The Scarabs RoboCup 2002 Rescue Robot Team

K. Bird, J. Fonda-Bonardi, R. Fooroghi, I Forte, D. Goodwin, J. Griffin-Roosth, V. Griffin, H. Liebowitz, M. Pettengil, N. Phillips, M. Randall

New Roads School
17213 Palisades Circle
Pacific Palisades, CA 90272
mr_mr@earthlink.net

Abstract

The Scarabs Team robot (Ringo) was designed and built by a team of high school students on a limited budget. The goal was to keep the design simple yet easy for a single human operator to control. The team uses a custom joystick interface to control robot movement using a PC. A Linux-based camera provides video to the operator and passes serial commands from the PC to the robot.

The goals of the Scarabs Team are: to build viable robots at minimal cost; to learn about math, computer science, electronic engineering, physics, artificial intelligence, system integration, international relations, character development, and teamwork; to have fun (!); and to make a positive difference.

Introduction

The Scarabs team is composed of high school students from the Los Angeles area. The goals of the Scarabs Team are: to build viable robots at minimal cost; to learn about math, computer science, electronic engineering, physics, artificial intelligence, system integration, international relations, character development, and teamwork; to have fun (!); and to make a positive difference.

When we first heard of the RoboCup competition to build soccer-playing robots, we decided that would be a good project to work on. As high school students, we have relatively little experience with sophisticated vision systems or autonomous control of robots. Initially, we built a single robot with an omnidirectional vision system. The user interface did not allow a lot of real-time control because it was meant to be replaced by an autonomous interface when we figured out how to do one. We would also have to build three additional robots. And we would have to figure out the best way to build a kicker.

We took that robot to RoboCup / AAAI in Seattle last year. A lot of people were favorably impressed by its speed. Several of them encouraged us to enter the robot in rescue competitions. In addition to solving current, real-world problems, the rescue competition only requires a single robot and does not require that the robot function autonomously. This made the rescue competition more accessible than the RoboCup mid-league soccer competition to us.

We modified the robot to function better in a rescue environment, added an operator interface, and prepared it for competition.

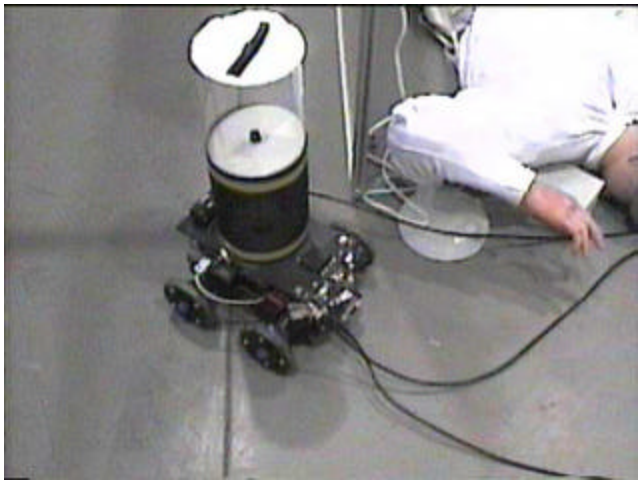
Robot Design

Our current robot entry, Ringo, is built around the chassis and motors of a radio-controlled race car. (Nikko Hercules). The car is a four-wheel drive vehicle with a flat chassis protruding beyond the wheels in the front and back. Such a chassis is fine for an event such as RoboCup where the wheels do not interfere with a kicker or other special devices which may be added to the robot. It is less suited for rescue events because it does not function well on uneven terrain. However, we chose to proceed with it due to budget constraints.

The first modification was to replace the radio control with a Parallax BASIC Stamp 2 (BS2) microcontroller. The BS2 was chosen for motor operation because of its low-cost, ease of programming, and proven reliability. The BS2 controls the left and right wheel motors via two R/C electronic speed controllers. The BS2 allowed a user to command the robot by entering single-character serial commands.

The robot was not sufficiently powerful to climb over obstacles, so we decreased the wheel diameter by approximately 1 in., and increased the battery voltage for the motors from 7.2 to 8.4 volts. (This is the maximum voltage the electronic speed controllers are designed to handle.)

Various methods were tried to give the wheels better traction. Because we were under severe budget constraints, we coated the wheels with hot glue and with rubber used to coat tool handles.



Cable control of the robot was chosen to avoid real-world problems of interference / loss of signal. When cable is used, it can become difficult to prevent it from tangling, especially if it's coiled. A common solution is to keep the cable as a figure-8, but this can be difficult to pre-stage and to transport. As a simple solution, we wrapped the cable around a piece of 3 in. PVC pipe about 2 ft. long using the pipe as a "kitestick," switching ends. The result is a figure-8 on the stick.

Vision System

For RoboCup, it is advantageous for the robot to always see what is happening in all directions. The original vision system was a small, lightweight camera connected to the PC using analog video. To achieve omnidirectional video input, the camera was mounted on top of the robot. A custom-made PET plastic cylinder, 6 in. diameter, and about 2 ft. tall was mounted around the camera. A convex mirror was placed at the top of the cylinder. This arrangement allowed the system to “see” in all directions although the image is very distorted. Still, it is a very cost-effective solution to the problem of omnidirectional vision.

Later, we requested and received a donation of several Axis 2120 web cameras. These cameras are much more accurate and flexible than the original camera, but they are also considerably bigger and heavier. This made the robot much more likely to tip over.

Software Interface

The Axis camera is based on Linux. It delivers video to the PC via Ethernet. It is also possible to establish a telnet session from the PC. A command is provided to send user-specified robot control data from a serial port on the camera.

When we started thinking about how to control the robot’s movement remotely, some of the team members were joking about driving it like an RC car. It didn’t really take long for the joking to become serious. We purchased a standard USB-interface joystick for the PC and implemented a “tank mixing” algorithm to convert the joystick into BS2 pulse widths for each of the motors. The tank mixing is a natural control method, immediately understandable to any user. Pushing the joystick forward or backward moves the robot forward or backward proportionally. Pushing it left or right turns it.

The camera includes an interface to place the data in an ActiveX control which can be put into a Visual Basic program.

The joystick control is in two parts. Visual Studio 6.0 puts the joystick interface into the multi-media routines. These routines are not directly available to the ActiveX interfaces. Therefore, we had to implement the “raw” joystick functions in a DLL which is then called from an ActiveX control. The joystick control displays the stick in an x/y square. The Z input can be used to scale the maximum value of the square. This permits the operator to set the robot to move more slowly in complicated areas. All buttons on the joystick are exposed. Buttons, exceptions, and significant changes in the joystick position cause events which can be captured externally. The joystick control can initiate a telnet connection to cause the camera to send serial data to the BS2.

One of the complications was that the BS2 microcontroller has to generate a pulse-width-modulated (PWM) signal every 20 msec. to control the motors. Because the BS2 cannot process interrupts or do multi-tasking, command inputs from the operator must be received in the ~18 msec interval between pulses. (Each pulse is 1-2 msec wide). At 9600 baud, it can process at most about 14 characters of data across the serial input in any 20 msec cycle.

The PC has to update its display of the camera image and any joystick input. This can require up to 200 msec, depending on the PC speed. The joystick interface includes the msec between its interrupts. The BS2 protocol includes the number of pulses to send between PC inputs. This permits both systems to retain appropriate frame times.

We noted several other shortcomings for the BS2 in this application:

- it does not accept input from the PC other than using its own development environment or mixed with output;
- it cannot do any floating-point math (therefore, the PC had to provide any data in a form simple enough for the BS2 to use almost directly);
- it appears to use BIOS features unique to "real" serial ports. (We were unable to get the BS2's development environment to recognize any connection using a USB serial interface. This means that any computer controlling the current version of the robot must have a serial port, a USB port, and an Ethernet connection. Only the oldest, slowest laptop owned by a team member actually has a serial port. As a result, the video display has a noticeable lag compared to the faster desktop on which development was done. The joystick also had to be slowed down to accommodate window display time);
- the timing issue prohibits use of output.

Results in Edmonton

At Edmonton, we were unable to communicate from the camera to the BS2 microcontroller. We also noticed that one motor had stopped working. We saw that a wire had come loose in the interface cable between the camera and the BS2. Even after replacing that cable with a manufactured, tested, cable, we were still unable to get that essential level of communication working.

We had a spare BS2 processor available. Replacing it caused the second motor to start working again but did not fix the communications problem. It appears likely that something was damaged in the handshaking process, possibly by the broken wire. This meant that the robot could not move under computer control.

Because the robot did not actually run, we also didn't get a chance to see how well our stick-based cable handling method works in practice. We believe we can feed the tether directly off the stick. If that's not possible, the stick can still be used to create a transportable figure-8.

Conclusions

We have learned a lot about robot design and building. This robot has some strong points in its vision system and human interface.

Perhaps the chassis could be modified to move the camera down into the robot body, exposing only the lens at the top. This would require a wholesale redesign because there isn't a large enough empty space to hold the camera.

A computerized vision system can be taught to recognize objects correcting for quantifiable distortion. While it is possible for a human operator to learn to deal with the distortion, it can never be easy or automatic.

In practice, it was much easier to drive the robot using a forward-facing camera. Pan and zoom kits are available for the camera. It might also be possible to mount additional cameras to the robot or to modify the display to reduce distortion for a human operator. (A forward-facing camera will also make the robot more stable).

It will be useful to provide video and snapshots to disk for later review. Of course the information would have to include location data and time.

The robot platform needs to be replaced with a system designed to work in a search and rescue environment. It needs to be able to deal with uneven terrain.

The on-board processor will be replaced with something more powerful, able to handle multiple processes, including input from additional sensors, and to assist a human operator in evaluating a disaster scene. (These sensors could include sound, temperature, inertial guidance, etc.)

The joystick control will be upgraded with an automatic mapping feature.

The robot also needs a light, possibly controlled by a joystick button.

Our goal is to create a simple, cost-effective robot with some level of autonomy. This robot would work with a human operator, displaying video of its surroundings and values of any sensory data. Eventually, some autonomy in movement may be added to enable a single human operator to control multiple robots.