11º WORKSHOP

ROBOTS SOCIALES

RoboCity2030

Universidad Carlos III de Madrid

UNED 40

Universidad de Alcalá

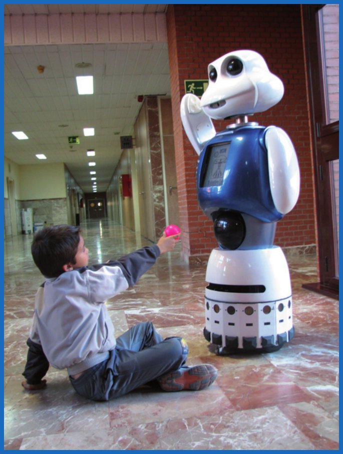
POLITÉCNICA

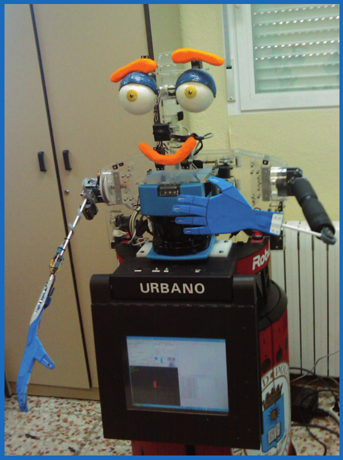
CSIC
Consejo Superior de Investigaciones Científicas

Universidad Rey Juan Carlos

La Suma de Todos
Consejería de Educación, Juventud y Deporte
Comunidad de Madrid
www.madrid.org

Unión Europea
Fondos Estructurales
Invertimos en su futuro





Editores:
Javier Fdez. de Gorostiza
Eduardo Silles
Sonia Mata
Miguel Ángel Salichs
Carlos Balaguer

Universidad Carlos III de Madrid

Marzo 2013

Edita: Universidad Carlos III de Madrid
Imprime:
Depósito Legal: M-7239-2013
ISBN: 978-84-695-7212-2

CAPÍTULO 5

Programming a Humanoid Social Robot Using the JdeRobot Framework

B. MENÉNDEZ[1], J. M. CAÑAS[2], E. PERDICES[3], J. VEGA[4],
R. SALAMANQUÉS[5], F.RIVAS[6], F.MARTÍN[7]

Robotics Group, Universidad Rey Juan Carlos.
[1]b.menendez.moreno@gmail.com, [2]jmplaza@gsyc.es,
[3]eperdices@gsyc.es, [4]julio.vega@urjc.es, [5]rubensalamanques@gmail.com,
[6]franciscomiguel.rivas@urjc.es,[7]fmartin@gsyc.es

Social robots awake great research interest as there are many potential applications in this field where robots can be beneficial. Humanoid robots are good platforms for social applications as their appearance facilitates their acceptance and natural interaction with humans. The Aldebaran's Nao is a cheap and robust humanoid platform which was chosen by the RoboCup Standard Platform League and has spread over many research centers in last years. We have developed a RoboTherapy software application for this humanoid, which has already been utilized in the therapy of real dementia patients. This paper shows three different frameworks that we have used to program it: the manufacturer's middleware, NaoQi, and two frameworks developed at our group: BICA and JdeRobot. In particular, we describe in detail the last one including several tools specifically developed to simplify the programming of this social robot (like its support in Gazebo) and also including some of our research in enabling technologies like walking gaits, visual localization and visual memory, that can be used in new humanoid applications.

1 Introduction

One field of growing interest in robotics is humanoids. Prototypes such as the Honda Asimo or the Fujitsu HOAP-3 are the basis for many research efforts, some of them designed to replicate human intelligence and manoeuvrability. Their appearance, being similar to people, facilitates their acceptance and natural interaction with humans as a personal assistant in the field of service robotics. As a representative sample, the functionality achieved in the Asimo humanoid has progressed significantly in recent years, allowing it to run, climb stairs, push carts and serve drinks.

QRIO humanoid robot was created by Sony but was never sold to the general public. A few years later came the Nao robot, replacing the quadruped AIBO from Sony in the RoboCup Standard League. Both robots, QRIO and Nao, have a height of about 60 centimeters, being a bit lighter Nao than QRIO (4.3 kg versus 7.3 kg, respectively). One new example of humanoid is Roboy[3], which will be presented in early March-2013. It has been developed by the Artificial Lab of the University of Zurich and its purposes will be helping with housework, job security, cleaning or even robotherapy in hospitals. Its creators also plan to use it as a service robot to help elderly or disabled people, thanks to its powerful autonomous behavior set.

One increasing application field of robotics is medicine. Beyond DaVinci robot used in surgery, there are also examples of robots used with elderly and in therapy for dementia patients. For instance, Paro is one therapeutic robot that has the form of a harp steal, intended to have a calming effect on and elicit emotional responses in patients of hospitals and nursing homes. This harp steal robot can be used with disabled or autism patients; it succeeded in reducing the stress of both patients and their caregivers, stimulating the interaction of the people.

Helping the humans suffering some type of dementia is a growing target market for robotics. Estimates point that by 2016 there will be 26.6 million people worldwide with Alzheimer′s disease, and this figure will be three times bigger by 2050 when Alzheimer's will affect 1 in 85 people of the total world population. In addition, 40% of them will be in an advanced state of disease, requiring a level of care that involves high consumption of resources (A. Tapus, 2009). Neurodegenerative dementia is a disease that progressively deteriorates brain functionality. One of its most common symptoms is memory loss. In addition, patients usually lose the ability to solve problems or control their emotions and present changes in personality and normal behavior. Current therapy aims to practice and stimulate the cognitive abilities of the patients to slow down the advance of the desease.

When interacting with people, robots should include strong social capabilities. Maggie robot (González, 2008), from Universidad Carlos III, is another example of social robot. It is able to interact with humans through its speech recognition, being also able to speak. It also offers facial recog-

[3] http://www.roboy.org

nition and can even feel when is touched. All these characteristics make this robot a very good candidate to work in robotherapy. Maggie can help visually impaired people reading books, navigate accompanying a person or act as playmate.

Regardless the final application, robot's hardware is composed of sensors, actuators and one or more computers. Their intelligence, behavior and capabilities mainly lie on its software (Brugali, 2007) on what they have been programmed to do. Some years ago the robot applications were typically developed using the manufacturer drivers to the sensors and actuators, usually over a specific operating system. In the last years general operating systems have been incorporated to the robots and several programming frameworks have been created to make simpler the development of robot applications (José M. Cañas, 2007).

Player/Stage, ERSP from Evolution Robotics, OROCOS, ORCA [(A. Makarenko, 2006), (A. Brooks, 2007)], ARIA from ActivMedia Robotics, Microsoft Robotics Studio (J. Jackson, 2007) and ROS[4] (Robot Operating System (M. Quigley, 2009)) from WillowGarage are some succesful frameworks. Maybe ROS has become a de facto standard. Some of them were created in private companies and others in research centers. Most of them are open-source. These platforms tipically provide (1) a Hardware Abstraction Layer for accessing to robot sensors and actuators, (2) a particular software architecture for the applications and (3) tools, libraries and software modules with common functionalities for developers. They favour code reuse speeding up the development time for new applications and reducing programming errors.

One common tool for robotics engineers are simulators. They allow testing the software on simulated environments and debug it before probing it on the real robots. Some of them only support 2D worlds; others fully support 3D worlds, complex sensors like cameras, laser, etc. and different robot geometries and platforms. Webots, Stage (Brian P. Gerkey, 2003), Gazebo[5] (Howard, 2004), V-REP, Morse (G. Echeverria, 2011) and USARsim (S. Carpin, 2007) are some successful simulator examples.

In this paper we present the Nao humanoid as a social robot, describe three frameworks that we have used to develop applications for it, including their relevant tools and some lessons learnt. All of them are component oriented frameworks which allow the robot programming in C++. First, NaoQi is the manufacturer's environment. Second, BICA platform has been developed at Universidad Rey Juan Carlos (URJC) and it has been used to develop the Nao RoboTherapy application, which has been used

[4] http://www.ros.org
[5] http://gazebosim.org

with real dementia patients in collaboration with medical experts at the Neurological Desease Researh Center (CIEN, Centro Investigación de Enfermedades Neurológicas). And third, we will make emphasis on the JdeRobot framework. It was developed at URJC for programming of several sensor and robot platforms. Its support for the Nao robot has been developed, several tools and components with enabling functionalities have been created and can be reused in new Nao applications.

The remainder of this paper is organized as follows. The second section exposes how to program the Nao humanoid using the manufacturer's platform. The third section presents RoboTherapy application and the BICA framework. The fourth section describes in detail the JdeRobot framework and its tools to develop applications for the Nao robot. Finally some conclusions are summarized.

2 Nao robot and NaoQi

The Nao robot is an autonomous, programmable and medium size robot, developed by the French company Aldebaran Robotics, based in Paris. On 2008 Nao replaced the robot dog Aibo from Sony as the official platform for the RoboCup Standard League. Since then the Nao platform has been continuouly improved, both in hardware and software, in part due to the feedback provided by its intensive use at RoboCup. Today there are two models of Nao, one designed for use in the RoboCup and a second type, which includes gripping capabilities in both hands, for universities and academic purposes.

Its main features are:

- 58 cm tall.
- 4.3 kg weight.
- Autonomy of 45 minutes (15 minutes walking).
- Degrees of freedom: 21 to 25.
- CPU: x86 AMD Geode at 500 MHz.
- Ethernet, WiFi.
- Inertial sensor.
- 4 ultrasonic sensors.
- 4 microphones, 2 Hi-Fi, 2 CMOS cameras.
- Voice synthesizer.

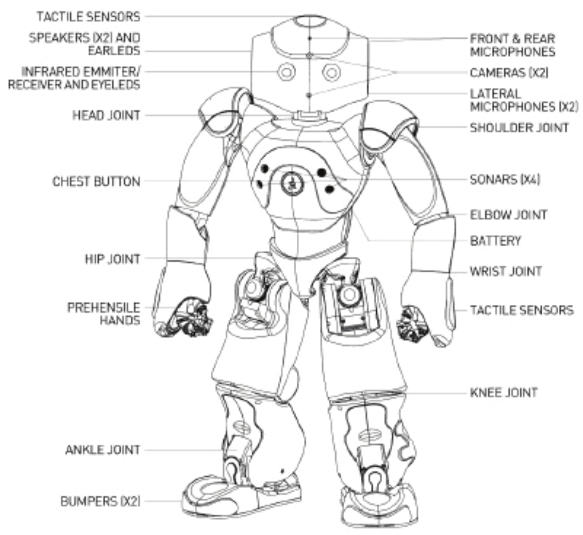


Figure 1: Nao hardware

One way to program the Nao robot is with the manufacturer's platform, named NaoQi. The Nao robot runs on the Linux platform and the NaoQi software development kit provides the ability to program it in C, C++, Ruby and Urbi. It is available for both Windows and Linux. This development kit is compatible with the Webots simulator from Cyberbotics.

This SDK is based on a client-server architecture where NaoQi itself acts as a server. NaoQi bases its programming architecture in separate modules called Brokers, which are executable programs connected via IP address and port. All these new Brokers are connected to a main Broker called MainBroker, as shown in Figure 2. This architecture allows to execute some code on the robot and some on remote machines.

Some of the modules used for programming social robots offered by NaoQi are:

- ALCamera: this module is responsible for communication with both cameras of the robot. Without access to the two cameras simultaneously is also responsible for managing the switch between cameras, and all the camera settings.
- ALMotion: this is the module responsible of robot's locomotion. It is responsible for the control of Nao movements, providing simple functions for controlling the actuators in space, the handling of center of mass and high-level motion such as: 10 cm

walks straight. This module offers different possibilities for the following features:

  - Solving the kinematic model of the robot.
  - Controlling the robot joint space.
  - Control the orientation of the torso.
  - Control the center of mass.
  - Create and control patterns.
  - Controlling parameters of the physical components of the robot, such as stiffness of the joints.

  This module gives access to gaits offered by the manufacturer, which are basically four: Straight line walking, Circular walking, Sideways walking and Turn on himself. In all cases, the walking speed is not set following a temporal-space parameters, but in the time it takes for the robot to take a step.

- ALMemory: event-based memory, used primarily to read sensor values. This implementation is very useful because it is capable of generating events or notifications whenever a stored value is modified.
- ALTextToSpeech: voice synthesizer module.
- ALSonar: module that provide access to the ultrasonic sensor.

The NaoQi SDK includes some relevant tools like Coreographe, which allows the definition of new movements, decomposing the robot motion in sequence of several joint movements.

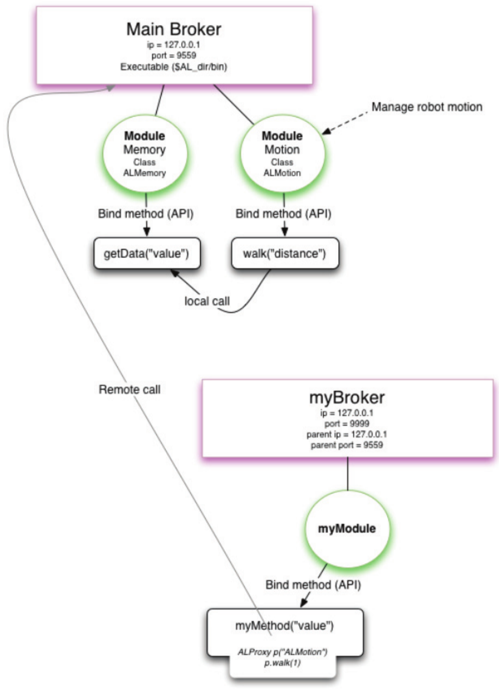


Figure 2: NaoQi software architecture using Brokers

3 BICA framework and the RoboTherapy application

The Robotics Group of URJC created a programming framework to develop autonomous applications for the Nao robot. It is named Behavior-based Iterative Component Architecture (BICA) (F. Martín, 2010), and has been used in research for several years around the RoboCup scenario, in teaching robotic courses.

BICA has been also used as the software base for the RoboTherapy application (F. Martín, 2013). In this application, developed in collaboration with medical experts at CIEN, the Nao robot has been used with real dementia patients in their therapy sessions. The main purpose of this pilot study is to test the use of the humanoid as a cognitive stimulation tool. The robot behaviors in therapy sessions are described mostly as a sequence of basic movements, music or text playing and light turning on-off operations. A file format syntax has been created to store these behavior descriptions (session scripts). Some specific components inside BICA have been developed, like one that runs session scripts or another that provides access to robot lights from the application software. In addition, some specific tools have also been created: a session script generator that allows easy and visual "programming" of robot behavior in therapy sessions, and the session monitor tool that helps the human therapist to control the session progress. They are all described in this section.

3.1 BICA Architecture

The software of our humanoid robot is organized as behavior-based architecture. It is implemented in component-oriented software architecture programmed in C++ language. Components are independent computation units which periodically execute control iterations at a pre-configured frequency. Every component offers an explicit interface to modulate its execution and to retrieve the results of its computations.

Some disadvantages of NaoQi are that module communication using SOAP was too slow for our requirements, and we did not find an efficient way to program the iterative nature of our Nao applications. BICA was created to overcome these limitations.

Behaviors in BICA are defined by the activation of perception components and actuation components. *Actuation components* take movement decisions, send commands to the robot motors, or locomotion system, or activate other actuation components. They run iteratively to periodically update their outputs. *Perception components* take data from the robot sensors or other perception components and extract information. They basically provide information to the actuation components. The output of a perception component is refreshed periodically and can be read from many other components in the system.

Beyond being a framework to integrate perceptive and actuation capabilities for autonomous behaviors, the BICA architecture also includes components that provide access to the basic sensors and actuators of the robot, a Hardware Abstraction Layer (HAL) for robot applications (`Body`, `Head`, `Music` and `Led` components). BICA is built on top of NaoQi and offers this HAL as a set of object method invocations.

In addition, BICA offers some tools for creating and debugging software components. For instance, the JManager tool allows to manage from an external computer the set of active or inactive components onboard the robot, and to monitor them in real time displaying their results. The Vicode tool allows the graphical definition of automata for robot behavior, and it automatically generates the corresponding BICA component in C++. More details of the architecture and its tools can be found at (F. Martín, 2010).

3.2 RoboTherapy application

A high level language has been developed to describe the robot behaviors in the RoboTherapy application. They can be stored in text files following a given syntax and read from them - they are called session scripts. The language includes three *basic instructions*: move, music and light. Two or three basic operations of different type can be grouped together, in *group instructions*, to be executed simultaneously. The robot behavior is a sequence of *basic instructions* and/or *group instructions*. In the script some synchronization points can be included to wait for the termination of all the basic instructions inside a group. In addition, the wait instruction causes the robot to stop execution until the human therapist provides the continue order, striking one button on the robot body or by using any monitoring tool. This allows the human therapist to control the session progress.

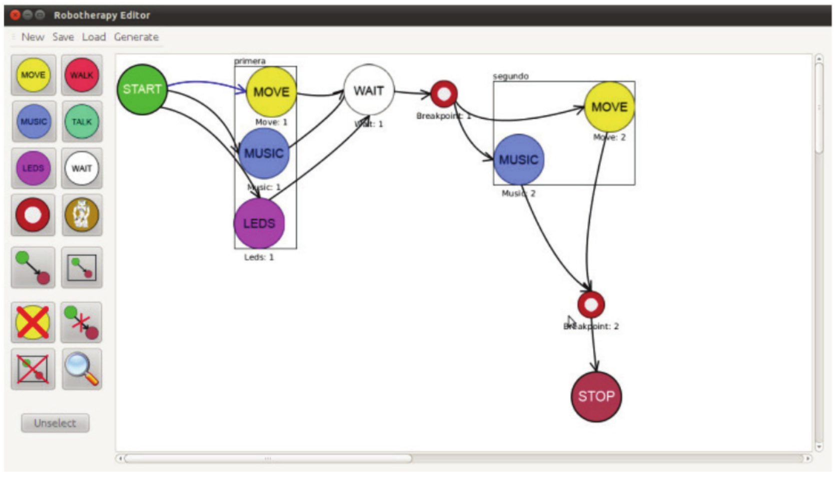


Figure 3: Session script generator

The scripts are generated and stored in text files. Their contents are designed by medical doctors and health assistants, attending to the desired stimulation in the dementia's patients. At the beginning they were created by directly editing text files, however more recently we have developed a graphical tool, the session script generator (Figure 3), that allows a fast and visual creation of these scripts.

One specific component has been developed inside BICA for the RoboTherapy application, Movie component. It accepts session scripts as input and runs the corresponding orders to robot motors and actuators, at the proper timing, unfolding the specified robot behavior. It uses several HAL components available in BICA, like the `Body`, `LED`, `Music` and `Head` components.

The therapist needs a way to communicate with the robot, for instance, to start a RoboTherapy session, to stop its execution while the patients answer one of the robot questions, to repeat any script step, among other tasks. The basic interface with the real robot was the set of buttons on the humanoid's feet and chest. At the beginning these buttons were used, but we developed two session monitor applications to allow an easier way to control the robot.

The first session monitor is an application running on a regular computer. It offers a GUI with sliders, selectors, visual buttons, etc. This allows the teleoperation of the robot body and head, so that the robot can approach the patients at the beginning of the sessions, for instance. It can be operated from an external computer or used in conjunction with a Wiimote. This Wiimote device is more convenient than the regular screen, keyboard and mouse configuration. In this case the session monitor reads the therapist's orders from the Wiimote buttons and accelerometers using Bluetooth.

In order to improve the tool usability, a second session monitor has been created. It runs on mobile devices like Android tablets or smartphones. With it the human therapist has full control of the progress of the therapeutic session without requiring any neither extra computer nor Wiimote, just the robot and the tablet or smartphone.

4 The JdeRobot tools for Nao programming

The third way to program the Nao robot that we have used is JdeRobot[6]. It is an open source software framework for robotic, computer vision and home automation applications created by Robotic Group of URJC. Robotic applications inside JdeRobot are a set of components that run simultaneously in parallel as individual processes. They perform simple specific tasks and interact with each other. The concurrent execution of multiple components results in a behavior. JdeRobot uses ICE as communications middleware between these components, which can be written in different programming languages (C++, Java, Python...) and run on distributed machines.

[6] http://jderobot.org

In this section we are going to present some specific components and tools developed for the Nao humanoid: `NaoServer` and the Nao support in Gazebo, an evolutive algorithm that learns motion gaits for the humanoid, the `Recorder` and `Replayer` tools to store and replay sensor data, the `VisualHFSM` tool to create Nao behaviors as hierarchical finite state machines, a component to teleoperate the humanoid from a smartphone, the `VisualMemory` component that builds a persistent local map from the images of the Nao camera and the `VisualLocalization` component that estimates the humanoid position inside a map using the images from its camera. The two last components include enabling technology that can be used in new Nao applications.

4.1 Real robot driver NaoServer

`NaoServer` is the JdeRobot component that provides access to sensors and actuators from other application components. It may serve several connections at the same time. It periodically makes calls to NaoQi API to query for sensor data and to send orders to the Nao actuators, and it offers a set of ICE explicit interfaces to receive sensor queries or motion commands from other JdeRobot components. It provides the following interfaces:

- Camera: for camera descriptions and the possibility of start and stop the streaming.
- Motors: it provides the motion interface. Instead of using an interface for each humanoid joint an abstract motion interface has been preferred, with three attributes: *v* for translation speed of the robot, *w* for rotation speed of the robot and *l* for side speed. This abstract interface has been matched to the corresponding walking gait offered by NaoQi.
- Pose3DEncoders: it provides position data of the head, its pan and tilt.
- Pose3DMotors: it provides motion orders to the head.

Exactly these interfaces have been also used with different robot platforms like the Pioneer from ActivMedia. This way some tools like the image monitorization tool (`CameraView` component) can be used with both robots without any change in code. The images come from the same interface, but provided by different servers.

4.2 Gazebo support for Nao

Gazebo is an open source 3D simulator that provides an environment to test and develop multi-robot systems quickly and even cameras that simulates realistically. It was born in the project Player/Stage (B. Gerkey, 2007) and WillowGarage has recently centralized its development as an auxiliary tool for ROS, becoming as an independent project. It integrates ODE physics engine, OpenGL rendering and support code for sensor simulation and actuator control.

The reference simulator for Nao with NaoQi and BICA frameworks is Webots from Cyberbotics. We wanted to avoid this dependence on proprietary software, so we developed a Gazebo plugin for the Nao robot. It includes the humanoid mechanical parts, with their weights and sizes, the joints and one camera sensor on the Nao head. It also includes the corresponding skins matching the real appearance of the robot. Figure 4 shows two instances of a Nao simulated in Gazebo with and without its skin.

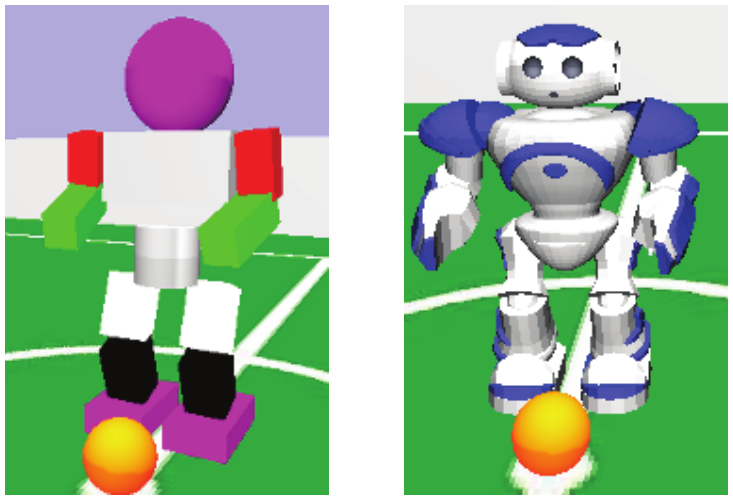


Figure 4: Nao robot in Gazebo (a) without skins and (b) with skins

In addition, the `GazeboServer` component has been extended to provide ICE interfaces to the simulated Nao. Initially `GazeboServer` was created for the support of simulated Pioneer robot, but it has been improved and now includes the same interfaces offered by `NaoServer`. The underlying code under the abstract motion ICE interface, `motors`, now moves the simulated joints. This way, the humanoid application can be tested seemlessly both on real robot (using `NaoServer`) and on the simulated Nao in Gazebo (using `GazeboServer`).

4.3 Walking gaits for Nao

Despite major advances in humanoid robots, locomotion of these is still an open problem, while still far from the flexibility, robustness and plasticity of natural movements of people. Generating ways of walking in humanoid robots is part of a more general problem: the coordination of N articulations. We currently use the locomotion subsystem provided by NaoQi, but we have also created an algorithm that explores the space of possible walking gaits for the humanoid and finds good ones to be used as locomotion subsystem in Nao applications.

The most widely used technique for humanoid locomotion has been for many years the ZMP (Zero Moment Point), which calculates the trajectory of the center of mass for the walk is stable. Complex robots as ASIMO or HRP use this algorithm. However, to operate properly it is necessary to perform a very accurate modeling of both the robot and actuators. Therefore, more and more authors are using bio-inspired models humanoids. We used the central pattern generators (CPGs) idea (Ijspeert, 1998), that simplifies the walking as a set synchronized waves (patterns) along the different joints of the robot arms and legs. This way each walking gait is represented by a set of parameters like angular offsets between the waves and relative amplitudes.

Our algorithm chooses a set of patterns and searches in the parameter space for the best walking gait. For evaluation of tentative walks we use a simulator where we let the model run for some time and numerically observe properties as stability, speed and linearity of generated movement.

All these factors are combined in a single health function that indicates the final quality of the walk. This health function is used both in systematic search o in an evolutionary algorithm along the parameter space.

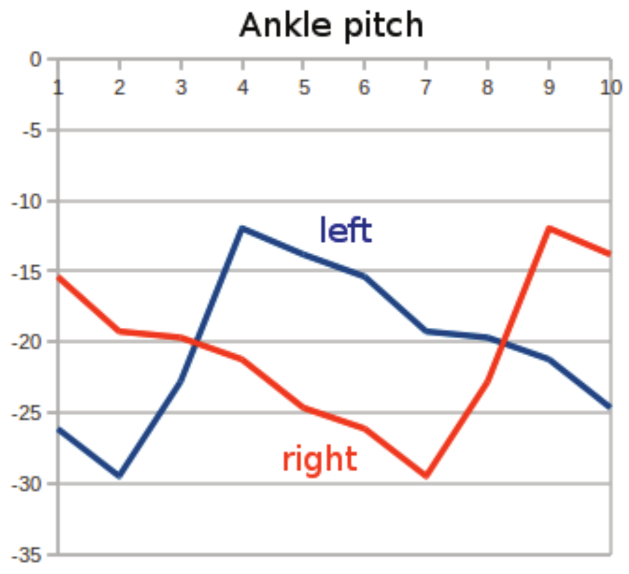


Figure 5: Ankle’s pitch of the Nao robot

The final result is a parameter set that describes a gait, with the patterns (sequence of joint positions) for the hips, knees and ankles of the Nao robot. This gait is stable and provides an advance speed close to the gait deployed by NaoQi locomotion subsystem. Figure 5 shows the angles of the robot's left and right ankles over time. As expected they are in antiphase. More details can be found at (F. Rivas, 2011). We are working more fine grained search to find faster gaits.

4.4 Recorder/Replayer

Recorder and Replayer are two JdeRobot tools. `Recorder` component is able to collect information provided by a robot or sensor server, like `kinectServer` or `GazeboServer`, and store it in a log file in the hard drive. `Replayer` component is able to reproduce recorded information saved in the log file.

Theese tools allow the offline testing of perceptive components which can be fed with exactly the same sensor data as input, because they have been stored in the log file. It is possible to configure recording speed and which ICE interfaces are selected for the log. For instance, many ICE interfaces are supported in `Recorder` and `Replayer`, like Camera, Laser, Encoders, Pose3DEncoders and cloud points from Kinect type devices. The replaying speed and which stored ICE interfaces should be provided off-line can also be configured.

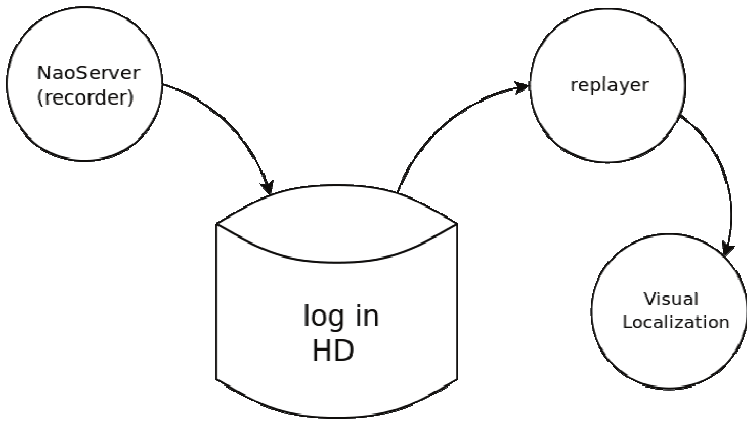


Figure 6: Recorder and replayer tools

The already mentioned `NaoServer` component can be configured to act as a recorder for Nao sensor data, as shown in Figure6. This capability has been used to test the `VisualLocalization` component, feeding it with off-line sensor and image data.

4.5 Generating Nao behaviors with VisualHFSM

We have created the `VisualHFSM` tool inside JdeRobot for the programming of robot behaviors using finite state machines with hierarchy (HFSM - Hierarchical Finite State Machine). It represents the robot's behavior graphically on a canvas composed of states and transitions. The source code to be executed at each state or when checking each transition can be also introduced. This tool decreases the development time of new applications, putting the developer into a more abstract visual language. It also increases the quality of these applications, automatically generating most of their code. The tool allows the engineer to focus on specific parts of his application, automatically generating the rest, getting a code more robust, less prone to failure.

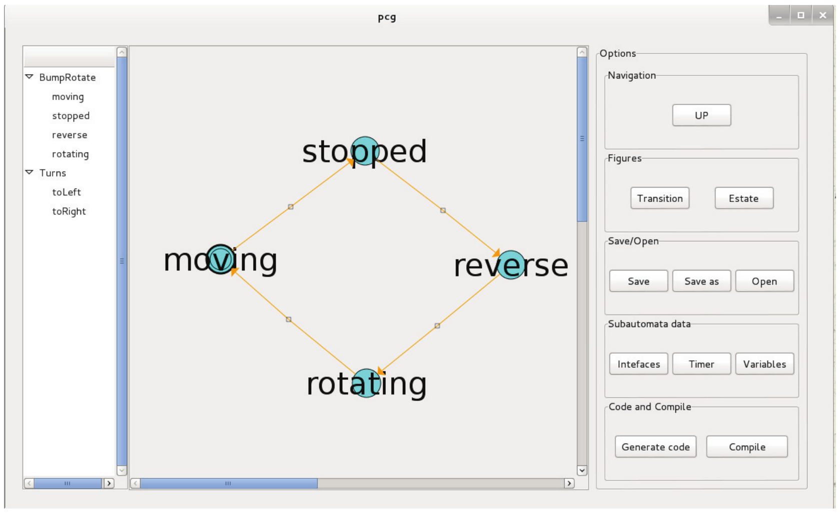


Figure 7: Graphical editor of the VisualHFSM tool

`VisualHFSM` is divided into two parts: the graphical editor and the automatic code generator. The graphical editor allows the user to visually represent, edit and add the states and transitions in a clear and simple way (Figure7). The GUI is divided into three parts: the tree view, the canvas and action buttons. The tree view is the area where you can see the hierarchical tree of the generated automata, the canvas is the area where it is drawn with circles for states and arcs for transitions, and the different action buttons allow editing and programming. The graphical editor saves in an XML file all the features of the developed component: the structure of the automata, the characteristics of each node and transition, etc.

The automatic code generator takes that XML file and generates the source code in C++ of a single JdeRobot component that implements the designed FSM (Figure 8). It uses a component template which has two parts: one thread for control and another one for graphics. The control thread iteration contains the entire code for the behavior of the robot. In every iteration of the control thread it checks in which state it is and executes the code. In each state it does the following steps: run perception code for action, run the code for concrete action, execute perception code for the transition, check the condition of the transitions and change to the corresponding state. The iteration of the graphic thread is used to display sensor data or internal structures at runtime in the component's GUI.

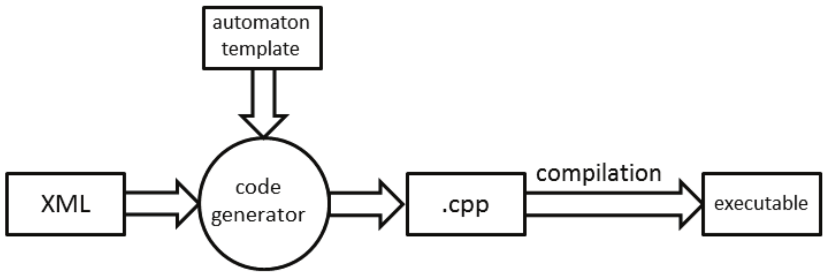
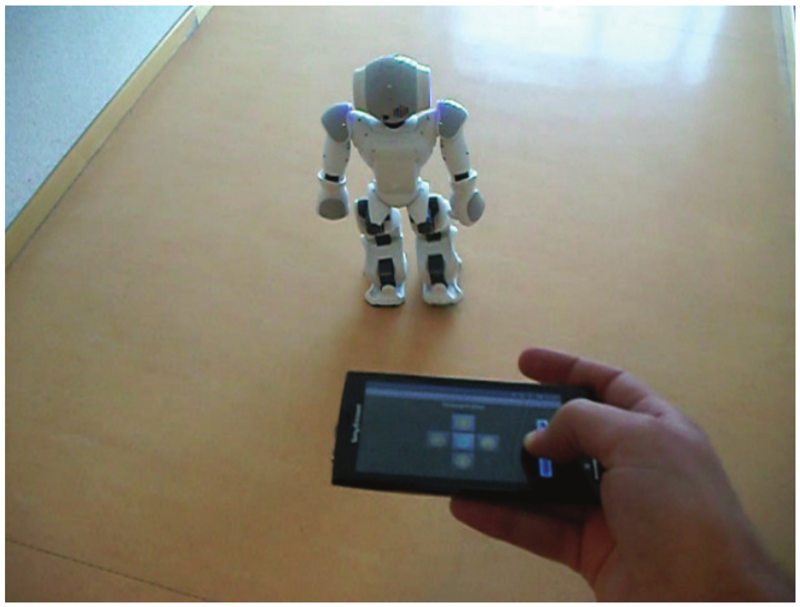


Figure 8: Automatic code generator of VisualHFSM tool

4.6 Mobile Teleoperator

`MobileTeleoperator` is a JdeRobot component for Android smartphones to teleoperate either a Pioneer robot through the `PlayerServer` or a Nao robot through the `NaoServer`. In Figure9 we can see `NaoServer` running on the Nao robot and providing its ICE interfaces and the `MobileTeleoperator` component interacting with them and with the human user.



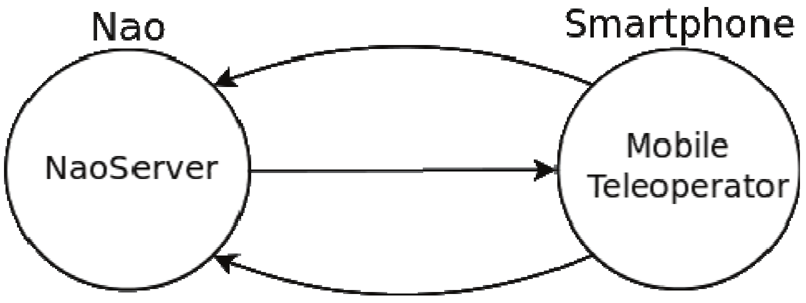


Figure 9: MobileTeleoperator controlling the Nao's movement

4.7 Visual memory

Cameras are one of the most relevant sensors in autonomous robots. However, two of their challenges are to extract useful information from captured images, and to manage the small field of view of regular cameras. The Nao humanoid has a pair of cameras (in a non-stereo setup) on its head as, which can be oriented at will, as its main sensors to perceive its surroundings.

We have developed one JdeRobot component, named `VisualMemory`, which receives data from robot cameras and encoders, and extracts a description of the objects around the humanoid even beyond the current field of view (Julio Vega, 2012). This information is provided to other actuation components like the navigation algorithm or other control units.

This component builds a local visual memory of objects in the robot's surroundings. The memory is built analyzing each camera image looking for relevant objects (like segments, parallelograms, etc.) and updating the object features already stored in the memory, like their 3D position. The memory is dynamic and is continuously coupled with camera images. The new frames confirm or correct the object features stored in memory, like their 3D relative position to the robot, length, etc. New objects are introduced in memory when they appear in images and do not match any known object. This component is also active, as it moves the humanoid head in order to cover several areas of the robot surroundings. It includes gaze control.

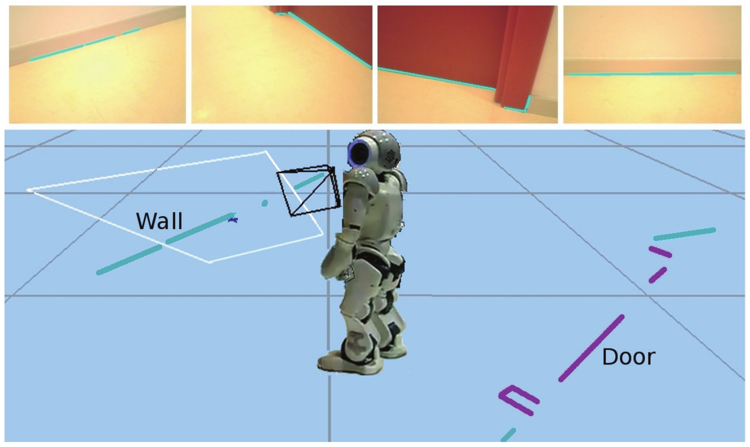


Figure 60: Nao visual memory from four camera images

Figure 60 shows a real experiment showing how Nao robot is viewing its world. In that figure we can see how the Nao robot is viewing some objects remembering how its world is, it is, the door and walls it saw before.

4.8 Visual autolocalization

Many Nao navigation applications need to know where the robot is inside a map. We have developed a vision-based localization component in JdeRobot to provide this enabling functionality. This component, named `VisualLocalization`, uses a population of particles and an evolutionary algorithm to manage them and estimate the current robot position.

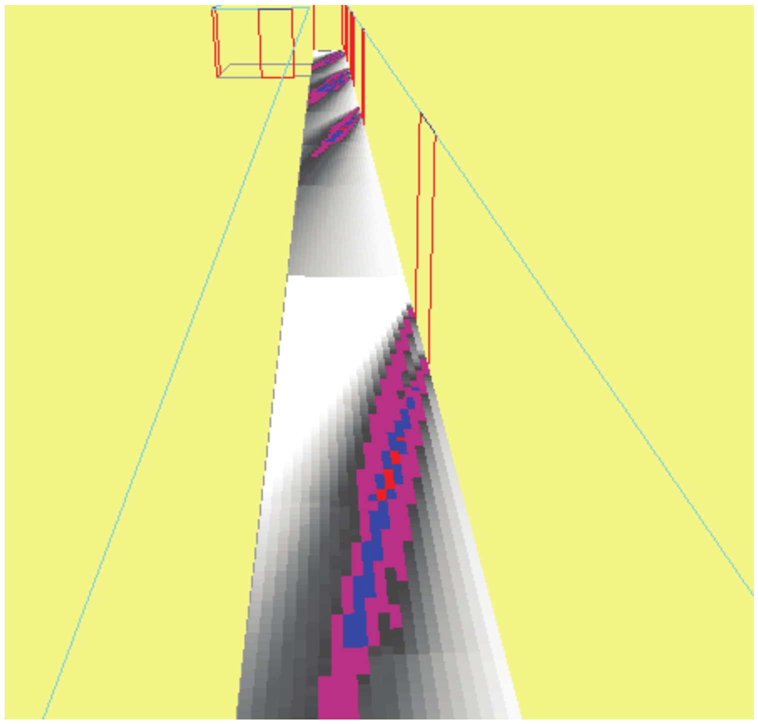


Figure 71: The likelihood of different positions of a corridor when the Nao perceives a door on the left side of its camera

Each particle or individual is a candidate solution. The health of each particle is computed comparing edge points in the current image with those in the predicted image from that location. The whole population of individuals evolves over time using genetic operators, such as mutation or crossover to generate new particles and elitism to keep alive those with highest health. More details can be found in (Julio Vega, Robot Evolutionary Localization Based on Attentive Visual Short-Term Memory, 2013). Figure 71 shows the health of different locations when a door is observed in the left side of the image of from the humanoid camera. Those locations close to any door have high likelihood using our health function.

5 Conclusions

In this paper we have presented three different frameworks to program a social robot, the Nao humanoid, and introduced some of their tools. First, we talked about the manufacturer's software, NaoQi, as the initial platform we used to program these robots. Second, we talked about BICA, a component-oriented software developed by the Robotics Group at URJC. BICA has been used to develop the social application RoboTherapy which has been used with real dementia patients.

Third, we discussed how to program the humanoid robot with the open source JdeRobot framework, also developed at the URJC. JdeRobot offers a hardware abstraction layer for the Nao robot: the `NaoDriver` component provides standard ICE interfaces that allow the applications (both onboard or at an external computer) receive sensor data like images, head encoders, etc. and send motion commands to the humanoid body or the Nao head. In addition, the support for the Nao in the 3D Gazebo simulator has been developed and the `GazeboServer` component provides those same ICE interfaces for the applications to use the sensors and motors of the simulated Nao. Some tools in this framework have been also described, like the `MobileTeleoperator` to move the humanoid from a smartphone and the `VisualHFSM` to visually program the robot behavior using hierarchical automata, efficiently and quickly. Finally, two components with enabling tecnologies have been presented: `VisualLocalization` provides the robot localization in a known environment and `VisualMemory` provides a description of the objects in the robot surroundings. They can be used in new applications of this humanoid social robot.

We are working on changing the BICA framework and organizing it as different ROS nodes. In addition, we are improving the RoboTherapy social application with new contents, doing code refactoring in order to download the therapies from a web server and allowing it to interact with other therapy tools like tablets. Regarding Jderobot we are updating the Nao support in Gazebo to the latest simulator release, preparing new social applications with higher human-robot autonomous interaction using vision and programming new humanoid behaviors useful when the robot moves at real people homes, like people following and autonomous navigation.

Acknowledgements

This work was supported by the project S2009/DPI-1559, RoboCity2030-II, from the Comunidad de Madrid and by the project 10/02567 from the Spanish Ministry of Science and Innovation.

References

A. Tapus, C. Tapus and M. J. Mataric. “The use of socially assistive robots in the design of intelligent cognitive therapies for people with dementia”. In Rehabilitation Robotics, 2009. IEEE International Conference on ICORR 2009, pp. 924-929, 2009.

A. Brooks, T. Kaupp, A. Makarenko, S. Williams and A. Orebäck. “Orca: a component model and repository”. In D. Brugali, Software engineering for experimental robotics, pp. 231-251, 2007.

A. Makarenko, A. Brooks and T. Kaupp. “International Conference on Intelligent Robots and Systems (IROS)”. In Orca: Components for robotics, pp. 163-168, 2006.

B. Gerkey and R. Vaughan. 2007. “Reusable robot software and the player/stage project”. Software Engineering for Experimental Robotics, pp. 267-289.

Brian P. Gerkey, Richard T. Vaughan and A. Howard. “The Player/Stage project: tools for multi-robot and distributed sensor systems”. In Proceedings of the 11th International Conference on Advanced Robotics, Coimbra, Portugal, pp. 317-323, 2003.

Brugali, D. 2007. Software Engineering for Experimental Robotics (Vol. 30). Springer.

F. Martín, C. Agüero, José M. Cañas and E. Perdices. “Humanoid soccer player design”. In V. Papic, pp. 67-100, 2010.

F. Martín, C. Agüero, J. M. Cañas, M. Valenti and P. Martínez. 2013. “Robotherapy with Dementia patients”. Int. J. of Advanced Robotic Systems. 1268-1299.

F. Rivas, José. M. Cañas and J. González. "Aprendizaje automático de modos de caminar para un robot humanoide". In Proceedings of Robot2011 III Workshop de Robótica: Robótica experimental. Sevilla, Spain, pp. 120-127, 2011.

G. Echeverria, N. Lassabe, A. Degroote and S. Lemaignan. "Modular OpenRobots Simulation Engine: MORSE". In Proceedings of the IEEE ICRA. 2011.

González, A. C. 2008. "Desde la teleoperación al control por tacto del robot Maggie". Leganés, Universidad Carlos III de Madrid.

Ijspeert, A. 1998. Design of artificial neural oscillatory circuits for the control of lamprey-and salamander-like locomotion using evolutionary algorithms. PhD Thesis, Department of Artificial Intelligence, University of Edinburgh.

J. Jackson. 2007. "Microsoft Robotics Studio: a technical introduction". IEEE Robotics & Automation Magazine, pp. 82-87.

José M. Cañas, V. Matellán, B. MacDonalds and G. Biggs. "Programming commercial robots". In Software Engineering for Experimental Robotics, Springer-Verlag, pp. 125-132, 2007.

J. Vega, E. Perdices and José M. Cañas. 2012. "Robotic Vision: Technologies for Machine Learning and Vision Applications". In Attentive visual memory for robot localization, IGI Global, pp. 406-436.
J. Vega, E. Perdices and José M. Cañas. "Robot Evolutionary Localization Based on Attentive Visual Short-Term Memory". Sensors. 2013.

M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler and A. Y. Ng. "ROS: an open-source Robot Operating System". In ICRA Workshop on Open Source Software. 2009.

N. Koening and A. Howard. "Design and Use paradigms for Gazebo, an open-source multi-robot simulator". In Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems. Sendai, Japan. 2004.

S. Carpin, M. Lewis, J. Wang, S. Balarkirsky and C. Scrapper. "USARSim:a robot simulator for research and education". In Proceedings of the IEEE 2007 International Conference on Robotics and Automation, pp. 1400-1405, 2007.