The failures of a self-reliant tour robot with no planner
or, What might a planner buy you?

Illah R. Nourbakhsh
The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
illah@cs.cmu.edu

Abstract

This essay very briefly describes a self-reliant, long-term mobile robot that does not have a planning module, then dives into the precise failures and shortcomings that this robot has faced in its operational life thus far. An analysis of these real-world failures may help to answer the question, "*What does planning buy you?*"

1 Introduction

"What fails when a totally autonomous robot lacking a planner is given the chance to run for months?" This is the question that this essay will address. We will describe the failures of a robot we have recently put into service at the Carnegie Museum of Natural History. The goal is to draw some conclusions about the sorts of failures that occur in long-term robotics, and to trigger some thoughts on how planning may or may not avert those failures. In the course of carrying on this discussion, you will need to have some background information on the robot and its algorithms. First, therefore, we present this background *very* briefly, lingering only long enough to mention a few essentials. This *Background* section is not a technical discussion by any means, as will be clear by the complete lack of references to similar work, of which there is a great deal.

Then, we will explore the failures that this robot has encountered, thereby beginning a discussion of what a planning module might buy this robot.

One caveat must first be presented: Sage is just a single data point in the space of socially useful, self-reliant robots. The conclusions based on Sage's accomplishments and failures are educational, but they obviously do not easily generalize to other problem domains or, indeed, other hardware instantiations.

2 Background

In early February, 1998, the Robotics Institute and the Carnegie Museum of Natural History set out to install a robotic exhibit in one of the museum's most popular attractions: Dinosaur Hall (Figure 1).

Figure 1: Dinosaur Hall

Our goal was to install a mobile robot that would wander among the dinosaurs and provide much more information about the dinosaurs and about paleontology than the visitors could glean by reading the posted summaries. The challenge was to create a tour robot that would be self-reliant, needing no human supervision during its operation nor requiring human aid to recharge its batteries at the end of the day.

We set May 22 as the deadline (which was met), making this an extremely rapid, 100 day development effort. By the time you read this, the robot has spent more time in operation than during coding and development, and so it offers us a rare source of data on the behavior and failures of a long-term mobile robot.

3 Sage: Implementation

We very briefly describe Sage's hardware, its obstacle avoidance and navigation algorithms, and its touring capabilities.

Figure 2: Sage with its target audience

Hardware

The robot consists of a modified XR4000 robot, which is commercially available from Nomadic Technologies, Inc. The XR4000 has four wheels, each with independent rotation and translation motors; so, it has the odd *holonomic* ability to pirouette and move along a line, all at the same time. The sensor suite consists of two rings of 24 sonars each, two rings of 24 active infrared rangers, and 24 touch-sensitive panels along the length of the robot. The processor is a single Pentium P166 running Linux.

On top of this base we mounted a Pioneer laserdisc player (connected to the PC for control via serial line), an Altec Lansing subwoofer/amplifier and tweeters, a 15” color LCD monitor and a video converter to translate the laserdisc player’s output to the monitor’s RGB input. Higher up still, we mounted a single color CCD camera that interfaces to a Matrox Meteor framegrabber.

Algorithms

Sage performs obstacle avoidance by brute sensor force, for it has so many sonar sensors that we have never seen the sonar fail to detect an obstacle, human or otherwise. The obstacle avoidance algorithm is purely functional, simply selecting a direction of travel that is within 45 degrees of the desired direction of travel (which is toward the end of the hall) based only on most recent sonar values. Recall that the robot is holonomic; no rotational velocity needs to be chosen, since the path changes are virtually instantaneous. Depending on the proximity of obstacles in the chosen direction of travel, Sage will move at an appropriately slow speed or stop.

Sage uses no sensor map or occupancy grid of any kind. However, each hallway has an associated width, and the obstacle avoidance algorithm will treat the edges of that virtual path as hard boundaries not to be crossed, ensuring that the robot stays within a desirable, preset distance from the centerline.

Navigation is gradually becoming a non-problem in the robotics community; however, Sage must navigate sufficiently reliably as to get lost, well, *never*. We chose a simple strategy that would accomplish this – and the chosen strategy has strong repercussions relating to the conclusions of this article, so beware.

Sage navigates by traveling on discrete, predefined virtual hallways, which may be viewed as highways with associated safe widths. The robot measures its longitudinal position in the hallway exclusively using its encoders. It never performs any other form of longitudinal localization. The robot measures its lateral position in the hallway exclusively using encoders as well. It simply calculates its theoretical offset from the centerline of the hallway.

The remaining uncertainty regards orientation, or angular localization with respect to the hallway. In some hallways, Sage uses solely its encoders for this as well. However, this strategy only works temporarily, for encoder orientation will gradually drift, eventually accumulating enough error as to be unusable. To prevent this, seven of Sage's ten hallways have a large, pink rectangle positioned at the end of the hallway. The pink rectangles are approximately 1 meter on a side, and they appear in the color CCD camera, at the furthest point, as a rectangle with 12 pixels of width. In these marked hallways, Sage simply measures its angular position to the hallway by measuring the displacement of the pink landmark in its CCD image, which has a field of view of 45 degrees.

Very simple filtering techniques, based on high-saturation and hue range selection, make it essentially impossible for Sage to miss its bright landmarks. Furthermore, Sage stores information about the expected size and position of these landmarks, using this information to double-check that the landmarks have not been manipulated or modified. In all of our development and operation, there have been no instances of Sage incorrectly picking out any of these landmarks.

Recall that Sage was to be self-reliant, and therefore it would have to be able to plug itself into the wall autonomously. After much research and testing, it became clear that building a compliant plug in order to compensate for the robot's inaccuracies was actually more expensive and more difficult than simply making the robot servo sufficiently accurately to plug itself into an unmodified plug.

Figure 3: Sage's docking landmark and regular plug

The resulting plug, which Sage has used without failure every day since May 22, includes a special, miniature landmark that Sage uses to visually align itself during docking (Figure 3). The landmark consists of a pink background square 11 cm on a side and a 1 cm square black square in the foreground, 10 cm in front of the background square. By measuring the relative position of the black square on the pink background, the size of the landmark, and the position of the entire landmark in the CCD image, Sage is able to compute its relative x-y and angular position from the plug, then simply servo in and plug itself in. This process takes approximately 7 minutes from a point 3 meters away from the wall receptacle.

Interactive Capabilities

Sage is a robot that delivers a canned tour for museum visitors. It does not customize tours to suit the interests of particular museum visitors, but rather travels between a predefined set of tour stops, in a predefined order, and plays an audiovisual presentation from its custom laser-disc at each stop.

The small amount of interactivity that Sage does possess is born out by what the robot does *between* its tour stops. Most noticeably, at a central location that serves as the beginning of the tour, Sage will stop and wait for visitors to approach it before beginning a tour. It has a large, blue button next to its LCD screen, and it will ask visitors that it has detected to push the button in order to start a tour.

A more subtle and far more interesting form of interactivity in which Sage engages involves its responses, over the long term, to the way in which humans treat it. Sage has a simple *mood architecture*, consisting of a set of emotional states (happy, lonely, tired, frustrated, confused, busy) and transitions between these states based on a slowly changing scoring function. Mood transitions are triggered, over time, by the perceptual events that Sage can identify: {*bumper-push*, *people-approach-behind*, *people-depart-behind*, *people-approach-front*, *people-depart-front*, *people-trap*, *people-cover-camera*, *battery-voltage-low*}.

As feedback to the humans, Sage exhibits its mood state primarily through its facility for speech: the tone, pitch, volume and speed of its voice all change based on its current mood. Most importantly, the actual content changes as well. For instance, a lonely Sage, when accosted by a human impeding its progress, might say "*Hello. I'm glad you're here. Come around and I'll give you a tour.*" In contrast, a busy Sage in the midst of a tour presentation, when encountering a human obstacle, might sternly say "*Excuse me. I'm giving a tour right now. Please let me pass.*"

4 Performance and Failure statistics

The following statistics were compiled by analyzing at Sage's diary from May 22, 1998 through July 24, 1998. The numbers below offer a snapshot of the overall reliability of the robot.

Table 1: A snapshot of Sage's performance

Total number of days in the period:	63
Total number of totally error-free days	47
Total number of errors	20
Total number of days Sage was down all day	5
Total number of hours Sage is operational	1,392 h
Total number of hours Sage is in motion, giving tours	348
Mean time between failure, hours	69.6
Mean time between failure, after malloc bug discovery	144
Average linear distance traveled by Sage per error free day	1.3 km
Approximate distance covered by Sage for the period	72.5
Total number of collisions (obstacle avoidance failures)	0
Total number of navigation failures	0

The statistics in Table 1 will give the reader of sense of the overall performance of Sage. One can expect to go to the Museum once every third day to answer a robot page. More recently, that number has increased to almost one week. However, failures continue to occur, and it is especially instructive to see every single failure. At this time, the cause of every failure has been determined; there are no mystery failures. In Table 2 below, each failure is summarized. In italics, the solution, if applicable, is summarized

Table 2: Comprehensive list of Sage's failures

05/25	Guards turn the lights in Dinosaur Hall off before the robot docks for the night. *Lecture.*
05/28	Robot is herded into and stuck in a concavity. *Narrowed width of the virtual hallway.*
06/02	A 68HC11 sonar controller resets unexpectedly, crashing the robot controller thread.
06/04	Sonar controller resets again. *No solution. Just rebooted.*
06/05	Someone moves the CCD camera's focusing ring, eventually Sage stops. *Lexan cover.*
06/06	Sonar controller resets again.
	Robot halts mid-tour. Bug found from yesterday's coding and corrected.
06/07	Sonar controller resets again. *Adjusted voltage regulator from 5.0 to 5.25 volts and added huge capacitors on the sonar controllers' power lines.*
06/18	Sonar controller resets again. *New thread added that restarts controller process.*
06/20	Sonar initialization fails in mid-tour. *Cause unknown at the time*.
06/21	Relay board (controls power to laserdisc player and monitor) fails. *Remove relay board*.
06/23	Sonar initialization fails. *Bug found: it's the new thread that was added on 6/18. Fixed.*
06/28	Sage is discovered facing completely wrong direction. *Cause unknown.*
06/30	Relay board was reintroduced and fails again. *Changed relay board driver code.*
07/01	Sage faces totally wrong direction again. *Cause unknown. In an attempt to identify cause all bells and whistles are turned off or removed.*
07/03	Sage faces totally wrong direction again, but this time prints out garbage, too. *Code pored over. Several small coding bugs found and corrected. One is a memory leak.*
07/09	Sage faces wrong direction. Again. *More poring over of code.* ***Fantastically major malloc() coding error found that explains everything.***
07/18	Sonar initialization fails after sonar controller resets, during error recovery routine. *Bug found in the error recovery routine in the case when the framegrabber also fails.*
07/22	Sonar initialization fails after sonar controller resets, d uring error recovery routine. *Another bug found in the corrections to the error recovery routine. This time, tested!*
07/22	Break-beam sensor IR transmitter fails. *Replaced.*

Sage has only experienced 8 unique errors. In fact, almost 50% of *all* of Sage's errors are sonar initialization errors, with most of the rest being caused directly by the malloc() bug. The nature of these 8 unique error types is informative: two are the results of unexpected human actions (the guard turns the light off; the visitor defocuses the camera). Three are results of straightforward programming errors—traditional bugs.

The three remaining errors are stereotypical robotics errors: the obstacle avoidance code was insufficiently agile, landing the robot in a stuck position; the relay board turned out to be somewhat hard of hearing; the break-beam sensor on one of the wheels physically failed.

None of these 8 error conditions would have been mitigated by a planner, with the possible exception of the focusing ring incident. Sage had noticed that it was having a particularly hard time finding its pink landmarks. If it had re-planned with this new information in mind, it could have decided to take fewer risks by going at once to its docking station, plugging in, and paging us. However, the standard argument for mobile-robot re-planning, to employ new routes because of unforeseen obstacles, simply does not happen in the case of Sage, situated as it is in the controlled setting of a museum environment.

These conclusions would seem to be obvious given Sage's architecture and its domain. But consider that this robot is performing a reasonably complex task on a day-to-day basis, in a social setting where humans have not been reprogrammed to treat the robot with special care. Yet, the failures or situations that one may imagine as requiring high-level re-planning simply don't occur.

Some of these failures have software solutions at least theoretically. But the failures are sufficiently abstruse that it would have been unthinkable to have designed sufficient detail into Sage's model as to capture that failure and thereupon re-plan. For instance, in the case of the sonar board reset problem, we the programmers were not even aware that a sonar board reset would cause the robot controller thread to freeze—this data was not even in *our* models until after it occurred.

5 Conclusions

It is instructive to examine the failure list of a long-life mobile robot that is truly self-reliant. We believe this hints at the types of failures we can all expect in the future. Of course, few generalizations can be made from such data for a single robot in a simple environment; even less for a robot that has such an unvarying relationship to the world--recall that Sage's tour is canned.

However, our next step is to enable Sage to query visitors regarding their interests, then take them on a customized tour that fits their interests best. A question that we will soon need to ask is whether this added interactivity and variation will mean that Sage will finally have a planning module. The answer to this question is still unknown. Until then, it seems plausible that, at the low level of navigation and immutable tours, Sage has no need for a planner.