Robotics Education using Embedded Systems and Simulations

Thomas Bräunl

The University of Western Australia *and* Technical University München
EECE, CIIPS, 35 Stirling Hwy. M018, Crawley WA6009, Australia
RCS, EI, Arcisstr. 21, 80333 München, Germany
Thomas.Braunl@IEEE.org

Abstract

We have taught a number of robotics courses at Australian and German universities, both on manipulator kinematics and on intelligent mobile robots. In all cases has the inclusion of lab components with hands-on embedded robotics systems and/or simulation systems proven to be very successful. Through this, students have gained a better understanding and achieved a higher retention of the subject material and have also been better motivated during the course.

Teaching Philosophy

We have experienced the differences in university teaching systems, both as a student and as a lecturer in the US, Australia, and Germany. Courses in Germany usually are not accompanied by a lab component in the same term. They are lectures with tutorials only, while some courses (or the combination of some courses) offer a optional lab-only course in the following semester. Contrary to this, most courses in Australia do have a concurrent lab component.

Both systems have advantages and disadvantages. If labs are delivered in the following semester, students will have covered all required course material and will have a broad overview of the subject area. On the down side, some students may have already forgotten some of the material, or even worse, as not every course offers a lab and students only have to enroll in a certain number of lab courses, only a fraction of the students taking a particular course will also enroll in the corresponding lab course. So only a minority of students will be exposed to the practical side of the subject area.

The system that combines lectures and labs in a single course presents a challenge for the lecturer, because he or she has to deliver the material at the right pace, so topics necessary for the labs will have been covered in time during the lectures. However, this system has been found to be very beneficial to the students, as all theoretical material is being backed up with practical experience in a near time frame.

Lab components for a course in Computer Engineering, or more specific in Robotics, usually do require some level of programming, both low-level and high-level. We use assembly and gnu C/C++ cross compilers. Low-level programming is done on the EyeBot embedded systems, while high level programming can be done either on the real robots (SoccerBots, driven by EyeBot controllers) or the simulation system, using the same application program. Students have typically taken a first year programming course on C before enrolling in either the Embedded Systems course or the Robotics and Automation course. However, we do provide optional weekly tutorial sessions throughout the semester for helping students to brush up on their programming skills. We consider the ability to program in Assembly and C/C++ to be a significant factor in a future graduate's employability. This fact should go without mentioning, but unfortunately is not recognized in every EE department.

Mobile Robot Systems

We use different mobile robot systems for the courses in "Embedded Systems" and "Robotics and Automation", reflecting their different levels. For the 2nd year course on Embedded Systems we use SoccerBot mobile robots, driven by EyeBot embedded controllers. For the final year course on "Robotics and Automation" we use Pioneer AT outdoor mobile robots driven by notebook computers.

SoccerBot Mobile Robot and EyeBot Embedded Controller

The EyeBot embedded controller, available from InroSoft, is based on a 400 MHz Intel X-Scale PXA255 (ARM) microprocessor with 16MB RAM, 4 DC motor drivers, 14 servo ports, 2 serial ports, 2 digital camera interfaces for stereo imaging, 2 USB slaves, 1 USB host, Bluetooth, LAN (extensible for WLAN), and on-board programmable FGPA, allowing to conduct on-board image processing on the robots. The controller further has a color graphics LCD and touch panel for user input, i.e. a complete user

Copyright © 2006, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

interface that significantly facilitates program development and debugging when compared to other controllers.
The controller runs the Linux operating system with our own RoBIOS extension, which comprises a monitor program, multi-tasking system, and a comprehensive system library with, among others, driver routines for various sensors and actuators.
The SoccerBot mobile robot is also available from InroSoft and measures only 16cm×10cm. It has a differential drive system using two Faulhaber precision motors with encapsulated gearboxes and encapsulated encoders. These not only allow precision driving, but also make labs possible for experimenting with feedback controllers, such as PID control.
Each SoccerBot is further equipped with a digital color camera and three Sharp infrared PSD (position sensitive device) sensors. This extremely rich sensor equipment in combination with the powerful embedded controller, both way above the average small mobile robot used in a lab environment, allows to conduct a virtually unlimited number of experiments in intelligent robotics.

Figure 1: SoccerBot mobile robot and EyeBot controller

Pioneer AT Mobile Robot

The Pioneer AT is one of the more high-end products from ActivMedia. It was chosen for a final year robotics course, because we wanted students to work in an outdoor scenario that in many ways is more realistic than indoor settings. Also, outdoor mobile robotics is closer to current developments in the automotive industry, such as driver-assistance systems [Maurer, Stiller 2005].
To control each robot, we use a standard notebook computer running Windows XP that is linked to the Pioneer's onboard controller via a serial interface (RS232). We further use a USB GPS and a USB web cam as the main sensors for navigation. Unfortunately, the Pioneer ATs are not equipped with sonar sensor as default and upgrading is quite expensive, so we are running them without these sensors.
The first valuable lesson the students learn when working with the robots is that all technical systems do have some problems that would never occur in a simulation or a mere software-based task. This may be frustrating at times, i.e. when the GPS again refuses to log on or the camera initialization fails, but it will develop their problem solving skills that will be invaluable when working in industry after graduation.

Figure 2: Students on the cricket oval with a Pioneer AT

Simulation Systems

We have developed a number of simulation systems over the years at Universität Stuttgart, Germany, and The University of Western Australia. We are using these simulation systems at various stages in our courses on "Embedded Systems" and "Robotics and Automation".

Hardware Simulation System Retro

Retro is a simulation system at register transfer level and allows the construction of complete CPUs from simple components [Bräunl 2000]. Students can construct a working CPU and execute their own machine programs on it. Retro greatly facilitates the understanding of the inner working of a CPU and thereby closes the gap between understanding of electronics at the transistor or gate level and understanding of software development.

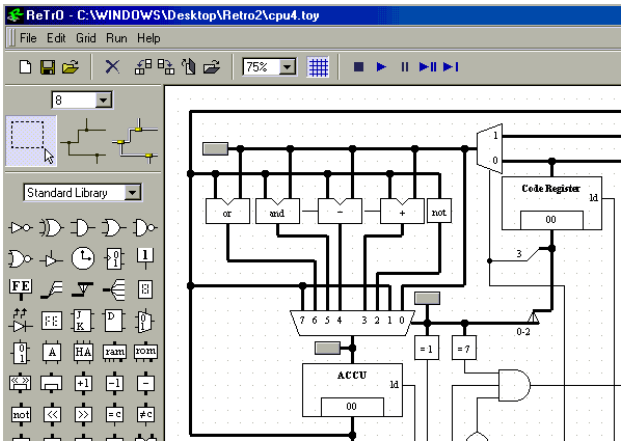

Figure 3: Sample CPU design in Retro

Retro provides a standard library of components, such as adders, multipliers, combinatorial logic gates, multiplexers and decoders, registers (banks of flip-flops) and memory modules (RAM and ROM). Programs and data can be placed at a memory location and saved as a file. Program execution can be step-by-step or continuous at various execution speeds.
Students have found this tool of great value. By actually building their own CPU (in simulation) and being able to run it with data, they achieve the best possible learning outcome. The simulation helps not only to understand the concept and inner working of a CPU completely, it also improves the understanding of any synchronous technical system.

Manipulator Simulation System RoboSim

RoboSim is only a demonstration implementation of a 6dof (degree of freedom) manipulator in Java that we provide as a standard reference to students. This simulator provides slide rulers to change manipulator position and orientation either in joint mode (forward kinematics) or coordinate mode (inverse kinematics).

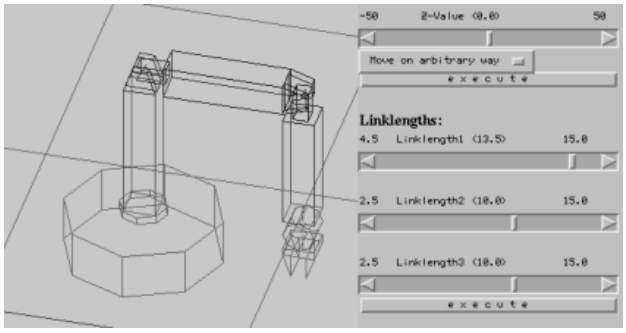

Figure 4: Manipulator simulation with RoboSim

One of the lab assignments students have to complete in the Robotics course is to implement their own simulation system of a simple 3-link manipulator. This combines application of the kinematics equations with graphical user interface programming, so students will gain essential skills in both areas.
By actually having to implement a robot software system, students will get a much deeper understanding of kinematics and control problems than by the traditional assignments that require the solving of kinematics equations. Having to write manipulator simulation software, or even better described as manipulator modeling software, students receive a very similar learning experience to using a real manipulator, but at a near zero cost.

Mobile Robot Simulation System EyeSim

EyeSim is a 2½ D driving simulator for multiple mobile robots with numerous sensors, including vision [Bräunl, Koestler, Waggershauser 2006]. EyeSim allows a robot application program to run in a "perfect world", thus allowing a student to test his or her algorithm, or to run in a much more realistic error setting that can be used for testing an application program's robustness [Koestler, Bräunl 2004].
Multiple robots can interact with multiple objects and each other in a common environment. Each robot can make use of multitasking, while each robot's application program is executed in parallel to all other robots' programs and the global environment update.
Care has been taken to re-implement the real robot operating system RoBIOS [Bräunl 2006] for the EyeSim simulation system. This required to re-implement every single RoBIOS system function, but has the invaluable advantage that robot application programs can be transferred from the real robot to the simulator and back without the need to change a single line of source code. All sensors of a SoccerBot robot, odometry, infrared distance sensors and the camera image, are being recreated in the simulation and fed back to the robot application program.

Figure 5: EyeSim mobile robot driving simulator

EyeSim allows to specify individual error profiles. These include on the actuator side Gaussian errors for driving functions (vehicle translation and rotation commands) and

on the sensor side Gaussian errors for the PSD distance sensors (position sensitive devices), various error models for the simulated robot camera (salt&pepper noise, 100s&1000s noise, Gaussian noise), and specific error models for the wireless communication between robots (message corruption and message loss).
EyeSim has become a central tool, not only for teaching, but also for research. Especially applications that require numerous repetitive experiments, such as training of neural networks or evolution with genetic algorithms can benefit from a realistic simulation system. Instead of having to repeat time-consuming experiments over and over with real robots and all the associated problems (placement accuracy, battery depletion, etc.), many of these experiments can now be executed on the EyeSim simulator in the first instance, and much faster than in real time.

AUV Simulation System SubSim

While EyeSim required only a minimum of physics calculations, SubSim contains a full rigid body physics subsystem, augmented by water (buoyancy) and propeller equations for simulating autonomous underwater vehicles (AUVs) [Boeing, Bräunl 2005], [Bräunl, Boeing, Gonzalez, Koestler 2006].
SubSim started as a research project sponsored by an industrial partner with the intention to make the system available to the public domain and use it for an AUV competition track, alongside a competition for real AUVs in the Australasian region (see [Bräunl 1999] for robot competitions in general). The simulator could then be used by groups that either do not have the funding (e.g. high-schools) for building their own AUV or traveling to Australia, or by groups that have not yet progressed far enough with their real AUV development.

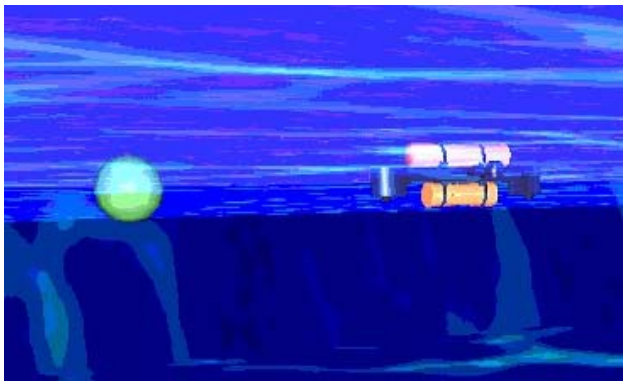

Figure 6: SubSim full physics simulator

A very basic and generic, low-level user interface has been created as the basis for SubSim. This allows access to all actuators and sensors and can in turn be used to implement high-level interfaces such as RoBIOS. We have in fact implemented our SubSim-RoBIOS interface on this low-level interface and interested parties can design their own operating system on this basis. AUV application programs can be written either using the low-level interface or the RoBIOS (or other, if available) high-level interface.
Care has also been taken to make the simulation system as extensible as possible. Therefore, it is possible to create one's own AUVs, both as a detailed structural description in XML, and as a graphics file for rendering purposes. It is further possible to add one's own actuators and sensors, and finally it is possible to replace the complete physics engine by a different package. So far we have tested and written interfaces/plug-ins for Dynamechs, Novodex, ODE, and Newton.

Summary

We have presented our experience on teaching methods for the subject of intelligent robotics. The inclusion of labs with a combination of real embedded systems and simulation systems has been found to be extremely valuable as it not only improved the students' understanding of the subject material and its practical applicability, it also gave them significantly higher motivation and satisfaction in completing their assignments.
Although having to schedule labs concurrently with lectures can be a challenge for lecturers in terms of presentation order as well as for timetabling, the benefits clearly outweigh the additional effort required.
Further details and downloads are available from our website http://robotics.ee.uwa.edu.au .

References

Boeing, A., Bräunl, T. 2005. *SubSim: An autonomous underwater vehicle simulation package*, International Symposium on Autonomous Minirobots for Research and Edutainment, AMiRE 2005, Fukui, Japan, pp. 33-38 (6).

Bräunl, T., Boeing, A., Gonzalez, L., Koestler, A., Nguyen, M. 2006. *Design, Modeling and Simulation of an Autonomous Underwater Vehicle*, International Journal of Vehicle Autonomous Systems (IJVAS), to appear Oct. 2006.

T. Bräunl, Koestler, A., Waggershauser A. 2006. *Fault-Tolerant Robot Programming through Simulation with Realistic Sensor Models*, International Journal of Advanced Robotic Systems (ARS), June 2006, vol. 3, no. 2, pp. 99-106 (8).

Bräunl, T. 2006. *Embedded Robotics - Mobile Robot Design and Applications with Embedded Systems*, 2nd Ed., pp. (XIV, 458), Heidelberg Berlin: Springer.

Bräunl, T. 2000, *Register-Transfer Level Simulation*, Proc. of the Eighth Intl. Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, MASCOTS 2000, San Francisco CA, Aug./Sep. 2000, pp. 392–396 (5)

Bräunl T. 1999. *Research Relevance of Mobile Robot Competitions*, IEEE Robotics and Automation Magazine, vol. 6, no. 4, Dec. 1999, pp. 32–37 (6).

Koestler, A., Bräunl, T. 2004 *Mobile Robot Simulation with Realistic Error Models*, International Conference on Autonomous Robots and Agents, ICARA 2004, Palmerston North, New Zealand, pp. 46-51 (6).

Maurer, M., Stiller, C. (Eds.) 2005, *3. Workshop Fahrerassistenzsysteme*, Walting, April 2005.