A Simple Reactive Architecture for Robust Robots

Rajiv S. Desai
desai@robotics.jpl.nasa.gov

David P. Miller
dmiller@ai.mit.edu

Jet Propulsion Laboratory / California Institute of Technology
Pasadena, CA 91109 USA

November 8, 1998

Abstract

This paper describes a simple reactive architecture that has been sucessfully implemented on a small outdoor robot that performs navigation and fetch tasks in rough terrain. The architecture uses simple behaviors combined with a simple sequencing system to provide apparently "intelligent" responses to its environment while maintaining ease of programmability and modification.

1 Introduction

Rocky III is a small mobile robot capable of autonomously and reliably navigating through rough outdoor terrain, acquiring a soil sample, and depositing the sample in a receptacle marked by a beacon[Miller92]. Rocky III was developed at the Jet Propulsion Laboratory as part of the Planetary Rover research program. Planetary rover research has often emphasized innovative design to reduce power (eg., [Simmons91]). Rocky III's architecture allows the robot to be made small, and therefore lower power in all of its subsystems. Small robots are cheaper to launch, but because they cannot support high-bandwidth communications, they must possess some level of autonomy.

Rocky III is the only example known to us at this time of an autonomous robot which operates off-road and performs both navigation and manipulation. The robot's performance has been demonstrated in dozens of tests in both an indoor laboratory setting and outdoor rough-terrain environments.

Rocky III is controlled by a small 8-bit processor using about 10K of memory. This is made possible through the use of a reactive behavior-control approach, where sensors are coupled directly to actuators through relatively simple computations. The control structure for Rocky III is extremely similar to that used on Tooth, our indoor object-collecting micro-rover [Miller90].

2 Behavior Control Architecture

Rocky III is controlled using a software paradigm which has come to be called behavior control, which can be characterized by the following features:

- Behavior control tightly couples sensors to actuators through fairly simple computational mechanisms.
- Complexity in behavior control mechanisms is managed by decomposing the problem according to tasks (e.g. collecting a soil sample) rather than functions (e.g. building a world model).
- Behavior control can be applied in situations where classical control theory is not applicable, either because the plant (i.e. the environment) cannot be modelled with sufficient precision, or because the complexity and dimensionality of the transfer function is too high to allow the mathematics to be carried through.
- Behavior control employs transfer functionals rather than the transfer functions of classical control theory, that is, the output of a behavior-based control mechanism can contain stored internal state (i.e. memory).

The behavior control architecture that we use consists of two main parts: behaviors and the master sequencer. The behaviors are a set of small reactive programs that allow the robot to do "something". For example, Rocky III has a behavior that is given a heading as input, and then turns the robot to that heading. These behaviors are reactive in that the heading behavior does not calculate the current heading, subtract it from the desired heading and then try and turn the robot to the resultant angle. Instead, the behavior compares the current heading to the desired heading to calculate which direction the robot should turn. The robot starts to turn, and the behavior continues to be active. If the robot turns too much, the same behavior will correct the heading. If the desired heading is changed, no new behaviors need to be activated. This very simple behavior is competent in its limited domain: keeping the robot pointed in the right direction, or turning it towards the right direction.

The other major component of the architecture is the sequencer. The sequencer can most easily be thought of a large case statement. The tests for each case are either particular sensor values, obtained from the environment, or state variables that have been set previously. The action clauses for each case are simply the (de)activation of various behaviors or the setting of state variables (by activated behaviors, new sensor data, or the sequencer itself). As Rocky III goes towards its goal, it may first need to pass through several predefined way-points. When the robot's position is within a specified epsilon of the current way-point, the test condition for the sequencer case that increments waypoints is met. The sequencer changes the value of the curent goal state variable, and all of the relevant behaviors then use those new values.

3 Rocky III Software

Rocky III is programmed in ALFA, a behavior language for designing reactive control mechanisms for autonomous mobile robots [Gat91]. ALFA programs consist of networks of computational modules which communicate with each other and with the outside world by means of communications channels. Modules and channels are nominally dataflow devices, that is, they continuously recompute their outputs as a function of their current inputs. Modules can also contain stored internal state, allowing sequential computations to be embedded within ALFA's dataflow framework.

The control software for Rocky III is made up of two types of behaviors, and the master sequencer. The basic behaviors control the motors based on input from the complex behaviors. For example, there is a behavior that reads two input channels, which output a nominal vehicle

speed and steering direction, and computes the settings for the individual drive and steering motors, there by implementing Ackerman steering at any desired speed.

The complex behaviors implement functionalities such as moving to a commanded heading and moving around obstacles. Moving to a commanded heading simply involves computing the difference between the desired heading and the current vehicle heading as reported by the compass, and generating an appropriate steering command. Moving around obstacles is currently accomplished by backing away from the obstacle and turning to one side. This is a fairly simplistic strategy and could be improved, but field tests have shown this approach to be quite robust.

The master sequencer performs high-level control over the mission. On Rocky III the sequencer moves the robot through a series of way-points (goal locations), stops the vehicle, collects a soil sample, and returns to the lander.

The master sequencer collects input from the original task description (goal and way-points) and from the beacon. The closest thing to a map in Rocky III is manipulated by the master sequencer. The "map" is simply a list of X,Y points that give the position of the lander, the goal point, and the waypoints. The master sequencer uses these points one at a time to compute the robot's current heading. When returning to the beacon, the signal received from the beacon is used to compute the input to the goal heading channel. Depending on the signal from the beacon the robot will maneuver directly towards the beacon, at right angles to it (so it can line up on the center line), or head towards the best estimate of the beacon's position (when the beacon is out of range or occluded).

4 Experiments

Dozens of tests were performed to verify the robot's abilities. The experiments took place in a large indoor laboratory, and outdoors in the Arroyo Seco outside of JPL (a dry wash strewn with rocks, boulders, sand, and hard packed soil). All of the experiments had the same basic format, though the details of the robot's starting position and orientation, the positions of obstacles, the sample site and way-points differed from test to test.

At the start of an experiment, the operator downloads the sample site and way-points (if any) to Rocky. Each point requires four bytes. The positions are given in X-Y coordinates with the X-axis aligned to the centerline of the lander, and the origin at the front of the lander. The robot is given its starting location and the compass orientation of the lander. The operator then tells the robot to start.

As Rocky starts moving forward, it compares its current heading to the heading needed to get to the first way-point (or the sample site if no way-point has been specified) from its current location. It calculates the proper direction of turn and turns in that direction. This behavior is continuously repeated so that should the robot overshoot the turn, Rocky will automatically correct its orientation. The robot keeps track of its position by using the wheel encoders and current compass heading to update its X-Y estimate.

As Rocky travels, it comes across rocks, ledges, and slopes of various sizes and degree. Any ledge or rock smaller than 13cm (one wheel diameter) is traversed by the mobility system. Ledges or rocks greater than a wheel diameter in size that are first contacted by a wheel, trigger one of the bogie switches. Rocks larger than a wheel diameter that go between the front wheels are detected by the skid plate or front contact switches. Severe slopes are detected by the roll and pitch clinometers in the belly pan of the robot.

When a section of untraversable terrain is detected, the robot executes an avoidance maneuver. It backs up, turns ninety degrees to the left or right (the opposite direction from the side of the vehicle at which the obstacle was first detected), moves forward a vehicle length (approximately a half meter), and then resumes its normal behavior. These obstacle detection and avoidance behaviors are active at all times and can override any other active behaviors.

When Rocky reaches the sample area, it deploys the sampling arm and tests the ground ahead of it for soft soil. If the ground is unsuitable, it lifts the arm, moves forward a few centimeters, and tests again. If suitable soil is not found within five trials, the mission is aborted, and the rover returns to the lander. When soft soil is found, the gripper closes around the sample, and the arm is retracted and stowed.

Using its current position estimate, the robot turns to a heading to bring it back to the lander. At the same time, it starts scanning for the lander beacon. The beacon consists of two sectors approximately forty degrees in angle, and a twenty degree center sector. Each sector's signal is modulated at its own characteristic frequency. When the beacon is detected by Rocky the robot heads straight towards the beacon if it detects the center sector, or it turns ninety degrees to the direction of the beacon, and moves towards the center sector, if a side sector is detected, until the center sector is detected. The beacon is then followed all the way to the lander. If sight of the beacon is ever lost (from occlusion), then Rocky reverts to navigation back to the lander by dead-reckoning.

The robot uses its front contact sensors to detect docking with the lander. At this point it deploys the arm and deposits the sample in the collection container.

After the software was debugged and tuned, dozens of runs were performed with only a few failures. Each of these failures involved hardware (a drive motor gave out, the beacon failed, a contact sensor was stuck on). In all cases the avoidance software succeeded in getting the robot through the obstacles and to the destination. For the outdoor runs, the vegetation was usually removed from the test area. But even during a run where the sample area was designated in a heavily vegetated spot, the robot eventually made its way around the large weeds and to the sample area.

5 Conclusions

This work adds to the body of evidence for the claim that complex symbolic computations and world models are not necessary to produce robust, intelligent behavior in autonomous mobile robots.

The success of the Rocky III experiments make another important point. Compared to most other reactive robots (eg., Herbert [Connell90]), Rocky III is sensor impoverished. The proprioceptive sensors (compass, encoders, and clinometers) tell the robot its orientation (and to some extent location) in space. However, all the information about the terrain that the robot is traversing comes from the bogie limit switches and the four contact switches on the front of the robot - eight single bit sensors. Rocky cannot sense the environment until it literally runs into it! Despite this handicap, Rocky is very capable of making its way through realistic and hazardous terrain.

It should be noted that natural terrain is seldom a maze. Terrain is rich with paths, and it is not necessary for the robot to select the optimal path, only a path that works. If placed in a maze, Rocky might never make it out, but in real terrain it will succeed in almost all circumstances (it has not failed yet!). By limiting the scope of the robot to those environments it will realistically encounter, we have been able to simplify the sensing and computation systems of Rocky beyond those typical even for reactive robots.

The exact nature of Rocky's program is somewhat ad hoc. It is difficult to prove, in a formal sense, what the robot is capable of. This could potentially make the robot difficult to program, since the lack of formal analysis means that lots of incremental experimentation is needed. Previous reactive architectures have had some difficulty in this area, because even if the individual behaviors were debugged separately, it would still be difficult to know how they will perform when put together. Under the architecture described here, this has not been a problem. Each behavior is independent and interfaces to others or to state variables only at prespecified inputs and outputs. One cannot drop an inhibiting "wire" into the middle of a behavior in this system [Brooks86]. While this may limit the flexibility of the architecture in

some extreme cases, we feel this is more than made up for in testability and ease of programming and modification.

Given the simplicity of the sensors and the programming, the question arises: "where is the intelligence?" The capabilities exhibited by this robot are a result of the entire robot system interacting with its environment. The sensors are simple, but they are the appropriate sensors for this robot and this class of activities. By mixing the sensing and reactive capabilities appropriately with the mobility hardware's capabilities, and the class of tasks assigned to the robot, we have a robot that operates intelligently over its domain. The intelligence is just as much hardwired into the selection and placement of the sensors and the actuators as it is in the executed code, but it works just as well. The experiments described above show that an intelligently acting system can be created where the intelligence is in large part encoded in the device structure, rather than totally in the control/planning system.

5.1 Acknowledgements:

This research was conducted at the Jet Propulsion Laboratory - California Institute of Technology under a grant from the National Aeronautics and Space Administration. The authors wish to thank David Atkinson, Donald Bickler, Jack Fraisier, Erann Gat, Robert Ivlev, John Loch, Nora Mainland, Jim Tran, and Brian Yamauchi for their significant contributions to this research.

6 References

[Brooks86] Rodney A. Brooks, A Robust Layered Control System for a Mobile Robot, *IEEE Journal on Robotics and Automation*, vol RA-2#1, March 1986.

[Connell90] Jonathan Connell, *A Colony Architecture for an Artificial Creature*, MIT Artificial Intelligence Laboratory TR#1151, 1990.

[Gat91] ALFA: A Language for Programming Robotic Control Systems, in *Proceedings of the IEEE Conference on Robotics and Automation*, May 1991.

[Miller92] D. P. Miller, R. S. Desai, E. Gat, R. Ivlev, and J. Loch, Reactive Navigation through Rough Terrain: Experimental Results. in the *Proceedings of the 1992 National Conference on Artificial Intelligence, AAAI*, SanJose, CA, July 1992.

[Miller90] David P. Miller, The Real-Time Control of Planetary Rovers Through Behavior Modification, in the *Proceedings of the 1990 SOAR Conference*, Alberquerque NM, June 1990.

[Simmons91] Reid Simmons, Erik Krotkov & John Bares, A Six-Legged Rover for Planetary Exploration,paper #AIAA-91-3812-CP in the *Proceedings of Computing in Aerospace 8*, AIAA, pgs 739-747, October, 1991.