1999

The XRCL Project: The University of Arkansas' Entry into the AAAI 1999 Mobile Robot Competition

Doug S. Blank
Bryn Mawr College, dblank@brynmawr.edu

Jared H. Hudson

Brian C. Mashburn

Eric A. Roberts

The XRCL Project: The University of Arkansas' Entry into the AAAI 1999 Mobile Robot Competition

Douglas S. Blank, Jared H. Hudson, Brian C. Mashburn, and Eric A. Roberts
University of Arkansas
Department of Computer Science and Computer Engineering
Engineering Hall Room 313
Fayetteville, Arkansas 72701
{dblank,jhhudso,bmashbu,eroberts}@comp.uark.edu

Abstract

This paper describes a new, open sources robot architecture initiative called the Extensible Robot Control Language (XRCL). This system is designed to provide a modern robot-programming interface which is adaptable and easy to learn. We describe a fuzzy logic control system built on this framework, and examine the system at work in two tasks at the AAAI 1999 Mobile Robot Competition.

Introduction

In the last few years we have seen the advent of a new market: sophisticated, relatively low-cost, commercial robots. Many smaller research groups that don't have the expertise, facilities, or budgets to build their own robots can enter the field of robotics by purchasing commercially built ones. As a result, there are now more laboratories capable of robotics research than there were just a few years ago.

However, a problem has arisen—each vendor has developed its own software architecture, and no two systems work with one another. This has created a robotics Tower of Babel.

Many vendors claim that it would be impossible for the situation to be any other way. They say that in addition to the obvious incompatibilities involved in controlling very different kinds of hardware, no single architecture could accomodate the needs of all researchers. For example, one researcher might need a formal planning system. Another might need a learning system. The development of a robot architecture, the vendors argue, is exactly what "doing research in robotics" is all about.

We, however, believe that there is much to be gained by there being some means for researchers to cooperate across platforms. Instead of creating an architecture which incorporates a researcher's specific control paradigm (such as neural networks or fuzzy logic), one could build an infrastructure from which these "architectures" could hang. To this end, we propose an open-sources Extensible Robot Control Language (XRCL), and discuss a specific implementation for it.

Extensible Robot Control Language

Our goal of creating a control language for robots immediately brought to mind XML, the Extensible Markup Language (Bray, Paoli, & Sperberg-McQueen 1998). XML is a description language that allows the representation of structure by the use of novel *tags*. XML is sure to be the successor of the Hypertext Markup Language (HTML), the current language of the World Wide Web. In addition to representing a hierarchy of structure, one may also represent attributes to the XML structural elements by including *attribute* = *value* pairs inside the tag, as shown in Figure 1.

```
<controller>
   <robot type="B21R">elektro</robot>
</controller>
```

Figure 1: Sample XML document. In this example, a controller is defined which is composed of a robot named *elektro*, which is of type "B21R".

We chose XML as a starting place for the XRCL (pronounced 'zircle') project for three main reasons as follows:

- it has emerged as a standard method of representing structure
- it provides many support tools (e.g., parsers, editors)
- it has become popular and well-known to programmers and non-programmers alike

Having identified a method of representing basic structure, we then examined the issue of representing control (i.e., program code). The Web has settled on a document organization that allows code (in the form of Java or Javascript) to be interspersed with HTML's tags. We decided to follow this standard. For our current implementation, we picked C++ to be our programming language; however, it would be easy to incorporate other languages, such as Java or Lisp. An example XRCL document can be seen in Figure 2.

```
<controller>
<robot type="B21R">elektro</robot>
<behavior name="Goto">
  <arg default_value="0.0"
       type="double">my_x</arg>
  <arg default_value="0.0"
       type="double">my_y</arg>
  <arg default_value="0.15"
       type="double">my_close</arg>
  <update>
    static int done = 0;
    if (! done) {
      xcSay("Here I go!");
      done = 1;
    }
    Fuzzy euclidean(0.0, 1.0);
    // degrees on left:
    Fuzzy left(5, 20);
    // degrees on right:
    Fuzzy right(-5, -20);
    double phi  = xcAngle(my_x, my_y);
    double dist = xcDistance(my_x, my_y);
    Fuzzy too_left = right >> phi;
    Fuzzy too_right = left >> phi;
    Fuzzy near_goal = euclidean << dist;

    // The fuzzy rules:

    IF too_left THEN Turn -.30;
    IF too_right THEN Turn .30;
    IF ! (too_left || too_right)
       THEN Speed xcSLOW*1.1;
    IF (near_goal || too_left
        || too_right)
       THEN Speed xcSTOP;

    TurnEffect(!near_goal);
    SpeedEffect( 1.0 );
    // within my_close cm
    if (dist < my_close) {
      xcPrint("\n[I'm there!]");
      xcSay("I am now at the
             specified location.");
      xcReturn();
    }
  </update>
</behavior>
</controller>
```

Figure 2: Example XRCL robot controller. This example shows a behavior-based program designed to allow a robot to go to a specific coordinate.

To test XRCL as an infrastructure upon which to hang architectures, we implemented a fuzzy logic, behavior-based control system based on work by (Saffiotti, Ruspini, & Konolige 1993) and (Konolige 1999).

A Behavior-based, Fuzzy Logic Controller

Although there is much work being done in *behavior-based robotics*, researchers have used the term in various ways. Currently this research falls into two main camps: vertical behavior-based, and horizontal behavior-based. The vertical behavior-based camp is represented by, and in fact defined by, (Brooks 1985). In systems of this type, behaviors are typically independent black boxes that are arranged into a hierarchy. Each element, or component, receives inputs, does some simple processing, and provides an output. Arranged in a proper manner, this hierarchy of independent elements can give rise to a fluid and flexible controller.

The horizontal behavior-based camp is also composed of independent components; however, this research aligns the elements at the same level, rather than in a hierarchy. The focus of such control systems is to *blend* all of components' outputs so that the controller acts as a single coherent, centralized system.

We have chosen to use the horizontal behavior-based paradigm, modelling our system after the fuzzy logic control architecture Saphira (Konolige 1999). We chose this variation for two main reasons:

- the control logic can be succinctly represented by IF-THEN syntax
- the resulting robot behavior is very often a nice blending of conflicting rules

In XRCL, we represent fuzzy rules in the the *behavior* sections of a controller. These behaviors are small nuggets of control code written to accomplish specific tasks, and are designed to be used in conjunction with other (possibly competing) behaviors.

Each behavior is composed of general C++ code, and a set of fuzzy rules. Each fuzzy rule takes the form *IF [fuzzyvalue] THEN [affecter] [amount]*. *[fuzzyvalue]* is a fuzzy real-valued truth value in the range 0 to 1. Currently, we have two controls: speed and turn. *[amount]* is the magnitude we wish this rule to effect the affecter, if the rule is true. Using fuzzy logic, the less true the rule is, the less it will have an effect.

Behaviors execute simultaneously (being implemented via native threads).[1] Each behavior sends a requested action to the underlying fuzzy logic system. The fuzzy logic system then computes each behavior's effect on its associated motor, and the robot adjusts its movement accordingly.

At any one time, any number of behaviors may be active in the system, with each trying to control the effectors. Figure 3 shows a graphical depiction of three

[1] Currently, we have XRCL and the fuzzy logic controller running on a dual-processor PC running Linux. We have used pthreads to implement the parallel components.

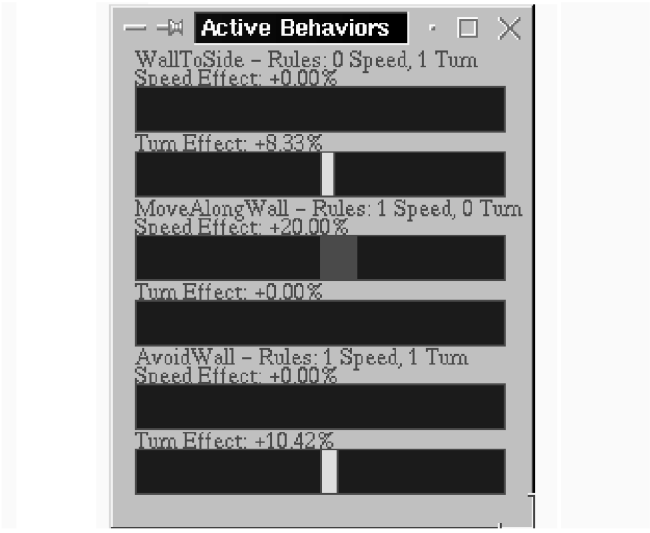

Figure 3: A graphical view of competing behaviors in the X Windows System. This window shows the active behaviors and their desired effects. The top two horizontal bars reflect the WallToSide behaviors' speed and turn effect, respectively from top to bottom. The remaining bars reflect two other competing behaviors' computed fuzzy desires for rotational and translational movements.

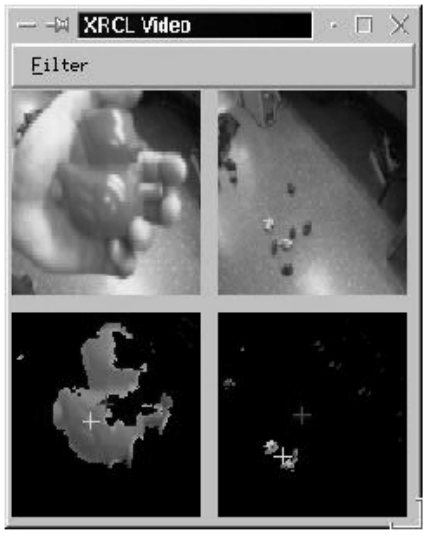

Figure 4: A bird in the hand. The top images show the live view from a robot's left and right cameras. The bottom images show the associated blobs after running the image through a set of blue duck filters. Crosses in the bottom images mark the computed centers of mass of the blobs.

behaviors competing to control the turn and speed of a robot. The actually movement made at each time step is actually a blending of all of the behaviors' desires.

In our fuzzy controller system, behaviors have the ability to get sensor readings, activate and deactivate other behaviors, call other behaviors as functions with parameters, and communicate with other behaviors through a built-in message passing system.

Although we have used XRCL to implement a fuzzy logic controller, XRCL is, of course, not tied to just this mode of operation. Other paradigms could easily be expressed in the system. For the AAAI 1999 Mobile Robot Competition, we also built two other subsystems: a blob object system, and a neural network object system. These are briefly described below.

Blob object system

The blob object system was created to provide an easy-to-access interface to video data. A *blob* is a two-dimensional array of black and white pixels produced via a set of filters through which we run the robot's live video data. The white section of a blob indicates areas which we are interested, and black indicates background. A filter can be as simple as an exact color matching filter, or as complex as a non-linear neurally-trained face detector. Figure 4 shows the interface for creating a color matching filter in XRCL.

Once a filter has been created, one can access the associated blob anywhere in XRCL by referring to specific blob fields. For example, if one has a set of filters which detects blue ducks, then one can develop a fuzzy logic rule based on *BlueduckFilter-Evidence* or *BlueduckFilter-CenterOfMass*.

Neural network object system

As an additional test of XRCL's flexibility, we implemented a back-propagation-based neural network object system (Rumelhart, Hinton, & Williams 1986). This provides an interface to a powerful learning system (Blank 1999). Although we only used this system in the AAAI competition for speech recognition, it could also be used as a filter for the blob system, or as data for fuzzy logic rules.

XRCL GUI Interface

To visualize and evaluate controllers in XRCL, we needed a graphical display. To that end, we developed a interface with multiple display modes, including textual, ncurses, and X Windows System. Figure 5 shows a simulated robot and representations of its sensors running in the X Windows System mode.

The graphical user interfaces also provide access to simulated robots, which alleviates the need for all work to be performed on actual robotics hardware.

The Competition

We decided to test XRCL by having it compete in two of the AAAI 1999 Mobile Robot Competition tasks: the *hors d'oeuvres* and the scavenger hunt competitions.

The *hors d'oeuvres* competition was designed to allow real people (AI researchers and their families) to interact with state-of-the-practice robotic waiters in the

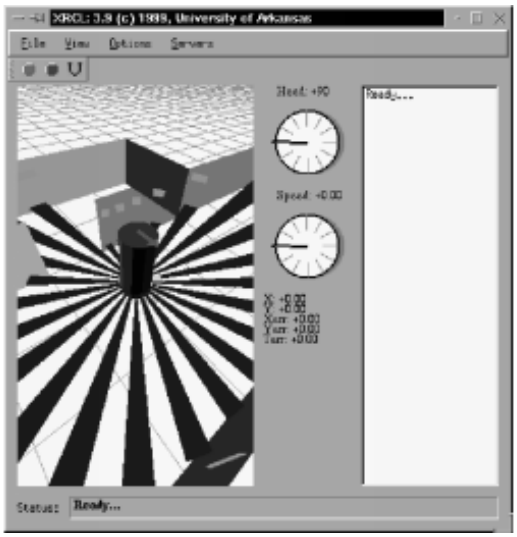

Figure 5: The X Windows System interface has a 3D graphics view of the robot that displays readings from sensors, including sonar, laser and tactile. The 3D graphics are created using our own rendering system based the Qt widget set.

Figure 6: Elektro the robot. Elektro is a Real World Interface, Inc. B21 robot. It has sonar, laser, infrared, and tactile sensors, as well as a high-performance stereo vision system. Here it is shown without its candy dispenser, which sits directly behind the cameras.

wild. The competitions were not held in any special area, but rather in places where people were able (in fact, encouraged) to wander. This marked a radical departure in the competition's design from previous years where contests were confined to specific robot-only areas. This year's goals were:

- to serve food to a general group of people with no special knowledge on methods of interaction
- to refill food tray when necessary
- to cover a large area
- to not hurt anyone too badly[2]

This last item was actually a controversial, yet necessary, addition to this year's competition. In years past, a robot could get completely surrounded by people, and had no recourse but to stay in its trapped spot. This often prevented a robot from refilling its tray as it could never get out of the crowd of people. This year, the judges added this "bump clause", which allowed a robot to gently nudge its way back to the service station.

The second event we competed in was the Scavenger hunt competition, which was also held in open spaces. The robots' goals were to find as many of a set of objects as possible in a short amount of time. For example, many toys and balls where spread throughout the conference center and all of the robots were sent out to find and identify them.

[2]These are no doubt the "rules of robotics" Asimov would have given, were he a caterer.

Elektro

For the competition, we used a B21 robot from Real World Interface, Inc., named Elektro (see Figure 6). Elektro is equipped with 4 main sensor modalities: sonar, infrared, laser, and tactile. It also has a stereo vision system that is comprised of dual high-resolution cameras, mounted on a 180-degree pan and a 45 degree tilt unit. This pan-tilt unit is mounted to a pedestal on the top of the robot. Images are gathered through two Matrox Meteor PCI frame grabbers.

Elektro's "brain" is an on-board, standard PC: a dual 400MHz Pentium II computer system with 128M RAM, 9G Hard drive space, and 8 port serial card. The computer also has standard peripherals, such as a sound card and speakers. In addition, we utilized an HC11-based HandyBoard micro-controller (Martin 1999), attached via a serial cable, which controlled small motors and lights.

The Plans

In devising a plan for serving *hors d'oeuvres*, we decided to focus on the interaction between human and robot. Given the open nature of the competition, this gave a us a huge niche to explore. To exploit fully all modes of interaction between humans and Elektro, we decided on a plan to dispense *M&M* candies based on a proposed neural network-based voice recognition system. After some initial discussion, we switched from *M&M*'s to *Reese's Pieces* as the latter come in only three colors.

Because of the sequential nature of our task, we created several behaviors to control the movements of the robot. A state machine was used to load behaviors in and out of the XRCL fuzzy logic system. In this way, we

would have only the necessary behaviors running and competing for any particular state. We then scripted a set of states to implement the following sequence of operations.

Upon startup, Elektro would search for a person to whom it could deliver candy. This was accomplished using behaviors based on sonar and laser sensors to find legs. The robot would then switch to an approach behavior, stopping in front of the (suspected) person. The robot would then change to a "ServePerson" behavior. Upon introducing himself, Elektro would ask which of the three colors of candy the person would like. A neural network speech recognition system recorded the input from the microphone, propagated the preprocessed data through its simulated neurons, and produced a best-guess of the spoken color.

The perceived choice activated signals sent via a serial connection to a HandyBoard micro-controller which controlled the dispensing hardware. Depending on the perceived color of candy requested, a series of motor controls would grab candy from the appropriate bin of candy affixed to the top of Elektro, and drop them down a tube leading to a cup hanging from the front of the robot.

The night before the competition, we also added the ability for Elektro to automatically update the neural network's weights used in the speech recognizer. The operation worked as fllows. Once the candy was delivered, Elektro would ask the person if the correct candy color was given. If Elektro was correct, the person was requested to press a large blue button on Elektro's front. If the correct color was not identified, the person was to do nothing. Pressing the button would cause the weights to be adjusted using the last voice input as the learning sample. In this manner, Elektro was capable of adapting its voice recognition ability to the specific acoustics in the environment during the competition.

Once the entire interaction between robot and human was complete, Elektro would thank the person, swap behaviors, and began the sequence over again by looking for legs in another part of the room.

Elektro would be relatively aggressive in its attempts to travel a fair distance from its current position. This was done to prevent getting trapped by a circle of people, as mentioned above. Over time, Elektro was capable of wandering around a large portion of the room.

The *Reese's Pieces* dispenser was built to deliver small quantities of candy at a time. The dispenser was built with a variety of common materials (i.e., PVC pipe and wood) and could provide feedback to the robot when it was low on a particular color of candy.

The dispenser was composed of two motor/gearbox assemblies, two discs with appropriately sized delivery holes, three tube containers for storing the candy, three photocells, two potentiometers, one housing box, one HandyBoard with serial board, and one delivery tube.

The motor/gearbox assemblies were mounted inside the housing with their drive-shafts mounted vertically. The discs were placed on the drive shaft so that they would turn like a record on a an old-fashioned record player. A potentiometer was mounted above the center of the disc and connected such that when the disc turned, the knob of the potentiometer would also turn. The variable resistance of the potentiometer would be fed into the HandyBoard to provide feedback on the position of the disc with respect to the pickup and delivery holes.

Containers were mounted on the edge of the disc so that when the hole in the disc would go under the tube, a small amount of candy would fall into the hole. As the disc continued to turn, the hole in the disc would line up with a hole in the bottom of the housing and the candy would fall into the delivery tube and out to the cup on the front of the robot.

Using photocells, we measured the light level at the bottom of the containers when empty. If enough light entered the container, the dispenser would notify XRCL that the amount of a particular color of candy was running low. We could then remove this color choice from the color options and notify the robot handler that it needed refilling.

The scavenger hunt competition required far less specific design, but still required that several behaviors be written in XRCL to control the robot. The actions required to solve the problem were two-fold. First, the robot was to move about the playing field in search of scavenger hunt objects. Secondly, the robot was to identify these items according to shape and color.

Approximately a dozen objects were photographed prior to the competition. Simple color-matching blob filters of these objects were then created and stored to files. During the competition, when a filter produced a large blob, Elektro would identify the object as the object of the filter. Some color manipulations were done to enhanced Elektro's ability to correctly identify a color in the changing lighting conditions of the conference hall.

Results

During the competition, we noticed that many of the other teams had used commercial voice recognition systems, such as IBM's ViaVoice. The night before the competition, we were struggling with getting our system to learn and considered switching. However, later that night we finally got our neural network-based speech recognition system to correctly perceive our spoken color choices, and we decided to go with it.

The neural network used was a simple recurrent network, or SRN (Elman 1990). For each word spoken, we normalized the auditory data into 1500 floating-point values between 0 and 1. We then fed those 1500 values into a SRN, 25 values at a time. This created a sequence of length 60. Although we didn't have much time to train it, we decided to let it train during the competition via reinforcement learning. We believed that it might start out performing poorly, but as the competition wore on, it would actually perform better. Although we didn't keep statistics on its performance

during the competition, the judges found it to indeed perform poorly early in the competition, and better as the night wore on.

Although we experienced communication problems between our host computer and Elektro, our solution for the scavenger hunt competition yielded a 3rd place finish. The judges awarded our team a technical innovation award for our adaptive voice recognition system in the *hors d'oeuvres* competition.

Summary

The 1999 AAAI Mobile Robot Competition provided an excellent testbed with which to try out our implementation of an Extendible Robot Control Language (XRCL). Although we wrote the entire system in the three months prior to the competition, we were able to successfully apply it to two tasks: the *hors d'oeuvres*, and scavenger hunt competitions.

The competition provided a unique challenge for us to test our general architecture in the wild. Although the experience was quite exhausting, it was a great learning experience for all involved. We look forward to future competitions.

Acknowledgements

The authors wish to thank the members of the Artificial Intelligence and Robotics Laboratory, especially Chris Craig, Genet Cramlet, Nick Farrer, Jim Gage, Dylan House, and Vikki Kowalski for their help on this project.

References

Blank, D. 1999. *Con-x Users' Manual, on-line at http://dangermouse.uark.edu/cx/.*

Bray, T.; Paoli, J.; and Sperberg-McQueen, C. 1998. *Extensible Markup Language (XML) 1.0 on-line at http://www.w3.org/TR/1998/REC-xml-19980210.*

Brooks, R. A. 1985. A layered intelligent control system for a mobile robot. In *Third International Symposium of Robotics Research*, 1–8.

Elman, J. 1990. Finding structure in time. *Connection Science* 14:179–212.

Hofstadter, D., and the Fluid Analogies Research Group. 1995. *Fluid concepts and creative analogies.* New York, NY: Basic Books.

Konolige, K. 1999. *Saphira Users' Manual.*

Martin, F. 1999. *The HandyBoard User's Manual.*

Rumelhart, D.; Hinton, G.; and Williams, R. 1986. Learning internal representations by error propagation. In McClellend, J., and Rumelhart, D., eds., *Parallel Distributed Processing: Explorations in the microstructure of cognition.* Bradford Books/MIT Press.

Saffiotti, A.; Ruspini, E.; and Konolige, K. 1993. Blending reactivity and goal-directedness in a fuzzy controller. In *Second IEEE International Conference on Fuzzy Systems*, 134–39.