Learning Computer Science Through Robotics

Maria Gini
University of Minnesota

The main purpose of this paper is to describe how we are integrating in our undergraduate curriculum a variety of projects in robotics and describe, in particular, an undergraduate project that culminated at the robot competition at the International Joint Conference on Artificial Intelligence in August 1995. This is part of a large effort aimed at exposing undergraduate students to a variety of projects in robotics, computer vision, and 3D modeling. We have chosen these topics as the sources of projects because of their interdisciplinary nature and because they provide a wide variety of problems where system integration, communication, and cooperation are important. This effort is intended to:

- motivate the study of advanced applied mathematics by demonstrating its importance to solving real world problems;
- teach students how to build complete systems (as opposed to write isolated programs;
- allow them to experiment with the object oriented programming paradigms they learn in class for a variety of complex problems;
- expose students to application areas (Virtual Reality, 3D modeling for manufacturing, graphical interfaces, real time operating systems, etc.) where new opportunities for employment or product development might exist;
- provide hands-on experience with distributed systems and with fundamental issues in communication and real-time control;
- familiarize the students with hardware interfaces, low level input/output, device drivers, and basic electronics, with hands-on experience;
- show students how to develop their own independent projects;
- foster development of leadership skills among students in project teams.

Robotics, including computer vision, graphics, and 3D modeling have been selected as application areas because of their interdisciplinary nature and because they provide a wide variety of problems where system integration, communication, and cooperation are important. We have had positive experience over the years with a number of undergraduates working in Robotics and we have observed how it becomes much easier for them to assimilate their course knowledge around projects.

Sample Short Projects

Here we describe briefly a few of the short projects we have used in some of our courses.

Object-Oriented Programming: In the introductory courses in Algorithms and Data Structures students learn the fundamentals of programming. In this project, a limited set of primitive procedures is used to control the movements of a Lego robot. The project exposes students to the simple protocols involved in verifying that a task was performed as programmed, and about handling error conditions. Various data structures need to be used to save information about the environment, such as the locations of objects. The hardware used is a set of Lego Technic with a hardware interface from Hyperbots that plugs into the serial port of a workstation or PC. After experimenting with the simple control interface directly from the Scheme language to the hardware (available from the MIT Scheme software repository) students developed a similar program in C++, so learning about a different style of object-oriented programming.

Building Device Drivers: Students in the course Structure and Programming of Software Systems learn machine architecture issues such as timing, interrupt handling, and I/O system programming. Their knowledge is then put to work to design device drivers for physical devices, such as serial ports used to connect external devices such as a GPS system, a conveyor belt, a rotary table, and other computer-controlled devices.

Cooperative Control of Equipment: The purpose of this project is for students to develop and test software that controls multiple pieces of equipment. In Introduction to Operating Systems students study process scheduling and interprocess communication. Problems, such as writing software for interprocess communication among the microprocessor boards used in our mini-robots, expose students to the subtleties of communication and synchronization.

Object Recognition For Pick And Place Tasks: In the course Computer Vision, students learn about basic image processing, 2-D region processing, object recognition systems, techniques for the derivation of shape from motion, texture, and stereo, and the different representations of solids and surfaces. As part of this course, students use cameras to obtain images of objects and have to recognize different objects and determine their pose with respect to the camera, so that a robot can be commanded to grasp them.

Sample Large Projects

We believe that students greatly benefit from the ability to complete large projects. At the same time, we recognize that most large projects require skills that students acquire in several courses and far more time than can be justified within the framework of existing courses. To address this need, we allow students to work on long term projects that span over multiple quarters. Students typically propose a project to a faculty advisor, and work with the advisor to develop a plan for completing the project over the course of three or more quarters. Some students receive support for this type of work from the University as part of the Undergraduate Research Opportunities Program. Theses grants require students to submit a formal proposal, and are awarded competitively. Here are some projects we have used.

Cooperation and Collaboration: this project allowed students to apply material learned in data structures, systems software, and operating systems. The complete project is to control a pair of robotic arms, along with associated sensors, to keep a set of objects flowing evenly on a pair of conveyor belts. The goal is to detect the objects and control both arms to prevent the objects from falling off of the conveyor belt. During the early stages of the project, students wrote software to control the arm for simple tasks such as pick up a stationary object, put it down on a conveyor belt, turn on the conveyor belt. Students then worked on the problems of controlling a pair of robots. By completing this project, students gained experience with designing and implementing device driver routines, building object-oriented abstractions, integrating real-time inputs into an application, programming with real-time constraints, and interprocess communication. The equipment used consists of a pair of Scorbot-Er VII robot arms with 5 degrees of freedom, a linear conveyor, a rotary table, and a slide base for one of the robots.

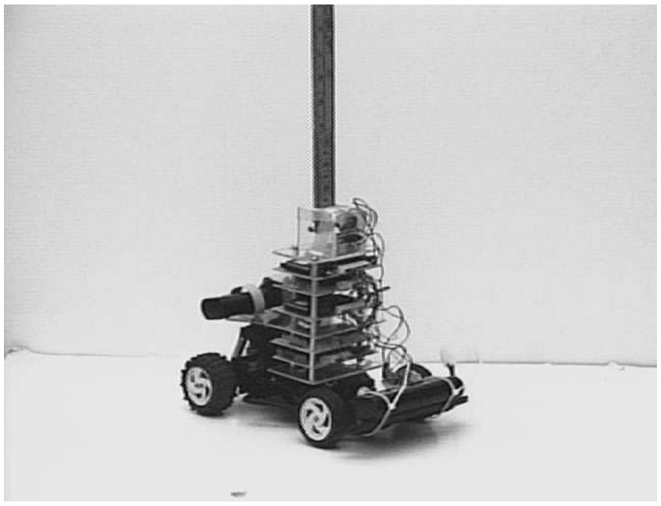
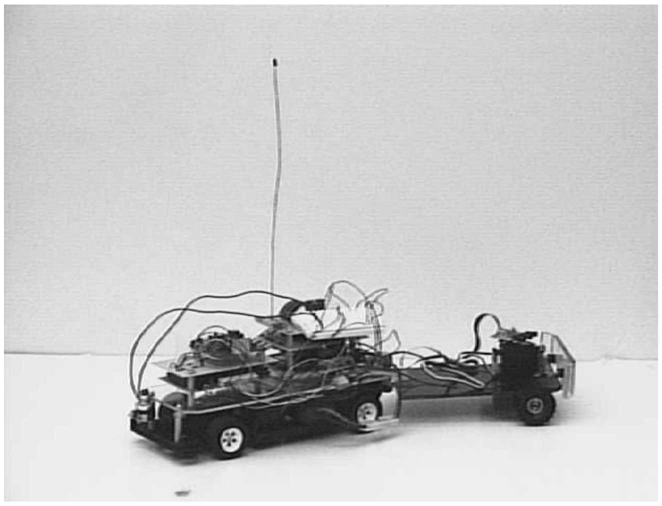

Figure 1: PBMin, the pole balancing mini-robot (on the left) and TBMin, the trailer backing mini-robot (on the right)

Learning to Balance a Pole: this project, that involved building a minirobot, focused on learning the classical control task on balancing a pole [4]. Since this is an advanced research project, it was done by a team including a graduate student in addition to two undergraduates. Pole-balancing grew out of a classic dynamics problem (see e.g. [2]) and is a well-studied control-learning problem. The system must learn to keep a long rod or pole, hinged to a cart and free to fall in a plane, in a roughly vertical orientation by applying forces to the cart. If the pole passes a certain value from the vertical (defined to be 12° by most authors), or the cart goes beyond a certain fixed distance from its starting point, a failure signal is generated. The robot we built, PBMin (Pole Balancing Mini-robot), shown in Figure 1 uses two sensor inputs, the pole angle and the car position. The pole-angle sensor is a variable resistor attached to a common yard stick. The pole is limited to move in a plane. The car position is sensed using a rudimentary position sensor that uses two infrared sensors placed slightly less than half way around the wheel. A paper disk, with white and black sectors, is mounted on the wheel. As the car moves, the disk on the wheel rotates and the infrared sensors sense a different amount of infrared light, depending on whether the white part or the black part of the disk is presented to them.

Learning to Backup a Trailer: for this project we studied the problem of backing a car and trailer rig to a target location by steering the front wheels of the car. Figure 1 shows the autonomous mini-robot *TBMin* that we have designed and built for this problem. TBMin uses two inputs: the angle from the spine of the trailer to the goal, and the angle of the hitch. The angle to the goal is computed using a light-tracking head (mounted on a servo) that tracks a 100W light bulb. The angle of the hitch is sensed using a variable resistor. If the rear of the trailer reaches the goal, success is signaled. If the angle to the target exceeds 90° or the angle of the hitch exceeds 45°, failure is signaled. The system is clocked to operate in discrete time units. The system is able to learn extremely quickly, despite noise, and with limited computing power and feedback [3].

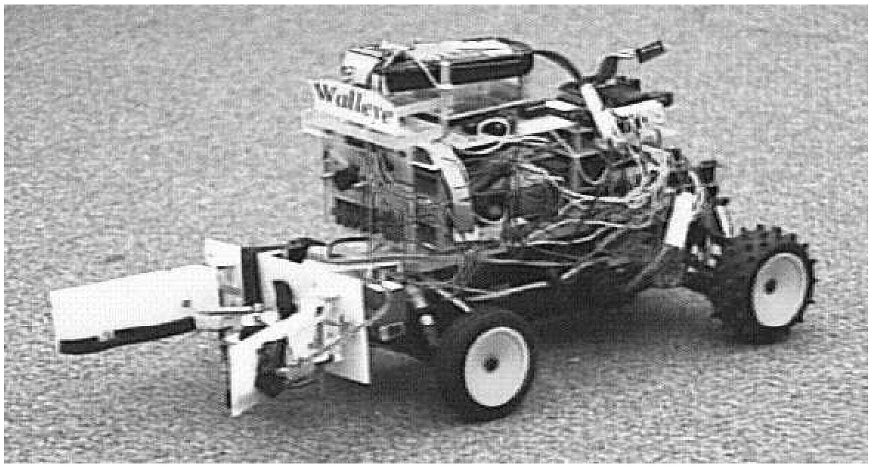

Figure 2: Walleye, the trash collecting mini-robot

A large team project

As part of the project we describe here, a team of five undergraduate students designed and built an autonomous minirobot that was able to find cups and cans on the floor, recognize them, pick them up, and bring them to the appropriate trash or recycling bin. The robot is named Walleye, the state fish of Minnesota that is known for his voracious appetite.

Walleye, shown in Figure 2 is built out of a radio controlled car with the original electronics replaced by boards designed by the students. All boards are built around the 68hc11 microcontroller, and have 16k of ROM and 32k of RAM. The vision system uses a CCD chip with digital output, a wide-angle lens, and a frame grabber board on which all the vision processing is done. Two 7.2 volt rechargeable batteries are used, one for the motors, one for the computer boards. The batteries last approximately 15-20 minutes. All software is written in C, with the exception of a few routines written in assembly. The software is developed under Linux and cross compiled using the "gcc" cross compiler. A variety of tools have been developed to download programs and upload results through the serial port of the workstation or PC. Walleye was built using off-the-shelf components at a cost of approximately five hundred dollars.

Walleye was built explicitly for the Fourth Annual Mobile Robot Competition that took place at the International Joint Conference on Artificial Intelligence in August 1995, even though we had already designed the computer boards and used them for other projects before. This was the first project in which we needed to do vision processing on minirobots, and vision turned out to be very challenging.

The limited size of the memory and the limited speed of the microcontroller have dictated most of the design choices. Walleye was, by far, the cheapest of all the entries at the competition, and performed well obtaining the third place. More important, working on the project has been a tremendous educational experience for the students in the team, and we plan on repeating the experience next year. The project required a variety of skills, from mechanical design (for the gripper of the robot), to hardware design (all the boards used to control the minirobot have been designed and built by the students), to programming (at all levels, from the low level code used to control the motors and read the infrared detectors, to the high level image processing routines), and to software and hardware testing.

The strategy used to achieve the task is based on decomposing the task into several stages, each as simple as possible. The strategy has been inspired by the strategy reported in [1]. Walleye does not keep track of its own position and does not even attempt to construct a map of the world. The position sensor

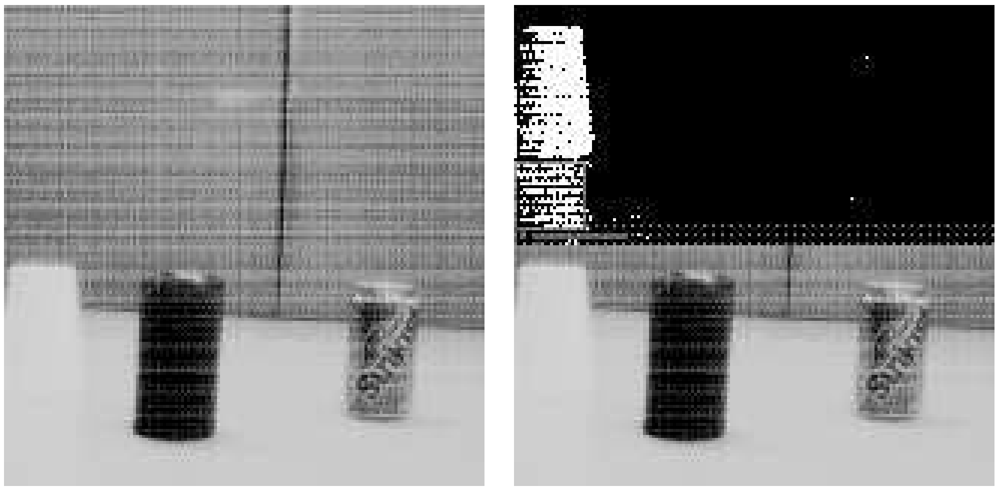

Figure 3: A picture before and after thresholding when looking for cups

we use is too crude for the map to be of any use. This allows objects to be moved around, and trash bin to be rearranged dynamically during the execution of the task. The presence of other obstacles, such as chairs or furniture, does not affect Walleye.

The most important part of Walleye is its vision system. We designed the vision software to be fast (which was a major challenge considering the limited computing power of the microprocessor). The other major problem we had to face was the limited amount of memory available. With only 32K of RAM and an image of 160 x 160 pixels there isn't too much memory to spare for intermediate results. So, instead of transferring the image to the main microcomputer board we do all the vision processing on the microprocessor on the frame grabber board. This saves transfer time, allows us to use the memory on the main microcomputer for the control stages, and keeps the vision software independent on the control part of the program. The frame grabber appears, to the program runing on the main microcomputer, as a collection of routines to perform operations such as grab an image, find a cup in the image and report is x and y coordinates in the image plane, or find a trash bin.

The camera height on Walleye was chosen to make the markers of the trash bins to always appear in the upper half of the image. No matter what the distance from the robot, the markers would never fall below the mid-point of the image. Similarly, since cups and cans are lower that the position of the trash markers, they will always appear in the bottom part of the image. So, if the robot is searching for cups or cans, only the bottom half of the image is utilized, and the top half becomes a work area. If trash bins are being searched for, then only the top half of the image is searched. This is extremely important because, first, we do not need to process the whole image (only the half that contains what we are looking for), and, second, we can use the half of the image not needed as work area (and so we have some memory on which to store intermediate results of image processing).

All of the cans used are of a dark color, and all of the styrofoam cups are white. The trash bins are designated by a large black "T" and the recycling bins by a large black circle. The assumption we make is that the color of the floor is sufficiently different from the color of both the cups and the cans, so that cups or cans can be found by performing different threshold operations on the image.

The initial operation is to transform an image into a binary image, by using a threshold that transforms all pixels to either black or white, depending on each pixel's brightness intensity. Since we need to distinguish white and dark objects, we use two different types of threshold on the image, a white threshold to search for cups, and a black threshold to search for cans and bins.

First, we do a white threshold to find out if there are any cups. The threshold changes all pixels that fall below a certain brightness value to black, and all pixels above to white. The system then searches through this image to look for white pixels. Once a white pixel is found, the pixels around it are checked to see if they are white. If they are not, the white pixel is deleted (changed to black), and the search continues. If a white pixel is found to have neighboring white pixels, the neighbors are examined, until the whole white area is determined. After finding a white area, the area is analyzed. If it conforms to a preprogrammed height-to-width ratio, we assume the white area is a cup, and the robot begins to track it. If not, the area is deleted and searching continues. This is shown in Figure 3.

If the whole image is searched, and no cup is found, a black threshold is taken to look for cans. In this threshold, all pixels below a certain brightness value are changed to white, and all pixels above a certain value are changed to black. In effect, what happens is that all dark items in the original picture are changed to white in the work area, and all light items are changed to black. This overall approach makes the system quite robust.

Acknowledgements

John Fischer originally proposed the idea of building minirobots, designed all the microcomputer boards used in the project, and wrote a large part of the software. Without him this project would have never started. We wish to thank the other members of the competition team, Elena Beltran, Luis Ortiz, Maxim Tsvetovatyy, and Paul Zobitz, for their enthusiasm and hard work in the Walleye project, and all the other students who have worked on projects in robotics.

We would like to acknowledge the support of NSF under grant NSF/DUE-9351513, the Undergraduate Research Opportunities Project of the University of Minnesota, the AAAI, the AT&T Foundation, and NIH through the Summer REU program at the University of Minnesota.

References

[1] T. Balch, G. Boone, T. Collins, H. Forbes, D. MacKenzie, and J. C. Santamaria. Io, ganymede, and callisto: A multiagent robot trash-collecting team. *AI Magazine*, 16(2):39–51, 1995.

[2] R. Cannon. *Dynamics of Physical Systems.* McGraw-Hill, New York, 1967.

[3] D. Hougen, J. Fischer, M. Gini, and J. Slagle. Fast connectionist learning for trailer backing using a real robot. In *Proc. IEEE International Conference on Robotics and Automation*, 1996.

[4] Dean Hougen, John Fischer, and Deva Johnam. A neural network pole-balancer that learns and operates on a real robot in real time. In *Proc. MLC-COLT Workshop on Robot Learning*, pages 73–80, 1994.

Biographical Information

MARIA GINI is an Associate Professor in the Department of Computer Science at the University of Minnesota. Her research interests are in using Artificial Intelligence to build autonomous entities, such as robots and intelligent software agents.