CMRoboBits: Creating an Intelligent AIBO Robot

Manuela Veloso, Paul E. Rybski, Sonia Chernova, Nidhi Kalra
Scott Lenser, Douglas Vail, James Bruce, Nick Aiwazian
Computer Science Department
Carnegie Mellon University
Pittsburgh PA, 15213-3891

Introduction

Since 1997, we have researched teams of soccer robots using the Sony AIBO ERS-210 robots as the robot platform (**?**; **?**; **?**; **?**; **?**). 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. We have since refined the course and taught it again in 2004. The course, which we entitled *CMRoboBits: Creating an Intelligent AIBO Robot*, introduces students to all the concepts needed to create a complete intelligent robot. We focus on the areas of perception, cognition, and action (illustrated in Figure **??**), and use the Sony AIBO robots to help the students 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 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.

The AIBO is a remarkable piece of commercially-available robotic hardware. An AIBO has fifteen degrees of freedom (DOF) in its legs and head, a

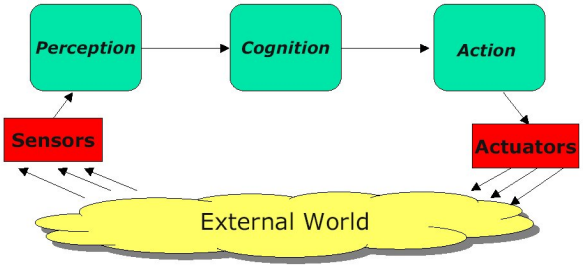

Figure 1: The modules used in the complete robot.

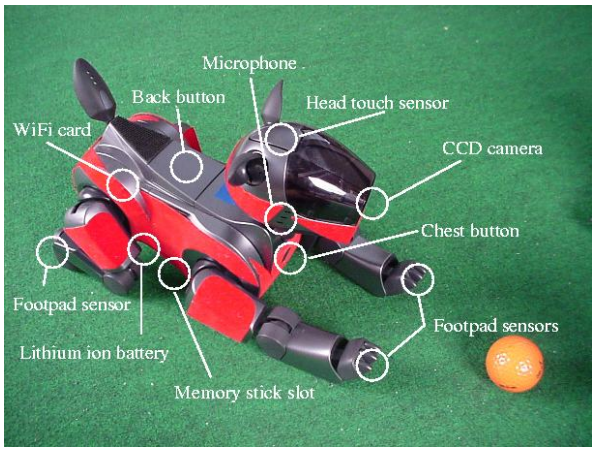

Figure 2: Annotated AIBO ERS-210

color CCD camera that can process images at 25-30 frames/second, a 3-axis accelerometer for body pose estimation, buttons on its back, head, and footpads, LEDs for visual debugging, and a wireless ethernet (802.11b) card for inter-robot communication. AIBOs are programmed using a free SDK called OPEN-R (found at http://openr.aibo.com/) which lets one compile control code on a workstation with a MIPS cross-compiler (available for GNU Linux, Microsoft Windows, and Mac OSX). The AIBO's low cost (approximately $1,600 US) allows an instructor to purchase several of them for the price of a more traditional research robotic platform.

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

Behaviors: The primary purpose of this course it to familiarize the students with the concept of behaviors for robot control. Every component, from sensors to localization, is caste in the framework of how a mobile robot can use those techniques in its 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. Figure **??** shows an illustrative example of decompositional and sequential behaviors used in the class.

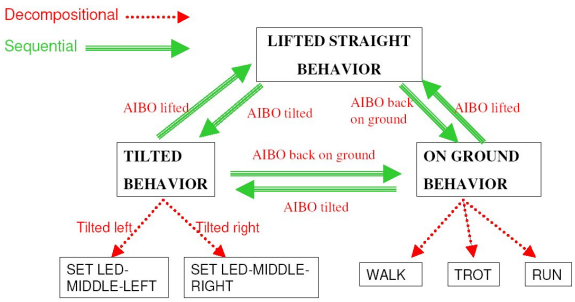

Figure 3: Description of a behavior hierarchy

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. Figure **??** shows an example plot of the 3-axis accelerometer data that students can use to determine what "state" the robot is in.

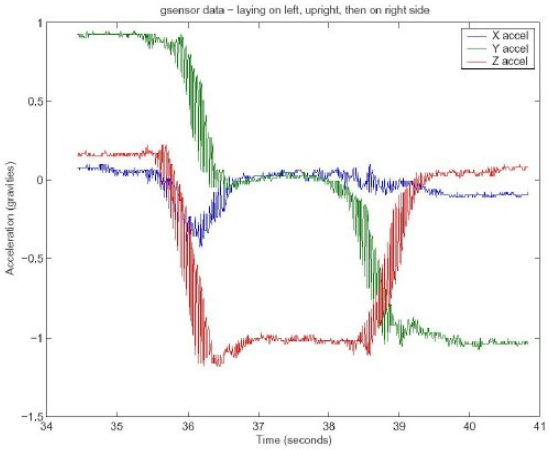

Figure 4: 3-axis accelerometer signature for a robot that starts on its left side, is rotated to an upright position, and then is rotated to its right side.

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 pos-

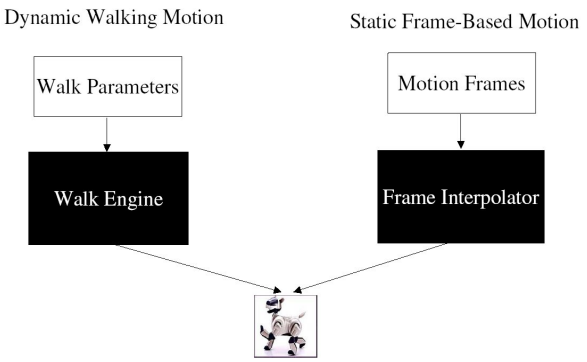

Figure 5: The two motion systems used in the CMRoboBits code base for controlling the AIBO's limbs.

itives 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. Figure **??** shows an example frame of video that has been processed by the AIBO's real-time color segmentation algorithm.

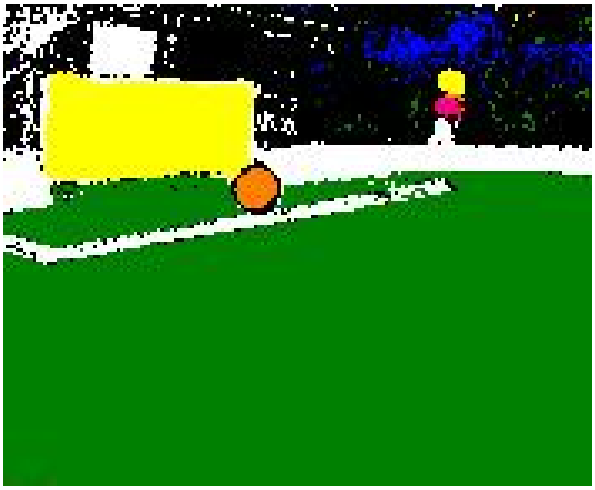

Figure 6: An example frame of video from the AIBO's camera after color-segmentation post-processing.

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.

Multi-Robot Cooperation: Once the students understand how to program a single AIBO to do interesting behaviors, we teach them how to use the AIBO's on-board 802.11b wireless ethernet system. This allows the robots to communicate between each other. For the remainder of the course, we teach the students how to solve problems with multi-robot behaviors, discussing the challenges and presenting several approaches for multi-robot communication and coordination.

Figure 7: Students learn some of the challenges with programming behaviors for cooperative multi-robot tasks.

Homeworks

REWRITE THIS TO BE MORE GENERAL, MORE OF A HOWTO FOR INSTRUCTORS?

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.

HW1: Introduction to Development

The first homework served as an introduction to the development 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 submissions 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, determine 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 development 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 solely 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 propagate 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 first 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 localization. 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.

Example Assignment: Mastermind

This section provides provides the text for an actual homework that was assigned in the Fall 2004 CMRoboBits class. This homework gave the students an opportunity to work with vision and motions, and required them to reason about complex behaviors that involved interacting/communicating with a human in a deterministic fashion.

The homework was to program the AIBOs to play the game of Mastermind with a human and do so with either the robot as player and the human as moderator, or vice versa. Mastermind is a guessing game where the player has to guess an ordered sequence of colors that the moderator has chosen in secret. The moderator provides simple but ambiguous clues to the player that must be used to infer the correct colored sequence.

Rules for Mastermind

Initialization

- Moderator randomly determines a sequence of length 'n' drawn from a set of 'm' possible colors (with repeated colors allowed)

Play

- Player guesses a sequence
- Moderator tells player how many colors are:
  - Correct and in the right position
  - Correct but in the wrong position
- Continue until the player has guessed the right pattern of colors

In the following homework description, a program by the name of "chokechain" is mentioned. This is a Linux-based console debugging tool which connects to an AIBO via TCP/IP. Students can use this program to view debug print statements (referred to in the homework as "pprintf"), as well as color-thresholded frames of AIBO video.

Introduction

This homework is geared towards switching between different behaviors (state machines in state machines) and also contains an opportunity to experiment with learning. There will also be some basic vision. You will be playing the game **Mastermind** with the dogs. If you're not really familiar with the game, check it out here:
http://www.kongtechnology.com/index.asp?im=mastermind
You can see the rules and play.

Game Setup

You will be playing a simplified version of the game with colored squares. There will be two positions and three colors to choose from. Colors can be repeated in the slots. The moderator will pick a sequence while the player will try to use the responses from the moderator to guess the sequence. You will interact with the dogs through simple motions, buttons, and LEDs.

Part 1: AIBO as Moderator

In the first part, the AIBO will have to come up with a random sequence of two colors and you will have to guess it. Here's how the game will go:

- Press the back button to unpause the robot.
- The AIBO starts in the play position, in this case a resting position on its belly.
- You press the button under its chin to start the game. It will select a random sequence and blink both middle LEDs twice to indicate the start of game play. It will also pprintf this sequence so that we can see the sequence using chokechain.
- You will then place two squares in front of it (your choice of colors) at a distance specified by the tape marks on the field. When you are ready for a response from the AIBO, you will press the back head button.
- The AIBO will see the two balls and respond to you by:
  - showing n green LEDs to illustrate the number of squares that are of the correct color and in the right place. (This corresponds to the black pegs in Mastermind)
  - showing m red LEDs to illustrate the number of balls that are of the correct color but in the wrong place. (This corresponds to the white pegs in Mastermind)
  - Nodding the head to let you know it is done evaluating your choice.
- You will continue to place squares and then request a response with the head button until you guess the right sequence.
- The AIBO should roll over when you have guessed the correct sequence (or you can use some other cute motion instead).

Part 2: AIBO as Player

Now it is the AIBO's turn to play. You will now have to implement the logic for Mastermind formally in the AIBOs. This will require keeping some history of what the AIBO has already seen and the responses it got. Here's how the game will go:

- Press the back button to unpause the robot.
- The AIBO again starts in the play stance. Now, you will come up with a sequence of colors and press the button under its chin to let it know you're ready to play.
- You will retrieve the AIBO's guess by showing it different colored squares and eliciting a position response for each square. Specifically:
  For each square of color C= C1, C2, C3
  - Show the AIBO the square C
  - If it wants square C, it will nod "yes" (up and down). It will also raise its right or left paw to indicate if it wants the square in the left or right position. If it wants the color in both slots, it will raise both paws. If it does not want color C, it will shake its head "no" (left and right).
- You will then give feedback to the AIBO by:
  - pressing the rear button on its head to tell it "start input"
  - pressing its left front footpad n times, where n is the number of squares of the correct color in the correct position
  - pressing its right front footpad n times, where n is the number of squares of the correct color in the wrong position
  - pressing the rear button on its head to tell it "stop input"
- You will repeat this sequence of square guesses and input until the AIBO correctly guesses the sequence. Then, you should press the button on its back to tell it that it has guessed correctly. It should roll over again or do some cute thing.

Regarding the code, the only new thing will be the vision. The real challenge will be to manage lots of states with lots of history. The vision code will be provided for you; you will not have to calibrate or do anything fancy. We expect you to simply use our code. You are advised to test the behavior in stages rather than doing it all at once. (In the case of AIBO as moderator,

you could have a "getMastermindGuess" behavior and a "respondToMastermindGuess" or something). This will be easier to debug.

Because some AIBOs might have broken foot buttons, please be sure to check the list inside the cabinet to make sure that you don't use one that has malfunctioning buttons. When testing your program, you might try lighting up an LED every time a footpad has been pushed to be sure that it works.

Questions

Playing Mastermind with 3 colors and 2 slots is pretty easy. But suppose we want you to play a much harder game of some n colors and m slots. Answer the following questions related to candidate elimination.

- As we discussed in recitation, candidate elimination by enumerating all the values can become ridiculous for large problem spaces. For arbitrary m and n, how can you perform candidate elimination concisely?
- With large problems, it is desirable to guess those hypotheses that will get you the most information and thereby reduce the total number of guesses to win. Explain how you would choose a hypothesis from your remaining set.

Figure 8: Mastermind: the AIBO on the left is the moderator and the AIBO on the right is the player. The human manipulates the colored squares for the player. In this picture, the player has notified the human that it would like to place the colored square on the left.

Class Final Project

Students in the 2003 and 2004 CMRoboBits courses were required to complete a final project worth 30% of their overall grade. In 2003, the students were asked to propose a project which would demonstrate some interesting behavior of the AIBOs, whereas in 2004, the students were all required to do the same project. We felt that both approaches had specific advantages. The open-ended 2003 project encouraged creativity and allowed students to explore what they learned and enjoyed most in the course, while the fixed 2004 project tested the student's complete knowledge of the course materials.

2003 Final Project

In 2003, the students turned in a written proposal of their project and then were coached by the professor to make sure that their project was feasible in the time remaining in the course. Because the projects were so open-ended, we felt that this encouraged the students to express their creativity and do a project that was interesting to them. The resulting scope of the final projects in 2003 was very broad. Some of the more interesting projects included:

Obstacle Course: An AIBO was programmed to navigate an obstacle course that consisted of an archway, a straight corridor, a tunnel, and a step. The robot was programmed to identify each obstacle by recognizing a colored strip on the ground. When encountered, the robot would perform a custom motion to move past the obstacle.

Figure 9: Student final project: an AIBO navigating through an obstacle course.

Tag: This project made use of localization and inter-robot communication to allow two robots to play a game of tag. Both robots used the six landmarks placed around the carpet to localize themselves. They would use this information to transmit their (x, y) locations to each other. With this shared information, the robot that was "it" would try to come within a minimum distance of the other robot while that other robot tried to evade. When the minimum distance was reached, the AIBOs switched roles.

Maze Learning: An AIBO was programmed to explore a maze of t-junctions and dead-ends using the robot's visual sonar sensor module to identify distances to the maze walls. The robot was started at

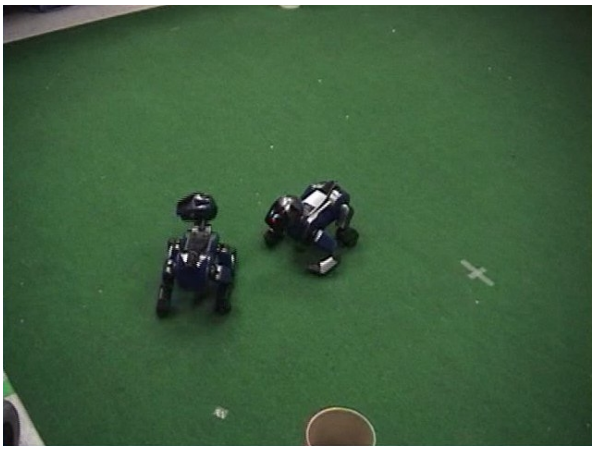

Figure 10: Student final project: two AIBOs playing tag.

one part of the maze and had to explore until it found an orange ball at the end. Whenever the robot encountered a t-junction, it would remember the last direction that it took through it. After the goal had been reached, the AIBO could be restarted at the beginning and would always take the same correct path through the maze to the goal.

Figure 11: Student final project: an AIBO learning a maze through exploration.

AIBO Dance: Two AIBOs were programmed with a set of choreographed motions to dance in time to a custom song mixed by the students. This project demonstrated the wide range of motions that are achievable by the AIBO. Some motions included flipping over to walk on their backs, sitting and waving both front paws in the air, and rolling completely over.

Figure 12: Student final project: two AIBOs dancing in time to music.

2004 Final Project

Inspired by the variety of student projects from 2003, some of the more interesting aspects from those projects were selected and merged to create a final project for the 2004 class. This final project tested the student's cumulative knowledge of the AIBOs and the CMRoboBits source code by having two AIBOs cooperatively find their way through a maze. Two identical mazes were constructed side-by-side, as shown in Figure **??**.

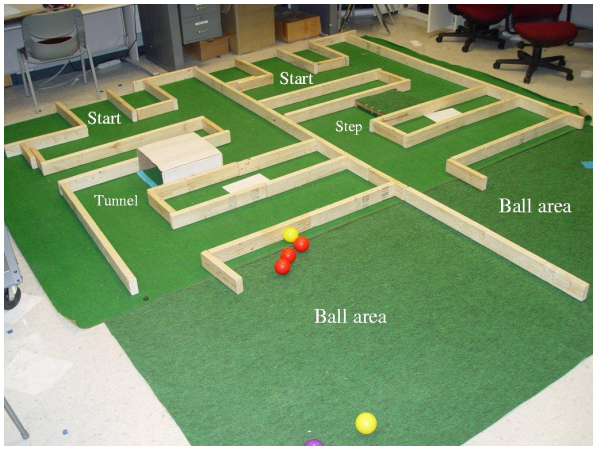

Figure 13: Final project maze in 2004

Each AIBO would be started in its own maze and both would have to find their way through to the end. Inside each maze was an obstacle that the AIBO would have to either climb over, crawl under, or crawl through. At the end of the maze was an open area filled with colored balls that the robots had to cooperatively empty. The emphasis on this project was on navigating the maze, overcoming obstacles with motions, and cooperating through wireless communication.

Cumulative Skill Testing For the final project, students had to make use of their knowledge of both single and multi-robot behaviors.

Single robot behaviors

- Navigation and obstacle avoidance
- Visual recognition and interpretation of landmarks
- Motions to climb over obstacles
- Visual navigation to colored targets

Multi-robot behaviors

- Synchronization of world model for maze navigation
- Negotiation and agreement on shared action for ball task

Navigation in the Maze In order to navigate the maze, the AIBOs were given clues at each of the t-junctions as to which direction they should turn to reach the end of the maze. These clues consisted of two-color signs that the robots would have to interpret. If the sign was tilted to the left, the robot should take the left path. If the sign was tilted ton the right, the robot should take the right path. If the sign was horizontal, the robot should turn around and take the path behind them. In order to read the signs, the robots would be required to face the sign so that the orientation had the correct meaning.

Cooperation in the Maze In order to encourage cooperation between the robots, not each t-junction in a specific robot's maze would have a sign. However, between the two mazes, each t-junction would have a sign. Therefore, students could program their AIBOs to communicate with each other to share the information as to which direction they should turn when they arrived at the same t-junction.

Cooperative Cleaning When both robots found their way through the maze, they were required to work together to remove all of the colored balls from an open area (each robot had its own separate area). In order to remove a ball, both robots would need to navigate to a ball of the same color and stop with their head over it. If both balls were the same color when the robots stopped, then they would both be removed and the robots could continue collecting. If the robots stopped over two balls of different colors, the balls would be moved to a new area and the robots would have to continue until they found two balls of the same color. Communication was necessary in this case since the robots were unable to see each other's activities.

Future Final Projects

The final project in 2004 required a firm grasp of the cumulative course materials. However, for 2005, the final project will include a learning component whereby the robots will have the opportunity to improve their performance through experience. We will offer the same final project to all students again, but will allow exceptional students to propose projects. These project proposals will be accepted so long as they exhibit the same breadth of coverage over the course materials.

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.

We seek to find a good balance between the theoretical aspects of the in-class lectures and the hands-on labwork. We feel very strongly that both are required to achieve a well-rounded learning experience. As of the 2004 class the students did not have any in-class midterm or final exam in lieu of a more in-depth final project, we have decided to include them the next time the course is taught. There are too many concepts to fit into a cumulative final project. A more traditional exam schedule will fill this gap. Currently, all of our software on our web page runs only on the AIBO ERS-210. Future work includes making the necessary changes to run the software on the new AIBO ERS-7s which have higher resolution cameras, slightly different kinematics, and more processing power.

The current course materials, including some final project videos, are available on the course Web page listed in Figure **??**.

Acknowledgements

We would like to thank Sony for their remarkable support of our research, specifically by making the AIBO robots accessible to us since their first conception in 1997. Sony has continued their support through these years, and is very interested in following the impact of an AIBO-based course. We would also like to thank the Carnegie Mellon Computer Science Department for approving this new course in the Fall of 2003 and providing the lab facilities.

Web site	URL
Course web page for the current year	http://www.andrew.cmu.edu/course/15-491/
Archived course materials from previous years	http://www.cs.cmu.edu/~robosoccer/cmrobobits
CORAL research group web page	http://www.cs.cmu.edu/~coral
SONY OPEN-R SDK home page	http://openr.aibo.com/
AIBOs at Robocup	www.robocup.org
Tekkotsu - another AIBO programming API	http://www.cs.cmu.edu/ tekkotsu

Table 1: Web resources for using AIBOs in education