SPL Portuguese Team: Team Description Paper for RoboCup 2011

A. J. R. Neves[1], N. Lau[1], Luís Paulo Reis[2], António P. Moreira[3], Alina Trifan[1], Bruno Pimentel[1], Carlos Sobrinho[1], Edgar Domingues[1] . . .

[1]IEETA/DETI – University of Aveiro, 3810-193 Aveiro, Portugal
[2]LIACC/FEUP – Faculty of Engineering of the University of Porto, Portugal
[3]INESC-P – University of Porto, Portugal

Abstract. Portuguese Team intends to participate in the Standard Platform League at the RoboCup 2011 for the first time with a team composed by members of the three Portuguese teams that achieved best results in RoboCup European and World championships: FC Portugal (Simulation 2D, Simulation 3D, Coach Competition, Rescue Simulation, Rescue Infrastructure and Physical Visualization/Mixed Reality), CAMBADA (Middle Size League and RoboCup@HOME) and 5DPO (Small-Size and Middle-Size leagues). Researchers from two different universities (University of Porto and University of Aveiro) in Portugal join efforts to build a new research oriented and competitive SPL team. Concerning scientific results achieved by the three teams, its members have a combined publishing rate of more than 100 papers regarding Robotic Soccer and related issues in international journals and conferences. The members have had an increasing publishing rate during the last years. For example, in the last two RoboCup Symposiums (2009 and 2010) the team members published a total of 7 papers. Based on previous experience, Portuguese Team developed an agent similar to the one developed for FCPortugal 3D and uses a similar distributed architecture as the one used in the CAMBADA robots. This distributed architecture is based on several processes, namely Communications, Vision and Agent, centered in a Real-time database. The agent is based on several modules, each one with a specific purpose: WorldState, AgentModel, Geometry, Optimization, Skills, Utils, Strategy and DeviceManager. This paper describes the Portuguese Team for the purpose of the qualification for RoboCup'2011.

1 Introduction

Portuguese Team is a RoboCup Standard Platform League (SPL) team composed by members of the three Portuguese teams that achieved best results in RoboCup European and World championships: FC Portugal, CAMBADA and 5DPO.

FC Portugal is a joint project of the Universities of Porto and Aveiro from Portugal. The team has been participating in several RoboCup competitions since 2000, including: Simulation 2D, Simulation 3D (humanoid simulation), Coach Competition, Rescue Simulation, Rescue Infrastructure and Physical Visualization/Mixed Reality. The team's research is mainly focused on creating new coordination methodologies such as tactics, formations, dynamic role exchange, strategic positioning and coaching. The research-oriented development of FC Portugal has been pushing it to be one of the most competitive over the years (Simulation 2D World champion in 2000 and European champion in 2000 and 2001, Coach Champion in 2002, 2nd place in 2003 and 2004, Rescue Simulation European champion 2006, Simulation 3D European champions in 2006 and 2007 and World Champions in RoboCup 2006).

CAMBADA is the Middle-Size League RoboCup team and project from IEETA/University of Aveiro that started in 2003. The team's main research interests cover most of the Middle-Size League challenges such as robot vision, sensor fusion, multi-agent monitoring, and multi-robot team coordination. The team won RoboCup 2008, achieved 3rd place in RoboCup 2009 and RoboCup 2010 World MSL competitions and was 2nd in the RoboCup German Open 2010. This team will participate for the first time in the RoboCup@HOME competition at RoboCup 2011.

5DPO is a project of INESC-P/FEUP aiming at researching in topics such as vision based self localization, data fusion, real-time control, decision and cooperation. The team is active in RoboCup since 1998 in the Small-Size and Middle-Size leagues. The team's results have been quite good in the Small-Size League. 5DPO won three European/GermanOpen Small-Size League championships and achieved a 3rd place at RoboCup 1998 and a 2nd place at RoboCup 2006 world championships in this league.

The Portuguese Team, besides several PhD senior members, with strong background in cooperative robotics and a large number of RoboCup participations in the context of the previously mentioned teams, includes several MsC students and PhD Students working with biped walking, robotic vision, sensor fusion, coordination in multi-robot systems, cooperative planning of tasks and trajectories in multiples robots.

This paper describes the current development stage of the team and is organized as follows: Section 2 describes the general architecture of the software running on the robot. Section 3 describes the necessary changes made in the FCPortugal 3D agent to the one used in the NAO robot. Section 4 addresses the research done in the walking behaviors. Section 5 describes the vision system developed for the NAO robot and finally, Section 6 presents some conclusions.

2 Agent architecture

The software developed for the Portuguese Team robots uses a similar distributed architecture as the one used in the CAMBADA robots [1]. This distributed architecture is based on several processes, namely Communications, Vision and Agent, centered in a Real-time database (RtDB).

Following the CAMBADA software approach, the software used in the robot is also distributed. Therefore, five different processes are executed concurrently. All the processes run on the robot's processing unit in Linux.

Inter-process communication is handled by means of a RtDB which is physically implemented as a block of shared memory. The RtDB is divided into two regions, the local and the shared one. The local region allows communication between processes running on the robot. The shared region implements a Blackboard communication paradigm and allows communication between processes running on different robots. All shared sections in the RtDB are kept updated by an adaptive broadcasting mechanism that minimizes delay and packet collisions.

The processes composing the Portuguese Team robot software are:

- **Vision**: is responsible for acquiring the visual data from the robot cameras.
- **Agent**: is the process that integrates the sensor information and constructs the robot's worldstate, taking the decisions based on this information .
- **Communications**: handles the inter-robot communications, receiving the information shared by other robots and transmitting the data from the shared section of the RtDB.

Portuguese Team agent is similar to the one developed for FCPortugal 3D. The agent is based on several modules, each one with a specific purpose: WorldState, AgentModel, Geometry, Optimization, Skills, Utils, Strategy and DeviceManager.

- **WorldState**: Contains classes to keep track of the environment information. These include the objects present on the field (fixed objects as is the case of flags and goals and mobile objects as is the case of the players and the ball), the game state, (e.g. time, playmode) and game conditions (e.g. field length, goals length);
- **AgentModel**: Contains a set of classes responsible for the agent model information. This includes the body structure (body objects such as joints, body parts and perceptors), the kinematics interface, the joint low-level control and trajectory planning modules;
- **Geometry**: Contains useful classes to define geometry entities as is the case of points, lines, vectors, circles, rects, polygons and other mathematical functions;
- **Optimization**: Contains a set of classes used for the optimization process. These classes are a set of evaluators that know how each behavior should be optimized;
- **Skills**: This package is associated with the reactive skills and talent skills of the agent. Reactive skills include the base behaviors as is the case of walk in different directions, turn, get up, kick the ball and catch the ball. Talent skills are some powerful thinking capabilities of the agent, which include movement prediction of mobile objects in the field and obstacle avoider;
- **Utils**: This package is related with useful classes that allow the agent to work. This includes classes for allowing the communication between the agent and the server, communication between agents, parsers and debuggers.

– **Strategy**: Contains all the high-level functions of the agent. The package is very similar to the team strategy packages used for other RoboCup leagues.
– **DeviceManager**: Our approach to allow the agent architecture access the robot hardware, like many others SPL teams, was to create a simple NaoQi module to communicate with the Device Communication Manager (DCM). This module reads the actuators values from the shared memory, sends them to be executed by the DCM, and copies the sensors values from the DCM to the shared memory. By accessing the shared memory, our agent becomes independent from the Aldebaran software.

3 Humanoid Behaviors: From Simulation to a Real Robot

The architecture of the agent that uses the shared memory is identical to the agent used in the simulation league. The only modifications made were in the low level communications. In simulation, the agent communicates with the server to send the actuator values and receive the sensor values. In the real robot, instead of communicating with a server, the agent sends the actuator values and reads the sensor values from the shared memory. The real robot has a PID controller for each joint as opposed to the simulated robot, whose PID controllers must be implemented by the user. Therefore it is unnecessary to copy this software PID controller from simulation to the real robot.

3.1 Behavior models

The high level behaviors can be ported from the simulated robot to the real robot with no modification, as long as the low level behaviors are developed for both the simulated and the real robot. When adapting the behaviors from simulation to a real robot we tried to keep the architecture of the agent similar on both. Since all behaviors on simulation produce joints angles which are then passed to a software PID controller, this was replaced with some software to send these angles to the shared memory (as explained on Section 2), and convert them to the angle range of the real robot joints if needed.

Slot Behaviors

A Slot Behavior is defined by a sequence of slots. Each slot has a time duration and a target angle for each joint to be controlled. When a slot is executed the joints are moved with a sinusoidal trajectory, from the angle they have at the beginning of the slot, to the target angle that is achieved at the end of the slot. The sinusoidal trajectory is used since the initial and final speed are zero, and it assures the lowest second derivative maximum, hence the acceleration will be minimized [2]. This produces a smooth motion for the joints. Slot behaviors are defined in Extensible Markup Language (XML) files, so they can be easily manipulated without need to recompile the agent.

Adapting the Slot Behavior algorithm was easy, since it only generates joint trajectories based on XML files. Only the behaviors themselves needed to be adapted.

The get up behaviors (after falling forward and backward) were developed from scratch because the get up behaviors used on the simulation execute motions that are not possible on a real robot. The sequence of poses of our get up behaviors were based on Aaron Tay's thesis [3] and on the B-Human 2010 Code Release.

The kick forward behavior was adapted from simulation and had to be changed to be stable on the real robot. Some slot durations were increased to make the behavior slower. The angles of the joints that move on the coronal plane (hip roll and ankle roll) were also increased, so the center of mass is better shifted to the support foot.

A kick to the side that did not exist on the simulation was also created.

CPG Behaviors

Central Pattern Generator (CPG) Behaviors execute, on each targeted joint, a trajectory defined by a sum of sine waves [4]. Each sine has four parameters: amplitude, period, phase, and offset. Like Slot Behaviors, CPG Behaviors are defined using XML files, to be easily manipulated without the need to recompile the agent.

The CPG behavior model was used to create periodic behaviors: walk, turn in place, rotate around the ball, etc.

In the CPG Behaviors, like the Slot Behaviors, the algorithm itself was easily adapted but the behaviors needed to be changed. The common modification was to slowdown the behaviors and reduce the joint amplitudes. However, each behavior needed to be independently adapted to the real robot.

4 Walking

Walking is one of the most important basic behaviors. In our robots we adapted the walking behaviors developed for the FCPortugal 3D team.

4.1 OmnidirectionalWalk Behavior

This behavior, based on the Sven Behnke's walk [2], produces the desired motion calculating the trajectories of three parameters for each leg, and then uses the Leg Interface to convert these parameters into joint angles. To adapt this behavior from the simulation to the real robot we modified the Leg Interface according to the dimensions of the real robot.

A difference from the Nao robot to most humanoid robots is that the hip yaw pitch joints in both legs are controlled by a single motor. This means that any rotation is applied to both joints at the same time. In the simulation the two joints can be controlled independently. So, when adapting this walk from simulation to the real robot, we need to combine the two motions into one that produces the desired stable gait. When, in the simulation, the two motions are equal, we can apply them directly to the real robot. When these two motions are different, the task is more difficult and each specific case must be analyzed.

For this walk each leg is composed by two kinds of movements. A sinusoidal movement, when the leg is in the air, and a linear movement, when it is on the ground. The two motions are significantly different and cannot be directly executed by the common joint of the real Nao robot. We needed to combine the two motions creating one that is a continuous function (so the joint can follow it smoothly) and that produces a stable walk. Our choice was to follow a linear-like motion which results from selecting the motion of the leg that is on the ground. To do this we switch between left and right leg yaw trajectory followed in the intersection of the two motions when both legs follow linear movements.

4.2 TFSWalk Behavior

This gait was optimized on the simulator, resulting in the fastest gait we have. However, the ground on the simulator has low friction and so the gait learned to use this as an advantage, not lifting the feet too much and almost not using coronal movement. When adapting this behavior to the real robot it only worked on slippery ground. On the official SPL field, the carpet has more friction, making the robot stumble and fall.

The first approach we took to solve this problem was to add coronal movement, allowing the robot to better shift its Center of Mass (CoM), resulting in a more stable gait. The coronal movement used is based on [5]. It consists of rotating the hip roll joint of the support leg (the one on the ground). This lifts the other leg (the swinging leg) from the ground. The ankle roll joint of the swinging leg is rotated with the same angle as the support leg hip, so the foot is always parallel to the ground.

The other approach we took, instead of using coronal movement to shift the CoM, lifts the feet higher and rapidly making use of the robot dynamics to keep the robot balanced. In order to increase the height of the feet trajectories, the knee, hip and ankle joints trajectories should be changed in a coordinated way. However, the specification of TFSWalk, by defining each joint trajectory independently of the others, does not provide a good model for controlling the foot trajectory. To achieve a more controllable model, the TFSWalk specification was converted to use the Leg Interface. With this new model it is easy to control foot height trajectory just changing the leg extension parameter. In addition, the velocity of the robot, controlled by the leg angle amplitude, becomes independent on the foot trajectory height.

As an example, the feet may be kept moving up and down in the same place by setting the leg angle amplitude to zero while keeping the normal value of the leg extension. This up and down movement is very useful to initiate and finish the walking behavior. The increase in the foot height trajectory diminished foot collisions with the ground and resulted in a stable walk without needing coronal movement.

Feedback was also added to make the behavior more robust against external disturbances, such as uneven ground and collisions with obstacles. The feedback is calculated using a filtered value of the accelerometer in the x direction (front). This value is summed to the two hip pitch joints, to balance the torso. This way, when the robot is falling to the front, the accelerometer will have a positive value that is added to the hip pitch joints, making the legs to move forward, to compensate. The inverse is also true, when the robot is falling backwards.

5 Vision of the robot

The architecture of the vision system can be divided into three main parts: access to the device and image acquisition, calibration of the camera parameters and object detection and classification. Moreover, apart from these modules, two applications have also been developed either for calibrating the colors of interest (NaoCalib) or for debugging purposes (NaoViewer). These two applications run on an external computer and communicate with the robot through a TCP module of the client-server type that we have developed. The current version of the vision system represents the best trade-off that the team was able to accomplish between processing requirements and the hardware available in order to attain reliable results in real time.

The camera is accessed using V4L2 API, a kernel interface for analog radio and video capture and output drivers. The V4L2 driver is implemented as a kernel module, loaded automatically when the device is first opened. The driver module plugs into the "videodev" kernel module.

The calibration module is not continuously running on the robot because of the processing time limitations. It is run just once whenever the environment or the lighting conditions change, having the purpose of setting the parameters of the camera so that the images acquired give the best possible representation of the surrounding world.

For the detection process, with the use of a look-up table, and by means of the OpenCV library, the raw buffer acquired by the camera can be converted into an 8-bit grayscale image in which only the colors of interest are mapped using a one color to one bit relationship (orange, green, white, yellow, blue, pink and blue sky, while gray stands for no color). The next step is the search for the colors of interest in the grayscale image, which we call an index image, by means of vertical or horizontal scan lines, and the formation of blobs from pixels that have the same color. The blobs are then marked as objects if they pass the validation criteria which are constructed based on different features extracted from the blobs (bounding box, area, center of mass of the blob). An example of the color segmentation and object detection are presented in Fig. 1.

Having the possibility of running the vision module as a server, the two applications that we have developed, NaoCalib and NaoViewer can act as clients that can receive, display and manipulate the data coming from the robot. Thus, NaoViewer is a graphical application that allows the display both of the original image as well as the corresponding index image containing the validation marks for each object of interest that was found. This application was essential in terms of understanding what the robot "sees" since NAO does not have any graphical interface that allows the display and manipulation of images. Also considering the limited resources of the robot the choice of building a graphical interface on

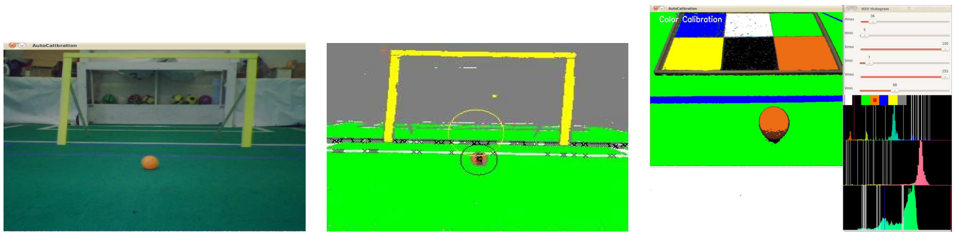
(a) (b) (c)

Fig. 1: In (a), an image captured by the NAO camera. in (b), the same image with the colors of interest classified. The marks over the color blobs represent the detected objects. In (c), an example of the classification of the colors of interest, by means of the NaoCalib application.

the robot was out of the question. NaoCalib is a very helpful application that we developed for the calibration of the colors of interest (see Fig. 1).

6 Conclusions

This paper describes the current development stage of the SPL Portuguese Team for the purpose of the qualification for RoboCup'2011. Most of the work was done in the development of an efficient vision system and on the adaptation of some modules from the software used on the CAMBADA robots. Moreover, the agent from FCPortugal 3D was adapted, mainly the low level behaviors and walking.

The team wants to thanks the support of the institutions involved on the project, IEETA, Universitity of Aveiro, FEUP, INESC and University of Porto.

References

1. Neves, A.J.R., et al.: CAMBADA soccer team: from robot architecture to multiagent coordination. In Papić, V., ed.: Robot Soccer. InTech (2010) 19–45
2. Behnke, S.: Online trajectory generation for omnidirectional biped walking. In: Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, Orlando, Florida, USA (2006) 1597 –1603
3. Tay, A.J.S.B.: Walking NAO. omnidirectional bipedal locomotion (2009)
4. Picado, H., Gestal, M., Lau, N., Reis, L., Tomé, A.: Automatic generation of biped walk behavior using genetic algorithms. In Cabestany, J., Sandoval, F., Prieto, A., Corchado, J., eds.: Bio-Inspired Systems: Computational and Ambient Intelligence. Volume 5517 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2009) 805–812
5. Shafii, N., Reis, L., Lau, N.: Biped walking using coronal and sagittal movements based on truncated fourier series. In Ruiz-del Solar, J., Chown, E., Plöger, P., eds.: RoboCup 2010: Robot Soccer World Cup XIV. Volume 6556 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2011) 324–335