Journal of Control, Automation and Electrical Systems (2018) 29:470–479
https://doi.org/10.1007/s40313-018-0390-y

Humanoid Robot Framework for Research on Cognitive Robotics

Danilo H. Perico[1] · Thiago P. D. Homem[1,2] · Aislan C. Almeida[1] · Isaac J. Silva[1] · Claudio O. Vilão Jr.[1] · Vinicius N. Ferreira[1] · Reinaldo A. C. Bianchi[1]

Received: 29 November 2017 / Revised: 26 March 2018 / Accepted: 1 May 2018 / Published online: 31 May 2018

Abstract
This paper presents a humanoid robot framework, composed of a simulator and a telemetry interface. The framework is based on the Cross Architecture, and it is developed aiming for the RoboCup Soccer Humanoid League domain. A simulator is an important tool for testing cognitive algorithms without handling issues of real robots; furthermore, a simulator is extremely useful for allowing reproducibility of any developed algorithm, even if there is no robot available. The proposed simulator allows an easy transfer of the algorithms developed in the simulator to real robots, as long as it uses the Cross Architecture as its software architecture. Then, in order to evaluate the cognitive algorithms in real robots, a telemetry interface is proposed. From this interface, it is possible to monitor any variable in the robot's shared memory. The framework is open source and has low computational cost. Experiments were conducted in order to analyze both, simulator and telemetry interface. Experiments performed with the simulator aim to validate the high-level strategy development and the portability to a real robot, while experiments with telemetry interface aim to evaluate the robot behavior using, as input, the information received from the shared memory passed by all processes. The results show that the simulator can be used to test and develop new algorithms, while the telemetry can be used to monitor the robot, thus validating the framework for this domain.

Keywords Humanoid robot framework · Cognitive robotics · Robot simulator and telemetry

1 Introduction

The humanoid robot framework proposed on this work is based on the Cross Architecture (Perico et al. 2014a, b). The Cross Architecture is a software architecture composed by independent modules, that was proposed in order to allow a humanoid robot, equipped with a computer, to perform several concurrent tasks, including servomotor control, decision, localization, vision and so on.

Several modules of the Cross Architecture are dedicated to the development of cognitive algorithms, i.e., algorithms aimed at building intelligent robots, where the input–output behavior of the robot matches corresponding human behavior (Russell and Norvig 2010). Nevertheless, one of the main problems of developing cognitive algorithms in real humanoid robots is the physical issues, such as backlashes, broken parts, bad connections. Another problem of developing cognitive algorithms in unique humanoid robots, i.e., those ones that are not standard, is the lack of reproducibility of any proposed code.

Thereby, a simulator becomes an important tool for debugging and testing high-level algorithms without the need of dealing with real robot problems. Furthermore, a simulator allows the reproducibility of any proposed algorithm, even if there is no real robot available. The usage of simulators also allows the performance of a large number of experiments, without the risk of damaging robots.

There are already several kinds of robot simulators available, such as Gazebo (2016), Webots (2016), Virtual robot experimentation platform (2016) and RoboCup Soccer 3D (2017). However, those 3D simulators often have high computational cost and they need to have the specific desired robot model, which is not always available by default. In the existing 2D simulators, such as the RoboCup Soccer 2D (2017), some of the difficulties arise from the integration of the Cross Architecture. Beyond that, it is not an easy task to

✉ Danilo H. Perico
dperico@fei.edu.br

Thiago P. D. Homem
thiagohomem@ifsp.edu.br

[1] Centro Universitário FEI, São Bernardo do Campo, SP, Brazil

[2] Instituto Federal de São Paulo, São Paulo, SP, Brazil

transfer an algorithm developed in a simulator to a real robot. Usually, several adaptations and changes are necessary.

Moreover, after the code has been transferred from simulator to the real robot, it is tough to monitor inward robot variables during execution time, which is a desirable task in order to check and evaluate whether the robot is following the expected behavior or not.

The approached domain of this paper is the RoboCup Soccer, more precisely, the Humanoid League, where humanoid robots have the goal of playing soccer autonomously and cooperatively. Hence, the objective of this work is to propose a humanoid robot framework composed by a 2D humanoid robot soccer simulator, called RoboFEI-HT Simulator (Perico et al. 2016), and a telemetry interface for real robots. The simulator allows the abstraction of real-world problems during the development of high-level algorithms based on the Cross Architecture. Besides, these developed algorithms can be easily transferred from the simulator to a real humanoid robot, as long as this robot uses the Cross Architecture as its software architecture. Meanwhile, the proposed telemetry interface allows the monitoring of the transferred algorithms behavior in a real robot. By using the telemetry interface, one can check some important robot features, such as its localization, battery level and decision taken.

There are other software frameworks that are used for humanoid robots, such as the one developed by UT Austin Villa Team (Barrett et al. 2013), which, as well as the Cross Architecture, is composed of logical processes and shared memory to allow communication between processes. Austin Villa also developed a visualization tool, mainly focused in showing the robot localization. However, it is not clear if their architecture is hierarchical, reactive or hybrid, which is very clear in the case of Cross Architecture. Another software framework was developed by Team NimbRo-OP (Allgeuer et al. 2013), that used the Robot Operating System (ROS) (Quigley et al. 2009) as the base of their architecture and the RViz plugin to create a visualization tool. The Cross Architecture was not implemented in ROS, because ROS has high computational cost when combined with all processes required by the robot.

So, in addition to the existing RoboFEI-HT Simulator (Perico et al. 2016), this article brings new relevant experiments performed in order to prove the accuracy of the simulator in comparison with the real robot for the development of cognitive algorithms, as well as the telemetry interface which, together with the simulator, composes the proposed framework.

This work is structured as follows: Sect. 2 briefly describes the Cross Architecture. Section 3 gives an overview about telemetry. Section 4 shows the proposed framework, including the simulator and the telemetry interface. In Sect. 5, some case studies are presented and Sect. 6 provides the conclusions and indicates avenues for extend this work.

2 The Cross Architecture

The Cross Architecture is based on the hybrid architecture (Arkin 1998). Hybrid architecture argues that a robot has to deliberate while it senses and acts, and it is done by decomposing a task into sub-tasks. Tasks that are reactive, such as protecting a robot from a falling or adapting the walking to sloped terrains (Silva et al. 2015), do not need to be planned or deliberated. When one of those situations occurs, the robot should simply sense and react, leaving complex behaviors, like deciding what to do or localizing itself, to the hierarchical structure.

The Cross Architecture allows an abstraction layer between the processes required for a humanoid robot soccer player due to the blackboard concept (Hayes-Roth 1985), that is a shared memory area that allows inter-process communication. The architecture states that independent processes can communicate between each other in order to achieve an intelligent system. It is composed by eight processes: vision, localization, decision, communication, planning, inertial measurement unit (IMU), control and management (Perico et al. 2014a).

- *Vision* is responsible for data acquisition; it can be monocular, when only one camera is used, or stereo, when two cameras are adopted. Inside the Cross Architecture, vision process is used to recognize objects that are in the robot's field-of-view and send the relative distance and direction to the localization and decision processes, via blackboard. Vision process is the main sensor used by the robot to perceive the world in RoboCup Humanoid League.
- *Localization* is the process in charge of estimating the robot's pose. Localization is currently made by Monte Carlo localization method (MCL) (Fox et al. 1999; Dellaert et al. 1999), where particles represent the robot's confidence on its pose. MCL works as a recursive Bayes filter, where the robot's localization is updated considering its motion and its measurements.
- *Communication* is responsible for receiving and sending data to the Game Controller. Besides, it receives and sends information to other robots in the team. Communication broadcasts information to everything on the same wireless network, using predetermined ports. Each port is attached to a robot.
- *Decision* is responsible for defining which actions the robot must perform and send them to the planning process, which, in turn, send the actions to the Movement Control process. From the data received by the vision, localization and communication processes, decision can use several AI approaches to define which action the robot should perform. The simplest decision implemented is a

naive one, where the robot searches for the ball, walks toward it and kicks the ball.

- *Planning* receives, from decision process, the actions the robot must perform. As the robot soccer is a dynamic environment, the planning process is used to allow a robot to move itself from an initial position to a target position, avoiding collisions with robots and other obstacles, in order to perform the received action.
- *Inertial measurement unit* is the process that manage the IMU sensor. IMU is composed of accelerometer and gyroscope, where each sensor has three axis (x, y and z). IMU allows the estimation of the robots orientation by using an extended Kalman filter (EKF) (Russell and Norvig 2010).
- *Control* is the process responsible for executing the movements of the robot. Generally, humanoid robots use a gait pattern generation in order to perform a walking motion. Our real robots use the gait pattern developed by Ha et al. (2011) for the DARwIn-OP robot. The control process allow the real robot to perform movements such as: walk forward, turn to right and to left, get up when a fall happens and to stay stand-up. The control process is connected to the IMU, as found in a reactive paradigm (Brooks 1986). When the IMU detects a fall, for example, the robot get itself up without deliberate about this action.
- *Management* is the process that initializes, monitors and finishes all other processes. Additionally, in case any process stops working, management will reboot this process, working like a watchdog timer.

Figure 1 depicts all processes and how they are interconnected by the blackboard (*bkb*) to exchange data among them, considering a robot n. There are some researches with humanoid robots that already use the Cross Architecture as their basic architecture for the development of cognitive behaviors, for example Silva et al. (2015), Vilão et al. (2014), Perico et al. (2016) and Homem et al. (2017).

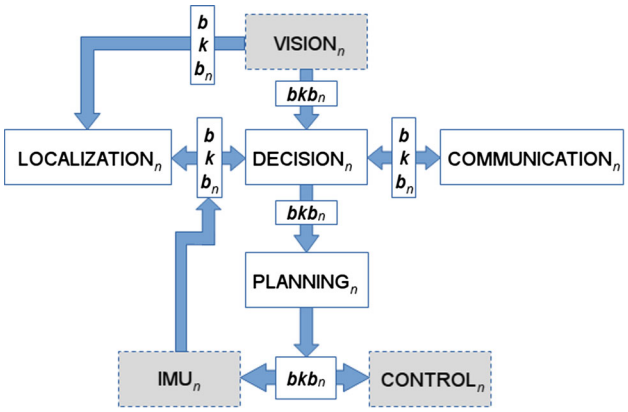


Fig. 1 The Cross Architecture. Adapted from Perico et al. (2014a). Gray boxes are the simulated processes

Fig. 2 RoboFEI-HT Simulator

3 Telemetry

Telemetry is an automatic way for communicating measurements of relevant data from a remote equipment to a monitoring agent, usually by the usage of wireless sensors. In robotics, telemetry is necessary in order to supervise robots behavior and its internal data during the robot's execution time. Oftentimes, telemetry and teleoperation can be seen as complementary processes, once a person can control the robot while follows the value of the robot's variables.

So, communication can be the main issue of telemetry in robotics, because, commonly, there is a great time delay of exchanging data. Another problem of communication is the fact that, in some situations, it needs a special protocol in order to be able to communicate in the right way (Ingrand et al. 1996).

In this paper, telemetry will be taken as a tool for monitoring the robot data remotely.

4 RoboFEI Framework for Humanoid Robots

4.1 RoboFEI-HT Simulator

RoboFEI-HT Simulator (Perico et al. 2016), depicted in Fig. 2, is a 2D simulator built upon the Cross Architecture, that simulates three processes from the Cross Architecture: vision, control and IMU. Figure 1 depicts the simulated processes in gray boxes.

All the processes linked to the cognitive algorithms of the robot, such as decision, localization, planning and communication, run in both platforms: real or virtual robots. As already mentioned, this abstraction is possible due to the blackboard.

So, from the point-of-view of cognitive processes, it does not matter whether the robot is real or virtual; decisions will be taken following the data published in the blackboard.

The proposed simulator supports multi-robot systems, where the robots can exchange data between them via socket (UDP or TCP), following the same procedure used by real robots.

RoboFEI-HT Simulator brings some contributions in relation to the already available simulators. The first one is the possibility to create prototype decision-making algorithms and the ease portability of these algorithms from the simulation to a real robot that works with the Cross Architecture, where only few parameters may need to be lightly adjusted in the actual robot. Another advantage is the fact that the proposed simulator is free, open source and it does not require a powerful computer to work.

4.1.1 Physics Simulation

The purpose of this simulator is to simulate the physical environment of the soccer competition in order to test the high-level algorithms developed by the team. The simulated world has two dimensions, allowing the robots to move forward, backward, to the left, to the right and to rotate around its own axis, representing the movements of real robots.

The objects' movements are simulated using the concept of speed. For robots, it is applied a rotation model, which changes the orientation θ_t of the robot by summing a constant turning factor $\Delta\theta$ to the previous robot's orientation θ_{t-1}, as shown in Eq. (1). The velocity $\hat{v}_t$ is computed by applying the current orientation θ_t to the previous speed $\hat{v}_{t-1}$, as shown in Eq. (2), where a_f and a_s are the forward acceleration and the sideways acceleration, respectively. Then, the current position $\hat{s}_t$ of the robot is computed by the sum of the previous robot's position $\hat{s}_{t-1}$ and the current robot's speed $\hat{v}_t$, as in Eq. (3). The simulator runs 60 frames per second; however, this value can be easily changed if needed.

$$\theta_t = \theta_{t-1} + \Delta\theta. \tag{1}$$

$$\begin{pmatrix} v_t^x \\ v_t^y \end{pmatrix} = \begin{pmatrix} v_{t-1}^x \\ v_{t-1}^y \end{pmatrix} + \begin{pmatrix} \cos(\theta_t) & -\sin(\theta_t) \\ -\sin(\theta_t) & -\cos(\theta_t) \end{pmatrix} \times \begin{pmatrix} a_\mathrm{f} \\ a_\mathrm{s} \end{pmatrix}. \tag{2}$$

$$\begin{pmatrix} s_t^x \\ s_t^y \end{pmatrix} = \begin{pmatrix} s_{t-1}^x \\ s_{t-1}^y \end{pmatrix} + \begin{pmatrix} v_t^x \\ v_t^y \end{pmatrix} \tag{3}$$

Robots' movements are affected by errors. These errors are simulated by normal distributed random numbers, that are added to the movements speed. There are three kinds of errors: speed error, which causes the robot to move forward or backward; drifting error, which causes sideways movements; and rotation error, which causes robots to rotate while moving.

When a robot kicks the ball, a new velocity is given to the ball. The orientation and direction are determined by the relative angle between robot and the ball, and its value is determined in function of the distance between ball and robot. This velocity decreases over time by a factor f, called friction. Note that the factor f ranges from 0 to 1, as shown in Eq. (4). Then, the ball's position is updated following Eq. (3).

$$\begin{pmatrix} v_t^{b,x} \\ v_t^{b,y} \end{pmatrix} = \begin{pmatrix} f & 0 \\ 0 & f \end{pmatrix} \times \begin{pmatrix} v_{t-1}^{b,x} \\ v_{t-1}^{b,y} \end{pmatrix}. \tag{4}$$

Collisions are interactions between objects, which can be classified in four categories: interactions between robots; interactions between robot and ball; interactions between robot and static objects; and interactions between ball and static objects. All interactions between robots, ball and goalposts were approximated by an algorithm of collisions between circles. To prevent robots and the ball to move beyond the limits of the simulation, the borders of the simulation were considered as walls.

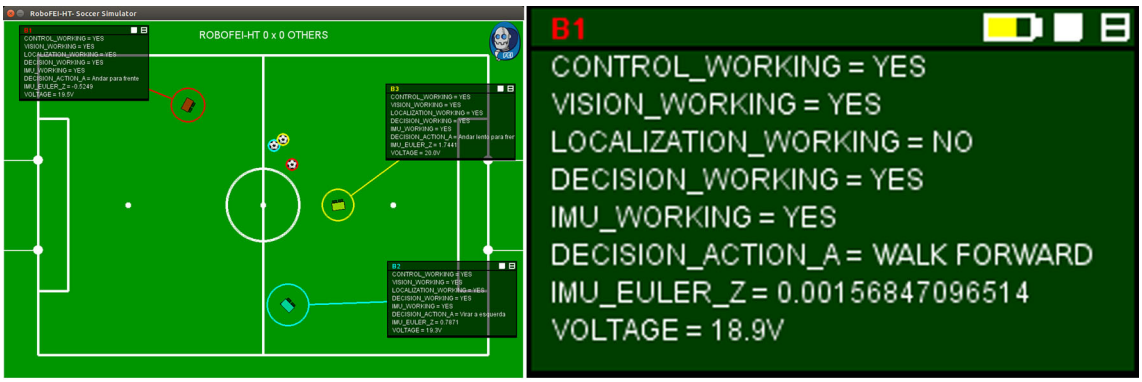
4.1.2 Movement Control

Two processes of the Cross Architecture are responsible for the robots movement: decision, which chooses the action performed by the robot, and control, which executes the movement. They communicate with each other through the blackboard. So, control reads a variable published by decision in the blackboard and acts following this received instruction. In order to simulate this behavior, each simulated robot has its own control process, which reads the same variables from the shared memory that the control of the real robot would read, and executes the same actions the real robot would execute.

4.1.3 Vision

The vision process is responsible for recognizing objects in the field, like the ball, teammates and opponents. The field-of-view of the robot is set to 70° and the pan is limited to 270° , following the rules of the RoboCup Humanoid League (2016/2017).

The search for objects in the soccer field is controlled by the decision process, that defines what kind of object should be searched and determines the beginning and the end of the process. During the search, objects positioned in the field-of-view and far up to 3 m from the robot can be identified. Finally, the calculated distance value (in meters) and angle value (in degrees) with respect to the robot are published in the blackboard. Normal distributed random numbers can be also added in these calculated values in order to simulate vision errors.



(a) (b)

Fig. 3 Telemetry graphical user interface. **a** Graphical user interface showing three robots with its respective status windows, **b** status window in details

4.2 RoboFEI-HT Telemetry

The motivation of the telemetry came from the problem of not knowing what was happening to the robot when it had a problem. The idea was to develop a graphical user interface which can represent the inner states of the robot in a friendly way, and not just a log screen. It was developed as an extension to the RoboFEI-HT Simulator, but works in a different environment.

4.2.1 Motivation

After the use of the simulator to develop cognitive algorithms for robotics, the natural next step is to test the same algorithm in real robots. As already cited, real robots present a lot of problems, such as walking problems, overheating servomotors or battery drain issues; thus, a telemetry interface has great value by allowing the monitoring of these problems during robot execution time.

4.2.2 Graphical User Interface

The function of the RoboFEI-HT Telemetry is to be an output screen for the team of robots, instead of a remote controller. The simulator was designed to be used for the RoboCup Soccer environment and, as a rule of the competition, teams must not teleoperate the robots. Following this rule, the telemetry only receives information.

Among it's capacities, the telemetry interface can read the Game Controller information (referee information), such as quantity of scored goals and time of match, and display it on screen. The telemetry presents a screen, which shows the environment where the robots are working on, as depicted in Fig. 3. Each robot is represented by a different color, and the robot's position is drawn based on the believes obtained by the localization process. The circles around each robot represent their believes in such position: smaller circles means higher believes. Another utility is to present the position the robot perceives the ball.

Each robot in the telemetry interface has a floating screen, which presents the robot's name, its battery condition and the internal status of the Cross Architecture processes. All these data are taken from the robot's blackboard.

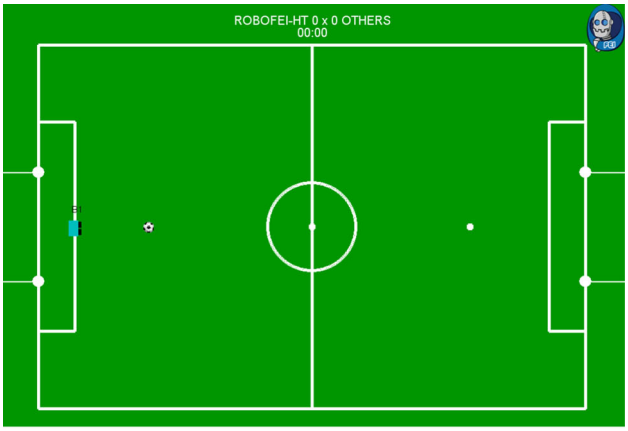
Table 1 Communication structure

Robot number	Localization	Processes	Others

4.2.3 Communication Protocol

An example of communication protocol is presented by Table 1, where robot number is a number which represent the robot, localization represent the set of variables used to draw the robot on screen, processes is a set of variables which represents the working processes, and others are a set of all other variables, which the user is willing to see in the telemetry screen.

These variables are sent through UDP protocol as a text message. Each robot is attached to a specific port, in a way the telemetry will always read this same port in order to get information from the given robot. Since the protocol converts everything into text messages, it is good to take care on how the information has been sent, depending on the precision used by the float numbers, message can became too big, causing the communication to be ineffective. Also, the computer with the telemetry and the robots need to be in the same network. Note that the simulator and the telemetry can work simultaneously in the same computer, any simulated robot can send information to telemetry in order to test the communication, or even making preliminary tests before testing in real robots.



(a)

(b)

Fig. 4 First proposed scenario for both domains. **a** Simulation, **b** real robot

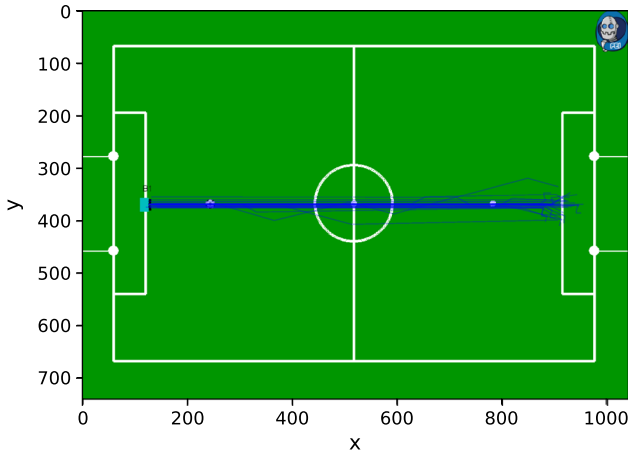
5 Case Studies

This section aims to validate the complete humanoid robot framework. So, the experiments performed in Sect. 5.1 aim to validate the high-level strategy development in the proposed simulator and its portability to a real robot, while Sect. 5.2 aims to evaluate the robot behavior within the telemetry interface using, as input, the information received from the localization process.

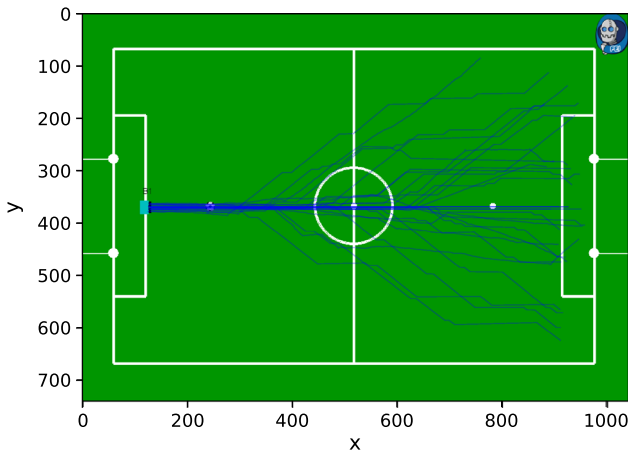
A humanoid robot inspired on DARwIn-OP (Ha et al. 2011) was used in the experiments. This robot uses an Intel Core i5 1.66 GHz computer with 8 GB SDRAM, running Ubuntu 14.04 LTS. The same computer was also used in the simulated experiments.

5.1 Transference of Algorithms Developed in the Simulator for the Real Robot

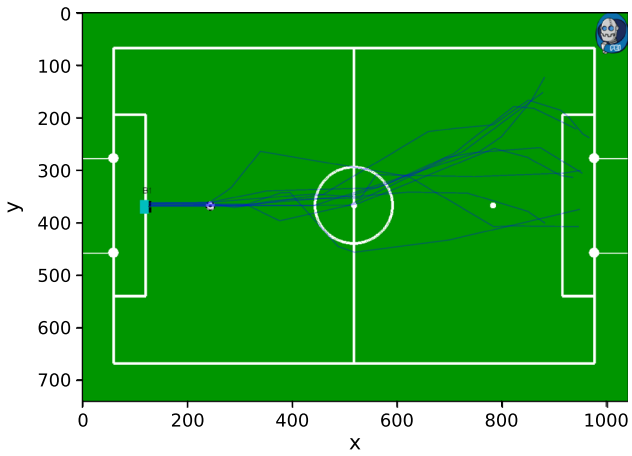
This section illustrates the use of the decision process on the simulator and its extension to real robots. The development and analysis of the decision was chosen considering that the control, IMU and vision are also implemented and available for the real robot. In order to validate the transference of the developed source code from the simulator to a real robot, the



(a)



(b)



(c)

Fig. 5 Paths performed by the virtual and by the real robot. **a** Simulated environment without considering error rate, **b** simulated environment with error rate, **c** real environment

Table 2 Results comparing the virtual robot and the real humanoid robot

	Average distance traveled (cm)	Standard deviation (cm)	% of scored goals
Real humanoid robot	848.40	42.82	50
Simulated robot-no error	791.17	20.70	100
Simulated robot-with errors	879.77	101.40	53

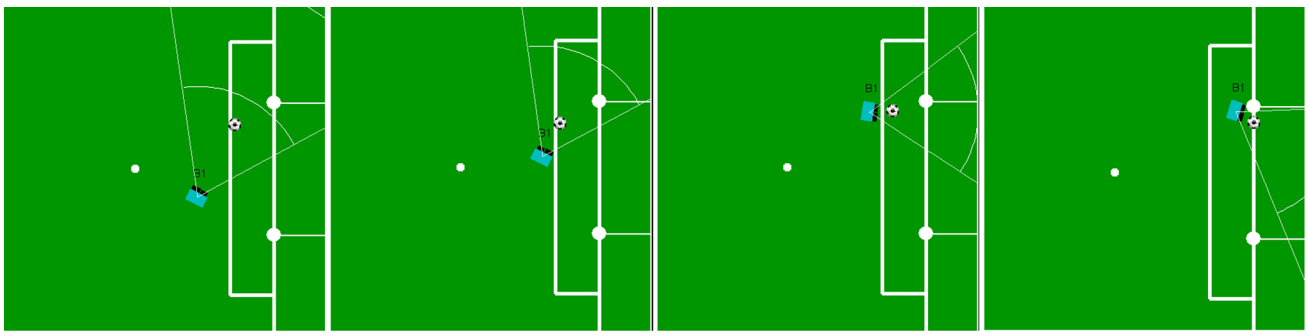


Fig. 6 Simulation: path performed in the second scenario

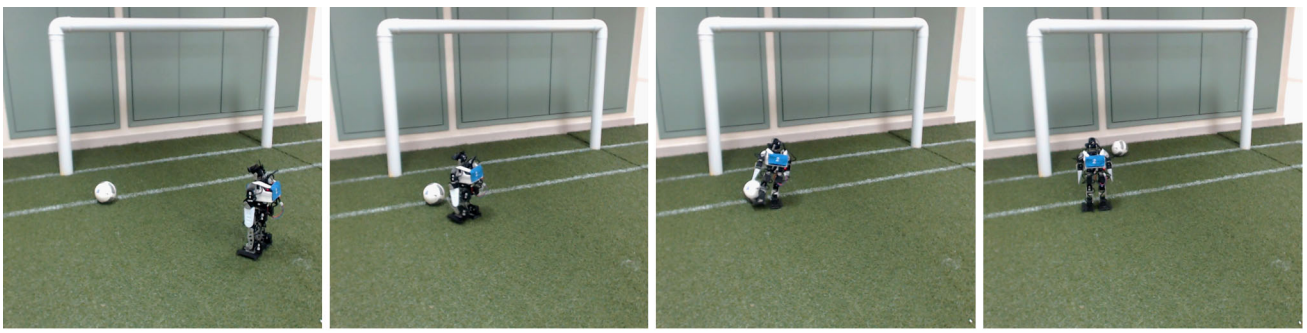


Fig. 7 Real world: path performed in the second scenario

decision process was implemented using the naive concept, in which the robot searches for the ball, walks toward it, and kicks the ball.

Two scenarios were considered for the experiments. Figure 4 shows the first scenario, in which the ball is positioned at the penalty mark and the robot is positioned near the ball, at the penalty area. So, due to the naive decision, it was expected that the robot keep itself kicking the ball and walking until a goal was scored, or until the ball was kicked outside the field. In order to allow the analysis of the portability of the algorithms between simulation and real world, the trajectory, as well as the traveled distance and the number of scored goals were stored.

Experiments with the real robot were performed 10 times, while experiments with the virtual robot were performed 30 times. Thus, three graphs of traveled paths have been generated and can be seen in Fig. 5. In the case of the virtual robot, two analyzes were performed, one without and other with the inclusion of errors in the robot. The errors were included in the robot velocity, in lateral displacement and in rotation and were generated randomly from a normal distribution, considering mean equal to zero and the following standard deviations: 0.2; 0.2; 0.1. In addition to the errors, a Gaussian noise was also included in the data received from IMU. This noise was generated with mean equal to zero and a standard deviation of 0.01. Errors and noise are updated at each iteration of the simulator. These deviation values were empirically obtained.

As it can be seen in Fig. 5a, the average path performed by the simulated robot, without considering error rates, was basically a straight line, where the robot scored 100% of the goals and traveled an average distance of 791.17 cm. Figure 5b depicts the traveled path performed by the virtual robot considering error rates, where the robot scored 53%

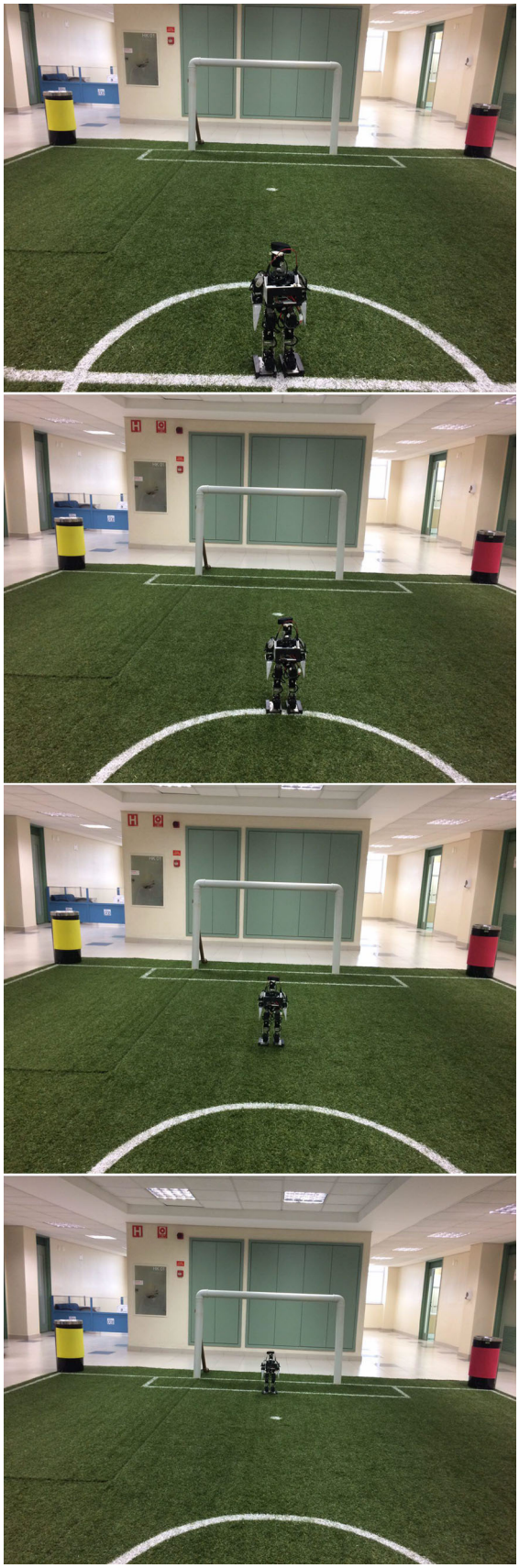


Fig. 8 Real robot positioning

Fig. 9 Real robot telemetry

of the goals and traveled an average distance of 879.77 cm. Finally, Fig. 5c shows the paths performed by the real robot, which scored 50% of the goals and traveled an average distance of 848.40 cm. Table 2 summarizes the results obtained in this experiment.

The t-test with unequal variances was performed in order to analyze whether the data obtained during the simulated experiments, considering the robot with error rates, were statistically different from the data obtained during the real robot experiment. The t-value calculated was 1.37; considering a confidence of 95%, t critical one-tail was 1.69 and t critical two-tail was 2.03. As t value is lower than both t critical values, we can conclude that the t-test failed verifying the null hypothesis, which can mean that is not possible to prove that data are different with confidence 95%.

The results demonstrate that the actions, paths and distances performed by robot in the simulator environment (considering error rates) are similar to the actions performed by the real humanoid robot, as described in Table 2.

The second analyzed scenario was shown in Figs. 6a and 7a, where the ball is positioned in front of a goalpost and the robot must walk toward it and kick the ball. The virtual robot was setup with the same error rates used during the first scenario, then Figs. 6 and 7, from (a) to (d) depicts the movement performed by the virtual and by the real robot, respectively.

Both experiments indicate that a high-level decision algorithm, implemented in the simulator, can be reproduced in real robots with minor changes. This is possible because the output of the simulated processes (IMU, control and vision) is very similar to the output of these same processes in real robots.

5.2 Telemetry in a Real Robot

This section illustrates the usage of the telemetry interface in order to show the robot's belief pose in the field. All the input data received by telemetry interface came from the localization process, that was running inside the real robot.

In order to evaluate the data representation of the localization and telemetry processes, the robot walked to different random positions in the field and its position was updated in these processes. The localization process has performed the robot localization using two landmarks in the corners of the field; the robot's position was sent to the telemetry interface, while it was updated automatically.

Figure 8 depicts the random positions of the real robot during the experiment, while Fig. 9 presents the data received by telemetry interface. Besides pose information, telemetry also presents the status of several processes of the Cross Architecture and battery level.

6 Conclusion

This paper describes the RoboFEI framework for humanoid robots developed for the RoboCup Humanoid Soccer domain, which allows cognitive algorithms to be implemented, simulated, transferred to real humanoid robots and evaluated by using the monitoring offered by telemetry interface.

The proposed framework has some advantages, for instance, it is open source[1] and it has low computational cost. With respect to the simulator, the main advantage is the fact that it allows the development of cognitive algorithms without having real robots, which also allows the reproducibility of any new proposed source code. In addition, the proposed simulator allows the transference of reasoning, learning and localization algorithms to real robots that uses the Cross Architecture with only few adjusts in some parameters. Furthermore, the proposed telemetry interface allows the actual time monitoring of an implemented algorithm in a real robot.

The performed experiments showed that framework can be a valuable tool during the development of high-level strategies for humanoid robots, since the framework helps: (1) the development and testing of cognitive algorithms in simulator; (2) the transfer of codes to real robots and; (3) using the telemetry, to evaluate the experiments in real robots. Despite the chosen domain, the framework can be adapted for almost every environment where a humanoid robot needs to work.

As future work, it is possible to improve the telemetry interface in order to allow it to be more generic in regard to the kind of information shown on the screen. With respect to the simulator, more work is needed in order to allow it to run in an accelerated mode, which would allow the simulation to be performed directly in the robot's computer. This last feature could be desirable because of several reasons, for instance, the execution of the simulator could be done in parallel with the real robot, reproducing the actual game situation in a simulated environment; thus, during an actual game, the virtual robot could learn some behavior in the simulator (running faster than the real robot) and this learned behavior could be used by the real robot.

Acknowledgements The authors would like to thank CAPES, CNPq and FAPESP (grants 2016/21047-3 and 2016/18792-9) for their financial support.

References

Allgeuer, P., Schwarz, M., Pastrana, J., Schueller, S., Missura, M., & Behnke, S. (2013). A ROS-based software framework for the NimbRo-OP humanoid open platform. In *Proceedings of the 8th workshop on humanoid soccer robot. IEEE-RAS, Atlanta.*

[1] The source code of the framework is available at: https://github.com/danilo-perico/robofei-ht-framework.

Arkin, R. C. (1998). *Behavior-based robotics*. Cambridge: MIT Press.

Barrett, S., Genter, K., He, Y., Hester, T., Khandelwal, P., Menashe, J., et al. (2013). UT Austin Villa 2012: Standard Platform League World Champions. In *RoboCup*, China.

Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, *2*(1), 14–23.

Dellaert, F., Fox, D., Burgard, W., & Thrun, S. (1999). Monte Carlo localization for mobile robots. In *1999 IEEE international conference on robotics and automation. Proceedings* (Vol. 2, pp. 1322–1328). IEEE.

Fox, D., Burgard, W., Dellaert, F., & Thrun, S. (1999). Monte carlo localization: Efficient position estimation for mobile robots. *AAAI/IAAI*, *343–349*, 1999.

Gazebo. (2016). http://gazebosim.org. Accessed May 19, 2016.

Ha, I., Tamura, Y., Asama, H., Han, J., & Hong, D. W. (2011). Development of open humanoid platform DARwIn-OP. In *SICE annual conference 2011* (pp. 2178–2181).

Hayes-Roth, B. (1985). A blackboard architecture for control. *Artificial Intelligence*, *26*(3), 251–321.

Homem, T. P. D., Perico, D. H., Santos, P. E., Bianchi, R. A. C., & de Mantaras, R. L. (2017). Retrieving and reusing qualitative cases: An application in humanoid-robot soccer. *AI Communications*, *30*(3–4), 251–265.

Ingrand, F. F., Chatila, R., Alami, R., & Robert, F. (1996). PRS: A high level supervision and control language for autonomous mobile robots. In *1996 IEEE international conference on robotics and automation. Proceedings* (Vol. 1, pp. 43–49). IEEE.

Perico, D. H., Bianchi, R. A. C., Santos, P. E., & Lopez de Mántaras, R. (2016). Collaborative communication of qualitative spatial perceptions for multi-robot systems. In *Proceedings of 29th international workshop on qualitative reasoning (IJCAI), New York*.

Perico, D. H., Homem, T. P. D., Almeida, A. C., Silva, I. J., Vilão, C. O., Ferreira, V. N., & Bianchi, R. A. C. (2016). A robot simulator based on the cross architecture for the development of cognitive robotics. In *2016 XIII Latin American robotics symposium and IV Brazilian robotics symposium (LARS/SBR)* (pp. 317–322).

Perico, D. H., Silva, I. J., Vilão Jr., C. O., Homem, T. P. D., Destro, R. C., & Tonidandel, F. (2014a). *Joint conference on robotics, LARS 2014, SBR 2014, robocontrol 2014. Revised selected papers, chapter Newton: A high level control humanoid robot for the RoboCup Soccer KidSize League*. Berlin: Springer.

Perico, D. H., Silva, I. J., Vilão, C. O., Homem, T. P. D., Destro, R. C., Tonidandel, F., et al. (2014b). Hardware and software aspects of the design and assembly of a new humanoid robot for robocup soccer. In *Robotics: SBR-LARS robotics symposium and robocontrol (SBR LARS Robocontrol)*.

Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., Leibs, J., Wheeler, R., & Ng, A. Y. (2009). ROS: An open-source robot operating system. In *ICRA workshop on open source software*.

RoboCup Humanoid League. (2016/2017). RoboCup Soccer Humanoid League Laws of the Game. https://www.robocuphumanoid.org/materials/rules/. Accessed Nov 20, 2017.

Robocup Soccer simulation. (2017). http://wiki.robocup.org/Soccer_Simulation_League. Accessed Nov 20, 2017.

Russell, S. J., & Norvig, P. (2010). *Artificial intelligence: A modern approach*. Englewood Cliffs: Prentice Hall Press.

Silva, I. J., Perico, D. H., Homem, T. P. D., Vilão, C. O., Tonidandel, F., & Bianchi, R. A. C. (2015). Using reinforcement learning to improve the stability of a humanoid robot: Walking on sloped terrain. In *12th Latin American robotics symposium and 2015 3rd Brazilian symposium on robotics (LARS-SBR)*.

Vilão, C. O., Perico, D. H., Silva, I. J., Homem, T. P. D., Tonidandel, F., & Bianchi, R. A. C. (2014). A single camera vision system for a humanoid robot. In *SBR-LARS robotics symposium and robocontrol*.

Virtual robot experimentation platform. (2016). http://www.v-rep.eu/. Accessed May 19, 2016.

Webots. (2016). http://www.cyberbotics.com/. Accessed May 19, 2016.