Programming Robotic Assistants
Extended Abstract

Bruce MacDonald, Geoff Biggs, Toby Collett
Department of Electrical and Computer Engineering, University of Auckland
Email: {*b.macdonald, g.biggs, t.collett*} *at auckland.ac.nz*

March 15, 2005

Context

Robot programming systems have not kept pace with developments in general purpose programming environments. Robot researchers face difficulties developing medium to large software systems for robots that are to assist humans in everyday human environments. Robot systems present special demands and as a result much of the software infrastructure is proprietary, much is necessarily targeted at specific hardware, robot software development kits may be limiting, there is a lack of open standards to promote collaboration, code reuse and integration, and there is a lack of techniques for bringing the human in to the robot's programming infrastructure.

Problem

Robot systems have special demands related to the complex interactions robots have in real environments, and the complex sensors and actuators that robots use, including:

- A large number of devices for input, output and storage, which far exceed human programmers' familiar senses and effectors, compared to the few devices in a desktop or server.
- Simultaneous and unrelated activity on many inputs and outputs.
- Real time requirements, as the automation system must operate in the real world.
- Unexpected real world conditions.
- Wide variations in hardware and interfaces, as opposed to the highly commoditized desktop.

Programmers of robot arms and other complex articulated automatic devices, must also deal with non–intuitive geometry. Programmers of mobile robots must deal with widely varying and unpredictable conditions as the robot moves through its environment. Standard debugging tools give programmers access only to program data. This makes debugging robot programs difficult because program data is at best an indirect representation of the robot and environment.

Forces

Robots are increasingly used in service roles, where the robot's environment and tasks are more varied and unpredictable, requiring more complex programming. At the same time robot sensors and controllers provide for more sophisticated decision making, adding to the robot capability but further increasing the demands on human programmers and the programming environment. Advances in human–robot interaction promise improved communication between robots and humans, and these advances should also be capable of improving the programming process.

Solution

We view robot programming systems as having three important conceptual components that are of interest to their designers:

- The programming component, including designs for programming language/s, libraries and application programming interfaces (APIs), which enable a programmer to describe desired robot behaviour.
- The underlying infrastructure including designs for architectures that support and execute robot behaviour descriptions, especially in distributed environments.
- The design of interactive systems that allow the human programmer to interact with the programming component, to create, modify and examine programs and system resources, both statically and during execution. The human programmer may also interact with the infrastructure component, to examine, monitor and configure resources, and directly with robots as they perform tasks.

There are other components that are not of particular concern to designers of robot programming systems, such as the robots themselves, operating systems, compilers, robot hardware drivers and so on. A few aspects, such as real time operating system performance, will be of concern.

Programming component Some of our previous work includes problem solving and instructable systems for teaching robots [1, 2, 3, 4, 5, 6, 7]. For example, a robot must model and reason about the human programmer's intentions, and be able to recognise plans presented by the human [2]. Robot programming systems can be combined with direct human–robot interaction, where the programmer provides a text skeleton of the program and details are completed interaction during the robot's initial executions of the task [4]. Geoff Biggs' work [8] is providing better programming language tools, and his paper recently submitted to IROS05 describes the use of dimensional analysis specially tailored to robotic programming applications.

Underlying infrastructure Evan Woo's [9] three layer CORBA based, service broker application architecture provides a distributed programming infrastructure, including tests on robots ranging from those with only a single microcontroller (LEGO Mindstorm, Khepera) to those with a considerably more sophisticated platform (B21r). Oscar Kuo's work [10] extends this to add real-time aspects using real-time CORBA and adds a services layer and other classes to make programming easier. Barry Hsieh has studied the difficulties of reusing robot software in a university environment and proposes a system architecture and simple templates for CORBA use by new postgraduates (recently submitted to IROS05).

Interactive systems for programming Félix Trépanier's [11] Graphical Simulation and Visualisation (GSV) tool aims to help humans visualise robot behaviours operating in different environments. It is integrated in a broader robot programming environment supported by the service based architecture mentioned above. The GSV tool can display any robot model controlled by a robot behaviour using a state-of-the-art game engine to render the virtual world thus giving an accurate 3D perspective of the behaviour of the robots in the virtual environment. Toby Collett's work is providing an augmented reality tool to aid programmers, giving them the capability to interact more directly with the robot's view of the world during programming (recently submitted to IROS05).

References

[1] Rosanna Heise and Bruce A. MacDonald. Robot program construction from examples. In *Proc. National Irish AI Conf.*, Dublin, Ireland, September 1989. Also in book form, edited

by A. F. Smeaton and G. McDermott (Eds.), AI and Cognitive Sciences '89, Springer–Verlag, 1990, pp 254–271.

[2] John D. Lewis and Bruce A. MacDonald. Machine learning under felicity conditions: exploiting pedagogical behavior. In *Proceedings of AI / CS '92*, Limerick, Ireland, September 1992. Also presented at the AAAI Workshop on Constraining Learning with Prior Knowledge, July 1992, San Jose, CA.

[3] Bruce A. MacDonald and David Pauli. Adaptive robot training by programming and guiding. *Journal of Intelligent Manufacturing Systems*, 4:385–404, 1993.

[4] Bruce A. MacDonald. Instructable systems. *Knowledge Acquisition*, 3:381–420, December 1991.

[5] Bruce A. MacDonald and Jacky Baltes. Learning, planning and understanding human instructions. In *Proceedings of the Machine Learning Workshop at AI/GI/VI'94*, pages vii–1–vii–10, Banff, Alberta, Canada, May 1994. Proceedings available as Research Report No. 94/539/08 from the Department of Computer Science, University of Calgary, 2500 University Drive NW, Calgary, Alberta, Canada, T2N 1N4.

[6] Natascha O. Schüler and Bruce A. MacDonald. Learning repetition in string transformations. In *Proceedings of the Tenth Canadian Artificial Intelligence conference*, pages 39–46. Canadian Society for the Computational Studies of Intelligence, May 1994.

[7] Jacky Baltes and Bruce A. MacDonald. A distributed architecture for an instructable problem solver. In *Proceedings of the 27th Hawaii International Conference on Systems Sciences*, pages 63–72, Wailea, Maui, 4–7 January 1994. Emerging Paradigms for Intelligent Systems minitrack in the Decision Support and Knowledge–based Systems Track.

[8] Geoffrey Biggs and Bruce MacDonald. A survey of robot programming systems. In *Proceedings of the Australasian Conference on Robotics and Automation*, CSIRO, Brisbane, Australia, December 1–3 2003.

[9] Evan Woo, Bruce A. MacDonald, and Félix Trépanier. Distributed mobile robot application infrastructure. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 1475–80, Las Vegas, October 2003.

[10] Yuan hsin (Oscar) Kuo and Bruce MacDonald. A distributed real-time software framework for robotic applications. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'05)*, Barcelona, 18–22 April 2005. To be presented.

[11] Félix-Étienne Trépanier and Bruce A. MacDonald. Graphical simulation and visualisation tool for a distributed robot programming environment. In *Proceedings of the Australasian Conference on Robotics and Automation*, CSIRO, Brisbane, Australia, December 1–3 2003.