from an initial overturned state to its nominal upright driving configuration. Due to the wide range of possible motions permitted by its mobility mechanism, considerable time and effort could be spent designing general self-righting motion sequences for the nanorover. The problem is complicated further when resource limitations (e.g. available power, time, etc) or certain flight constraints must be considered in the solution. A control software design approach is proposed that is aimed at reducing the effort involved in developing self-righting behaviors that are sensitive to on-board resource limitations. The approach is based on the integration of evolutionary computing with a dynamics simulation environment for evolving and evaluating suitable motion behaviors. The

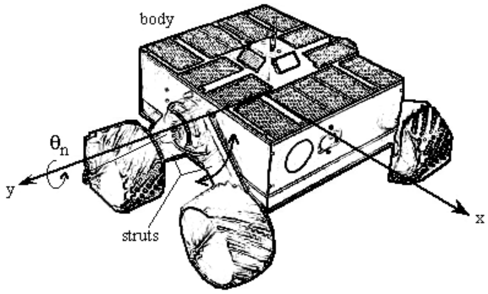

FIG. 1: Articulated nanorover prototype.

automated behavior design approach is outlined and the software infrastructure necessary for implementing the strategy is described.

2 NANOROVER MOBILITY

The current nanorover prototype is illustrated in Fig. 1. The rover's mobility mechanism is comprised of four wheels on articulated struts. Each wheel and strut can be actuated independently. The largest dimension (length) of the rover is 20 cm which makes it 30% the size of Sojourner, the Mars Pathfinder microrover. Each aluminum wheel contains a drive motor within, and is cleated with a helical tread on the outer surface to enhance traction and skid-steering performance. In addition to basic functionality for forward/reverse driving and turning, the high-mobility articulated mechanism provides the rover with the capability to self-right, as well as operate with its body/chassis upside down. This implies the ability to recover from overturning, and allows body pose control for preferential pointing of on-board science instruments. Aside from the rover's apparent miniature size, it is the capability to self-right which distinguishes it from many other planetary rover designs. Moreover, this capability enhances its survivability, and hence, the likelihood of mission success.

The rover has an on-board computer that can be programmed to execute autonomous sequences of strut, body, and wheel motions, which cause the vehicle to self-right (as well as perform other useful behaviors). Its suite of attitude sensors and motor actuators permits simultaneous coordinated control of strut articulation and body pose. As indicated in Fig. 1, the four struts can rotate in two directions about a common pivot axis (y-axis in the figure), however, struts on either side cannot rotate past one another. In addition to strut rotations, the body can be actuated to pitch about the same axis. These rotations constitute the articulation degrees of freedom θ_n (n = 1–5); the wheel motions provide four rolling degrees of freedom. Strut angles are measured by potentiometers; wheel rotational displacements are measured by encoders. The flight rover design includes sensors at each wheel for detecting proximity to, and contact with, the ground. It also includes a sun sensor for detecting body orientation relative to the sun. A variety of pose configurations that are possible with this mechanism are shown in Fig. 2.

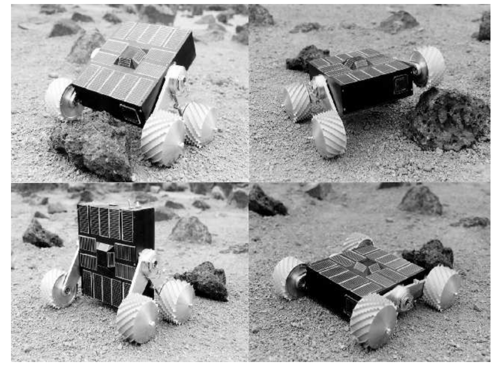

FIG. 2: Posable-strut and chassis configurations.

Due to the flexibility of the mobility mechanism and chassis, a number of feasible motion sequences can be executed that result in successful self-righting from an initially overturned state. One possible sequence is illustrated in Fig. 3, in which the motion progresses from (a)–(f). From the initial overturned state in (a), the rover actuates its struts towards the terrain until its wheels make contact, (b). The same strut motion continues until the configuration in (d) is achieved. At this point, the body is actuated to its nominal upright configuration, (e)–(f). A single fixed sequence such as this may be inadequate as a general self-righting solution. While effective on relatively flat local terrain, it may fail if attempted in very close proximity to large rocks. A more general solution calls for an algorithm or set of control rules that assesses the overturned configuration via sensory perception, and produces expedient actuator controls. For completeness, the behavior should be able to prescribe control responses for the range of possible sensor stimuli. This can be achieved efficiently with behavior control rules that accept inputs that are partitioned into intervals, or even fuzzy sets [4].

2.1 Practical Issues

Some of the motion sequences that can be executed with the posable-strut mechanism are more favorable than others with regard to the total number of motions necessary (and therefore, power required), and the required execution time. Nanorovers used for flight missions rely on solar energy as their primary electrical power source. The flight nanorover

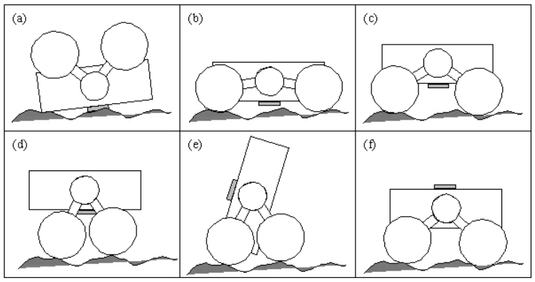

FIG. 3: Example self-righting sequence.

is designed to have most sides of its chassis populated by solar panels, with the primary solar panel located on the nominal top side. This design ensures that sufficient power will always be available for actuation of motors needed to self-right. The maximum size of the primary solar panel for these rovers is limited by the small footprint of the vehicles. As such, nanorovers must operate within the constraint of relatively low power budgets. Sufficient available on-board power for mobility actuators, science instruments, and communications is of primary concern for nanorovers. Designs for self-righting and other motion behaviors must be sensitive to on-board power constraints. Some of the most intuitive solutions (such as that in Fig. 3) may not sufficiently account for realistic on-board resource limitations. Therefore, it behooves the rover control engineer to explore the space of feasible solutions for behaviors that would minimize power consumption and comply with other operational constraints or flight rules. Execution time required for self-righting is also of concern since the frequency of unintentional overturning may be significant for nanorovers operating in certain environments and terrain-types. The cumulative time spent recovering from frequent overturning could easily detract from time allotted for science data gathering and navigation goals. An additional concern for nanorovers is the negative impact that dusty environments can have on solar panel efficiency. Due to their low profile relative to the terrain, dust could accumulate over time on the rovers' solar panels. The problem is only compounded each time the rovers overturn. This issue is currently being addressed by a dust mitigation approach planned for the flight rover, which is based on the use of an electronic dust rejection apparatus.

As an alternative to the tedious effort of examining all of the possible motion sequences, an automatic computational method of self-righting behavior design is proposed in the following section. The goal and expected result of the approach is the discovery of one or more viable self-righting behaviors that can be used as is, or as a starting point for further refinement. The advantage is a savings in time and effort that would otherwise be spent searching the space of possible motion sequences.

3 SELF-RIGHTING EVOLUTION

In this section, we outline an approach to artificial evolution of self-righting behaviors. More specifically, we propose genetic programming for off-line learning of self-righting behaviors for nanorovers. A genetic programming (GP) system [5] computationally simulates the Darwinian evolution process by applying fitness-based selection and genetic operators to a population of candidate solutions, which are represented as computer programs or subroutines. The main distinction between genetic programming and genetic algorithms is that the former adapts hierarchical *symbolic* data structures (e.g. computer programs), while the latter adapts linear numerical data structures (e.g. bit strings or arrays of integers or reals). For our purposes, the computational structures undergoing adaptation are sets of condition-action rules of dynamically varying size and structure. That is, the population consists of behavioral rule sets, each represented as a tree data structure, of different numbers of rules. Tree nodes, or *genes*, may consist of components of a generic if-then rule construct and common logic connectives (e.g. `AND`, `OR`, and `NOT`), as well as input/output variables and parameters associated with the problem. Each set of rules constitutes a motion behavior that maps articulation, orientation, and wheel-contact sensor values into strut and body motions.

The objective of the GP system is to create a population of candidate self-righting behaviors, evaluate behaviors via dynamics simulation, and improve the population through artificial evolution until one or more highly fit solutions is discovered. All behavioral rule sets in the initial population are randomly created from syntactically valid combinations of genes. Descendant populations are created by genetic operators — primarily reproduction and crossover. For the reproduction operation, several behaviors selected based on superior fitness are copied from the current population into the next, i.e. the new generation. The crossover operation starts with two parental rule sets and produces two offspring that are added to the new generation. This operation selects a random portion of each parental tree structure and swaps them (while maintaining valid syntax) to produce the two offspring. GP cycles through the current population evaluating the fitness of each behavior based on its performance in computer simulations of the control system. After a numerical fitness is determined for each behavior, the genetic operators are applied to

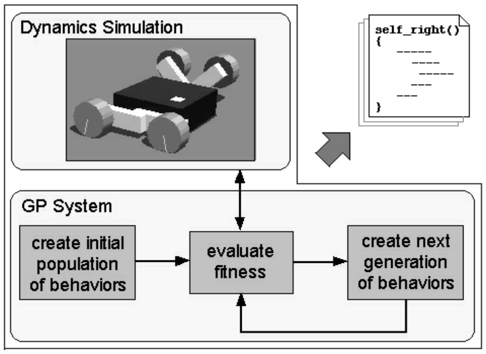

FIG. 4: Behavior evolution architecture.

the fittest behaviors to create a new population. This cycle repeats on a generation by generation basis until satisfaction of termination criteria (e.g. discovery of a highly fit behavior, lack of improvement, maximum generation reached, etc). The end result is the best-fit behavior that appeared in any generation.

The overall process is summarized as illustrated in Fig. 4. Candidate self-righting behaviors in the population evolve in response to selective pressure induced by their relative fitnesses for implementing the desired motion behavior. This population-based approach is particularly suitable for global search and optimization in large and/or multi-modal search spaces. The key distinction between such evolutionary search methods and a conventional gradient descent based approach is that, in the former, multiple points in the search space are sampled in parallel. The approach has been verified through numerous examples reported in the literature. In the definitive GP text [5], Koza has applied genetic programming to evolve computer programs that solve a number of interesting control problems. The same techniques have been successfully applied to search and optimization of robot manipulator trajectories [6], mobile robot control and navigation behaviors [7], and collective behaviors for multi-robot systems [8]. Each implementation differs in various problem-dependent ways. However, for robotic system applications, a common characteristic is the formulation of a fitness measure that drives the evolution and is coupled to a motion simulation. The viability of evolved behaviors is a function of the thoroughness of the evaluation process. Performance is based solely on evaluation of behavioral responses predicted by the simulator, and is computed by a user-prescribed fitness function. As such, the success of the approach depends in large part on the fitness function employed and the fidelity of the simulation environment. Each of these integral aspects is discussed further below.

3.1 Behavior Evaluation

In order to apply evolutionary algorithms for behavior design, a measure of behavior fitness must be formulated to drive the process. It is important that the fitness function map observable parameters of the problem into a spectrum of values that differentiate the performance of behaviors in the population. If the spectrum of fitness values is not sufficiently rich, the fitness function may not provide enough information to guide GP toward regions of the search space where improved solutions might be found. For problems involving simulation of controlled behavior, a variety of performance attributes can be considered for inclusion in the fitness measure. Examples include a maximum number of time steps, explicit error tolerances, terminating physical events such as task success or failure, and penalties/rewards thereof. In general, selected performance attributes can be weighted to emphasize their relative importance in the search for candidate solutions. The fitness function is analogous to the performance measure of optimal control theory, or more generally, the objective function of optimization theory.

One approach to evaluating evolving candidate self-righting behaviors is to test them against a number of fitness cases, tabulate a performance score for each case, and average the scores to determine an overall fitness value. The initial postures for each fitness case should be chosen to represent an overturned configuration that can occur in the target environment. The number of fitness cases should be chosen such that they represent the search space sufficiently to allow the evolved strategy to generalize (i.e. handle unforeseen initial conditions). Fig. 3a is one example of a fitness case for the self-righting problem. A few additional examples are illustrated in Fig. 5. For each fitness case the goal is the same — recovery from an initial overturned state to achieve the nominal upright driving configuration.

Given the practical points expressed in Sect. 2.1, it would be prudent to formulate a fitness score based primarily on the estimated power consumed by motors (p), the time elapsed during execution (t), and the percentage of progress made ($\psi \leq 100$). Each of these performance attributes is measurable at the end of each fitness case. It is possible, however, to formulate the fitness evaluation such that performance is measured during fitness case execution. This was done in [8] where a reinforcement learning function was coupled with fitness evaluation to install a progress indication during fitness trials. Power consumption can be estimated from knowledge of motor performance characteristics and usage during execution. Elapsed time is determined based on simulation

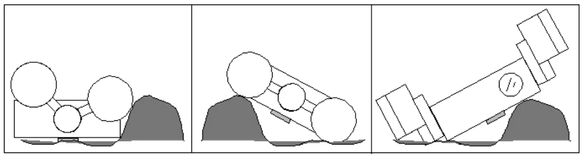

FIG. 5: Example fitness cases.

ticks starting from the beginning of the self-righting maneuver to the end of the trial. The amount of progress made is indicated by the percentage of angular displacement achieved by the chassis from the initial posture towards the desired nominal driving configuration. Secondary performance attributes from among the aforementioned examples can also be included in the formulation. With selected attributes defined, a fitness score $f(p, t, \psi)$ can be computed for trial runs through each fitness case. The *overall* fitness of a candidate self-righting behavior would be computed by averaging the scores over the total number of fitness cases defined. Suitable fitness formulas for self-righting would reward behaviors that consistently achieve (or come close to) the desired upright configuration in a timely manner, while minimizing power consumption.

3.2 Dynamics Simulation

A simulation environment is a key component of the approach described above. This is particularly true for evolution of rover behavior(s). One of the challenges of evolutionary robotics is the successful evolution of robust controllers in simulation. It was pointed out in [9] that the use of simulation environments of questionable fidelity tend to result in evolved behaviors that are not easily transferable to real robots. However, for developing rover systems designed to operate in unknown space environments, evolution in simulation is often the most practical option. Behaviors evolved in simulations must, however, be validated and verified to some extent on real rovers. The use of rover and environment simulators of reasonably high fidelity can mitigate such concerns. Preexisting simulators are particularly useful in streamlining rover control and navigation software development efforts when prototype/flight hardware is unavailable or inaccessible.

A high-fidelity dynamics simulation system is available at JPL for use in this work. It is based on the JPL-developed Darts/Dshell [10] simulation tools. Darts/Dshell is a multi-mission spacecraft simulator with a real-time computational engine for flexible multi-body dynamics. It includes libraries of hardware models for various sensors, actuators, and motors. Its simulation infrastructure allows for interfaces to a 3D animation viewer and rover research/flight software. The interface between rover software and the simulator enables software to issue control updates to the simulator and receive state/sensor data from the simulator. The computational engine computes dynamics of multi-body systems based on inertial properties of the bodies in the system and forces applied to those bodies. In this dynamics simulation system, the nanorover is modeled as a multi-body system of wheels, struts, and a chassis. Different friction models can be created to simulate characteristics of wheel-terrain interactions, and the gravitational acceleration can be varied as well. Currently, the Darts/Dshell spacecraft simulation tools are being leveraged to develop a related software simulation toolkit that is more germane to rovers [11]. These systems provide suitable environments for rover/terrain modeling and simulation that are useful for flight software design and development. When integrated with a genetic programming system, as described above, high-fidelity simulators provide a fitness evaluation medium for artificial evolution of rover behaviors.

4 ISSUES FOR SMALL BODIES

The approach as described thus far is nominally focused on the basic discovery of self-righting behaviors that might be feasible on Earth and Mars. The importance of a self-righting capability is magnified in the case of surface exploration on small bodies like asteroids. In this case, the gravitational fields are substantially weaker than those of Earth or Mars, and the likelihood of unintentional overturning is substantially higher. Before the proposed approach can be applied to evolve effective behaviors for small-body exploration, additional considerations must be factored into the dynamics simulation. Most notable among these are appropriate gravitational effects and terrain characteristics.

When accurate data about small bodies of interest are unknown, assumptions about gravity and terrain characteristics must be made. In a recent preliminary study [12], the mobility performance of a nanorover operating within a small-body gravity field was examined using a commercial dynamics simulation software package. In that study, assumptions were made about the environment of the near-Earth asteroid Nereus (4660), the primary target of the MUSES-C flight mission, which is less than one kilometer in diameter. The surface gravity of Nereus is expected to be 8–80μg [3]. In [12], 20μg was assumed. The aim of this small-body mobility study was to predict the rover's ability to maintain adequate tractive forces with the ground surface to achieve forward progress. Two wheel-terrain interaction models were considered. The first was based solely on Coulomb friction (with a friction coefficient of 0.5); the second was a combination of Coulomb friction and adhesive forces (thought to arise due to electrostatic attractions between the wheels and a dusty surface). To computationally evolve self-righting behaviors for such environments, the simulator used for behavior evaluation must be capable of representing different gravity fields and terrain types. The dynamics simulator mentioned above offers this flexibility.

Until additional facts are learned about Nereus, data presented in [3] and assumptions made in [12] will be used as a baseline for our computational behavior evolution experiments. For the upcoming flight mission, relevant new findings will be factored into the design of control and navigation behaviors for mobility on the target asteroid. The various desirable attributes of viable evolved behaviors will be identified for possible realization on the flight rover. This activity will be supported by high-fidelity computer simulations as well as hardware-based low-gravity simulations that focus on evaluating behaviors in the context of relevant mission scenarios and constraints.

5 SUMMARY AND CONCLUSIONS

Nanorovers with articulated mobility mechanisms are capable of a variety of maneuvers besides conventional rolling and turning. This paper has focused on the problem of autonomous self-righting and has expressed some of the practical aspects of the problem. An automated software design approach has been proposed for developing rover control behaviors for self-righting. Genetic programming is advocated as a means for offline learning using a high-fidelity dynamics simulation of the rover and environment. The proposed approach can be used to synthesize self-righting behaviors and optimize them based on performance feedback from the simulator, which can be interfaced with prototype rover control software. The integrated system would be beneficial for streamlining rover software design and development efforts.

In addition to self-righting behaviors, the approach can be applied to develop other functionalities for which solutions are not already well-defined. The necessary software infrastructure consists of an evolutionary computation kernel and a simulator of reasonable fidelity. The interested reader can find source code for implementing GP in the LISP programming language in [5]. Public domain implementations that are written in C or C++ are also available on the World Wide Web.

ACKNOWLEDGMENTS

The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration. The dynamics simulation environment is being developed by Jack Morrison and Jeffrey Biesiadecki of the Jet Propulsion Laboratory.

REFERENCES

[1] B. Wilcox et al. Nanorovers for planetary exploration. In *AIAA Robotics Technology Forum*, pages 11–1–11–6, Madison, WI, Aug 1996.

[2] E. Tunstel, R. Welch, and B. Wilcox. Embedded control of a miniature science rover for planetary exploration. In *7th Intl. Symp. on Robotics with Applications, WAC'98*, Anchorage, Alaska, May 1998.

[3] R. Jones et al. NASA/ISAS collaboration on the ISAS MUSES C asteroid sample return mission. In *3rd IAA Intl. Conf. on Low-Cost Planetary Missions*, IAA-L98-0506, Pasadena, CA, Apr 1998.

[4] E. Tunstel. Mobile robot autonomy via hierarchical fuzzy behavior control. In *6th Intl. Symp. on Robotics and Manufacturing, WAC'96*, pages 837–842, Montpellier, France, May 1996.

[5] J.R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, MA, 1992.

[6] Y. Davidor. *Genetic Algorithms and Robotics: A Heuristic Strategy for Optimization.* World Scientific Publishing Co., Teaneck, NJ, 1991.

[7] E. Tunstel and M. Jamshidi. On genetic programming of fuzzy rule-based systems for intelligent control. *International Journal of Intelligent Automation and Soft Computing*, 2(3):273–284, 1996.

[8] S. Calderoni and P. Marcenac. Genetic programming for automatic design of self-adaptive robots. In W. Banzhaf et al., editors, *Genetic Programming: Proc. of the First European Workshop.* Springer, Paris, France, 1998.

[9] M. Matarić and D. Cliff. Challenges in evolving controllers for physical robots. *Robotics and Autonomous Systems, Special Issue: Evolutional Robotics*, 19(1):67–83, Oct 1996.

[10] J. Biesiadecki, A. Jain, and M.L. James. Advanced simulation environment for autonomous spacecraft. In *Intl. Symp. on Artificial Intelligence, Robotics and Automation in Space*, Tokyo, Japan, Jul 1997.

[11] J. Yen, A. Jain, and J. Balaram. ROAMS: Rover analysis modeling and simulation software. In *Intl. Symp. on Artificial Intelligence, Robotics and Automation in Space*, Noordwijk, Netherlands, Jun 1999.

[12] E.T. Baumgartner et al. Mobility performance of a small-body rover. In *7th Intl. Symp. on Robotics with Applications, WAC'98*, Anchorage, Alaska, May 1998.