Description of Team Dynamo-Pavlov Uppsala

Jens Alén, Olle Gällmo, Rikard Hansson, Rani Khalil,
Maria Olsson, Paul Pettersson*, Arsenij Vodjanov,
Samuel Waxin, and Joakim Örtbrant

Department of Information Technology, Uppsala University
P.O. Box 337, S-751 05 Uppsala, Sweden.

April 23, 2003

Abstract

Team Dynamo-Pavlov Uppsala is an effort at the Department of Information Technology at Uppsala University, aimed as an serious attempt to accept the robocup challenge to making Sony AIBOs play soccer.

1 Introduction

Team Dynamo-Pavlov Uppsala was founded in the fall of 2002, when a group of 25 computer science students started their fourth-year projects at the Department of Information Technology at Uppsala University. Their task was to complete two larger assignments using Sony AIBO 210A as the hardware platform. As one of the assignments, 13 students start the development of a Sony AIBO soccer team to compete in the legged league at German Open and Robocup.

1. Research interest: The research direction in the project is mainly influenced by the research areas of the two lecturers Ph.D. Paul Pettersson and Fil.Lic. Olle Gällmo, which include modelling and analysis of real-time system, software testing, and learning systems. In addition, the development work has involved other computer science, in particular distributed systems and image analysis.

2. Proposed approach to address the RoboCup challenge: Our current software is described in Section 2 of this paper.

3. Background of the principal investigator: The curriculum vitae of the lecturers Ph.D. Paul Pettersson and Fil.Lic. Olle Gällmo can be found at the web page `http://user.it.uu.se/~paupet/gu/projdv02/rc03-application/`.

*Team leader. Email: {`Paul.Pettersson`}`@it.uu.se`

4. Description of the team organization and effort to be spent: The long-term plan in terms of organisation is to annually have groups of fourth-year computer science students working on projects further developing team Dynamo-Pavlov Uppsala.

5. Pointers to relevant publications: The publications of the lecturers Ph.D. Paul Pettersson and Fil.Lic. Olle Gällmo can be found in their curriculum vitae.

6. Statement of commitment to enter RoboCup-2003 in Padova, including traveling expense and registration fee for participation: All necessary funding for registration fee etc. have been granted, the travel has already been organised. We are currently in the process of reserving hostel rooms for the students. In addition, we have organised with Sony to change the shell-parts of eight of our AIBOs, to comply with the rules of RoboCup 2003.

2 Overview of Completed Work

Currently, the system is divided into six modules. Each module them run as a single OObject with message passing communication. The the following we give a brief description of each module.

2.1 Module Vision

The vision module is responsible for analysing the images taken by the camera in the AIBO's nose. Once a picture is taken, the AIBO will try to find key objects on the field, such as the ball, a goal or other players. When the image has been analysed, the gathered data is sent to the strategy module which will work out what to do with the information.

It also tries to determine distance to these objects as well as assertain its own position on the field. Localisation is done by triangulating its position with the six uniquely colored beacons around the field.

From AIBO's camera we get an image in YCrCb–color-space. This image is then run through a BitMask Classifier that gives us a new image with a more convenient color-space. The classifier strips off uninteresting colors, other colors are classified in the sense that we are told what kind of object this color can be associated to.

The new image is then given to an edge–detection function that returns a bitmap containing the edges of the given image. The image and the edge–bitmap are the base for the next step in our object detection, the flood-filling. The flood-fill–function is not only a flood-fill–algorithm though, it has evolved to an object–detector and as it fills an object in the image it gathers information about the object's height, width, color, position, area and perimeter. Very small objects are thrown away, as they contain too much uncertain information. Objects that seem to form a beacon are combined.

2.1.1 Distance

Approximation The distance from the robot to objects are all approximated based on the size of different properties of these objects. The size is expressed in number of pixels of each object in the picture. The approximation uses a mathematical function which is calculated from the relationship between object size and real distance.

- Ball
  The size of the ball is based on its width. This is because it is common that the top of the ball is to bright to be recognized as the ballcolor, which is partly due to our lighting conditions. So the height of the ball will most of the time be non accurate, and hence the width is better suited to represent the size of the ball.
- Beacon
  The size of each beacon is based on the euklidian distance between the center of the two beacon parts. We first used the height and width of each beacon part, but the euklidian distance d seems to give better distance approximations.
- Goal
  The goal size is equal to its height, because the AIBO is more likely to see the whole height than the whole width, when there are other robots obstructing the view.

Recalibration Our distance approximation to objects is quite good at close range. But due to the poor resolution of the camera picture the approximation doesn't give us distances that are accurate enough when object are far away from the robot. This inaccuracy sometimes makes the localization of the robot unsatisfactory, since it is based on the distances to visible beacons. The solution is to use *triangulation*.

When the robot sees three landmarks, either beacons or goalposts, we can use triangulation to recalibrate the distances to these landmarks. This will enhance the accuracy of the localization.

2.2 Module Strategy

It receives data continuously, evaluates that data and determines a behaviour that it wants to perform. This behaviour is then sent on to the motioncoordinator module.

2.2.1 Behaviours

The strategy module has a set of commands that it can send to the motioncoordinator:

- GoToBall, *the AIBO goes to the position of the ball*

- GoToXY, *requires a point to be given*
- Aim, *requires an angle that the AIBO should spin*
- KickBall, *the AIBO kicks the ball*
- LockBall, *the AIBO lowers its frontlegs and head to lock the ball*
- PushBall, *the AIBO pushes the ball lightly*
- ScanField, *the AIBO scans the field without moving its body*
- SpinBody, *requires an angle that the AIBO should spin*
- MakeSave, *the AIBO makes a save*
- ContinuePrevious, *if the new calculated action is the same as the previous one, we don't send the command again, but send a ContinuePrevious instead.*
- Stop, *stops the current command*

2.2.2 Decision trees

The behaviour is determined using a decision tree that consists of a hardcoded binary tree. This tree looks slightly different depending on wheather the AIBO is a goalkeeper or an ordinary player.

2.3 Module Worldstate

The Worlstate module is our map module, it handles and stores the data received from the vision- and the communication module. The data received from the vision module is a message that contains what the vision module has seen. The first thing we do is that we convert this message into a worldstruct. The worldstruct contains all objects allowed on the field. We the send a copy of the message to the strategy module.

The worldstate module doesn't only get data from the vision module. Since the dogs cooperate and have a boss they also send messages, i.e their own worldstruct, to the boss. The messages are sent via the communication module. When the boss receives messages from the dogs he combines the data from the different players and broadcasts the new combined data to all the dogs. The idea with this is that each dog should have the same view of the gamestate.

2.4 Module Motion Coordinator

Motion Coordinator module acts as an abstraction layer between Strategy and the two motion modules (head– and leg– motion). A finite set of commands (with parameters) can be sent in sequence from Strategy to Motion Coordinator. Motion Coordinator's task is to convert these abstract strategic actions into lower–level commands understood by the motion modules, and deliver these commands in sequence to the appropriate motion module.

2.5 Module Motion

The Motion module is responsible for translating commands from the Motion-Coordinator module into movements. This includes walking, rotating, moving the head, kicking, saving etc.

2.5.1 Walking style

We have chosen to use a walking style known as *trot*. In this walking style two legs move simultaneously which makes the walking style quite fast.

The basic idea is that we specify a rectangle in the 3D-room in which each paw should move. We could then choose points in the rectangle and transfer the coordinates of those points into joint angles by using inverse kinematics.

The coordinate system is individual for each leg and origo is placed in the shoulder. This method can be used both for walking forward, backward, sideways (strafing) and rotating. We choose to divide the rectangle into six positions, one in each corner and another two at the bottom. This will ensure that the time in which the paw is lifted, is equal to the time in which the paw is dragged along the floor. All positions are relative to the AIBO's starting position.

2.6 Module Communication

The communication module handles network connections for the AIBO. It uses the ANT (abr. for OPEN–R Network Stack) for TCP/IP communication with other team members and our debug tool.