Auton Robot (2008) 25: 383–404
DOI 10.1007/s10514-008-9102-y

A multimode navigation system for an assistive robotics project

**Andrea Cherubini · Giuseppe Oriolo ·
Francesco Macrí · Fabio Aloise · Febo Cincotti ·
Donatella Mattia**

Received: 23 October 2007 / Accepted: 22 July 2008 / Published online: 21 August 2008

Abstract Assistive technology is an emerging area, where robotic devices can help individuals with motor disabilities to achieve independence in daily activities. This paper deals with a system that provides remote control of Sony AIBO, a commercial mobile robot, within the assistive project ASPICE. The robot can be controlled by various input devices, including a Brain-Computer Interface. AIBO has been chosen for its friendly-looking aspect, in order to ease interaction with the patients. The development of the project is described by focusing on the design of the robot navigation system. Single step, semi-autonomous and autonomous navigation modes have been realized to provide different levels of control. Automatic collision avoidance is integrated in all cases. Other features of the system, such as the video feedback from the robotic platform to the user, and the use of AIBO as communication aid, are briefly described. The performance of the navigation system is shown by simulations as well as experiments. The system has been clinically validated, in order to obtain a definitive assessment through patient feedback.

A. Cherubini (✉) · G. Oriolo · F. Macrí
Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Via Ariosto 25, 00185 Rome, Italy
e-mail: cherubini@dis.uniroma1.it

G. Oriolo
e-mail: oriolo@dis.uniroma1.it

F. Aloise · F. Cincotti · D. Mattia
Laboratorio di Imaging Neuroelettrico e Brain Computer Interface, Fondazione Santa Lucia, IRCCS, Via Ardeatina 306, 00179 Rome, Italy

F. Aloise
e-mail: f.aloise@hsantalucia.it

F. Cincotti
e-mail: f.cincotti@hsantalucia.it

D. Mattia
e-mail: d.mattia@hsantalucia.it

Keywords Assistive robotics · Vision-based navigation · Brain-computer interfaces · Legged robots

1 Introduction

The development of robots for socially assistive applications for elderly or disabled persons is a growing and increasingly popular research area. Feil-Seifer and Matarić (2005) define *Socially assistive robotic systems* (SARS) as the intersection of *Assistive robotic systems* and *Socially interactive robotic systems*.

Assistive robotic systems give aid or support to a human user. These systems can be assessed by considering their effects on the quality of the user life. Projects undertaken in this field, range from navigation aids for the visually impaired (Kulyukin et al. 2004), to robots for assisting motor disabled individuals (Caselli et al. 2003). These systems should assist the patient in everyday tasks, e.g., managing home appliances, carrying objects, or monitoring the environment (Harmo et al. 2005). An important objective in this field is to design versatile systems, which adapt to the user level of disability by offering various human-robot interfaces and various levels of interaction. Semi-autonomous navigation systems for wheelchairs, which adapt to the patient autonomy level (Fioretti et al. 2000; Seki et al. 2000; Kitagawa et al. 2001), or provide users with driving assistance (Yanko 1998) are an example of this approach. Moreover, it should be possible to collect signals for controlling robots in an 'intelligent home' from different sources depending on the user residual abilities (e.g., vision based signals Rao et al. 2002, electro-encephalographic brain signals Belic et al. 2005, or hand gestures Do et al. 2005).

On the other hand, socially interactive robots have been defined by Fong et al. (2003) to describe robots whose main task is some form of interaction. Fong categorizes these robots by the aspects of social interaction (speech, gestures, etc.) they use. Concerns regarding human perception of robotics, particularly the difference in social sophistication between humans and social robots, are addressed, and long-term interaction is an area worthy of future research. Thus, these systems should be validated by experiments on potential users, as in Wada et al. (2005). Besides, in these applications, an important role is played by the robot design, which should be familiar and friendly-looking, in order to facilitate everyday interaction (Irie and Shibata 1997; Kanamori et al. 2003).

SARS share with assistive robotics the goal to provide assistance to human users, but specify that the assistance is through social interaction. In SARS, the robot goal is to create close and effective interaction with a human user for the purpose of giving assistance and achieving measurable progress in convalescence, rehabilitation, etc.

In this paper, we present the integration of a mobile robot in the ASPICE (Assistive System for Patient's Increase of Communication, ambient control and mobility in absence of muscular Effort) project (Cincotti et al. 2008). One central feature of the ASPICE system is the possibility, for the user, to remotely control the motion of a mobile robot (a Sony AIBO) by means of a reduced set of commands, including commands from a Brain-Computer Interface (BCI). While moving, the robot should assist the user by monitoring the environment, and by communicating specific requests to the caregiver. Depending on the residual abilities of the user, as well as on the desired task, it is possible to choose between three different navigating modes for controlling the robot motion. Automatic obstacle detection and avoidance is integrated in the system to guarantee safe, collision-free motion in cluttered environments. Human-robot interaction during the ASPICE experimentation has been assessed by patient feedback, in order to evaluate the social aspects of the system. Hence, both physical and social interaction between the user and the robot are considered in the project. A comparative study of our technique with respect to other techniques developed for different applications (and in particular those cited in the above survey) is however very difficult, since the characteristics of our robot navigation system were specifically designed according to the ASPICE requirements (e.g., the remote robot control via a low-frequency device such as a remote BCI, and the use of a low-cost robot with closed hardware and characteristic locomotion).

The development of the ASPICE robot navigation system has been described in part in Cherubini et al. (2007). Here, we provide a more complete overview of the system over (Cherubini et al. 2007). First, we add more details on the robot driver primitives, on the BCI, and on other robot features implemented in ASPICE. Second, extended experiments showing various aspects of the system are described. The experiments are used to emphasize:

- the performance of the obstacle avoidance algorithms,
- the performance of the BCI-based navigation by comparison with standard devices,
- the localization performance in the map-based navigation mode,
- the utility of the designed navigation modes for accomplishing everyday tasks.

Finally, this article presents a deeper insight into the clinical validation of the ASPICE navigation system.

The article is organized as follows. In Sect. 2, the architecture of the ASPICE system is briefly illustrated. In Sect. 3, the main features of the AIBO robot are described. Section 4 presents the primitives developed for the robot framework, at perception and motion levels. The robot navigation modes that we implemented, on the basis of the ASPICE requirements, are outlined in Sect. 5. Other aspects of the ASPICE robot driver are described in Sect. 6. Simulations and experiments are reported in Sects. 7 and 8, respectively. In the conclusion, we summarize the results.

2 The ASPICE project

2.1 Overview

The ASPICE project received in 2004 a two-year funding grant from TELETHON, an Italian medical research charity foundation. The project involved three partners, among which the Clinical Neurophysiopathology Laboratory of the Fondazione Santa Lucia IRCCS and the Robotics Lab of the University of Rome "La Sapienza".

The project was aimed at the development of a technological aid which allowed neuromotor-disabled users to improve or recover their mobility and communication within the surrounding environment. The project was particularly addressed towards those patients in which the residual muscular strength was low and practical obstacles or security concerns did not allow a displacement from the bed (see Cincotti et al. 2008 for more details). Therefore, the major requirements were: adaptability to different levels of disability, low cost, and robustness to the setting. Depending on the user requirements, the assistive device would be a program running on a common low-power PC, on a palmtop, or on a powerful workstation.

The ASPICE architecture, with input and output devices, is summarized in Fig. 1. Some key elements of the system are:

- a variety of input devices for easy access to the Control Unit: these include standard input devices (mouse, joy-

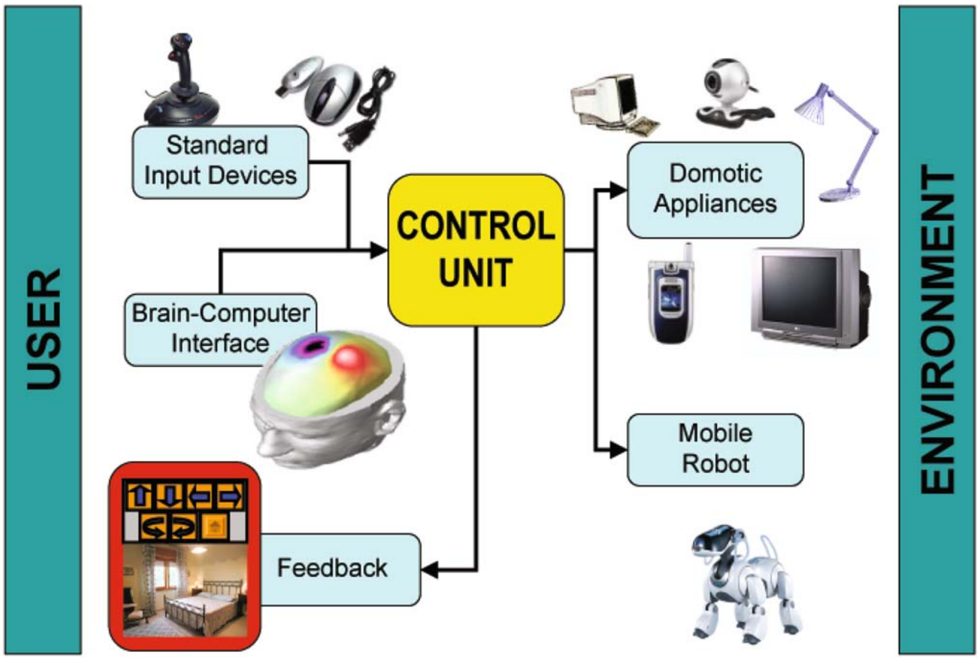


Fig. 1 The ASPICE architecture

stick, eye tracker, voice recognition) as well as a Brain-Computer Interface;
- the Control Unit, which receives signals from the input devices through a Graphic User Interface (GUI) and converts them into commands that drive the output devices (either the domotic appliances or a mobile robot);
- the mobile robot;
- a number of domotic appliances, which must comply with the patient's need for ambient control (e.g., TV, lights, video camera, telephone, personal computer);
- visual feedback (either through the fixed video camera or through the robot vision system) to provide the user with an increased sense of presence in the environment.

The Control Unit contains drivers for all output devices; in some cases, previously existing drivers are utilized, whereas in other cases (e.g., the mobile robot) the driver has been designed "ad hoc" for the specific system. Note that all the signals between the input devices and the Control Unit, and between the latter and the output devices (including visual feedback) are transmitted over a wireless connection. In this paper, we do not focus on ASPICE input devices other than the mouse and BCI, nor on output devices other than the mobile robot AIBO. For further details, the readers should refer to Cincotti et al. (2008). In the remainder of this section, the BCI which is used as input device and the Robot Driver that we implemented for controlling AIBO are briefly described.

2.2 Brain-computer interface

The ASPICE system input devices are customized on the severely motor impaired patients' residual abilities, based on the aforementioned technologies. Users can utilize the aids they are already familiar with, and on the other hand, the variety of input devices provides robustness to the worsening of the patient abilities, which is a typical consequence of degenerative diseases.

When the patient is not able to use any of the standard input devices, or when a degenerative disease likely implies that in the future he/she will no more be able to use them, a BCI should be utilized to access the Control Unit. The BCI gives the user communication and control channels that do not depend on the brain normal output channels of peripheral nerves and muscles (Wolpaw et al. 2002). In other terms, a BCI can detect the activation patterns of the brain, and whenever the user induces a voluntary modification of these patterns, it is able to detect it, and to translate it into an action that is associated to the user will. BCI technology has substantially improved in the last decade, and it is reasonable to expect that, in the near future, a wider class of users will profit from it. Though the number of completely paralyzed patients is rather small (some hundred thousand worldwide), BCIs have the relevance that derives from being the 'only' option for such users, who would otherwise be locked in their bodies.

As it emerges from a concise review of related work, real time control tasks based on human EEG have been addressed to simple applications, such as moving a computer cursor on a screen (Wolpaw et al. 1991), opening a hand orthosis (Pfurtscheller and Neuper 2001), controlling a wheeled robot (del Millán et al. 2004), or driving a wheelchair (Rebsamen et al. 2006). Recently, an experimental BCI which was implanted into the brain motor cortex, enabled a tetraplegic to move a computer cursor (Hochberg et al. 2006). However, to our knowledge, application of non-invasive BCI technology to interaction with a wider set of devices has not been explored yet, and represents one of the goals of the ASPICE system.

The BCI used in ASPICE can be based alternatively on time domain (i.e., P300 evoked potentials Farwell and Donchin 1988) or on frequency domain features (i.e., sensorimotor rhythms Wolpaw and McFarland 1994) of the EEG signal. The performance of these two BCI versions varies on an individual basis, and the most reliable features are chosen depending on the user predisposition. In both versions, a visual interface (on a screen) aids the user in choosing the command to be sent to the ASPICE Control Unit. The EEG potentials are captured by means of an electrode cap, amplified, digitized and transmitted to a personal computer. Processing is handled by the BCI2000 software package (Schalk et al. 2004). After a preliminary signal conditioning phase, which includes a linear mixture of channels implementing a high pass spatial filter, the features (either in the time or frequency domain) are extracted and used to identify the user desired command.

When time domain features are employed, a sequence of icons corresponding to possible ASPICE commands is shown on the screen to the user. The icons are highlighted successively one by one (ca. 3 icons are highlighted per second). After each sequence, classification is implemented, and the command corresponding to the identified target icon is forwarded to the ASPICE Control Unit. Time domain features are the result of an averaging procedure: the mean of time samples at equal latency from the stimulus (i.e., the icon display) is computed, for each channel, and for each stimulus. Averaged potentials at specific latencies and channels are fed into a linear classifier. The latency/channel choice, and the weights for classification are determined in advance by training the classifier. A threshold is used to assess reliability of the classification.

When frequency domain features are employed, two targets, positioned at the top and bottom edge of the screen, are shown to the user. Two actions (*scroll* along all possible ASPICE commands, and *select* the desired command) are associated with the targets. The subject controls the vertical velocity of a cursor on the visual interface by modulating the amplitude of his EEG sensorimotor rhythms above or below a dynamically adapted threshold value. When the cursor reaches either the top or the bottom target, the corresponding action (scroll or select) is performed in order to choose the command to be forwarded to the ASPICE Control Unit. Frequency domain features are computed using a parametric estimation, which takes into account the latest 300 ms of signal and is updated every 100 ms. The power spectral density values at specific channels and frequency bins (which are identified in advance during the training phase) are linearly combined. The output is detrended using a moving average value, which avoids drift of the control signal if the EEG amplitude is increased or decreased (e.g., due to non-voluntary arousal effects).

2.3 The robot driver

In any assistive robotics project, a major requirement is the user friendliness of the robotic platform. In fact, although in recent years users are becoming, on the average, more acquainted with technology, characteristics such as low cost, safety, and low request for maintenance are still fundamental needs of any biomedical robotic application. Moreover, clinicians have often emphasized the importance of working with a familiar, friendly-looking robot, in order to limit its psychological impact on patients (Irie and Shibata 1997). In our case, these considerations led to the choice of the dog-like robot Sony AIBO ERS-7 (described in Sect. 3) for inclusion in the system. Besides, studies on improvement of quality of life, among elderly, using AIBO, have given good results (Kanamori et al. 2003).

AIBO should be driven around the user home with a small set of commands, depending on the residual abilities. It should also assist the impaired patient in visually monitoring the environment and in communicating with the caregiver. Partial autonomy should be implemented in order to avoid collisions with unexpected obstacles present in the environment. Another requirement is that AIBO should be able to charge its battery when needed without any user intervention. As aforementioned, one of the objectives of the ASPICE project is compatibility with a variety of users and their level of disability. In this spirit, three navigation modes have been developed: *Single step*, *Semi-autonomous* and *Autonomous* mode. The user is expected to choose single step navigation when he/she wants to retain complete control of the robot motion; e.g., for fine motion in cluttered areas. In semi-autonomous navigation, the user specifies the main direction of motion, leaving to the robot the task of avoiding obstacles. Finally, in the autonomous navigation mode, only a target point in the environment is assigned by the user, and the robot travels to the target; this is useful for quickly reaching some important locations (a window, the front door, the kitchen). This mode of operation is expected to be particularly useful for severely impaired patients, which are unable to send frequent commands. All three navigation modes must contain some level of obstacle avoidance.

Each navigation mode is associated to a GUI in the ASPICE Control Unit. The three GUIs are shown in Fig. 2. By selecting the corresponding button from the single step GUI, the user can control the direction of the step. From the semi-autonomous mode GUI, the user can select one of six directions—the same of the single step mode—or stop the robot. Instead, from the autonomous navigation mode GUI, each button that the user can select corresponds to a destination in the user apartment (here, the bedroom, the living room, the bathroom, and the kitchen).

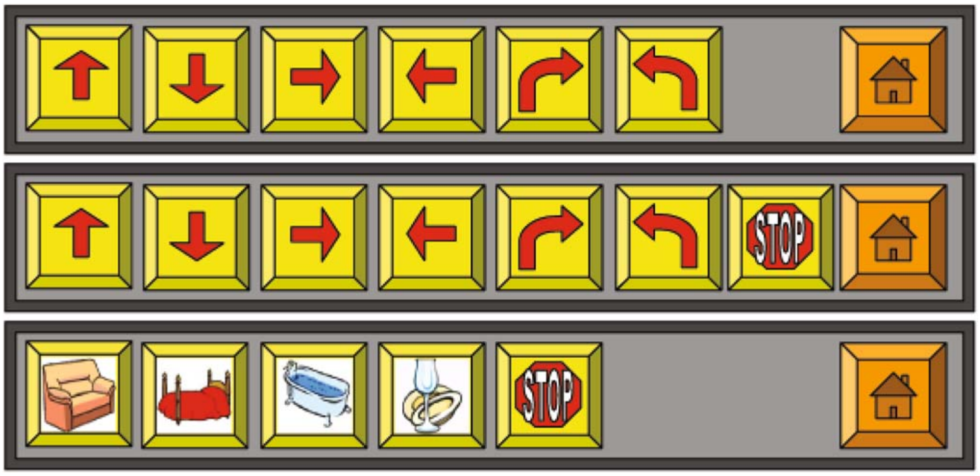


Fig. 2 The ASPICE navigation GUIs: single step (*above*), semi-autonomous (*center*) and autonomous (*below*) modes. In each GUI, the home button brings back to the ASPICE main GUI

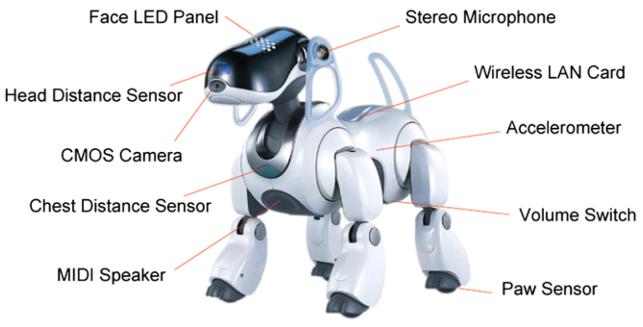


Fig. 3 The Sony AIBO ERS-7 used in the ASPICE project

3 The robot platform: AIBO

The platform used in this work is a quadruped robot, Sony AIBO ERS-7, pictured in Fig. 3. AIBO is a very interesting low-cost robot, widely used for research as well as entertainment purposes. The robot is equipped with 20 actuated joints, a CMOS camera, two distance sensors (on the head and on the chest), an accelerometer, a stereo microphone, a MIDI speaker, a set of leds and pression sensors. A wireless LAN card enables remote control and debugging. The actuated joints are: 3 for each leg, 3 for the head (head tilt, head pan, and neck tilt), 2 for the tail, 1 for each ear and 1 for the mouth. AIBO's real-time operating system APERIOS runs a specialized layer called OPEN-R, a cross-development environment based on C++. The robot behavior is programmed by loading all executable and configuration files on a memory stick which is read by the on-board processor. In spite of the above features, the AIBO robot presents many limitations, which made its use within the ASPICE project a real challenge. The most severe are the following:

- the closed hardware prevents the addition of sensors and/or actuators;
- since Sony does not release the code of its driver, we had to realize from scratch an *ad hoc* driver for this work;
- the head distance sensor and the CMOS camera move in accordance, making it impossible for the distance sensor to detect obstacles in directions other than the one pointed by the camera: a tradeoff between moving the head for video feedback and moving it for obstacle detection/avoidance had to be reached;
- the chest distance sensor is constrained to the robot body and peculiarly oriented, thus limiting its effective utility;
- vibrational and slipping effects during the quadruped gait cycle make odometric reconstruction very inaccurate in the long run;
- the variable attitude of AIBO during its gait precludes the use of an external sensory system (e.g., based on infrared triangulation with a detector placed on the robot) for solving the localization problem.

4 Primitives

In order to utilize AIBO, specific primitives have been developed and integrated in the driver framework. The primitives have been designed to fulfill the robot driver requirements, i.e. obstacle detection/avoidance, motion control, and path planning.

Let us define the three reference frames which will be used in this work:

- the *robot frame* (Figs. 4 and 6) with origin fixed at the robot center projection on the ground, x-axis in the forward direction, y-axis pointing the left side of the robot, and z-axis in the vertical direction;
- the *image frame* (Fig. 5) with origin fixed at the top left corner of the image, horizontal ${}^{i}x$-axis pointing right, and ${}^{i}y$-axis pointing downward—the coordinates of the image center in this frame are noted: $[{}^{i}\bar{x}\ {}^{i}\bar{y}]^{T}$;
- the *camera frame* (Figs. 4 and 6) with origin fixed in the camera center, horizontal ${}^{c}x$-axis pointing right, ${}^{c}y$-axis pointing downward, and ${}^{c}z$-axis pointing forward—in

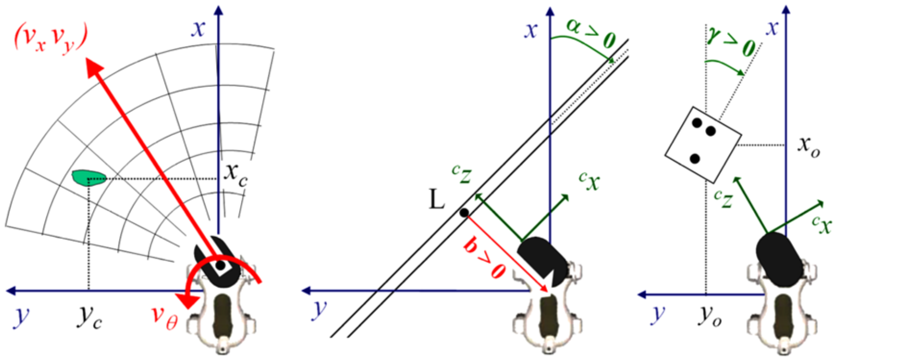


Fig. 4 Relevant variables utilized in: (**a**) occupancy grid generation, (**b**) straight white line tracking, and (**c**) coded square tracking

Fig. 6, we also noted the robot neck tilt, head pan, and head tilt joint positions respectively with: $q_H = [\psi_1\ \varphi\ \psi_2]^T$.

4.1 Perception primitives

The main features that the robot should perceive are the obstacles that it should avoid, and the landmarks that it needs for localization and path planning purposes. We chose to use the robot range sensors to detect obstacles, and the camera to recognize visual landmarks. We use a local two-dimensional occupancy grid to represent the detected obstacles, built by the *occupancy grid generator*. The visual landmarks that we use are straight white lines and coded squares placed on the floor. Thus, a *straight white line extractor* and a *coded square extractor* have been developed. Moreover, the visual landmarks should be located in sequential scenes. This task is accomplished by a *visual landmark tracker*.

4.1.1 Occupancy grid generator

The robot should be able to recover robust and useful spatial descriptions of its surrounding obstacles, using sensory information. These descriptions should be used for short-term planning in the environment. To do this, we use a tesselated two-dimensional representation of spatial information called the occupancy grid. In the past years, the occupancy grid framework proved to be extremely efficient for performing path planning and obstacle avoidance in unknown and unstructured environments, and researchers proposed different functions for updating the grid cells (e.g. Fuzzy (Oriolo et al. 1997); Bayesian (Howard and Kitchen 1996); Gaussian (Elfes 1989)). The occupancy grids have also been used for obstacle detection on Sony AIBO. Fasola et al. (2005) make use of the robot camera to generate a local occupancy grid, used for taking navigation decisions. A similar approach is used in Hoffmann et al. (2004), where recent information is integrated along with current information, based on odometric data. However, both works were used in the Robocup application, where free space can be easily identified by the green color (the color of the soccer field). Instead, in unknown environments, using visual information for obstacle detection is very challenging. The AIBO range sensors are a better tool, although only two are available. In Hugel et al. (2003), a scanning motion of the AIBO head distance sensor is used to map the obstacles locally, and the position of the barycenter of the sensor readings after every scan is used for robot navigation.

For our application, a simple local instantaneous map without prior information is sufficient. In our approach, the two range finders (head and chest) are used to detect obstacles, although the chest sensor, due to its limited range and particular orientation (see Fig. 3), can only detect near obstacles and therefore is not used to compute the occupancy grid. Thus, only the head sensor is utilized to build the local occupancy grid by moving the head pan joint along a sinusoidal profile spanning an angular width of 90°. While the origin of the occupancy grid is always on the head pan axis, and its longitudinal extent is limited by the range of the head distance sensor (1 m), its orientation (i.e., the direction of its bisectrix) is the same as the direction of motion $(v_x\ v_y)$ (see Fig. 4a). As is shown in the figure, 30 grid cells are used. The cells are annulus sectors of width 15° and height 0.2 m. Obviously, due to the joint limit, it is impossible to build occupancy grids for backward motions. The grid may be built with the robot either stationary or in motion. In the second case, the head pan movement is synchronized with the gait cycle, and odometric data (reconstructed through the leg joint encoders) are used to build a consistent map: in practice, at every time frame, the previous range readings are displaced on the map according to the robot measured motion. When the pan cycle is complete, a cell in the grid is considered to be occupied if there is at least one sensor reading indicating an obstacle inside that cell. Although the grid is built locally, our approach is effective for various reasons. First, it does not require excessive exploitation of the robot computational resources. Second, since the robot builds more than 10 occupancy grids while traveling the distance corresponding to the longitudinal extent of the distance sensor (1 m), obstacles on the route (including moving obstacles) are unlikely to be missed. Thirdly, due to its low weight, AIBO has a small inertia, which enables it to be very reactive as soon as obstacles are perceived.

4.1.2 *Straight white line extractor*

A requirement of the robot driver is straight white line extraction. In order to be independent from color classification, the straight white lines are detected by search only on the luminance signal $I({}^ix,\ {}^iy)$. Thus, line edges are searched at pixels with a strong variation of luminance with respect to that of adjacent pixels. For each pixel located in $p = [{}^ix\ {}^iy]^T$ the gradient of luminance $\nabla I(p)$ is computed, using the Roberts operator (Roberts 1965) as in Rofer et al. (2005):

$$\begin{aligned}
&s(p) = I\left({}^ix+1\ ,\ {}^iy+1\right) - I\left({}^ix\ ,\ {}^iy\right)\\
&t(p) = I\left({}^ix+1\ ,\ {}^iy\right) - I\left({}^ix\ ,\ {}^iy+1\right)\\
&|\nabla I(p)| = \sqrt{s(p)^2 + t(p)^2}\\
&\angle \nabla I(p) = \mathrm{ATAN2}(s(p), t(p))
\end{aligned} \tag{1}$$

where $|\nabla I(p)|$ is the magnitude and $\angle \nabla I(p) \in (-\pi, \pi]$ is the direction of the pixel luminance gradient (with respect to line ${}^iy = -{}^ix$ and positive CW, see Fig. 5. Edges are then detected by applying a threshold test to $|\nabla I(p)|$. We use a threshold T dependent on the mean value (noted $\mu_{|\nabla I|}$) of $|\nabla I(p)|$ on the given image. This adaptive thresholding makes the edge detection algorithm more robust to varying light conditions, as compared to similar works implemented in environments where light conditions had to be controlled. The threshold test may be written:

$$\begin{aligned}
&p \in P_e \quad \text{if } |\nabla I(p)| \geq T\left(\mu_{|\nabla I|}\right)\\
&p \notin P_e \quad \text{else}
\end{aligned} \tag{2}$$

where P_e is the set of image edge pixels (marked in yellow in Fig. 5).

Afterwards, by applying threshold tests to relative distances and to luminance gradient directions of the edge pixels belonging to P_e, subsets of line pixels are derived. Indicating with N_{SWL} the total number of straight white lines extracted on the image, the line pixel subsets are noted: $P_{SWL,j}$ ($j = 1, \ldots, N_{SWL}$). Each $P_{SWL,j}$ defines a line detected on the image frame, and must contain at least $n_{SWL,min}$ pixels.

We tested several other edge detection algorithms (Russ 1999; Smith and Brady 1997), but the Roberts operator showed better results, although the aforementioned methods are more efficient for color space based extraction (Wesolkowski et al. 2000). Besides, due to its low computation time, this line extraction method was preferred to the Hough Transform technique, which we also experimented, and which is widely used for line detection in noisy images, with extensions also accounting for line connectivity and thickness, as in Yang et al. (1997).

In conclusion, the straight white line extractor algorithm returns the coordinates of the pixels belonging to the N_{SWL} lines extracted on the image frame (Fig. 5):

$$\begin{aligned}
&[{}^ix_r\ {}^iy_r]^T_{SWL,j} \in P_{SWL,j}\\
&\quad r = 1, \ldots, n_j,\ \ j = 1, \ldots, N_{SWL}
\end{aligned} \tag{3}$$

4.1.3 *Coded square extractor*

Along with the straight white lines, we have chosen to use as visual landmarks a set of white coded squares laid on the ground. The identity and orientation of each square is uniquely identified through a black dots code, similarly to Carreras et al. (2003). The choice of binary coding, i.e., black and white, is aimed at using luminance variation, instead of color classification, for extracting the square characteristics. We arranged from 1 to 7 black dots on the border of the squares, in order to generate configurations which uniquely define the landmark identity (defined by its label: *ID*) and orientation. The 15 landmarks which we used can be seen in Fig. 7. Note that all the used landmarks are unambiguous with respect to multiple 90° orientation errors. Hence, the landmark identity and orientation can be uniquely identified in spite of the square rotational symmetry.

In practice, edges of squares are searched within the set of edge pixels P_e derived as in the straight white line extractor. The robot frame coordinates of all edge pixels are derived from their image frame coordinates, with the projection which will be presented below. Then, the projected edges are compared with a reference square with the same dimensions of the coded square, so that the square perimeter pixels are identified. These define a subset of P_e for each square (see Fig. 5). Indicating with N_{CS} the total number of coded squares extracted on the image, the subsets of edge pixels of each square l are noted: $P_{CS,l}$ ($l = 1, \ldots, N_{CS}$). Each $P_{CS,l}$ defines a coded square detected on the image frame, and must contain at least $n_{CS,min}$ pixels. Afterwards, pixels on the segments (scanlines) leading from the center to the perimeter edges are classified by using a binary segmentation which uses the mean value of I on the image as threshold. Corke (1996) showed how binary segmentation is affected by noise, threshold selection and edge gradient. However, in this application, the choice of binary coding and the use of binary segmentation only in a small image window, and with adaptive thresholding, reduces these problems. Finally, black dots are extracted by associating a sufficient number of near black pixels found on the scanlines.

In conclusion, the coded square extractor returns, for each of the N_{CS} detected coded squares: the image coordinates of the square center o and of the centers of the n_{dots} black dots (respectively marked in red, and in cyan

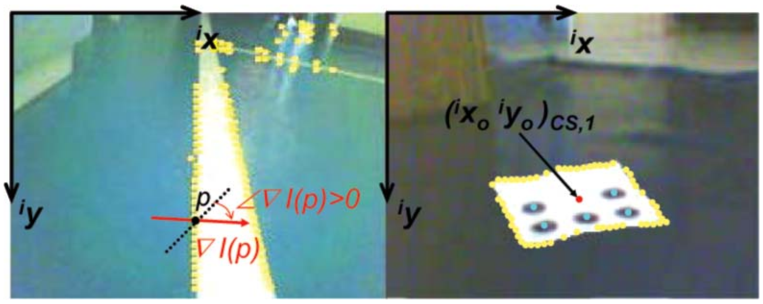


Fig. 5 (Color online) Extracting edges (in *yellow*), for *straight white line* (*left*) and coded square (*right*) detection. The detected coded square center is marked in *red*, and the *dot centers* are marked in *cyan*

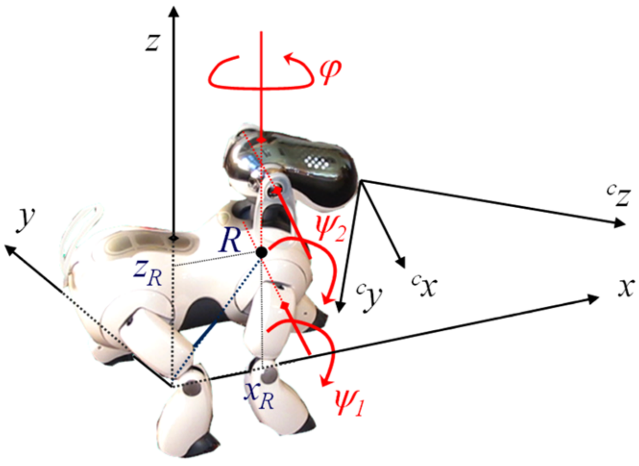


Fig. 6 The camera frame and head joint positions

in Fig. 5):

$$[{}^i x_o\ {}^i y_o]^T_{CS,l} \qquad [{}^i x_m\ {}^i y_m]^T_{CS,l},$$
$$l = 1, \dots, N_{CS},\ m = 1, \dots, n_{dots},\ n_{dots} = 1, \dots, 7 \tag{4}$$

4.1.4 Visual landmark tracker

The straight white line extractor and coded square extractor only take into account information from the current image, and give no long-term knowledge. Thus, consistent landmarks must be obtained by comparing the extracted landmarks over consecutive images. This is done by projecting the characteristic points of each extracted visual landmark $VL = SWL_1, \dots, SWL_{N_{SWL}}, CS_1, \dots, CS_{N_{CS}}$ from the image frame (coordinates $[{}^i x\ {}^i y]^T_{VL}$) to the robot frame (coordinates $[x\ y\ z]^T_{VL}$). Such mapping is not one-to-one, and can only determine the projecting ray of the point. However, in our application, since all the visual landmarks are on the ground plane, the problem can be solved in closed form.

In fact, given the intrinsic parameters of the camera:

α_x	scaling factor in pixels/mm for the ${}^i x$-axis
α_y	scaling factor in pixels/mm for the ${}^i y$-axis
${}^i \bar{x}$	image plane center abscissa in pixels
${}^i \bar{y}$	image plane center ordinate in pixels
f	focal length in mm

the point coordinates in the camera and image frames can be related (Corke 1996) by:

$$\begin{pmatrix} {}^i x \\ {}^i y \end{pmatrix} = \begin{pmatrix} {}^i \bar{x} \\ {}^i \bar{y} \end{pmatrix} + \frac{f}{{}^c z - f} \begin{pmatrix} \alpha_x {}^c x \\ \alpha_y {}^c y \end{pmatrix} \tag{5}$$

This equation can be rewritten:

$$\begin{pmatrix} -f\alpha_x & 0 & {}^i x - {}^i \bar{x} \\ 0 & -f\alpha_y & {}^i y - {}^i \bar{y} \end{pmatrix} \begin{pmatrix} {}^c x \\ {}^c y \\ {}^c z \end{pmatrix} = f \begin{pmatrix} {}^i x - {}^i \bar{x} \\ {}^i y - {}^i \bar{x} \end{pmatrix} \tag{6}$$

Besides, given the homogeneous transformation matrix ${}^r\mathbf{T}_c$ representing the camera frame pose with respect to the robot frame, the coordinates of a point in the two reference frames are related by:

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = {}^r\mathbf{T}_c \begin{pmatrix} {}^c x \\ {}^c y \\ {}^c z \\ 1 \end{pmatrix} \tag{7}$$

We neglect the distance between the robot head tilt and head pan axes, assuming they intersect in a point which stays in a constant position $R = [x_R\ 0\ z_R]^T$ in the robot frame (see Fig. 6), and that the robot body maintains constant orientation around the y axis, and null orientation around the two other axes, during motion. Under these assumptions, ${}^r\mathbf{T}_c$ can be easily computed at every frame through the three head joint positions $q_H = [\psi_1\ \varphi\ \psi_2]^T$, by using the Denavit and Hartenberg method (Hartenberg and Denavit 1955) as in Rofer et al. (2005). Since landmarks are on the ground plane (i.e. $z_{VL} = 0$) the third equation from (7) can be expanded and used, along with (6):

$$\begin{pmatrix} t_{31} & t_{32} & t_{33} \\ -f\alpha_x & 0 & {}^i x_{VL} - {}^i \bar{x} \\ 0 & -f\alpha_y & {}^i y_{VL} - {}^i \bar{y} \end{pmatrix} \begin{pmatrix} {}^c x \\ {}^c y \\ {}^c z \end{pmatrix}_{VL} = \begin{pmatrix} -t_{34} \\ f({}^i x_{VL} - {}^i \bar{x}) \\ f({}^i y_{VL} - {}^i \bar{x}) \end{pmatrix} \tag{8}$$

where the t_{pq} are the elements of ${}^r\mathbf{T}_c(q_H)$. Inverting this equation away from singularities allows for computation of the landmark position $[{}^c x\ {}^c y\ {}^c z]^T_{VL}$ in the camera frame, given the landmark position $[{}^i x\ {}^i y]^T_{VL}$ in the image frame returned by the straight white line extractor and coded square extractor. Finally, substituting $[{}^c x\ {}^c y\ {}^c z]^T_{VL}$ in (7) gives the landmark position $[x\ y\ 0]^T_{VL}$ in the robot frame.

The robot frame coordinates of all straight white line points $[x_r\ y_r\ 0]^T_{SWL,j}$ are then processed with a least square error algorithm in order to identify the parameters $[b\ \alpha]^T_{SWL,j}$ of each of the N_{SWL} lines (see Fig. 4b). Variable b is the signed distance between the nearest line point L and the robot (positive for positive y_L), and $\alpha \in (-\frac{\pi}{2}, \frac{\pi}{2}]$ denotes the orientation offset. A similar approach is used to process all the coded squares characteristic points $[x_o\ y_o\ 0]^T_{CS,l}$, and $[x_m\ y_m\ 0]^T_{CS,l}$, and obtain the identity $ID_l = 1 \dots 15$ and orientation γ_l, of each of the N_{CS} coded squares (see Fig. 4c). Visual landmarks extracted and projected at previous frames are displaced according to the robot measured motion and compared with the current projected landmarks for checking consistency and filtering out false positives.

The algorithm returns the characteristics of the visual landmarks, validated in a sufficient number of consecutive frames:

$$\begin{aligned} &[b\ \alpha]^T_{SWL,j} && j = 1, \dots, N_{SWL} \\ &ID_l\ \gamma_l\ [x_o\ y_o\ 0]^T_{CS,l} && l = 1, \dots, N_{CS} \end{aligned} \tag{9}$$

4.2 Motion primitives

From a kinematic viewpoint, AIBO can be considered as an omnidirectional robot, i.e., three velocities (forward v_x, lateral v_y, and angular v_θ around the robot center, positive for CCW rotation) can be independently specified (Fig. 4a). In all cases where the robot velocities are specified in workspace coordinates as $V = [V_x\ V_y]^T$ (e.g., when they are imposed by a user command), they must be mapped to the configuration space. To perform this conversion, we have tested two strategies. The first (*omnidirectional translational motion*), consists of simply setting $[v_x\ v_y\ v_\theta]^T = [V_x\ V_y\ 0]^T$. Instead, the second kind of conversion (*nonholonomic-like motion*), consists in setting:

$$\begin{aligned} v_x &= V_x \\ v_y &= 0 \\ v_\theta &= \mathrm{ATAN2}(V_y, V_x) \end{aligned} \tag{10}$$

The characteristics of each strategy have determined their utilization for each robot behavior, as will be illustrated later.

Basic motion primitives for controlling the robot legs in order to obtain the desired motion $[v_x\ v_y\ v_\theta]^T$ are based on the quadruped parameterized walk inspired by the work of Hengts et al. (2001), which is widely used in the four-legged robot community, and which we will not discuss in detail. Velocity commands computed by the motion primitives are suitably scaled if any of them exceeds the physical limits of the actuators.

We also developed three primitives: *Landmark fixer*, *Landmark approacher* (LA), and *Straight line follower*, which use visual information returned by the perception primitives to guide the robot. In practice, the robot actuators are driven by a *visual servoing* scheme. Since the visual landmark tracker returns the position of the visual landmarks relative to the robot, *position-based* visual servo control turns out to offer a better solution than *image-based* servoing. The three vision-based motion primitives are explained below.

4.2.1 Landmark fixer

Referring to Corke (1996), "fixation" is defined as motion aimed at keeping one point in the scene (the "target", noted TG) at the same location in the image plane. In this work, it is of great interest to apply this control scheme to a visual landmark, by keeping it centered in the image plane. Advantages include: reducing chances of losing sight of the landmark during motion, reducing motion blur, and reducing the effect of geometric distortion in the lens (since the optical axis will be pointed at the landmark). In many *visual servoing* works, the knowledge of camera motion during fixation is used to determine the 3D position of the target: $[x\ y\ z]^T_{TG}$. Instead, in this application, since the 3D position of the target is returned by the visual landmark tracker algorithm outlined above both for straight white lines and for coded squares, it can be used as is for fixation.

For fixation of a straight white line with parameters $[b\ \ \alpha]^T$, we choose the 3D position of the target to be the position of the line point L nearest to the robot: $[x\ \ y\ \ z]^T_{TG} = [b \sin\alpha\ \ \ b\cos\alpha\ \ \ 0]^T$. Instead, for fixation of a coded square with center $[x_o\ \ y_o\ \ 0]^T$, we choose: $[x\ y\ z]^T_{TG} = [x_o\ y_o\ 0]^T$. In both cases, the position $[x\ y\ z]^T_{TG}$ is used for solving the inverse kinematics problem of finding the head joint coordinates for fixation. In practice, given the target coordinates in the robot frame, the fixation task consists in finding the robot head joint positions $q_H = [\psi_1\ \ \varphi\ \ \psi_2]^T$ such that the target coordinates in the camera frame are $[{}^c x = 0\ \ \ {}^c y = 0\ \ \ {}^c z]^T_{TG}$ (corresponding to centering the target in the image plane). This is equivalent to solving equation:

$$\begin{pmatrix} 0 \\ 0 \\ {}^c z \\ 1 \end{pmatrix}_{TG} = {}^c T_r(q_H) \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}_{TG} \tag{11}$$

for q_H. ${}^c T_r(q_H)$ is the homogeneous transformation matrix representing the robot frame pose with respect to the camera frame coordinates. It is derived by inverting ${}^{r}T_c(q_H)$ away from singularities.

Apart from singular configurations where no solution can be determined due either to joint limits (e.g. target "behind" the robot head: in this case a different target point must be chosen) or to singularities of ${}^{r}T_c(q_H)$, for most configurations the solution of (11) is non-unique. In fact, although the head pan joint position φ can be derived from the first equation in (11):

$$\varphi = \mathrm{ATAN2}(y_{TG}, x_{TG} - x_R) \tag{12}$$

replacing it in the two other equations does not guarantee a unique solution for the head tilt and neck tilt joint positions. In our implementation, we choose, whenever possible (i.e. when x_{TG} is "sufficiently" large) to fix ψ_1 to its lower bound (in order to maximize scene depth ${}^{c}z$), and derive ψ_2 from (11). If it is not possible to adopt this strategy due to the ψ_2 joint limits, we fix ψ_2 to its limit, and derive ψ_1.

4.2.2 *Landmark approacher*

When the robot finds a landmark (with the extractors described in Sect. 4.1), it should approach it, in order to get a better perception, which can be useful for localization purposes. In practice, it is a *posture stabilization* task with reference configuration defined by the absolute position and orientation of the landmark. As we suggested previously, some tasks can be accomplished more effectively if nonholonomic-like motion is enforced. However, no smooth state-feedback control law can solve the non-square posture stabilization problem for a nonholonomic mobile robot (Canudas de Wit et al. 1996). Alternative control approaches (e.g. smooth time-varying (Samson and Ait-Abderrahim 1991) and discontinuous feedbacks) have shown limitations such as slow convergence and oscillatory transient. These considerations, along with the requirement of minimizing the path to the target, led us to the choice of omnidirectional motion, instead of nonholonomic motion, in the implementation of the landmark approacher.

The omnidirectional walk that drives the robot to the landmark implements a proportional closed-loop control strategy for reducing the robot relative distance and orientation with respect to the nearest landmark perceived.

In the case of straight white line approaching, this is done by setting robot velocities:

$$\begin{cases} v_x = \kappa_{\mathrm{T}}\ b\ \sin\alpha \\ v_y = \kappa_{\mathrm{T}}\ b\ \cos\alpha \\ v_\theta = -\kappa_{\mathrm{R}}\,\alpha \end{cases} \tag{13}$$

A similar controller is used for coded square approaching; in this case the robot velocities are set to:

$$\begin{cases} v_x = \kappa_{\mathrm{T}}\ x_o \\ v_y = \kappa_{\mathrm{T}}\ y_o \\ v_\theta = -\kappa_{\mathrm{R}}\,\gamma \end{cases} \tag{14}$$

In both cases, κ_{T} and κ_{R} are positive given gains.

4.2.3 *Straight line follower*

This primitive should solve the *path following* problem for a straight white line, i.e. find a control law such that

$$\lim_{t\to\infty} b(t) = 0 \qquad \lim_{t\to\infty} \alpha(t) = 0 \tag{15}$$

For this problem, we decided to adopt a nonholonomic model for the robot, in order to obtain more effective obstacle avoidance (as stated further), and a more "natural-looking" walk. Moreover, the *path following* problem differs from the *posture stabilization* problem in that both linear and non-linear smooth state-feedback control solutions exist for a nonholonomic mobile robot (Canudas de Wit et al. 1996). On the other hand, since the task is less stringent than *posture tracking*, it can be achieved by using only one control variable. In this work, we utilize only the angular velocity v_θ. AIBO is modeled as a unicycle robot, with velocities $[v_x \quad v_y = 0 \quad v_\theta]^T$ and it is rather simple to verify that the following kinematic equations hold:

$$\begin{aligned} \dot{b} &= -v_x \sin\alpha \\ \dot{\alpha} &= v_\vartheta \end{aligned} \tag{16}$$

Linear feedback control can be realized by tangent linearization of the two equations above, in the neighborhood of $(b = 0, \alpha = 0)$. This gives the second order linear system:

$$\begin{aligned} \dot{b} &= -v_x\alpha \\ \dot{\alpha} &= v_\vartheta \end{aligned} \tag{17}$$

which is clearly controllable, and thus asymptotically stabilizable by linear state feedback on v_ϑ, when v_x is constant and strictly positive. Thus, let us fix $v_x = v_f > 0$. It is rather simple to verify that a stabilizing linear feedback is of the form:

$$v_\vartheta = (k_2 b - k_3\alpha)\, v_f \tag{18}$$

with:

$$\begin{aligned} k_2 &= a^2 \\ k_3 &= 2\xi a \end{aligned} \tag{19}$$

where $a > 0$ must be chosen so as to specify the transient "rise distance" and $\xi \in (0, 1)$ is the damping coefficient. The validity of this approach at walking and jogging speeds has been proved by similar works, e.g. for hexapod robot line following, as in Skaff et al. (2003).

As an alternative approach, consider the *nonlinear* feedback control:

$$v_\vartheta = \left(k_2 b \frac{\sin\alpha}{\alpha} - k_3\alpha\right) v_f \tag{20}$$

with k_2, k_3 given by (19). Control (20) asymptotically stabilizes $(b = 0, \alpha = 0)$ for any initial robot configuration. The proof can be derived by considering the Lyapunov function:

$$V(b,\alpha) = k_2\frac{b^2}{2} + \frac{\alpha^2}{2} \tag{21}$$

5 Robot navigation modes

All three navigation modes are based on the algorithms presented in Sect. 4. The Single step mode and the Semi-autonomous mode utilize only the Occupancy grid generator. The first sequentially uses the occupancy grid and implements motion control, whereas in the latter occupancy grid generation and motion control are executed simultaneously. The Autonomous mode utilizes *all* the primitives presented in Sect. 4.

5.1 Single step mode

With single step motion, the robot can be driven, with a fixed step size, in any six directions (forward, backward, lateral left/right, CW and CCW rotations). Before performing the motion command, the robot generates the appropriate occupancy grid (oriented along the intention of motion) from its stationary position and verifies whether the step can be performed without colliding with obstacles. The collision is checked by applying a threshold test to the sum of the inverse distances of all the occupied grid cells. If:

$$\sum_c \frac{1}{\|c\|} > T_{SS}$$

(with $\|c\|$ distance of the occupied cell center from the robot center) the collision check is positive, and the robot will not step in the desired direction. Otherwise, the step will be performed. Clearly, the contribution of near occupied cells to the above sum is greater than that of far occupied cells. Note that, since no occupancy grid can be built for backward or rotational motions, the corresponding step commands should be used with care.

5.2 Semi-autonomous mode

With semi-autonomous motion, the user specifies general directions of motion, which the robot should track as closely as possible. Instead of executing a single step, the robot walks continuously in the specified direction until it receives a new command (either a new direction or a stop). If the specified direction is forward or lateral[1] (i.e., the user desired direction of motion in the workspace is $V_{des} = [V_{des,x}\ 0]^T$ or $V_{des} = [0\ V_{des,y}]^T$), autonomous obstacle avoidance is obtained by the use of potential fields. The algorithm used in this case is explained below.

In fact, the use of potential fields has proved to be a powerful technique for controlling robot motion (Khatib 1986). Some researchers have used potential fields to address the problem of real time action selection both for navigation and manipulation purposes on the AIBO (Johansson and Saffiotti 2001). Others (Prestes et al. 2001) calculate, from an occupancy grid, the potential fields needed for autonomous exploration of an environment. We decided to use the latter approach by generating the occupancy grid as the robot moves, and then using it to compute the robot velocities. Our algorithm uses a composition of vortex and repulsive fields to build the velocity field. In particular, for each occupied cell on the occupancy grid, centered at $c = [x_c\ y_c]^T$, with x_c and y_c cell coordinates in the robot frame (see Fig. 4a), define the repulsive potential as:

$$U_r(\|c\|,\eta) = \begin{cases} K_r\left(\dfrac{1}{\|c\|} - \dfrac{1}{\eta}\right)^2 & \text{if } \|c\| \le \eta \\ 0 & \text{else} \end{cases} \tag{22}$$

where $\|c\|$ is the distance of the occupied cell from the robot center, η the radius of influence of the potential, and K_r a given gain. The repulsive field induced by each cell is simply obtained as the gradient of this potential, i.e.,

$$f_c^r = \begin{pmatrix} f_{c,x}^r \\ f_{c,y}^r \end{pmatrix} = \begin{pmatrix} \dfrac{\partial U_r(\|c\|,\eta_r)}{\partial x_c} \\ \dfrac{\partial U_r(\|c\|,\eta_r)}{\partial y_c} \end{pmatrix} \tag{23}$$

while the vortex field (De Luca and Oriolo 1994) is defined as:

$$f_c^v = \begin{pmatrix} f_{c,x}^v \\ f_{c,y}^v \end{pmatrix} = \begin{pmatrix} \pm\dfrac{\partial U_r(\|c\|,\eta_v)}{\partial y_c} \\ \mp\dfrac{\partial U_r(\|c\|,\eta_v)}{\partial x_c} \end{pmatrix} \tag{24}$$

Note the different radii of influence η_r and η_v of repulsive and vortex fields, respectively. By choosing $\eta_v > \eta_r$, we obtain velocity fields that are essentially vortices at large distances, and become increasingly repulsive at close range. The signs of $f_{c,x}^v$ and $f_{c,y}^v$ depend on the position of the

[1] As for the single step mode, no obstacle avoidance can be performed when executing backward or rotational motions.

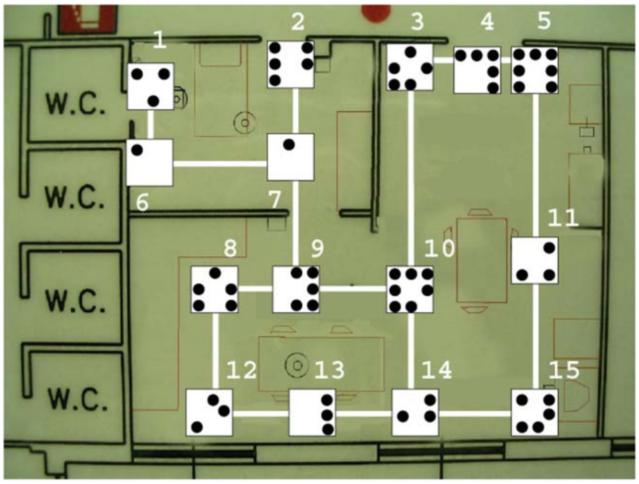


Fig. 7 The roadmap used in autonomous navigation mode. The ID labels of each coded square are indicated. Note that crossings appear larger than they are

occupied cell with respect to the robot sagittal plane: a cell in the right (left) half of the grid will induce a CW (CCW) vortex.

The fields generated by all the occupied grid cells are then superimposed with the desired workspace velocity in order to obtain the total velocity field:

$$V = \sum_c f_c^r + \sum_c f_c^v + V_{des} \tag{25}$$

This velocity must be mapped to the configuration space velocities either with the omnidirectional translational motion conversion or by enforcing nonholonomic-like motion (10). The first is consistent with the objective of maintaining as much as possible the robot orientation specified by the user. Instead, with the second kind of conversion, the orientation of the robot is always tangent to the path; the grid provides more effective collision avoidance since the direction of its angle bisector coincides with the x-axis (because v_y is null). After having implemented and tested both modes in a preliminary phase, the omnidirectional mode was chosen for semi-autonomous navigation.

5.3 Autonomous mode

For the Autonomous navigation mode, we designed a physical roadmap (shown in Fig. 7) to reach and connect all the relevant destinations in the experimental arena, and utilized a more sophisticated visual servoing scheme.

The roadmap is formed by streets and crossings, all realized in white adhesive tape and laid on the ground. The perception primitives described in Sect. 4.1 are used to identify the streets (i.e. straight white lines) and crossings (i.e. coded squares) while the motion primitives described in Sect. 4.2 are used to drive the robot on the map. When approaching a landmark or following a street, the robot concurrently implements landmark fixing, in order to keep the landmark centered in the image plane.

The robot autonomous behavior is represented by a Petri Nets (Murata 1989) based framework which has been successfully deployed on the AIBO Platform in the Robocup field (Ziparo and Iocchi 2006). The Petri Net Plan formalism allows for high level description of complex action interactions that are necessary in programming a cognitive robot: non-instantaneous actions, sensing and conditional actions, action failures, concurrent actions, interrupts, action synchronization.

The autonomous navigation plan uses the following actions (note that at all times during the actions, the perception primitives are also executed for searching and updating perceived data):

- *Seek streets*: The robot seeks streets by exploring the environment, while avoiding collisions. Motion directions are predefined: AIBO alternates forward and rotation steps.
- *Approach the nearest street*: When it perceives some streets with the straight white line extractor, and tracks them with the visual landmark tracker, the robot uses the landmark approacher to walk (using omnidirectional motion) towards the nearest.
- *Follow the street*: When the robot is sufficiently close to the street, it implements the linear straight line follower for walking on the street (using nonholonomic-like motion), until at least one crossing is detected.
- *Plan the path to destination*: When a crossing is detected with the coded square extractor, and tracked with the visual landmark tracker, the robot has univocally identified its *ID* and orientation. This information, along with the coded square positions in the robot frame, and with the map, identifies the robot pose (position and orientation). The robot then utilizes a Dijkstra-based graph search (Dijkstra 1959) to find the shortest path to the destination. Depending on the result of the graph search, the robot will approach and follow another street (repeat the corresponding actions in the plan), or stop if the crossing corresponds to the desired destination.

The autonomous navigation plan repeats the above actions until the destination is reached. Transitions that start or terminate the actions represent events (e.g. *Street seen*, or *Crossing near*) which are triggered by conditions on sensed information (e.g. distance from a line). The plan must also deal with action failures. For instance, whenever the robot loses visual contact with a street it was approaching, the system aborts the current action and moves to the state where the street is not seen, and so on, until the robot reaches the street again.

6 Other aspects of the ASPICE robot driver

In this section, we will describe the implementation of four features of the ASPICE robot driver which have not been detailed in Cherubini et al. (2007).

First, a fundamental requirement for effective remote controlled navigation is the feedback of the robot camera to the Control Unit (i.e., to the user). This is not only an essential aid for the user to drive the robot in the environment, but is also helpful for exploration and extension of virtual mobility. By driving the robot in a desired spot of the apartment, the disabled person can monitor, with AIBO's on board camera, that area. Thus, an algorithm for feeding back to the ASPICE GUI the image captured by the robot has been developed. Each new image captured by the robot camera is compressed on board using the standard techniques provided by the Independent JPEG Group libraries,[2] before being streamed over a wireless connection to the ASPICE Control Unit. However, we had to develop ourselves a simple algorithm for luminosity adaptation, since the quality of the AIBO camera could not cope with luminosity changes. In practice, the image luminosity is adaptively incremented/decremented before compression, depending on the image average luminosity. This simple approach reduces the variations in the average image luminosity during video stream.

To improve ambient exploration with the camera, a GUI for head control has also been developed. This GUI allows the user to directly move the robot head and point the camera in a desired direction. With this GUI, the user can control the head pan and tilt angles (φ and ψ_2) with fixed steps, in order to point the camera in a desired direction. The head control GUI is shown in Fig. 8 (top).

Another feature of the system is a GUI for vocal requests, which has been included to improve the communication of the patient with the caregiver. This feature covers an important issue in ASPICE, enabling the robot to be *socially interactive*. When the robot receives a vocal request (e.g., 'I am thirsty') from the control unit, it plays the corresponding prerecorded audio file with its speakers in order to attract the caregiver attention. The vocal request GUI is shown in Fig. 8 (bottom). In the GUI, each button that the user can select corresponds to vocal request (here, 'turn on/off the light', 'Please come', 'I am thirsty', 'Close the window' and 'I am hungry').

Finally, another issue that deserved attention is AIBO's walking gait, which has been modified on the basis of the ASPICE requirements (e.g., reducing the noise caused by foot contact with the ground). The basic principle of the algorithm is to move the robot feet alternatively (in a "waddle" style) in rectangles. The directions and sizes of such rectangles are determined by the motion command (v_x, v_y, v_θ). The joint angles required to drive the feet on the rectangular trajectories are calculated by solving the inverse kinematics problem for each leg. Nevertheless, many walking styles can be implemented, and a set of parameters (e.g., the position of the rectangles with respect to AIBO's body, or their height) defines each possible walking style. Hence, a set of such parameters was chosen and tested to fulfill ASPICE requirements. Clearly, the limitations in the robot hardware forbid usage in environments with steps (e.g. stairs) and steep slopes.

[2]www.ijg.org

7 Simulations

The possibility of testing the robot navigation system in a simulated environment has been crucial from the very early stages of the ASPICE project. To this end, we have adopted Webots, a mobile robot simulation environment developed by Cyberbotics, which is used by many universities worldwide for modeling and controlling mobile robots. In Webots, a complete mobile robotics setup can be defined, by creating complex environments and equipping a robot with sensors and actuators. For each object, one can define all physical properties, and the simulated robot can be programmed with the same development environment as the real one.

The simulation environment was built according to the real environment where the ASPICE experiments take place, a small apartment located at Fondazione Santa Lucia in Rome, intended for rehabilitation purposes, and the various navigation modes of Sect. 2 have been implemented in Webots. A typical simulation of the ASPICE semi-autonomous mode is shown in Fig. 9. Here, the user tried to drive AIBO to a given destination in the apartment (indicated by a red arrow in the figure), through a single direction command (forward) along the line joining the initial and final configuration. The built-in obstacle avoidance algorithm was able to successfully accomplish the task in spite of the many obstacles along the way. In the simulation shown in Fig. 9, the nonholonomic conversion entailed by (10) was being tested, for a preliminary assessment.

8 Experiments

In this section, we show the results of various experiments that were performed with the robot driver. The experiments cover all the characteristics of the driver mentioned in the paper. Moreover, we discuss the ASPICE clinical validation and its results on the quality of life of potential users. The two designed motion modes (omnidirectional and nonholonomic-like) are associated to the robot

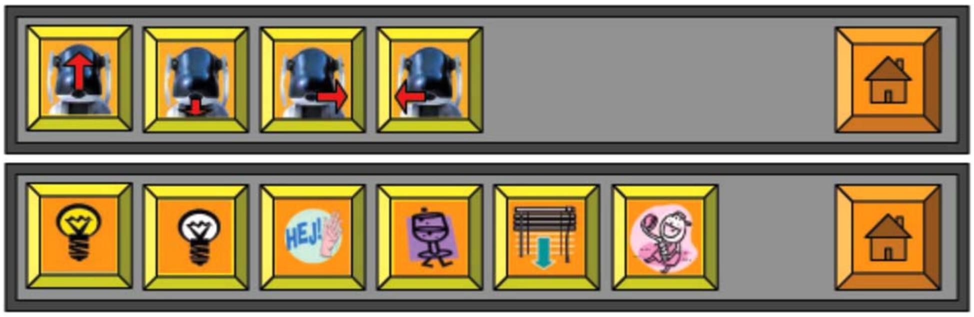


Fig. 8 The ASPICE GUIs for head control (*top*), and for vocal requests (*bottom*)

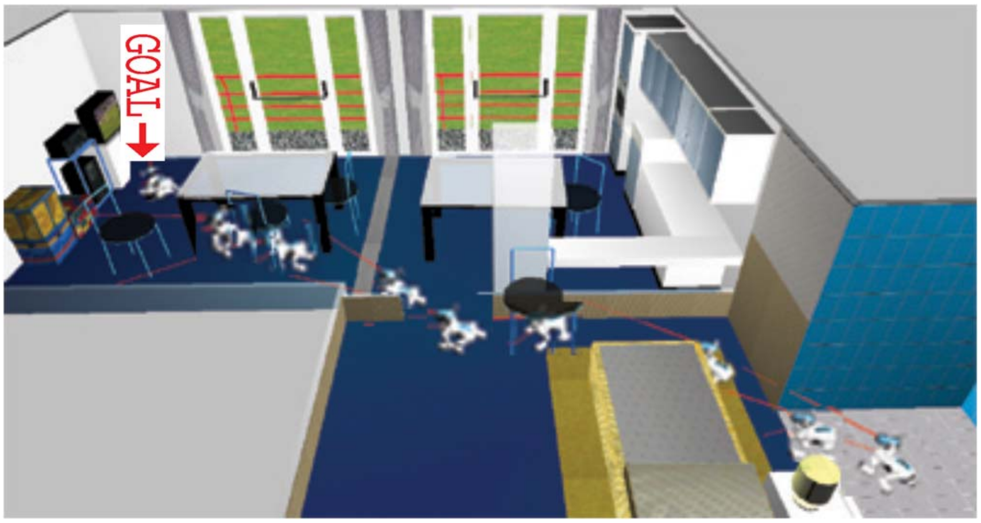


Fig. 9 A simulation result for semi-autonomous navigation

behavior as explained previously: omnidirectional motion is used for single step navigation, semi-autonomous navigation and landmark approach, while nonholonomic-like motion is used when seeking and following streets. Short video clips of the experiments can be viewed on the web site: http://www.dis.uniroma1.it/~labrob/research/ASPICE.html).

In a first series of experiments (shown in Fig. 10), the performance of the obstacle avoidance algorithms is tested by using the semi-autonomous navigation mode. Omnidirectional translational motion is used for mapping desired user velocities to the configuration space. In the first (top left in the figure) and third (bottom) experiments, a single 'go forward' command is used, and the robot is able to successfully avoid the obstacles while maintaining the orientation desired by the user. In the second experiment (top right in the figure), a single 'go right' command is used. Again, the robot avoids the obstacle while maintaining the desired orientation. The robustness of the obstacle avoidance algorithm is evident in the third experiment, where the obstacle scenario is more complex than in the two other experiments. In general, due to the vibrations in the quadruped gait cycle, and to the low-quality of AIBO's distance sensors, small and fast moving obstacles might not be detected by the robot. However, using the robot camera video feedback presented in Sect. 6, the user can detect such obstacles and intervene to prevent collisions. Our approach has given excellent results, and the robot has never collided during the experiments.

In another experiment, the localization performance in the map-based navigation mode is addressed. In the autonomous mode, robot localization, broadly speaking, consists of the four actions (see Sect. 5.3): *seek streets*, *approach the nearest street* (after having detected a street), *follow the street*, and *plan the path to destination* (after having detected a crossing). Line detection during the *seek streets* action is the major bottleneck in this localization scheme, because, once the robot has detected a street, the high accuracy in the three other actions ($> 95\%$) guarantees convergence to the desired destination. Hence, the autonomous mode localization performance is evaluated, by estimating the maximum distance from the roadmap at which the robot is able to detect the straight lines. In our opinion, this is the best way of assessing the localization in the map-based navigation mode: in fact, as soon as at least one line is detected, the robot is able to reach the roadmap and find its way to the requested destination. For the experiment, the robot is placed at a distance of 3 m from a straight white line, in various positions and orientations (some configurations are shown in Fig. 11). The maximum distance at which the robot detects the line is measured and averaged over all the experiments. After 20 experiments, the average

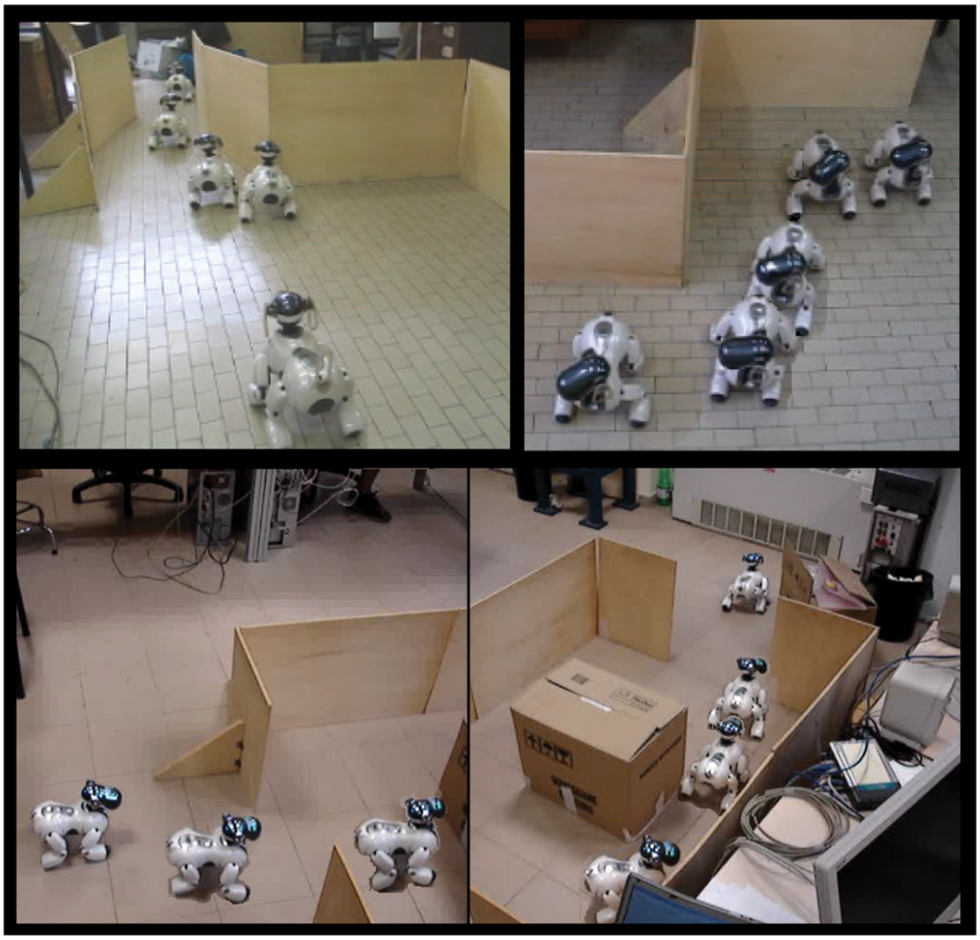


Fig. 10 Three experiments showing the performance of semi-autonomous navigation. The user drives the robot using a single command: 'go forward' (*top left* and *bottom*) and 'go right' (*top right*)

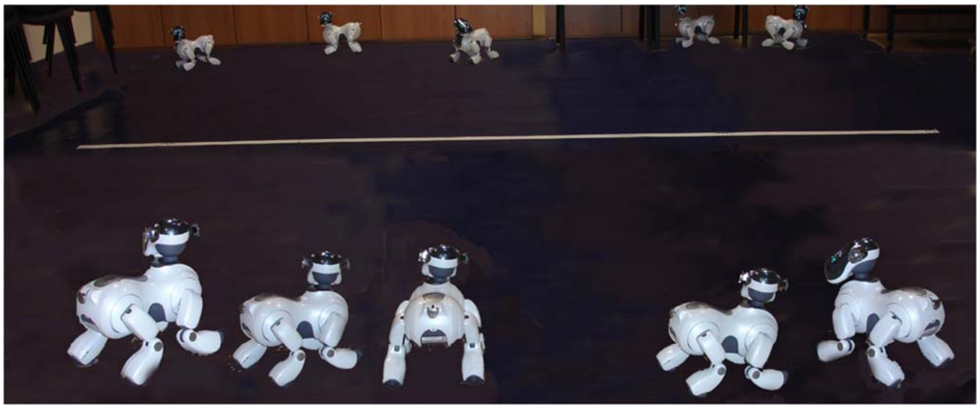


Fig. 11 The maximum distance at which the robot is able to detect the roadmap lines is measured by positioning the robot in various configurations around a *straight white line*

value is 2.14 m, with standard deviation 0.15 m. Clearly, these values are strongly related to the environment light conditions and possible visual occlusions. In our opinion a more 'quantitative' assessment of the localization performance is impossible, since too many factors should be taken into account: environment structure, light conditions, map topology, etc. In the roadmap used for the ASPICE experiments (see Fig. 7), since the maximum distance between lines is compatible with such results, the robot is capable of reaching the roadmap from every position in the environment. Moreover, even when the initial position is far from the roadmap, the *seek streets* action will lead the robot to a position where it is able to detect at least one line.

In a third series of experiments, the performance of the navigation system is evaluated by driving the robot from a start point to a goal point (noted respectively "S" and "G" in Figs. 12 and 13). The task is repeated 5 times for each of the three navigation modes (single step, semi-autonomous,

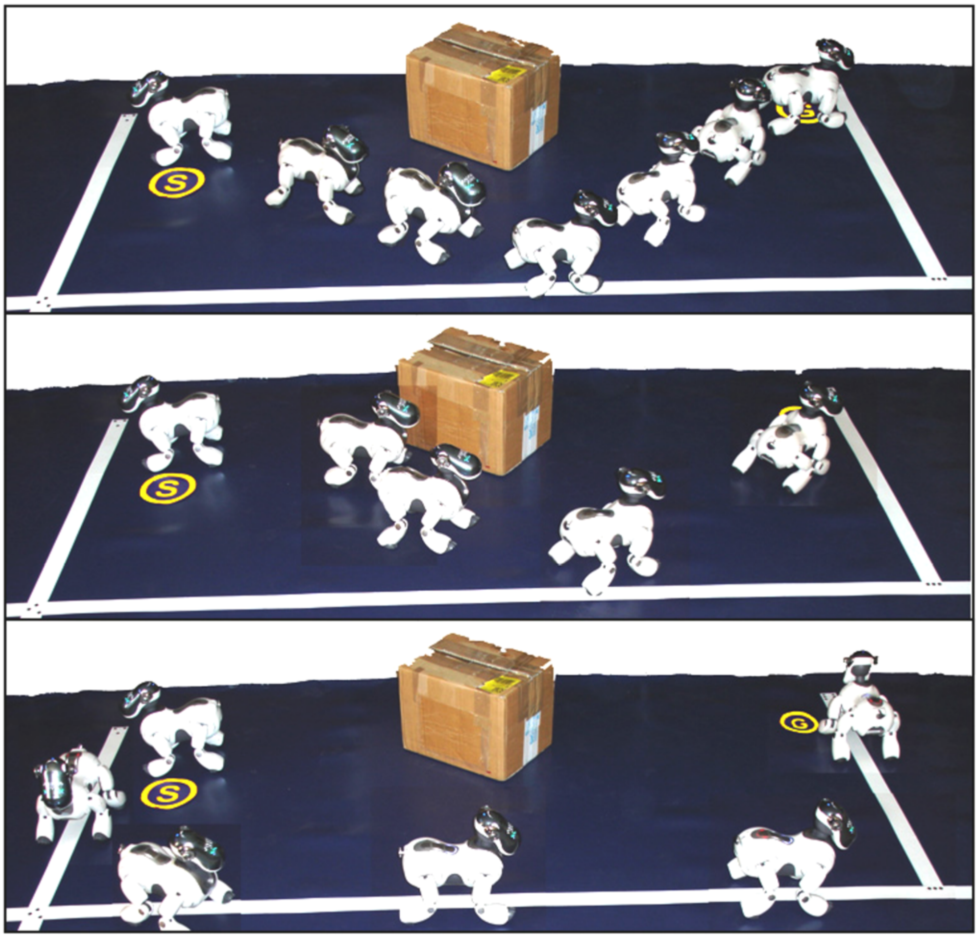


Fig. 12 Comparison between the three ASPICE navigation modes: with the mouse as input device, the user drives the robot from point S to point G using the single step (*above*), semi-autonomous (*center*) and autonomous (*below*) modes

and autonomous) by using first a mouse (Fig. 12), and afterwards a BCI (Fig. 13) as input devices. In semi-autonomous navigation, omnidirectional translational motion is used for mapping desired user velocities to the configuration space. In the BCI experiments, time domain features (see Sect. 2.2) are used to select the desired command. We decided to use the P300 as a BCI control signal because of its stability and because it does not require subject training. Based on these characteristics, we considered the P300-based BCI control as the more appropriated to be compared with the mouse control. The comparison between the three modes and between the two input devices is based on execution time and user intervention (i.e. number of times the user had to intervene by updating the commands) averaged over the experiments. As expected, the results (plotted in Fig. 14) confirm the qualitative properties expected for each mode. Note that the best choice depends not only on the user preference and ability but also on the specific task (e.g., position of the start and goal points in the environment, presence and position of obstacles). As expected, the user intervention diminishes as the robot autonomy increases (both using BCI and mouse). On the other side, note that in the specific configuration used in the experiment, the execution time does not always decrease as autonomy increases. For example, when the robot is mouse-driven, the task is executed in 1:23 minutes using the semi-autonomous mode, and in 1:30 using the autonomous mode. The differences may depend, among other factors, on the distances of the start and goal points from the roadmap. The use of the BCI introduces a time delay on each user intervention. This delay is due to the time required for selection of each command. In fact, to reduce the effect of the EEG signal noise, the command selection is validated after the user has ‘confirmed’ it various times (in our experiment, 6 times). Moreover, BCI control introduces errors during the navigation, due to EEG noise and large user inaccuracy, as compared to mouse control. Hence, the user intervention is greater with the BCI, since the user must apply extra commands to correct the errors. Consider, for example, the semi-autonomous experiment: with the mouse, four clicks were sufficient, whereas with the BCI, the user had to intervene 11 times. This specific result is related to the characteristics of the semi-autonomous mode, which requires precise timing when switching between walking directions. This requirement is hardly compatible with the BCI. For both the aforementioned reasons (time delay on each command and errors), the total execution time is par-

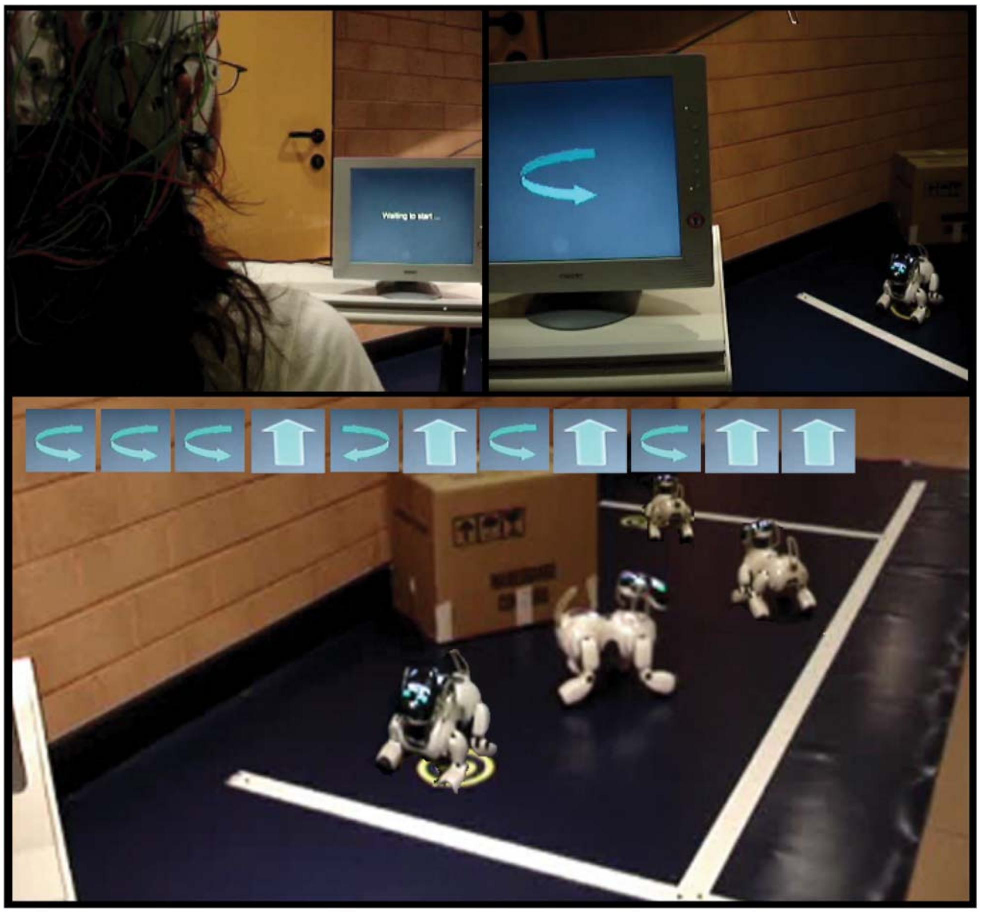


Fig. 13 BCI-driven single step navigation experiment. A feedback stimulus is provided on the screen. The commands used to drive the robot from point S to point G are displayed on the *bottom*

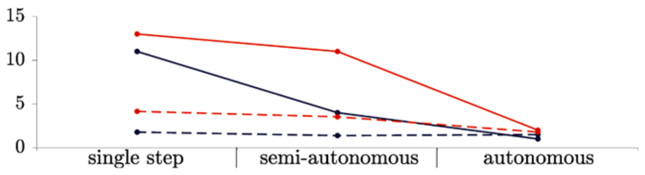


Fig. 14 (Color online) Comparison between the 3 navigation modes using 2 input devices: mouse (*blue*), BCI (*red*). Execution time (in minutes): *dashed*, user intervention (number of commands): *solid*

ticularly worsened when a high intervention is required. For instance, in single step mode, the experiment requires 2:22 more with the BCI than with the mouse, whereas in autonomous mode the difference is only 17 seconds. At the state of the art, there is no advantage in using a BCI when any other input device can be controlled by the user. Nevertheless, for all users that are unable to use standard devices, the BCI provides a useful solution, especially when it is associated with autonomous navigation. Hybrid approaches in which brain signals are employed at the same time as motor-based inputs are currently under investigation.

The usefulness of the navigation system to increase patient independence in everyday life is shown in an experiment where a domestic task is achieved by taking advantage of all the characteristics of the ASPICE robot driver. The task is shown in Fig. 15: the robot is used to verify whether the bottle of water is on the green table in a domestic environment. The visual feedback from the robot camera supports the user throughout navigation. The robot is initially located far from the roadmap with an obstacle on the way. The user perceives this information via the robot camera, and decides to control AIBO with the semi-autonomous mode to approach the roadmap (Fig. 15a). When the user perceives that the robot is sufficiently near the roadmap, the autonomous mode is used to drive the robot to the target nearest to the green table (Fig. 15b–d). Then, in order to obtain precise motion of the robot camera and check if the bottle is on the table, the user controls the robot with the single step mode, and finally displaces the robot camera with the head control menu (Fig. 15e).

Finally, in another experiment, autonomous robot battery charging, is implemented. This behavior is also present in the commercial Sony Driver, but since Sony does not release the code of its driver, we had to realize it *ad hoc* for this project. This experiment not only fulfills an important ASPICE requirement, but also provides an additional testbed for the autonomous navigation mode. In fact, the AIBO Charging Station is placed near a marked crossing on the

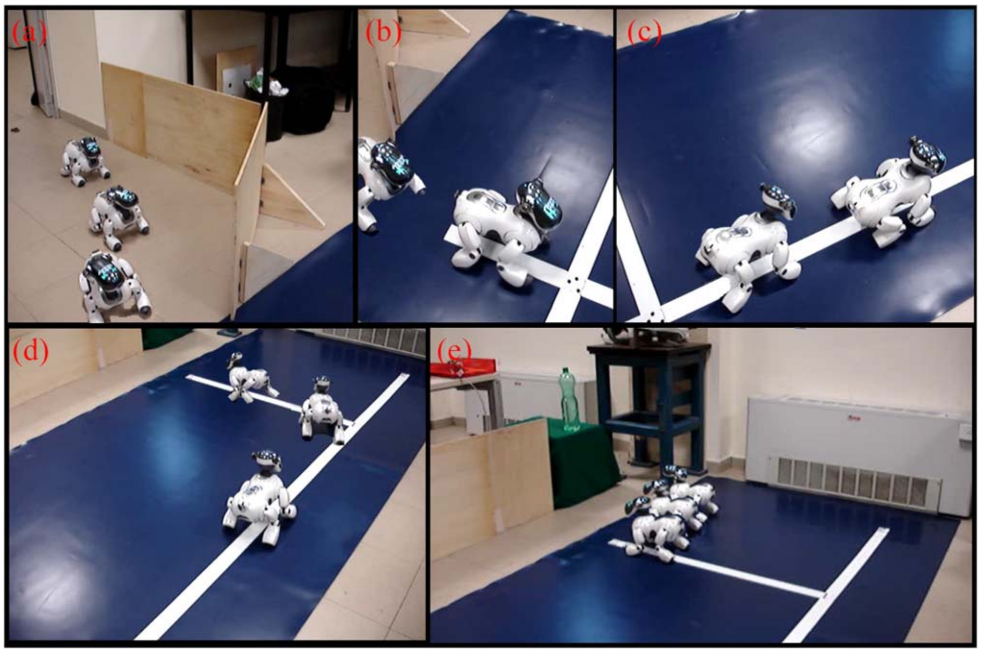


Fig. 15 Executing a domestic task (check if the bottle is on the green table) by taking advantage of all the characteristics of the ASPICE robot driver

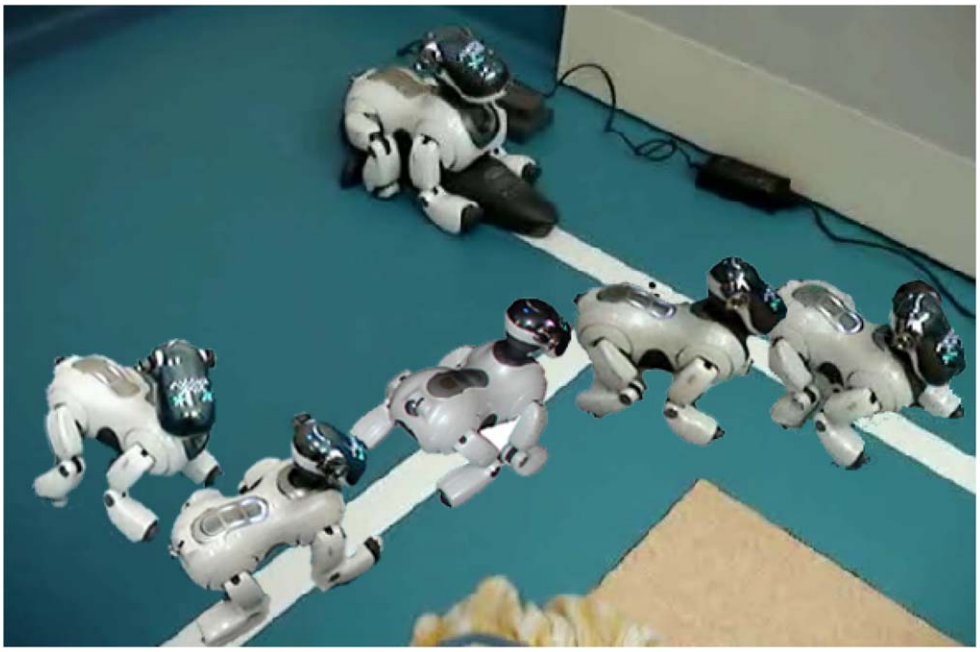


Fig. 16 Battery charging experiment

roadmap shown in Fig. 7, and as soon as the battery level is low, the robot autonomously moves to the station. The experiment is illustrated in Fig. 16. The robot position at consecutive time frames is shown while it approaches the roadmap, follows the street up to the battery charger crossing, detects it, and makes a turn in order to reach its destination (the charging station) on the basis of the plan.

At this stage of the project, the system has been implemented and is available at the Fondazione Santa Lucia in Rome for validation with patients (Cincotti et al. 2008). All domotic appliances used in ASPICE have been installed in the experimental apartment of the hospital. A portable computer runs the Control Unit program, and several input devices are available to cope with as many categories of users as possible. The subjects have been admitted for a neurorehabilitation program, and the whole procedure underwent the approval of the local ethical committee. The patients have been informed on the device, and have given their voluntary informed written consent. Furthermore, they have been interviewed and examined physically by the clinicians, for evaluating some variables of interest: the degree of motor impairment and of reliance on the caregivers for everyday activities as assessed by current standardized scale (Barthel Index *BI*), the familiarity with the system input devices, the ability to speak or communicate, and the level of informatics alphabetization measured by the number of hours per week spent in front of a computer. Then, for a period ranging from 3 to 4 weeks, the patient and (when required) her/his caregivers have been practicing weekly with the ASPICE system. During the whole period, patients had the assistance of an engineer and a therapist in their interaction with the system. The experiments have been carried out with eight subjects suffering from Spinal Muscular Atrophy type II (SMA II) and six subjects suffering from Duchenne Muscular Dystrophy (DMD). Depending on their level of disability, the users controlled AIBO with various ASPICE input devices.

According to the score of the *BI*, all patients were on almost complete dependence of caregivers, especially the six subjects suffering from DMD ($BI < 35$), who required artificial ventilation, had minimal residual mobility of the upper limbs and very slow speech. Because of the high level of muscular impairment, the DMD patients all had access to the system via joypads. As for the eight SMA II subjects, their level of dependency was slightly lower with respect the DMD patients, but they also required continuous assistance for daily life activity ($BI < 50$). These patients have accessed the system via joystick, touchpad, head tracker, keyboard and microphone, since the residual motor abilities were still functionally effective both in terms of muscular strength and range of movements preserved. Four of them interacted with the system via BCI. Both the frequency domain and the time domain versions of the BCI have been used, according to the user predisposition. All of the patients were able to master the system and control AIBO within 5 sessions. Details on ASPICE input devices other than the mouse and BCI, are provided in Cincotti et al. (2008). Although the successful integration of these devices emphasizes the robot driver flexibility, space limitation does not allow further discussion in this paper.

Data on user satisfaction, increase of independence, and reduction of caregiver workload, have been collected, structured in questionnaires, during the experimentation. For instance, the users were asked to indicate with a number ranging from 0 (not satisfied) to 5 (very satisfied) their degree of acceptance relative to each of the output devices. The average grade given by the patients to their 'personal satisfaction in utilizing the robot' was 3.04 on a 5-point scale. This is a very promising result, considering that the users were originally more accustomed to using and controlling the 'traditional' ASPICE appliances rather than the mobile robot.

According to the results of the questionnaire, all patients were independent in the use of the system at the end of the training and they experienced (as they reported) "the possibility to interact with the environment by myself". From the clinical viewpoint, the robot navigation system was useful in blending the "desiderata" of the patients (e.g., being in the living room with the relatives, reaching someone far from the patient's location) with their level of disability (which prevented them from acting as first-person in realizing such "desiderata"). For example, patients with a partial preservation of distal arm muscular ability (i.e., most SMA II patients) could have a complete control of the robot via single step navigation. On the other hand, severely impaired patients (i.e. DMD patients), who were unable to send frequent commands, could operate the robot in semi-autonomous and autonomous navigation modes. Further studies are necessary to confront a larger population of patients with ASPICE, in order to assess the robot navigation system impact on the quality of life by taking into account a range of outcomes (e.g. mood, motivation, caregiver burden, employability, satisfaction, Kohler et al. 2005; Bach et al. 2003; Natterlund et al. 2000). Nevertheless, the results obtained from this pilot study are encouraging for the establishment of a solid link between the field of human machine interaction and neurorehabilitation strategy (Hammel 2003). The ASPICE system cannot substitute the assistance provided by a human. Nevertheless, it can contribute to relieve the caregiver from a continuous presence in the room of the patient, since the latter can perform some simple activities on its own. The perception of the patients is that they no longer have to rely on the caregiver for each and every action. This provides the patient with both a sense of independence, and a sense of privacy. For both reasons, his quality of life is sensibly improved.

9 Conclusions and future work

In this work, we presented the development of a multimode navigation system for AIBO based on few simple primitives. In particular, three navigation modes have been devised to provide the user with different levels of interaction and control of the mobile robot. These operating modes have been integrated in the ASPICE system, and have been tested by patients in a neurorehabilitation program. All aspects of the system are covered: design, implementation, simulation, experiments and clinical assessment. The article contains major improvements and refinements over our previous works. In this paper, details on the robot driver primitives, on the BCI, and on other robot features implemented in ASPICE have been added, along with extended experiments showing the main system characteristics.

The main contribution of this work is the presentation of a system that provides remote control of a mobile robot via various input devices (including a Brain-Computer Interface), in an assistive rehabilitation project. In particular, the robot driver has been designed in order to be effectively operated by low-speed input devices, such as the ones developed in ASPICE (e.g., the BCI). Hence, in this work we did not attempt to provide a new control interface for robot navigation, but rather focused on the design and implementation of an appropriate robot navigation system, that can be matched to various forms of detection of the user's will. The study presented in this manuscript successfully showed feasibility of the proposed approach.

Acknowledgement This work is partially supported by the Italian Telethon Foundation Grant GUP03562.

References

Bach, J. R., Vega, J., Majors, J., & Friedman, A. (2003). Spinal muscular atrophy type 1 quality of life. *American Journal of Physical Medicine and Rehabilitation*, *82*(2), 137–142.

Belic, A., Koritnic, B., Logar, V., Brezan, S., Rutar, V., Kurillo, G., Karba, R., & Zidar, J. (2005). Identification of human gripping-force control from electro-encephalographic signals by artificial neural networks. In *16th international federation of automatic control world congress*.

Canudas de Wit, C., Siciliano, B., & Bastin, G. (Eds.). (1996). *Theory of robot control*. *Communication and control engineering*. London: Springer.

Carreras, M., Ridao, P., Garcia, R., & Nicosevici, T. (2003). Vision-based localization of an underwater robot in a structured environment. In *IEEE international conference on robotics and automation*.

Caselli, S., Fantini, E., Monica, F., Occhi, P., & Reggiani, M. (2003). Toward a mobile manipulator service robot for human assistance. In *2003 Robocare workshop*.

Cherubini, A., Oriolo, G., Macrí, F., Aloise, F., Cincotti, F., & Mattia, D. (2007). Development of a multimode navigation system for an assistive robotics project. In *IEEE international conference on robotics and automation*.

Cincotti, F., Mattia, D., Aloise, F., Bufalari, S., Schalk, G., Oriolo, G., Cherubini, A., Marciani, M. G., & Babiloni, F. (2007). Non-invasive Brain-Computer Interface system: towards its application as assistive technology. *Brain Research Bulletin*, *75*(6), 796–803. Special Issue: Robotics and Neuroscience.

Corke, P. I. (1996). *Visual control of robots: high performance visual servoing*. RSP Press.

De Luca, A., & Oriolo, G. (1994). Local incremental planning for nonholonomic mobile robots. In *1994 IEEE int. conf. on robotics and automation* (pp. 104–110).

Del Millán, J. R., Renkens, F., Mouriño, J., & Gerstner, W. (2004). Noninvasive brain-actuated control of a mobile robot by human EEG. *IEEE Transactions on Biomedical Engineering*, *51*, 1026–1033.

Dijkstra, E. W. (1959). A note on two problems in connection with graphs. *Numerische Mathematik*, *1*, 269–271.

Do, J. H., Jang, H., Jung, S. H., Jung, J., & Bien, Z. (2005). Soft remote control system in the intelligent sweet home. In *2005 IEEE/RSJ international conference on intelligent robots and systems* (pp. 2193–2198).

Elfes, A. (1989). Using occupancy grids for mobile robot perception and navigation. *Computer*, *22*(6), 46–57.

Farwell, L. A., & Donchin, E. (1988). Talking off the top of your head: Towards a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and Clinical Neurophysiology*, *70*(6), 510–523.

Fasola, J., Rybski, P., & Veloso, M. (2005). Fast goal navigation with obstacle avoidance using a dynamic local visual model. In *SBAI'05, The VII Brazilian symposium of artificial intelligence*, São Luiz, Brazil.

Feil-Seifer, D., & Matarić, M. J. (2005). Defining socially assistive robotics. In *Proceedings of the 2005 IEEE 9th international conference on rehabilitation robotics*.

Fioretti, S., Leo, T., & Longhi, S. (2000). A navigation system for increasing the autonomy and the security of powered wheelhairs. *IEEE Transactions on Rehabilitation Engineering*, *8*(4), 490–498.

Fong, T., Nourbakhsh, I., & Dautenhahn, K. (2003). A survey of socially interactive robots. In *Robotics and autonomous systems*.

Hammel, J. (2003). Technology and the environment: supportive resource or barrier for people with developmental disabilities? *Nursing Clinics of North America*, *38*(2), 331–349.

Harmo, P., Taipalus, T., Knuuttila, J., Vallet, J., & Halme, A. (2005). Needs and solutions—Home automation and service robots for the elderly and disabled. In *2005 IEEE/RSJ international conference on intelligent robots and systems* (pp. 2721–2726).

Hartenberg, R. S., & Denavit, J. (1955). A kinematic notation for lower pair mechanisms based on matrices. *Journal of Applied Mechanics*, *77*(2), 215–221.

Hengts, B., Ibbotson, D., Pham, S. B., & Sammut, C. (2001). Omnidirectional motion for quadruped robots. In *Lecture notes in artificial intelligence: Vol. 2377. Robocup international symposium*.

Hochberg, L. R., Serruya, M. D., Friehs, G. M., Mukand, J. A., Saleh, M., Caplan, A. H., Branner, A., Chen, D., Penn, R. D., & Donoghue, J. P. (2006). Neuronal ensemble control of prosthetic devices by a human with tetraplegia. *Nature*, *442*, 164–171.

Hoffmann, J., Jungel, M., & Lötzsch, M. (2004). A vision based system for goal-directed obstacle avoidance. In *Proceedings of 8th international Robocup symposium*.

Howard, A., & Kitchen, L. (1996). Generating sonar maps in highly specular environments. In *4th international conference on control, automation, robotics and vision*.

Hugel, V., Costis, T., Strasse, O., Bonnin, P., & Blazevic, P. (2003). *Robocup 2003 Les trois mousquetaires* (Technical Report). Laboratoire de Robotique de Versailles, Vélizy, France.

Irie, R., & Shibata, T. (1997). Artificial emotional creature for human-robot interaction—A new direction for intelligent systems. In *1997 IEEE/ASME international conference on advanced intelligent mechatronics*.

Johansson, S. J., & Saffiotti, A. (2001). Using the electric field approach in the Robocup domain. In *Proceedings of 5th international Robocup 2001 symposium*.

Kanamori, M., Suzuki, M., Oshiro, H., & HAAT Members (2003). Pilot study on improvement of quality of life among elderly using a pet-type robot. In *IEEE international symposium on computational intelligence in robotics and automation*.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, *5*(1), 90–98.

Kitagawa, L., Kobayashi, T., Beppu, T., & Terashima, K. (2001). Semi-autonomous obstacle avoidance of omnidirectional wheelchair by joystick impedance control. In *2001 IEEE/RSJ international conference on intelligent robots and systems* (Vol. 4, pp. 2148–2153).

Kohler, M., Clarenbach, C. F., Boni, L., Brack, T., Russi, E. W., & Bloch, K. E. (2005). Quality of life, physical disability, and respiratory impairment in Duchenne muscular dystrophy. *American Journal of Respiratory and Critical Care Medicine*, *172*(8), 1032–1036.

Kulyukin, V., Gharpure, C., Nicholson, J., & Pavithran, S. (2004). RFID in robot-assisted indoor navigation for the visually impaired. In *2004 IEEE/RSJ international conference on intelligent robots and systems*.

Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, *77*(4), 541–580.

Natterlund, B., Gunnarsson, L. G., & Ahlstrom, G. (2000). Disability, coping and quality of life in individuals with muscular dystrophy: a prospective study over five years. *Disability and Rehabilitation*, *22*(17), 776–785.

Oriolo, G., Ulivi, G., & Vendittelli, M. (1997). Fuzzy maps: a new tool for mobile robot perception and planning. *Journal of Robotic Systems*, *14*(3), 179–197.

Pfurtscheller, G., & Neuper, C. (2001). Motor imagery and direct brain-computer communication. *Proceedings of the IEEE*, *89*, 1123–1134.

Prestes, E., Idiart, M. A. P., Engel, P. M., & Trevisan, M. (2001). Exploration technique using potential fields calculated from relaxation methods. In *2001 IEEE/RSJ international conference on intelligent robots and system*.

Rao, R. S., Conn, K., Jung, S. H., Katupitiya, J., Kientz, T., Kumar, V., Ostrowski, J., Patel, S., & Taylor, C. J. (2002). Human robot interaction: application to smart wheelchairs. In *2002 IEEE international conference on robotics and automation* (Vol. 4, pp. 3583–3588).

Rebsamen, B., Burdet, E., Guan, C., Zhang, H., Teo, C. L., Zeng, Q., Ang, M., & Laugier, C. (2006). A brain-controlled wheelchair based on P300 and path guidance. In *Proceedings of the 2006 IEEE/RAS-EMBS international conference on biomedical robotics and biomechatronics*.

Roberts, L. G. (1965). Machine perception of three-dimensional solids. In *Optical and electro-optical information processing*. Cambridge: M.I.T. Press.

Rofer, T. et al. (2005). *Robocup 2005 German team* (Technical Report). Center for Computing Technology—Universität Bremen.

Russ, J. C. (1999). *The Image Processing Handbook*. Boca Raton: CRC Press.

Samson, C., & Ait-Abderrahim, K. (Eds.). (1991). Feedback stabilization of a non-holonomic wheeled mobie robot. In *Proceedings of IEEE/RSJ international workshop on intelligent robots and systems* (pp. 1242–1247).

Schalk, G., McFarland, D., Hinterberger, T., Birbaumer, N., & Wolpaw, J. (2004). BCI2000: A general-purpose brain-computer interface (BCI) system. *IEEE Transactions on Biomedical Engineering*, *51*, 1034–1043.

Seki, H., Kobayashi, S., Kamiya, Y., Hikizu, M., & Nomura, H. (2000). Autonomous/semi-autonomous navigation system of a wheelchair by active ultrasonic beacons. In *2000 IEEE international conference on robotics and automation*.

Skaff, S., Kantor, G., Maiwand, D., & Rizzi, A. A. (Eds.). (2003). Inertial navigation and visual line following for a dynamical hexapod robot. In *2003 IEEE/RSJ international conference on intelligent robots and system* (Vol. 2, pp. 1808–1813).

Smith, S. M., & Brady, J. M. (1997). Susan: A new approach to low-level image-processing. *International Journal of Computer Vision*, *23*(1), 45–78.

Wada, K., Shibata, T., Saito, T., Sakamoto, K., & Tanie, K. (2005). Psychological and social effects of one year robot assisted activity on elderly people at a health service facility for the aged. In *2005 IEEE international conference on robotics and automation* (pp. 2796–2801).

Wesolkowski, S., Jernigan, M. E., & Dony, R. D. (2000). Comparison of color image edge detectors in multiple color spaces. In *International conference on image processing* (Vol. 2, pp. 796–799).

Wolpaw, J. R., & McFarland, D. J. (1994). Multichannel EEG-based brain-computer communication. *Electroencephalography and Clinical Neurophysiology*, *90*(6), 444–449.

Wolpaw, J. R., McFarland, D. J., Neat, G. W., & Forneris, C. A. (1991). An EEG-based brain-computer interface for cursor control. *Electroencephalography and Clinical Neurophysiology*, *78*, 252–259.

Wolpaw, J. R., Birbaumer, N., McFarland, D. J., Pfurtscheller, G., & Vaughan, T. M. (2002). Brain-computer Interfaces for communication and control. *Clinical Neurophysiology*, *113*, 767–791.

Yang, M. C. K., Jong-Sen, L., Cheng-Chang, L., & Chung-Lin, H. (1997). Hough transform modified by line connectivity and line thickness. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(8), 905–910.

Yanko, H. A. (1998). Wheelesley: A robotic wheelchair system: indoor navigation and user interface. In *Assistive technology and artificial intelligence*. *Lecture notes in computer science*. Berlin: Springer.

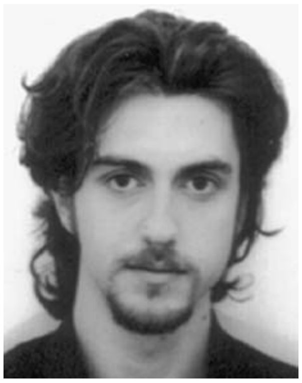
Ziparo, V. A., & Iocchi, L. (2006). Petri Net Plans. In *Fourth international workshop on modelling of objects, components, and agents (MOCA'06), Turku, Finland*.

Andrea Cherubini received the M.Sc. degree ("Laurea") in Mechanical Engineering in 2001 from the Università di Roma "La Sapienza", the M.Sc. degree in Control Systems in 2003 from the University of Sheffield, U.K., and the Ph.D. degree in Systems Engineering in 2008 from the Università di Roma "La Sapienza". During his Ph.D. programme (2004–2007), he was a visiting scientist at the Lagadic group at IRISA/INRIA in Rennes (France), where he is currently working as postdoctoral fellow. His research interests include: visual servoing for mobile robotic applications, assistive robotics, legged locomotion, nonholonomic robot navigation, and robot learning.

Giuseppe Oriolo received the Ph.D. degree in Control Systems Engineering in 1992 from the Università di Roma "La Sapienza", Italy. He is currently an Associate Professor of Automatic Control and Robotics at the same university, where he also heads the Robotics Laboratory. His research interests are in the area of planning and control of robotic systems, in which he has published more than 120 papers.

Francesco Macrí received the M.Sc. degree in Computer Science Engineering from the Università di Roma "La Sapienza", Italy. During his master, he has worked on computer vision issues within the Robocup Four-Legged Team for the same university. After receiving the master, he has been involved for one year in the ASPICE project with the "La Sapienza" Robotics Laboratory. He is currently working with Taitus Software Company to develop a satellite simulation tool in collaboration with the European Space Agency.

Fabio Aloise Fabio Aloise received the M.Sc. degree in Computer Science Engineering from the University "Unical" of Calabria, Italy. He is currently a Ph.D. student at the University of Rome "Tor Vergata". He is involved in different national and european projects with the Neuroelectrical Imaging and Brain Computer Interface Laboratory at Fondazione Santa Lucia IRCCS (Rome). His principal field of research is non-invasive brain computer Interface (BCI). Specifically, his research interests are in the design and implementation of aid systems for communicating with and controlling domotic environments via BCI. His focus extends to the development of multimodality feedback for use with these systems.



Febo Cincotti Febo Cincotti received his degree Laurea cum laude in Electronic Engineering in 1998, and the Ph.D. degree in Biophysics in 2003, from the Università di Roma "La Sapienza". He is a Research Scientist at the Department of Clinical Neurophysiology, "Neuroelectrical Imaging and BCI Lab.", at Fondazione Santa Lucia, IRCCS, Rome, Italy. His research interests include brain-computer interface system implementation and applications and development of algorithms for high resolution EEG processing.

Donatella Mattia Donatella Mattia received her degree as medical doctor from the Università di Roma "La Sapienza", Italy in 1987. In 1991, she completed her training as resident in Neurology and became neurologist at the same University. She also received the PhD degree in "Physiopathology of movement disorders" in 1996, from the Department of Neuroscience, Università di Roma "La Sapienza", Italy. During her PhD program (from 1992 to 1995), she was a research fellow at the Montreal Neurological Institute and Department of Neurology and Neurosurgery, McGill University, Montréal Québec, Canada. Currently she is an Assistant Researcher at the Department of Clinical Neurophysiology, "Neuroelectrical Imaging and BCI Lab.", at Fondazione Santa Lucia, IRCCS, Rome, Italy. Her research interests include cortical excitability and plasticity processes in different neurological disorders as evaluated by neuroelectric imaging techniques.