# A Modular, Interdisciplinary Approach to Undergraduate Robotics Education

Eric Welton Seth Hutchinson Mark Spong
University of Illinois

## 1 Introduction

The field of robotics is interdisciplinary by nature, and as a result, university robotics curricula tend to be distributed across various departments, typically within the College of Engineering. This arrangement poses a number of problems: (1) resources, particularly laboratory equipment, are difficult to obtain, since in many cases several departments are competing for a single, limited set of resources; (2) there is much duplication between the various introductory robotics courses, resulting in an inefficient utilization of faculty teaching resources; (3) even the introductory courses are not interchangeable, since each department's courses tend to spend considerable lecture time on specialty topics, of interest only to the specific discipline.

We have developed a set of modular robotics courses and laboratories to help solve the problems listed above. The key to our approach is the use of half-semester courses. The first half of the semester is used to provide a single, general introduction to fundamental concepts, and the second half is used to treat discipline-specific topics in detail.

Our current robotics curriculum (which is still in the experimental stage) consists of three courses. The first course in the sequence is a half-semester course covering the fundamentals of robotics (homogeneous transformations, kinematics, and dynamics). This course is supplemented by a laboratory section that exposes students to state-of-the-art robot simulation software and industrial grade robot arms. The current laboratory facilities for the course are a combination of resources from several departments, including Electrical and Computer Engineering, General Engineering, and Mechanical and Industrial Engineering.

For the second half of the semester, students are encouraged to register for one or more half-semester courses covering more specific topics. Currently, we are offering a course on dynamics and control, and a course on computer vision. We hope, in future semesters, to add a number of additional half-semester courses, sponsored by various Engineering departments. Finally, the Graduate curriculum contains a number of advanced courses, which will require as prerequisites an appropriate sequence of these half-semester courses.

The organization of this paper is as follows. We begin by briefly describing the three half-semester courses that are currently being offered. We then present a description of some of the laboratory facilities, and the associated educational software packages that we have developed.

0-7803-1482-4/93/$3.00 © 1993 IEEE

## 2 Course Contents

Our current robotics curriculum is illustrated in Figure 1. The first course, required as a prerequisite for all others, is the half-semester course entitled *Introduction to Robotics*, which is offered in the first half of the fall semester. Two additional half-semester courses, *Introduction to Computer Vision* and *Robot Dynamics and Control*, are offered in the second half of the fall semester. All three of these courses are cross-listed in both the General Engineering and Electrical and Computer Engineering Departments. Each of these three courses is described in more detail below.

There are now three graduate level courses in the robotics curriculum: *Computer Vision*, *Advanced Robotic Planning*, and *Advanced Topics in Robot Control*. These courses require much greater mathematical sophistication than the three half-semester courses.

In addition to these courses, we hope to add one half-semester class on off-line robot programming, and one half-semester class on artificial intelligence techniques for robotics.

The typical student in these courses is a senior or beginning graduate student in Electrical and Computer Engineering, General Engineering, Mechanical Engineering or Computer Science. The current course format is two 90 minute lectures per week, with weekly homework assignments, including laboratory exercises.

### 2.1 Course Overview: "Intro. to Robotics"

The introductory robotics course is the result of combining the first halves of two courses previously offered by the General Engineering and Electrical and Computer Engineering Departments. The course is currently under supervision of the Electrical and Computer Engineering Department. The contents, which are outlined below, are those concepts that are fundamental to the study of robotics, including rigid body motions, homogeneous transformations, kinematics, and basic dynamics.

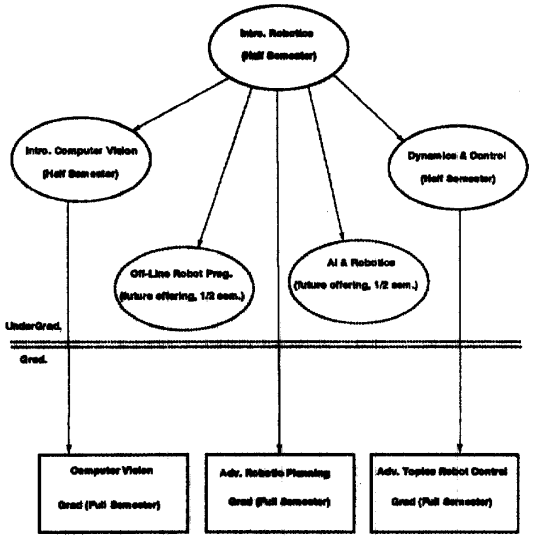



Figure 1: Organization of Courses

| Topic | Hours |
|---|---|
| **Introduction** | 3 |
| Historical development of robots; basic terminology and structure; robots in automated manufacturing | |
| **Homogeneous Transformations** | 3 |
| Rotations and their composition; Euler angles; roll-pitch-yaw; angular velocity and acceleration; homogeneous transformations | |
| **Forward Kinematics** | 3 |
| common robot configurations; Denavit-Hartenberg convention; A-matrices; T-matrices; examples | |
| **Inverse Kinematics** | 3 |
| Planar mechanisms; geometric approaches; spherical wrist | |
| **Velocity kinematics** | 4 |
| The Jacobian; singular configurations; manipulability; singular values; pseudoinverse | |
| **Dynamics** | 6 |
| Generalized coordinates; virtual work; Euler-Lagrange equations; manipulator kinetic and potential energies | |
| **Exams** | 2 |
| **Total** | 24 |

## 2.2 Course Overview: "Intro. to Computer Vision"

The introductory computer vision course was originally included as the second half of an introductory robotics course offered by the Electrical and Computer Engineering Department. The new course is under supervision of the Electrical and Computer Engineering Department. The content, which is outlined below, is basic image processing and computer vision, including image enhancement, smoothing, edge detection, and binary thresholding.

| Topic | Hours |
|---|---|
| **Introduction** | 1.5 |
| Computer vision in automation; inspection; lighting techniques | |
| **Two-Dimensional Image Formation** | 1.5 |
| Perspective geometry, pin-hole lens approximation | |
| **Three-Dimensional Image Formation** | 1.5 |
| Stereo vision by triangulation; active range sensors | |
| **Low Level Vision** | 4.5 |
| 2D discrete convolution; 2D digital linear filters; histograms; enhancement | |
| **Non-linear Operations on Images** | 3 |
| Median filtering; morphological operators; Medial Axis Transformation | |
| **Segmentation and Feature Detection** | 4.5 |
| edge detection; threshold selection; split/merge approaches; region growing; Hough transformation | |
| **Description** | 3 |
| Skeletons; signatures; moments | |
| **Recognition** | 3 |
| graph matching; subgraph isomorphism | |
| **Exams** | 1.5 |
| **Total** | 24 |

## 2.3 Course Overview: "Robot Dynamics and Control"

The robot dynamics and control course was originally included as the second half of an introductory robotics course offered by the General Engineering Department. The new course is under supervision of the General Engineering Department. The content, which is outlined below, includes more advanced topics than are treated in the introductory course, including trajectory planning, position control, and force control.

| Topic | Hours |
|---|---|
| **Dynamics** | 3 |
| review of Lagrangian dynamics actuator and sensor dynamics | |
| **Trajectory Planning** | 3 |
| Cartesian space; joint space; interpolation methods | |
| **Position Control** | 10 |
| independent joint control; PID and feed forward control computed torque and inverse dynamics resolved motion control control of orientation | |
| **Force Control** | 6 |
| stiffness and compliance; network models; hybrid impedance control | |
| **Exams** | 1.5 |
| **Total** | 24 |

## 3 Laboratory Overview

In the remainder of the paper, we describe the laboratory that accompanies the two courses *Introduction to Robotics* and *Introduction to Computer Vision.* These courses feature several laboratory exercises that reinforce fundamental concepts in robotics and computer vision (including homogeneous transformations, kinematics and inverse kinematics, trajectory planning, low level computer vision, object recognition, etc.). The students are also provided with a set of optional laboratory exercises, to be performed during laboratory open hours. These optional exercises allow additional exposure to robotic systems, and provide an atmosphere in which robotic/vision hardware can be used to further explore concepts taught in the lecture portion of the course.

The lab itself is equipped with a mobile robot, two robot work cells, and one vision work cell. The mobile robot is a TRC Lab-mate mobile base, outfitted with proximity sensors and linked to a non-mobile PC via a serial tether. Each robot work cell is comprised of a SUN Sparcstation 1 and a PUMA 260 robot arm equipped with an Assurance Technologies Force/Torque sensor and a Schunck pneumatic gripper. The PUMA robots are jointly controlled by the SUN and the PUMA's native controller which implements the VAL-II language. The vision work cell includes two Sony CCD cameras, a DataCube Digimax video digitizer, a DataCube Framestore frame buffer and a SUN workstation. Two additional workstations are also available.

## 4 Laboratory Software

We will now briefly describe two new software systems that are used in the lab (one for robotics and one for vision). Each of these systems is a comprehensive framework of stand-alone programs, libraries, simulators, and device controllers providing at least three advantages over earlier systems. First, each system is largely hardware independent, interposing simulators and actual hardware when appropriate, transparent to the user. This provides a great deal of freedom since all features of both systems are accessed through a unified C++ (and C) interface, allowing the student to develop project solutions regardless of hardware availability or lab access. Such hardware independence also allows educators to present robotics and vision topics in a concrete fashion even if they are not in a position to provide hands on experience. Second, the systems provide rich environments into which students can embed and test their own algorithms in parallel with the actual hardware and verified algorithms. Lastly the student can optionally access the hardware and software at different implementation levels, making it possible to retain an appreciation for the reality of implementation while simultaneously focusing on the elegance of full-project design in the context of semester solutions to real world problems.

The design of both software systems was motivated by two principal factors. First, we wished to provide an environment where students with minimal computer experience could readily access and explore the course topics while students with higher levels of experience could harness the power of the SUN workstations for a more challenging and motivating experience. Second, we wanted to allow optional interaction with the implementation details of a given robotic or vision system. We did not want to completely hide the reality of a robotics or vision system behind a high level interface yet we wanted to provide an environment where implementation issues were not a limiting factor. In this way student projects and lab exercises could span a wide range of physical complexity while maintaining a near constant level of implementation complexity. For example, a student interested in examining image processing algorithms could implement a project at a the bit-flipping, video-card level, while a student interested in visually-guided motion planning would be free of the details of implementing all of the necessary robot and video interfaces, concentrating instead on the actual information interaction necessary for such a project. It was decided that the best way to meet these two conditions was to design a software package which incorporated multiple levels of access and useability while remaining simple and easy to use.

We chose to develop the software in C++ (version 2.0 and above) using the g++ compiler (v1.4.5 and above). We chose C++ for instructional as well as practical reasons. Practical considerations ranged from increased application development speed to increased ease and range of portability. From the instructional standpoint standard C++ libraries provide a more natural and readable form in which to code operations such as matrix multiplication. Kinematics programs, for example, can be written more cleanly and concisely when matrix multiplications can take the inline form N=M1*M2, instead of function calls or, in the worst case, inline implementation of a multiplication algorithm. This language choice also provides the student with an opportunity to gain experience working with a generally useful language, a skill desirable for both industry and graduate studies. Lastly, when designed correctly, C++ libraries can be used with traditional C, giving the student the option of using whichever language they are more familiar with–if 'learning-a-new-language' presents an issue for the student. Our particular choice of compiler was motivated by the ready availability of new C++ features such as class and function templates.

The resulting software is comprised of several independent libraries, two daemon processes, an some miscellaneous stand-alone programs that use both the libraries and the daemons. While the underlying complexity of both software systems is substantial, the complexity is hidden by two software interfaces. Also, installation of the library and header files in standard locations and interfaces designed with novices in mind, mean that use of the new facilities presents no complexity above and beyond that associated with writing the 'hello world' program. Furthermore, in addition to the software developed solely for this project, many additional libraries available from concurrent work have been made available to the student.

Students (and instructors) are often daunted by the cryptic style of UNIX operating system. To help reduce this we have designed each system such that only a single header file needs to be included in source files and a single library needs to be linked in at compile time. We have also made template makefiles available to eliminate the additional compile time argument.

### 4.1 Vision Software

The vision software consists of a run-time daemon, C++ and C interface libraries, and a graphical user interface. The daemon handles all video hardware access and maintains a dynamic library of processing tools, which is helpful in two ways. First, since all video hardware requests are made from a central point, alteration and expansion of the video equipment becomes quite easy and transparent to the student. Second, since the daemon

maintains a *dynamic* library of processing tools, the library of tools available for students can grow as the lab grows, each year potentially increasing the possible scope of the laboratory. Yet another facet of the dynamic nature of the daemon library is the ability to personally tailor one's daemon session via simple calls to the interface library. When coupled with the graphical user-interface this allows the student to utilize a posh environment into which they can incorporate their own image processing algorithms. We feel that the ability of the student to juxtapose and/or merge the results of their algorithms with algorithms known to work correctly will greatly aid the students' understanding of the lecture material and will foster a willingness to investigate these topics further. An example of this juxt-ability is given later.

Figure 2 illustrates the underlying process structure for a potentially typical session. The figure depicts the process layout that would result from three users accessing the daemon functions as a library, without requesting camera access, one user accessing the daemon functions as a library and acquiring images from the digital video hardware, and one user accessing both the daemon services and their own image processing tools from the graphical user interface (GUI). The student code is labeled *user-code*. Note how all access to the vision system passes through the VidLink interface. An example of a typical GUI screen is shown in Figure 3 below.

As can be seen from Figure 2, the principal components of the vision software are the Image Transform Daemon, the Image Transform Processor, the Image Editor GUI, and the VidLink interface. This is somewhat misleading since the Image Transform Processor is actually a child process forked from the daemon and in that sense is a complete copy of the daemon only serving a single client instead of many. Thus the only real function performed by the daemon is the acceptance and distribution of connection requests. All real work is performed by the child Image Transform Processor (ITP). Consequently only the ITP/VidLink chain and the GUI/VidLink relationship is presented in significant detail.

### 4.2 Robotics Software

The robotics software also consists of run-time daemons, an interface library, and some stand-alone programs. While the final robotics system will contain two run-time daemons, one supporting non-real-time supervisory control and the other supporting real-time path control, only the supervisory (non-real-time) daemon is currently in operation. The supervisory system structure is shown in Figure 4 and several aspects of its function are described below.

The non-real-time system utilizes the VAL controller's 'supervisory' port, which offers access to controller functions for non-time-critical applications. The port itself is a 9600 baud RS232 serial port which utilizes the DDCMP protocol as defined in *DECnet Digital Network Architecture, Digital Data Communications Message Protocol Specification Version 4.0, March 1, 1978.* Communication with the PUMA via this facility consists of handling and responding to messages from six 'logical units' or LUNs. Direct use of the supervisory system requires knowledge of the DDCMP protocol and the correct protocol for dealing with the six LUNs. Each of the six LUNs is essentially an autonomous agent within the controller, and consequently a supervisor must interact with each according to its own specifications. Thus the primary difficulty in dealing directly with the supervisory system is ensuring that messages from each LUN are handled in a sensible and timely fashion. It is the responsibility of the supervisory daemon (SupD) to handle both the DDCMP protocol and the LUN protocol, transparent to the end user. While this may sound cumbersome at first, the complete implementation of SupD provides several useful features such as asynchronous position and status monitoring, unlimited VAL disk space via a remote 'virtual' disk on the local SUN server, and the ability to insulate both the robot and student programs from the failures and glitches of the other. By focusing robot access through a single point we are able to provide safety features such as joint motion limit checking and speed bounds, as well as provide the simulator transparency described below.

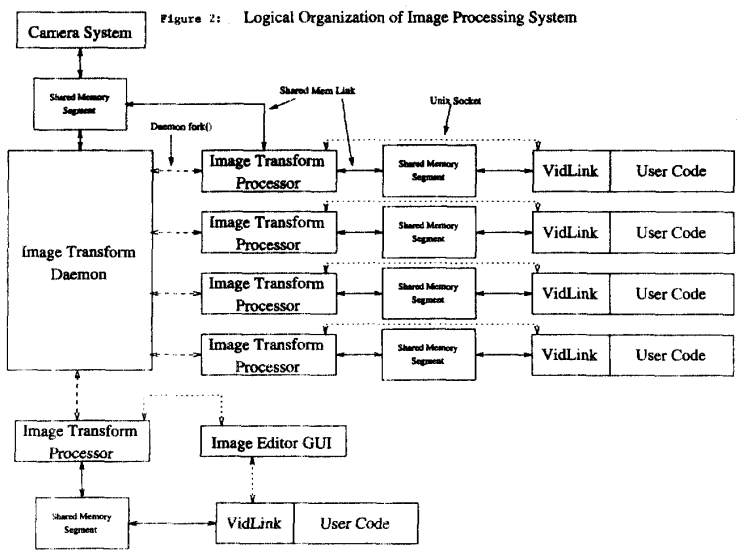



Figure 2: Logical Organization of Image Processing System

## 5 Summary

In this paper, we have briefly described three modular, half-semester courses that comprise the emerging undergraduate robotics curriculum at the University of Illinois. In addition, we have presented two software systems that are currently being tested in these courses. These systems allow students to make progress, unrestrained by occasional hardware inaccessibility and undaunted by implementation details which have been an issue in the past. Students are also able to run their own solutions in parallel with actual solutions. For example, a student could run a forward kinematics solver while simultaneously observing the isomorphic operation of the physical robot. Furthermore, use of these systems provides an opportunity for students to design and implement solutions to real-world problems within the course of a semester.

Figure 3: Typical GUI Screen

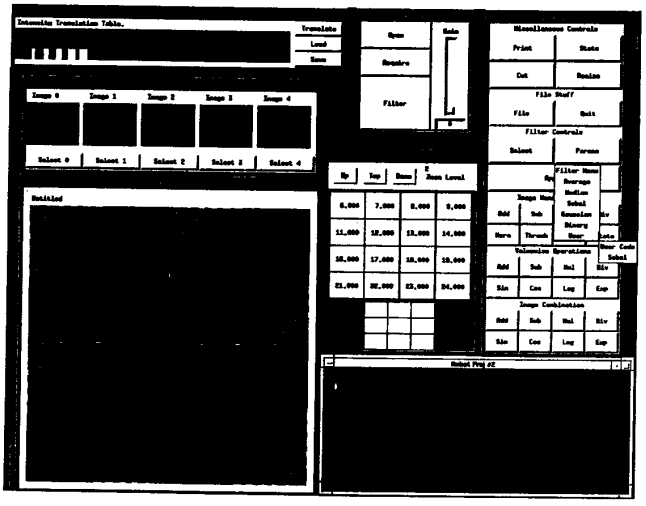
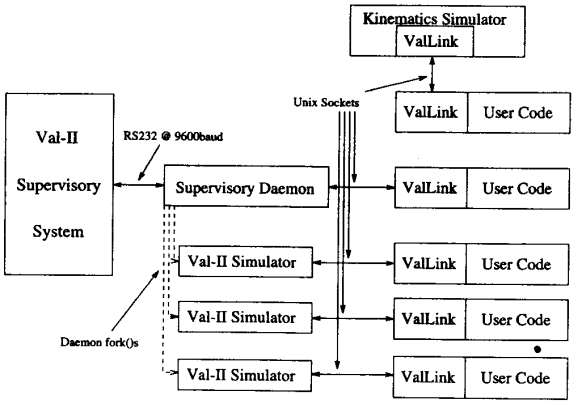


Figure 4: Logical Organization of ValLink



## Eric Welton

**Eric Welton** is an undergraduate at the University of Illinois at Urbana-Champaign, currently completing a double major in Computer Science and Philosophy.

## Seth Hutchinson

**Seth Hutchinson** is currently an assistant professor of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign.

## Mark Spong

**Mark Spong** is currently a professor of General Engineering at the University of Illinois at Urbana-Champaign.