DP-MS-88-94

EXPERT SYSTEM CONTROL OF A SIX-LEGGED WALKING TELEROBOT

Kevin R. DeVries
RoboticsTechnology
Savannah River Laboratory
Aiken, South Carolina 29808

ABSTRACT

The Robotics Technology Group of the Savannah River Laboratory has implemented a three-stage expert system for a six-legged walking telerobot. Remote operation of this machine requires the knowledge of a highly skilled operator. Stability, size, and mode of operation considerations must take place continuously, in general, much more so than with a typical wheeled vehicle. The technology employed provides for quasi-real-time computer control, manual control, and an expert advisor -- all in the same package, which runs on the IBM PC/AT.

Concurrently with, though not tied to, the emergence of versatile, highly sophisticated telerobots has been the explosion of the "expert system" tool market for personal computers. Typically, these shells are intended for the capture of an "expert's" knowledge of a subject and are used under static circumstances; that is, the user sits at a keyboard responding to inquiries, as the software steps through the same "rules-of-thumb" that the original expert had used to arrive at his solutions and gives advice to that user accordingly. There is a direct correlation between this form of program flow and the control of telerobots by "expert" human operators. We have explored that overlap by employing such a shell as a combination controller/ advisor to achieve quasi-realtime control of a six-legged walking robot built by Odetics, Inc. Anaheim, CA. What has resulted is a tool that turns an exclusively teleoperated robot, which requires much practice and dexterity to operate via joysticks, into a system in which operator demands are minimized due to: 1) automation of "normal" or everyday types of tasks and subtasks (primarily navigation and cursoring/keyboard requirements); 2) detailed advice on operations that cannot be automated; 3) explanations of system errors; and 4) advice on general operation and maintenance. In other words, we turn a complex machine into a semiautonomous, user-friendly robot. Knowledge from experience can very effectively be described in the form of "rules-of- thumb" for operating a particular robot. Rather than a technical or database expert, what exists is an expert on allowable operational sequences. This is exactly what rule-based programming can provide: the ability to translate the operator's decision making process into coded, sequence-oriented heuristics.

Extensive testing in our laboratory is providing us with quantitative results of the benefits such a system can provide to teleoperation. The results from this testing are discussed.

INTRODUCTION

The system which the Robotics Technology Group of the Savannah River Laboratory (SRL) has designed includes what we consider to be the necessary enhancements for a truly useful operator interface for a mobile telerobot. Using a low cost, expert system programming environment , with real time code running external to it, we have built a utility which provides computer control, vehicle and situational diagnosis, expert operational advice, and, when needed, straight teleoperation. All of these features are very naturally tied together by taking advantage of the rule based programming features of Insight 2D (Level Five Research, Inc., Indiatlantic, FL). To program an equivalent system with conventional coding techniques would be significantly more complex and memory intensive. Because of the modularity of our software structure, the system's growth potential is limited only by the size of the hard disk (we are running it on an IBM PC/AT). As new tasks are programmed and new features designed, they may be added to the existing architecture in a building block fashion, the new having no effect on the old. The software architecture is tailored not only to utilize the features of Insight 2D (Insight) but to sidestep its shortfalls as well.

INTERSECTION OF EXPERT SYSTEMS AND TELEROBOTICS

The Robotics Technology group of SRL is primarily concerned with enhancing the capabilities of existing vehicles for remote work. We pursue emerging technologies which hold potential promise for application in the expert mobile robot arena [Byrd, et al. 1987]. Consequently, we took interest in the virtual flood of the Personal Computer market with inexpensive "expert system" shells which offered rule/frame based programming syntax, forward/backward chaining, depth/breadth first searching, etc., etc. . . . was there any common ground between these and mobile robots? A search through the maze of products revealed that nearly all of the low cost of these were designed for the capture of an expert's knowledge of a subject in a format which allowed for future regurgitation of it in a static setting. Static systems, by definition, cannot operate in the real-time realm. Since our primary concern was for operational control, there was no direct fit. However, on a higher level, that of the operator, there is a natural match, since an operator is himself an expert at using "his" telerobot. This knowledge can be captured quite painlessly in a rule based format, therefore, a window to the lower levels of real-time code would appear to lend high hopes to the "new" technology. It should be well understood that this is not an endorsement for the products we chose to employ, but an application study of the potential for the technology contained within them and the techniques which can take advantage of their unique features.

Our choice of software packages was a direct result of such justifications. We chose to implement our tests using Insight 2D as our expert system top layer. Within Insight, the option is available to branch to external programs (Turbo Pascal, Borland International, Scotts Valley, CA), and chain to multiple knowledge bases, sharing variables, flags, and parameters between all. Although there are other shells just as flexible, Insight is representative of the programming techniques we sought to investigate, and its cost and availability were both in its favor, as well. What remained to be shown was how much the package could be pushed to perform.

THE PLATFORM

The platform to which we applied the software was a six-legged walking robot built for the Savannah River Laboratory by Odetics, Inc. of Anaheim, CA. The Odetics walking machine has been nicknamed "ROBIN" due to its resemblance to a large ROBotic INsect (See Figure 1). This mobile robot uses six legs to orchestrate its walking gate, atop which is mounted a seven degree of freedom arm and work package. Walking vehicles may one day bridge the currently awkward gap between the "humanized" process facility layout and mobile robots. They offer the theoretical advantage of navigation within areas accessible to humans, yet difficult if not impossible for the wheeled vehicle to negotiate. Particulars in this category include ascending/descending stairs and platforms, and clearing clutter, such as pipes -- all of which ROBIN is capable.

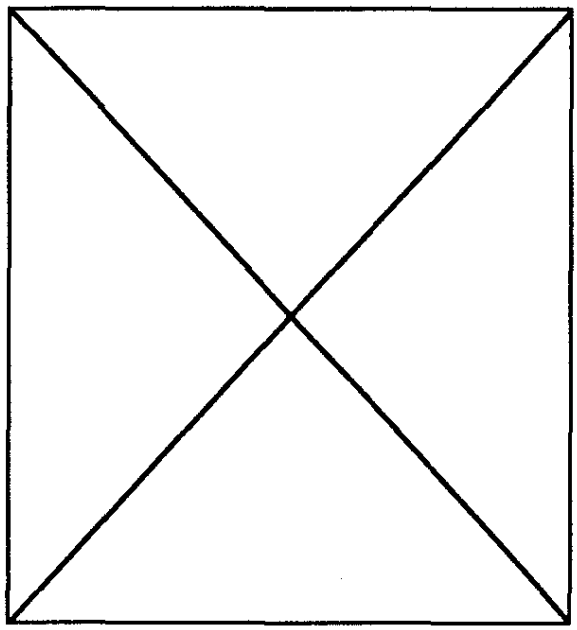

Figure 1. SRL Walking Robot.

ROBIN's walking stability is achieved by keeping at least three of the six feet in contact with the floor at all times and assuring that the center of gravity remains within the envelope of that supporting tripod. This is no trivial achievement, as a large portion of the nearly eight hundred pounds of metal, motors and electronics rests in the upper half of the body, not to mention the fact that the 5 1/2 foot telescoping arm can suspend 30 pounds at a point 6 1/2 feet from the actual center of the robot. Despite its top heavy stature, the walking algorithms do an extremely effective job of keeping the vertical tilt within the three degree maximum allowable error threshold.

Robin is tethered to an operator control station designed by Odetics and enhanced by SRL's E. E. Gillenwater. The console contains four external camera views, as well as views from any combination of two of the three onboard cameras. Also on the console, for manual operation, are three joysticks and one CRT on which multiple screens can be toggled between (See Figure 2). Manual control is executed by entering menu selections from either the transport (leg related) or the turret (arm related) menu screen and energizing the appropriate joystick. The obvious situation arises here, as it does with teleoperation, that where any work is to be performed through

ROBIN, it must not only be initiated by a human operator, but must necessarily consume 100% of his time and attention. This is adequate when the tasks being performed are sporadic, as in emergency response applications. If the tasks, however, are of the routine (ie. regular maintenance) variety, operator demand must be minimized, thereby freeing him to focus his attention on a larger number of duties, since he will be performing the same general tasks over and over. This inefficiency points directly to the operator interface. There is no need for onboard omniscience to achieve large operational time savings. Overcoming the bottlenecks, in general, does not require immense amounts of additional onboard computing power.

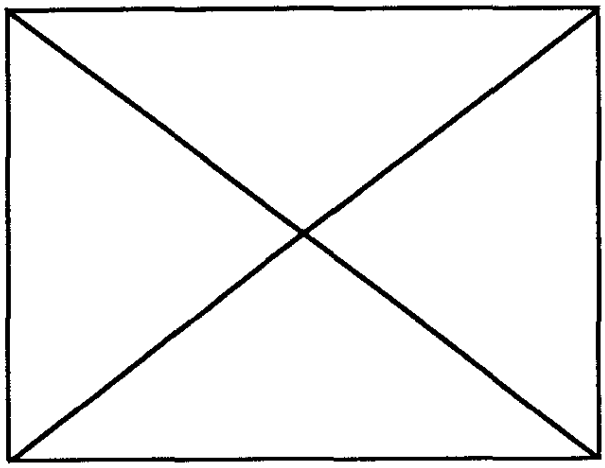

Figure 2. Operator control console.

NEED FOR IMPROVED INTERFACE -- MORE INTELLIGENCE

It is precisely this type of routine maintenance application, one which is an integral, normal piece of a remote process, toward which our developments are geared. Even complex operations requiring control by a human operator contain subtasks for which that operator's full attention is not necessary. Navigation is the most obvious. The only real "work" being done is more than likely at a specific location or a series of specific locations. In straight teleoperation, however, the operator is forced to get the robot to the scene of work, a task which is often as time consuming as the "work" itself, and yet there is realistically nothing to show for the operator's efforts. If then, the operator can be relieved from the burden of navigating the robot because the robot can navigate autonomously, a significant time savings is realized. Navigation is not the only arena where time savings are achievable with no loss in robot functionality. Eliminate enough of these, and the operator will be called upon for only the most dexterous of operations, rendering his energy significantly more focused. Furthermore, it becomes conceivable to introduce the concept of one operator controlling multiple telerobots simultaneously, because the commands he instigates are no longer handed down from a joystick, but from a set of high level commands necessitating only intermittent hands on control.

SOFT ARCHITECTURE

Insight "programs" are called knowledge bases. They are written in "Production Rule Language" (PRL), Level Five's name for their rule based programming syntax. Rules are written in the traditional IF..AND..OR.. THEN..ELSE format, and, once compiled, follow a primarily backward chaining, goal driven inference pattern. Straightforward development of small scale knowledge bases allows for rapid prototyping of both tasks and logic. Insight offers the ability to chain between these knowledge bases providing, therefore, the ability to retain modularity and to allow continuous growth as new capabilities are developed. When one knowledge base chains to another, for instance a subset of knowledge, any parameters and facts desired to be shared between the two are passed. Given the nature of the rule structure, however, it is very awkward to return to the original knowledge base. This is the primary reason for the final overall architecture which the system has taken on. It proved far cleaner to make the flow of knowledge unidirectional between levels of knowledge bases.

Each level, as shown in Figure 3, has a distinct function which is performed on every task initiated. Level 1 is the selection menu. Its sole purpose is simply that -- to give the operator the choice of things which he may initiate, be it physical action on the robot, manual operation, diagnostic information retrieval, or operational advice. The selection which he makes determines to which level 2, task specific module the system will chain. These level 2 modules are the workhorse knowledge bases of the system. Each of these perform the operations requested. It is this level which has the potential to enlarge merely by tacking on additional modules as they are developed. The success or lack thereof achieved in level 2, chains the system into either of a pair of knowledge bases in level 3. If all was performed to completion, the robot having no external disturbances or interruptions, it flows into a knowledge base which sends the operator a report of current status and exactly what was last performed. If, however, complications occurred, a knowledge base is entered which attempts to explain what was initiated, what went wrong, at what point it failed, and what the current state of the machine is. Note from Figure 3 that a common file of shared facts and parameter values is used by all levels. Program flow is indicated by the dark arrows.

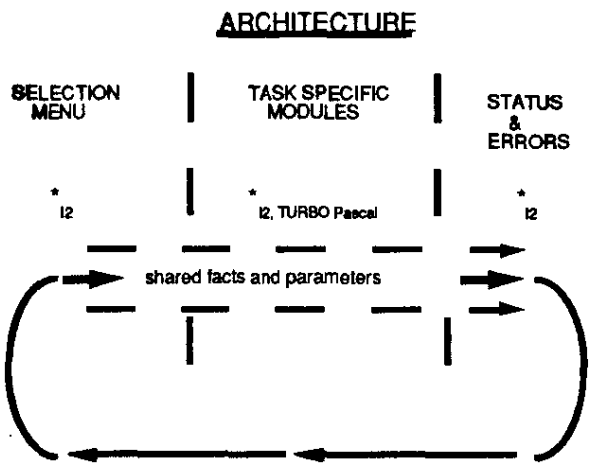

Figure 3. Software Architecture.

TURBO PASCAL -- THE BOTTOM LAYER

As mentioned earlier, we knew from the onset that the control portion of the software would have to reside external to the Insight knowledge base. What, in fact, the Insight rules of thumb are geared to duplicate is the logic which an operator has aquired from experience. When a physical action is instigated by a rule, that action is driven by a program written in Turbo Pascal (Turbo). The Turbo, in turn is sending out appropriate ASCII commands over RS232 to the console, which has been placed in "Remote Mode". In this mode, the console acts to translate the ASCII commands into the same format that joystick data is in when sent to alter ROBIN. The console also acts as a window to all critical data available on the current state of ROBIN which is condensed into frames of hex ASCII format. This data is used by the Turbo routines for decision basing. Also used by this low level of control code are data from a laser scanner (discussed later) and facts/parameters sent from Insight. Within a Turbo program, additional facts and parameters can be given values and returned to be used in the Insight knowledge bases. They are passed via files, though memory was an option, as well. This is about all that will be mentioned in reference to the Turbo level of software, other than to note that a good deal of effort was necessary to raise the level of the Odetics ASCII commands into high level procedures which do all of the necessary checking transparently so that any programmer can use them to build quick applications.

FLEXIBLE PROTOTYPING BY TESTABILITY

As mentioned, we are using the inference flow of PRL in an effort to duplicate the operational sequence knowledge of an experienced operator. We have, in fact, captured about three man years of experience into rules and advice. As an illustration of the advantage this programming syntax provides when rapid prototyping is involved, a typical rule is constructed to perform the docking operation within the workspace. The programmer simply writes in sentences the steps to perform a dock as follows:

```
RULE  to perform successful dock
IF    Read current x, y, and turret orientation
AND   Determine turret sweep destination
AND   Read radial distance and angle from Lasernet
AND   Compute the transport's current error
AND   Rotate the transport to eliminate the error
AND   Reset x, y, and orientation angle
AND   CHAIN status
THEN  Docked successfully

RULE  to sense a dock failure
IF    Dock could not be completed
AND   CHAIN errors
THEN  Errors while docking
```

In an actual Level 2 module, these two rules perform all of the necessary logic needed to dock. In development, the programmer stops with only rules which deal with the logic flow, none that branch to Turbo routines. He may then "play back" the knowledge base through the normal, static interactive session with his newly created knowledge base and check to assure that his rules will act in the order and reasoning that he wants. Once this is established (it is a fast session!), he then puts in the conditions required to fire each of the needed facts. Take the third condition from above, "Read radial distance and angle from Lasernet", as an example. In order to fire, there are some things which must interactively take place between the computer and ROBIN. Also, if something hangs that operation, in this case it could only be a bad exchange of data, then the program should chain to the "errors" knowledge base of level 3. This condition is proved in the following manner:

```
RULE  radials
IF    FORGET bad exchange
AND   ACTIVATE radials.com
DISK  param.i2
SEND  sweep destination
```

RETURN	radial distance
RETURN	radial angle
RETURN	bad exchange
AND NOT	bad exchange
THEN	Read the radial distance and angle from Lasernet

Note that the file "param.i2" is the means by which the variables, in this case, three numbers and one simple fact (true/false) are passed between Insight and Turbo. If "bad exchange" returns false from Turbo, then the condition "Read the radial distance and angle from Lasernet" fires true, and the next condition is executed. Successful proving of all of the conditions in the main rule cause a jump to the "status" knowledge base of level 3 and the program continues as described. This is a simple example, but it covers the basics of building a rapid prototype and program flow.

LABORATORY SETUP AND TESTING

Figure 4 depicts the general setup of the laboratory as well as the concepts used to carry out our testing. As drawn, the darker lines represent legal path segments and are connected by nodes, denoted by the letters A through K. The robot uses this type of map to navigate within its process building (in our case, the laboratory). ROBIN does, however, accumulate a significant amount of dead reckoning error during navigation as a function of body height, umbilical tension, distance traveled, and arm payload. These can be reasonably predicted, and this predicted error is tracked in the software. At various spaces in the floor plan are the areas which contain the bulk of the work to be performed. It is within these "Laser defined workspaces" that ROBIN performs the work which requires the most accuracy and multiple tool use. On the tool rack shown in figure 4 are various tools designed by E.E. Gillenwater which are adapted to the gripper. Also hung on the tool rack is an inexpensive laser scanner manufactured by Namco Controls of Mentor, OH. The beam from this scanner spans 90 degrees with a range of about 20 feet, achieved by reading the reflected beam from a 4"x4" passive retroreflector mounted on ROBIN's turret. Our workspace is actually about 10 x 10 feet. Anywhere within this workspace, ROBIN's controls can recalibrate and eliminate the accumulated deadreckoning error; therefore, any path is legal if its origination and destination are within the scanner's view. The operator is stationed behind his console which is out of sight of the workspace.

From the operator's standpoint, operation of ROBIN has become far less attention intensive. He now gives task level orders, such as "Find valve 2" and watches through the cameras until called upon by the computer to intervene. ROBIN proceeds to get first to the workspace using autonomous navigation techniques developed at

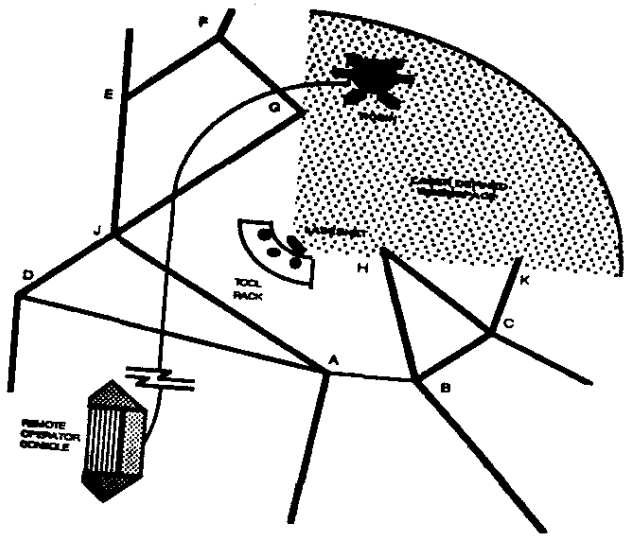

Figure 4. Laboratory layout.

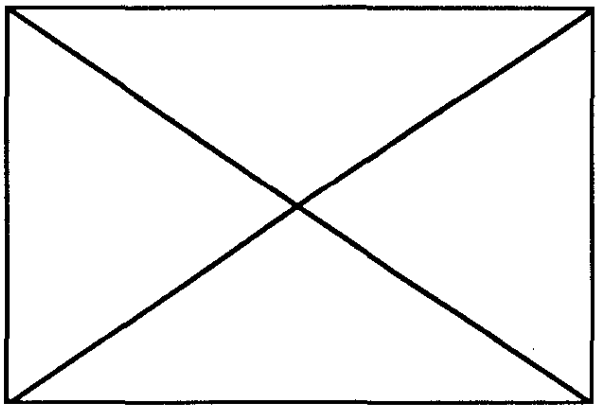

Figure 5. ROBIN performing a mock spill clean up within laserdefined work area. Right bottom is corner of tool rack.

SRL by J. J. Fisher, then enters the workspace, corrects its error, proceeds to the proper coordinates, points its wrist camera at the valve and awaits further instruction or "tweaking" by the operator. If used as a teleoperated machine, the operator is required to use the joysticks and cameras to first get the machine to the destination, navigating with the aid of camera views only, and move, servo by servo, the arm, to position it on valve 2. Several simulations have been used to compare the advantages and disadvantages of our system as compared to straight teleoperation. They include filter change out, emergency I-beam support, pipe repair, spill clean up (see Figure 5), object transport, and various types of valvework. All include

significant navigation and negotiation of stairs, doors, platforms, and grating. Under the computer control, the time elapsed when completing an operation is, in general, comparable to that of manual control in navigation speed, and more efficient (fewer wasted motions) with turret motions involved with real work tasks executed within the laser defined workspace; and yet the operator is actually free to focus his attention on any other work he desires. In addition, he may get any information available on his telerobot at any time by merely requesting it from the PC.

CONCLUSIONS

That which was explored in our work was an intersection between the low end of the line, PC based, static expert system shells, and the technology of teleoperation. Not only does the package offer the cost advantages of inexpensive software and hardware, but it buys efficient programming techniques, system modularity, low memory requirements, and an effective environment for rapid prototyping as well. The shell used in our work has limitations which forced us to use some awkward overall architecture constructs, however, these limitations are being overcome weekly by the constant barrage of new products with more flexible features. The expert system coding techniques are promising for the field of mobile telerobotics in the future and even extremely useful for the present. They offer manpower savings from the standpoint of operator endurance as well as the attention required of him without suffering the output of remote work achievable by the telerobot.

ACKNOWLEDGEMENT

The author acknowledges the outstanding work of E.E. "Dick" Gillenwater for the design of the tooling and the enhanced control console and for the many hours of dedicated operation of the equipment during the laboratory evaluation tests. The information contained in this paper was developed during the course of work under Contract No. DE-AC09-76SR00001 with the U. S. Department of Energy.

REFERENCES

Byrd, J. S. 1988 (June 19-23, Portland, OR) Mobile Robotics R&D at the Savannah River Laboratory. *Annual Conference American Society of Engineering Education.*

Byrd, J. S., Fisher, J. J., DeVries, K. R. and Martin, T. P. 1987 (Aug. 31-Sept. 2, Snowbird, UT) Expert Robots in Nuclear Plants. *Proceedings Topical Meeting on Artificial Intelligence and Other Innovative Computer Applications*, American Nuclear Society.