CMRoboBits: Creating an Intelligent AIBO Robot

Manuela Veloso, Scott Lenser, Douglas Vail, Paul Rybski,
Nick Aiwazian, and Sonia Chernova - *Thanks to James Bruce*
Computer Science Department
Carnegie Mellon University
Pittsburgh PA, 15213-3891

Introduction

Since 1997, we have researched teams of soccer robots using the Sony AIBO robots as the robot platform (Veloso & Uther 1999; Veloso *et al.* 2000; Lenser, Bruce, & Veloso 2001a; 2001b; Uther *et al.* 2002). Our experience runs across several generations of these four-legged robots and we have met increasing success every year. In the fall of 2003, we created a new course building upon our research experience with the AIBO robots. The course, which we entitled CMRoboBits, introduces students to all the concepts needed to create a complete intelligent robot. We focus on the areas of perception, cognition, and action and use the Sony AIBO robots to help the students to understand in depth the issues involved in developing such capabilities in a robot. The course has one two-hour weekly lecture and a one-hour weekly lab session. The course work consists of nine weekly homeworks and a larger final project. The homework assignments include written questions about the underlying concepts and algorithms as well as programming tasks for the students to implement on the AIBO robots. Evaluation is based on the students' written answers, as well as their level of accomplishment on the programming tasks. All course materials, including student solutions to assignments, are made available on the Web. Our goal is for our course materials to be used by other universities in their robotics and AI courses. In this paper, we present the list of topics that were covered in the lectures and include examples of homework assignments as well as the rational behind them.

The Goals of the Course and the Schedule

The main goal of the course is to learn how to create an *intelligent* robot, using the AIBO robot as a concrete example. We want the students to understand how *to program* the robots to perform tasks. Our aim is to *demystify* robot programming so that it becomes clear and accessible to all of our students. A parallel goal of the course, and mainly our own goal, is to move from our research code in robot soccer to modular code that can be used for any general robot task. We aim to provide course materials that are modular and well structured so that people at other universities can use the materials in their own courses. We further believe that reorganizing and cleaning up our robot soccer code will have several additional positive effects, namely facilitating both our own future research and the initiation of new students in their research.

We designed the 15-week course along five main components:

Sensors and actuators: Robots perceive the world using their *sensors* and they affect their environment with their *actuators*. All interactions between the robot and its environment are mediated by sensors and actuators; they are equivalent to input and output operators in robot programming. This component of the course introduces students to the idea of acting in the face of uncertainty. Unlike traditional programming where input values are completely known, robots must perform with only limited, noisy knowledge of their environment. Additionally, robots must cope with noise and uncertainty in their actions; motors do not always perform the requested movements and factors such as friction and slip are difficult to take into account when predicting the outcome of actions. Students must be introduced to the idea of uncertainty, which is central to robot programming.

Motion: The AIBO robots offer an interesting and challenging platform for exploring robot motion. AIBOs are interesting because they are a legged platform with fifteen degrees of freedom (DOF) in their head and legs. Each of the four legs has three DOF and the head has pan, tilt, and roll joints. This count only includes the major joints. The tail, mouth, ears, and eye LEDs can also be actuated to create more expressive behaviors. In this unit, we introduce students to the ideas of forward and inverse kinematics. We also include a practical introduction to our motion system on the AIBO. We describe our parameterized walk engine which uses approximately fifty numeric parameters to specify an entire gait for the robot. These parameters include factors such as robot body height, body angle, lift heights for each leg, and timings. We also introduce students to the idea of frame based motion where all joint angles are specified for a few *key frames* and the robot interpolates between them. This type of motion is useful for scripting kicking motions for soccer, dance motions, climbing, and other predefined motions.

Vision: The AIBO robots use vision as their primary sensor. Color images in the YUV colorspace arrive at a framerate of 25hz. The vision unit of the course acquaints students with the basics of robot visual processing. Students briefly learn about the YUV color space, which is commonly used by image capture hardware. Real time color segmentation and camera calibration are also discussed. Finally, higher level concepts such as object recognition from the color segmented images, including weeding out false positives is covered at length. Students also learn how kinematics ties back to vision for calculating the real world position of objects in the vision frames.

Localization: In order to act effectively, a robot often needs to know where it is in the environment. Localization becomes an essential component that interacts with perception, decision making, and motion. This unit introduces the ideas of probabilistic localization beginning with the basic ideas of Markov localization and including different methods of representing belief such as Kalman filters and particle filters. We also cover ideas such as recovering from errors in localization (e.g. the kidnapped robot problem) through sensor based resampling and the various tradeoffs that may be made between computational cost and resource consumption.

Behaviors: We teach students about behaviors at several places in the course since behavior is a basic component of virtually robot task. Initially, we introduce finite-state machines and incrementally address more complex behavioral structures, such as hierarchical behaviors and planning. We finish the course with multi-robot behaviors, discussing the challenges and presenting several approaches for multi-robot communication and coordination.

The schedule is organized along these five main components. Table 1 shows the current ongoing schedule for the Fall 2003.

Homeworks

In this section, we briefly describe the rational, requirements, and grading of the homework assignments in the course. Students were typically given one to two weeks to complete each assignment. They worked in groups of 2 or 3 students and kept the same groups for the entire semester. Assignments were due at the beginning of the lab period each week, although we often gave students until the next day. This allowed us to either have a demonstration session at the beginning of the lab or to go over the assignment with the students where the TA could look at the students' code and watch the robot to diagnose problems. It was vital to have both the robots and source code available while helping students with problems.

Date	Topic	Homework
09/03	Introduction - Intelligent Robots	1 out
09/08	Sensors and Basic Behaviors	
09/10	*Lab: Accessing sensors*	1 due, 2 out
09/15	Motion - parameterized, frame-based	
09/17	*Lab: Motion sensitivity to parameters*	2 due, 3 out
09/22	Vision - color spaces, calibration	
09/24	*Lab: Vision - color spaces*	3 due, 4 out
09/29	Vision - object recognition, filtering	
10/01	*Lab: Vision - debugging tools*	4 due, 5 out
10/06	Vision - Visual sonar	
10/08	*Lab: Obstacle avoidance*	5 due, 6 out
10/13	Behaviors - reactive, machines	
10/15	*Lab: Behavior implementation*	6 due, 7 out
10/20	Localization - modeling, filtering	
10/22	*Lab: SRL*	7 due, 8 out
10/27	Localization - ambiguity, tracking	
10/29	*Lab: Ambiguous markers*	8 due, 9 out
11/03	Behaviors - Hierarchical, multi-fidelity	
11/05	*Lab: Chase ball to goal*	9 due, 10 out
11/17	Behaviors - Planning and execution	
11/19	*Lab: Playbook implementation*	10 due, 11 out
11/24	Behaviors - Execution, learning	
11/26	*Thanksgiving Break (No Lab)*	
12/01	Behaviors: Multi-robot coordination	11 due, 12 out
12/03	*Lab: Push bar*	12 due

Table 1: CMRoboBits: Fall 2003 Schedule

HW1: Introduction to Developement

The first homework served as an introduction to the developement environment and brought students up to speed on how to access the source code from our CVS tree, compile the code using the OPEN-R SDK (freely available from Sony), and copy the final programs to memory sticks for use with an Aibo. This homework also showed students how to select which behavior runs using our framework and allowed us to test code hand-ins using a dropbox system. Creating a simple first assignment allowed us to iron out the wrinkles in how we had setup the course and student lab.

HW2: Basic Sensors

The second homework is designed to familiarize the students with the sensors on the robot. The background section covers how to subscribe to sensor messages, specifically, data from the robot's accelerometer and the touch sensors on its feet. The students then must use this information to set LEDs on the robots face every time a foot contacts the ground, to detect when the robot is lifted off the floor, and to display whether the robot is level, tilted toward its left side, or tilted to its right.

This assignment gives the students practical experience with a sense-think-act loop. They must read [noisy] sensor data from the robot, deterimine which actions to take based on this sensor data, and finally send commands to the robot to perform these actions. This sequence is repeated with a frequency of 25 hz on the robot.

HW3: Robot Motion

Robot motion involves a great deal of trial and error. In the third homework, students learned how to build up to a complete motion through incremental, trial and error experimentation. The assignment was broken down into two parts. In the first part, students created a set of walk parameters to describe a gait. Robot gaits are specified by 51 parameters that are used by a walk engine to generate the actual trajectory that the end of each foot follows over the course of a single step. The parameters include limits on how high each foot can rise above the ground, the desired angle of the robot's body, and other similar factors. Finding an effective walk is an optimization in this 51 dimensional parameter space. Parameters are often coupled together in certain portions of the space and there are many local minima. Typically we optimize for speed and stability, although other factors such as a walk with a high body height are possible.

The second part of the assignment required students to create a new motion from scratch using a key frame animation based approach. Specifically, students created a motion that made the robot perform a complete rollover and then climb back onto its feet. They learned how to convert between the positions of the robot's limbs in space and the corresponding angles of the robot's joints in their own coordinate frame. Since rolling over is a dynamic activity that depends on building up momentum and moving different legs in concert, the students also learned how to coordinate different joints simultaneously. An incremental, experimentation based approach was also important for being successful with this portion of the assignment.

HW4: Calibrating Vision

Since the focus of this course was to give students the practical knowledge that they'd need to program a working robot, we included an assignment on vision calibration. In this homework, students used the robot's camera to capture images of the environment. They transfered these images to a workstation and used standard image editing software to label the colors in the images. In other words, they would draw over the orange regions of an image with a solid orange, replacing the large set of YUV values that appear as orange with a single, predefined value for that color. These labeled images serve as training data for a supervised learning algorithm that learns a mapping between YUV color values and symbolic color values such as *yellow*, *orange*, or *blue*.

Part of the value from this assignment was showing students how much work goes into calibration. Taken with the fact that fast color segmentation algorithms that rely on mapping directly from pixel values to symbolic colors are brittle in the face of changing lighting conditions, this provides strong motivation to try other approaches; recalibrating vision for new lighting is a lot of work! Students also learned the tradeoff between taking more sample photos to improve accuracy versus the increase time spent labeling the photos. They learned that this type of lookup based segmentation is unable to disambiguate between colors that look different to humans but have the same YUV values to the camera. Students also learned how to adjust the weights assigned to the examples for different symbolic colors. For example, training images often contain fewer examples of colors associated with small objects and many pixels from larger objects. This creates a bias in learning where the end classifier wants to say everything is the same color as large objects. Finally, students learned to evaluate the final, learned mapping from pixel values to symbolic colors against a test set of images rather than against the training set. Realistic evaluation of how well algorithms will perform is important.

HW5: Object Recognition

Once students understand low level vision concepts such as color segmentation, they need to learn how to perform higher level tasks such as object recognition. This was the focus of the fifth assignment. Students learned to detect a bright orange ball, a colored bullseye, a small scooter, and a tower built from colored cylinders using color segmented images. The wheels of the scooter were the same shade of orange as the ball and additional towers built from colored cylinders were present so students needed to filter out false positives as well as avoid false negatives.

Although the training data was gathered using the robot's camera, this assignment was completed entirely on workstations using the same code that runs on the robots with an abstraction layer to allow it to run under Linux. The exact same vision processing is done starting with a raw YUV image, but the entire process can be observed using standard debugging tools. This allowed students to get under the hood of the vision process and try many more approaches than embedded developement would; the turnaround time to try new code is much lower on a workstation and the running program is much easier to observe. Once the algorithms are fine tuned, they can be ported to the robot by simply recompiling for a different target platform. This practical lesson is perhaps as important as teaching the students how to create heuristics for object detection.

HW6: Mounting a Charging Station

The sixth assignment built on the previous vision assignments. Students used object detection code to find the colored bullseye and tower beacon where were positioned on either end of a charging station. They programmed the robot to search and then climb onto the charging station before sitting down and shutting off.

This assignment brought together many of the past assignments and tied them together into a unified whole. The robot needed to sequence searching, seeking, and charging behaviors together relying on vision for sensing. The provided walk for the robots was too low to step onto the station so students needed to create custom motions to move the robot into position over the charger and settle themselves onto the contacts. This assignment tied vision, behaviors, and motion together into a coherent whole.

HW7: Maze Traversal

Students continued to create unified systems that rely on several basic components in the seventh assignment. In this assignment, students used a [provided] egocentric world model to track regions of free space around the robot. They created a behavior to traverse a convoluted path while controlling the robot's head to ensure that the local model contained accurate and up to date information. The path was not a true maze as it had no dead ends, but the robots did need to navigate through several turns without touching walls.

HW8: Written Localization

Localization requires more formal mathematical material than the rest of the material in the course. In order to give students experience with manipulating probability distributions this assignment consisted soley of written work. Students were given a uniform prior distribution of robot poses in a grid world and calculated the posterior probability after several moves through the world. The movements were nondeterministic and the students wrote out the complete prior and posterior distributions following each step. Several markers spaced across the grid gave the students a chance to incorporate observations using a sensor model as well as use a movement model to propage belief forward through time.

HW9: Hands on Localization

Hands on experience with localization is also important. Students created a behavior where the robots avoided a large square in the center of a carpet. When grading, the robot was firsted moved to a home position and told to memorize its position with a button press. Then the robot was picked up and moved to a new position (typically on the other side of the carpet) and replaced on the carpet. Evaluation was based on the robot detecting that it had moved and returning to its original position while avoiding the square in the center of the carpet by using localization. Six colored markers around the carpet were used for localzation. Students needed to create behaviors to control the head to seek out and fixate on these markers in order for the robot to be well localized. Additionally, students experimented with varying the number of samples used by the particle filter for localization. They made observations about the quality of the position estimates and convergence speed.

An Example Assignment

We present the full text of the second homework assignment. The full text of all assignments is available from the course webpage which is located at: http://www.andrew.cmu.edu/course/15-491/.

Homework 2 - Sensors

1. Introduction

You will learn how to subscribe to sensor messages, read data from the gsensor and foot pads, set the LEDs in the robot's face, and issue a simple motion command. It is also important to note the location of the header files that we use because they are valuable reference sources for additional information.

2. Background

You will need to access sensor data as a part of this lab. To do so, you subscribe to updates from the "SensorData" event processor. This is just like we went through in class for FeatureSet in the SpinDog behavior.

This subscription will provide a SensorData object with up to date information from the robots buttons, foot pads, and gsensor. The declaration of the SensorData class can be found in dogs/agent/headers/Sensors.h and /dogs/agent/shared_code/Sensors.cc. The header file is more interesting for the purposes of this assignment. Specifically, the fields of the SensorDataFrame structure and the SensorData::getFrame() method.

Recall that sensor frames arrive at 125hz while the camera operates at approximately 25hz. This means that there are multiple sensors frames available for each camera frame. In this lab, we will only worry about the most recent (unless you want to get more complicated - the assignment can be done using only the most recent frame).

To access foot pad data (assuming you have a pointer to a SensorData object and that you have included the file "../headers/Sensors.h" at the top of your source file so the relevant structures are available):

- bool pad_val = sensor_data_object_ptr->getFrame(0)->paw[foot_number]

The getFrame method takes an integer telling it which sensor frame you are interested in (because there will be several possible ones for the current vision frame). A value of 0 means use the most recent. A value of -1 means use the frame before the most recent frame. And so forth.

The paw field of the SensorDataFrame structure that is returned by getFrame is an array of 4 boolean values. Offsets 0 and 1 are for the left and right front legs respectively. Offsets 2 and 3 are for the left and right rear legs.

You will also need data from the gsensor to complete this lab. This information is also found in the SensorDataFrame structure that is returned by getFrame. The accelerations are contained in a vector3d structure

named accel. The vector3d class is an instantiation of the template found in
“dogs/agent/headers/gvector.h”. It has lots of useful utility methods available. However, you won’t need any of them in this lab; you’ll just want to access the individual fields of the vector. They are named x, y, and z. As a concrete example, the find the value of the acceleration along the robots z-axis in gravities, you would use the following:

- double z = sensor_data_object_ptr->getFrame(0)->accel.z;

You do the same for x and y. Recall that x is from the axis from the front of the robot to the back. Positive x is in front. Y is from left to right. Positive y is to the left. Z is up and down. Positive Z is up.

Finally, you will need to fill in a MotionCommand structure to send motions to the robot. This structure is defined in
“dogs/agent/Motion/MotionInterface.h”. The relevant fields are motion_cmd, which you must set to MOTION_WALK_TROT, and vx, which you should set to a possitive value to move forward. You will also need to set the led field (which is a bitmask) by ORing together LED constants. Three additional constants, MAX_DA, MAX_DX, and MAX_DY may be of use. They are the maximum velocities the robot can actually achieve when rotating and translating.

3. Procedure

- Go to your dogs directory and run the “cvs update” command to retrieve an updated walk and source code additions. Be sure to do a “stickit -a” in order to move the new walk to your memory stick (after you compile). Be aware that this will also overwrite run.cfg, so you may need to edit that file again. It is located in /memstick/config.
- Create a new behavior called “FootDog” in dogs/agent/Behaviors using the files SpinDog.h and SpinDog.cc as a template. This new behavior should act in the following way:
  - Set LED_LOWER_LEFT when the left front paw pad is depressed
  - Set LED_UPPER_LEFT when the left rear paw pad is depressed
  - Set LED_LOWER_RIGHT ... ... right front ... ...
  - Set LED_UPPER_RIGHT ... ... right read ... ...
  - Walk forward in a straight line at 1/2 max velocity.
  - The robot should stop walking when it’s lifted off the ground.
  - Set LED_MIDDLE_LEFT when the robot is off the ground and tilted to the left.
  - Set LED_MIDDLE_RIGHT when the robot is off the ground and tiled to the right. Neither of the middle LEDs should be set while the robot is on the ground.
- Remember to include “../headers/Sensors.h” in your behavior. You may also need to add “../Motion/MotionInterface.h” if it is not already present in order to use the LEDs.
- You MUST add your .cc file to dogs/agent/Main/Makefile in order for your code to be compiled. Find the section with behavior sources in it and made a new entry using that same format.

4. Questions

Answer the following using *no more than 3 sentences* for each answer.

- How did you detect that the robot was laying on its side?
- In class we used the example of maintaining an equilibrium distance from an object as an example of a place where hysteresis using two separate thresholds would be useful. This was actually a poorly chosen example. Explain why.

5. Grading

- Setting 4 LEDs for footpads - 4 pts
- Stopping the robot when it is lifted - 2 pts
- Setting 2 LEDs when the robot is tilted - 2 pts
- Questions - 2 pts

Conclusion

We are very interested in teaching Artificial Intelligence concepts within the context of creating a complete intelligent robot. We believe that programming robots to be embedded in real tasks illustrates some of the most important concepts in Artificial Intelligence and Robotics, namely sensing uncertainty, reactive and deliberative behaviors, and real-time communication and motion. This paper briefly describes a new course we have created this semester using the AIBO robots and building on our extensive robot soccer experience. The current course materials, including some videos of the results of some of homeworks done by the students, are available off the course Web page http://www.andrew.cmu.edu/course/15-491/.

Acknowledgements

We would like to thank Sony for their remarkable support of our research, especifically by making the AIBO robots accessible to us since its first conceptions in 1997. Sony has continued their support through these years, and is currently very interested in the potential impact of this AIBO-based new course. We would like to also thank the Carnegie Mellon Computer Science Department for approving this new course, and Professor Illah Nourbaksh for his generous share of lab space with this new course.

References

Lenser, S.; Bruce, J.; and Veloso, M. 2001a. CM-Pack'00. In Stone, P.; Balch, T.; and Kraetzschmar, G., eds., *RoboCup-2000: Robot Soccer World Cup IV*. Berlin: Springer Verlag. 623–626.

Lenser, S.; Bruce, J.; and Veloso, M. 2001b. CM-Pack: A complete software system for autonomous legged soccer robots. In *Proceedings of the Fifth International Conference on Autonomous Agents*. Best Paper Award in the Software Prototypes Track, Honorary Mention.

Uther, W.; Lenser, S.; Bruce, J.; Hock, M.; and Veloso, M. 2002. CM-Pack'01: Fast legged robot walking, robust localization, and team behaviors. In Birk, A.; Coradeschi, S.; and Tadokoro, S., eds., *RoboCup-2001: The Fifth RoboCup Competitions and Conferences*. Berlin: Springer Verlag.

Veloso, M., and Uther, W. 1999. The CMTrio-98 Sony legged robot team. In Asada, M., and Kitano, H., eds., *RoboCup-98: Robot Soccer World Cup II*. Berlin: Springer Verlag. 491–497.

Veloso, M.; Lenser, S.; Winner, E.; and Bruce, J. 2000. CM-Trio-99. In Veloso, M.; Pagello, E.; and Kitano, H., eds., *RoboCup-99: Robot Soccer World Cup III*. Berlin: Springer Verlag. 766–769.