Team Description: UW Huskies-01

D. Azari, J.K. Burns, K. Deshmukh, D. Fox, D. Grimes, C.T. Kwok, R. Pitkanen, A.P. Shon, and P. Tressel

Department of Computer Science & Engineering
University of Washington, Seattle, WA
{azari, jkburns, kaustubh, fox, grimes, ctkwok, pitkanen, aaron, tressel}@cs.washington.edu

1 Introduction

This paper provides an overview of the robot control system of the UW Huskies, our entry in the RoboCup-2001 Sony AIBO legged robot league. Our development team consisted of one faculty member (D. Fox, team leader), one technical support staff (R. Pitkanen), one undergraduate student (J.K. Burns), and six graduate students. Since this was our first participation in RoboCup, we relied on the implementation of well-established techniques for robot control. These techniques include a layered control system, particle filters for robot localization, and efficient computer vision techniques for object recognition.

2 Architecture

Flexibility was the overriding design goal for our system. We were especially careful to make our system as modular as possible. This would permit us to adapt to unanticipated difficulties or novel tactics. Implementation efficiency and maintainability were important secondary goals. Figure 1 illustrates the control system. The different modules communicate using the Aperios inter-object message passing facility. When large data (e.g. world model, camera images) are to be shared between multiple modules, shared memory is used. The control system consists of the following five modules:

Hardware Interface: This module receives input from the raw sensors and passes it to the other modules. The most important information consists of time stamped camera images coupled with the corresponding position of the robot's head.

Vision: The vision module analyses images on a frame by frame basis. Efficient algorithms are applied to detect objects such as the ball, goals, and markers. Distances to these objects are extracted from their size. Information about the robot's head position is used to transform from camera-relative to robot-relative coordinates. The detected objects are passed to the state estimator module.

State estimator: This module integrates information over time and stores it in the world model. Most important information is the position of the robot on the soccer field and the position of the ball relative to the robot. The robot position is estimated using a particle filter approach [2]. The position of the ball is based on smoothed estimates of the vision system. From this information, it is rather straightforward to derive information such as the absolute position of the ball, or the relative position of the opponent's goal.

A. Birk, S. Coradeschi, and S. Tadokoro (Eds.): RoboCup 2001, LNAI 2377, pp. 721–724, 2002.

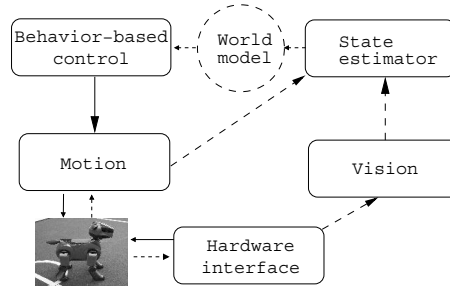

Fig. 1. System architecture. The arrows indicate the main flow of information (dashed) and commands (solid) between the different modules.

Behavior: Using the most recent information in the world model, this module decides which actions the robot should perform. Our control system follows a multi-layered approach (see also [3] and several chapters therein). Behaviors are modeled by hierarchical finite state machines (FSMs). Simple skills such as "kick" or "walk" form the basic skills of the hierarchy. At the top of the hierarchy a "goalie" or "forward" FSM determines the highest-level actions of the robot depending on a preassigned role.

Motion: We developed gaits and kicks offline by a combination of basic inverse kinematics and direct manipulation of the robot's joints. These gaits are saved on the memory stick as text files and read in at boot time. The motion module executes commands at the request of the behavior module. During the competition we relied on the gaits provided by Sony's OPEN-R robot control system.

3 Vision

Our basic vision package emphasizes efficiency and interface simplicity to maximize reuse of the code. The package possesses a memory footprint of less than 2MB and allows a frame rate of more than 100 frames per second on the robot itself (significantly faster than the camera update rate), including color clustering, blob identification, object recognition, and distance estimation. To achieve this high efficiency in combination with accurate object estimates, we follow a dual-resolution approach: Object detection is performed on the low-resolution images which are color-clustered by the robots vision hardware. Once the different objects are detected, we use the high-resolution image to get more accurate estimates for the positions and sizes of these objects.

The image processing pipeline begins with the hardware-clustered CDT image. The clusters are represented by rectangular boxes in YUV-space. These boxes are determined manually, using an offline tool that reads in images taken from the camera and allows a trainer to assign groups of pixels to specific color classes. During operation, the classified image pixels are grouped into blobs. We use a tree-based union find algorithm with path compression and 4-connectedness matching to locate blobs. The blobs of each color are sorted by size, and very small blobs (likely to be noise) are eliminated. Objects are detected by testing spatial relationships between the individual blobs (e.g. green blob above yellow blob for green/yellow marker).

All operations described so far are performed on the low-resolution CDT images. To get more accurate estimates for object sizes, we use the bounding boxes of the detected objects to focus on regions in a higher-resolution image. Color clustering of the higher-res images is performed in software using the same color bounding boxes as the CDT images. We found that the higher-resolution images tend to yield more accurate distance estimates, while the low-res pass allows reliable and efficient object detection[1]. Finally, before sending object information to the state estimator, the object positions are transformed from camera coordinates to positions relative to the robot.

4 State Estimation and World Model

The task of this module is to keep track of the dynamics of the environment. In our current system, each robot estimates its own position and the relative positions of the ball and the goals. The relative position of the goals is either determined by direct observation or extracted from the robot's location on the field. We estimate the robot position using particle filters, a highly efficient and robust approach to localization [2]. The key idea of this probabilistic approach is to represent the robot's position by sets of random samples. Position estimates are refined whenever the vision module detects a marker or a goal. Detections contain information about the distance and orientation of the objects relative to the robot. From these detections we can derive probabilities for the different robot positions on the field. Whenever the robot moves, the samples are shifted according to a model of robot motion. The noise parameters of this motion model were determined offline by measuring velocities for the different motion commands. Similar to [4], recovery from localization failures is achieved by adding random samples drawn according to the observation likelihoods.

Efficiency was one of the key goals of our implementation of particle filters. Using several approximations (such as pre-computing random samples and replacing Gaussian distributions by triangular distributions), one update cycle of our approach can be done in less than 0.05 seconds on average (400 samples).

5 Behaviors

This module receives world model updates from the state estimator, processes the information, and outputs commands to the motion module. The behavior module is implemented as a hierarchy of finite state machines. We have two top-level state machines, each corresponds to a different type of player, the Forward and the Goalie. A simplified Forward state machine is shown in Figure 5(a). Each state has a list of transitions and conditions associated with these transitions. Typical conditions test whether something holds true in the world model, for example, whether the ball is close to the robot. If true, the associated transition is made, during which motion commands may be issued. Each state in the state machine implements an abstraction called SKILL. Basic SKILLs

[1] To achieve more accurate estimates of the distance to the ball, we additionally use information from the IR sensor when the ball is already judged by the vision system to be relatively close and straight ahead.

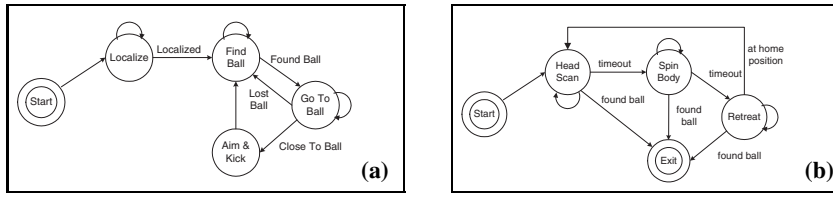

Fig. 2. Hierarchical finite state machines for robot control: a) Simplified version of the FSM implementing the Forward robot. Note shown are transitions from each state to the "Localize" state. The "Go to Ball" state is implemented by a docking motion field to guide the robot to the ball so that it can kick it toward the opponent's goal. b) The Find Ball skill of the Forward is implemented as a lower level FSM. When the exit state of this FSM is reached, control is transferred back to the main state machine.

are motion commands such as move head or walk. More complicated SKILLs are built out of state machines themselves, each state again corresponds to a SKILL.

Acknowledgments

This research is sponsored in part by NSF (contract number 0093406). The team wishes to thank Wolfram Burgard for his support and advice during the last weeks of preparation for RoboCup-2001.

6 Conclusions and Future Work

Our software system for RoboCup-2001 relied on well-established techniques from the robotics and vision community. Due to the lack of preparation time, our team was not competitive against the senior teams of our league. Our major problem was the fact that we were not able to develop fast and stable motion patterns within the given time. Since fast motion is of utmost importance for success in robot soccer, our next efforts will focus on the development of motion patterns. Another thread of our research will explore issues in multi-robot collaboration and the combination of information collected by different robot platforms (communication will be available in RoboCup-2002). Further research will involve reinforcement learning and efficient state estimation in the context of real-time robot control (see e.g. [1]).

References

1. D. Fox. KLD-sampling: Adaptive particle filters and mobile robot localization. In *Advances in Neural Information Processing Systems (NIPS)*, 2002. To appear.
2. D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte Carlo Localization: Efficient position estimation for mobile robots. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 1999.
3. D. Kortenkamp, R. P. Bonasso, and R. Murphy, editors. *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems*. MIT/AAAI Press, Cambridge, MA, 1998.
4. S. Lenser and M. Veloso. Sensor resetting localization for poorly modelled mobile robots. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2000.