A Robot Simulator Based on the Cross Architecture for the Development of Cognitive Robotics

Danilo H. Perico*, Thiago P. D. Homem*†, Aislan C. Almeida*, Isaac J. Silva*,
Claudio O. Vilão Jr.*, Vinicius N. Ferreira*, Reinaldo A. C. Bianchi*
* Electrical Engineering Department
Centro Universitário FEI
São Bernardo do Campo, São Paulo, Brazil
Email: {dperico, rbianchi}@fei.edu.br
† Department of Informatics
Instituto Federal de São Paulo
São Paulo, São Paulo, Brazil

***Abstract*—This paper presents a new 2D robot simulator based on the Cross Architecture for RoboCup Soccer Humanoid League domain. A simulator is an important tool for testing cognitive algorithms in robots without the need of handling with real robot problems; moreover, a simulator is extremely useful for allowing reproducibility of any developed algorithm, even if there is no robot available. The proposed simulator allows the direct application of the algorithms developed in the simulator into a real robot that works with the Cross Architecture. Besides, the simulator is freely available, it is open-source and has low computational cost. Experiments were conducted in order to analyze the portability of a decision code developed in the simulator to a real robot. The results allowed us to conclude that the simulator can be used to test new algorithms, since the decisions performed by the robot in simulation and in the real world were quite similar.**

I. Introduction

The Cross Architecture [1] 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 without the need of microcontroller boards and without performance loss.

Several modules of the Cross Architecture are dedicated to the development of the Artificial Intelligence algorithms, for instance, robot's decision, localization and planning. Nevertheless, one of the main problems of developing cognitive algorithms in real humanoid robots are the physical issues, such as backlashes, broken parts, bad connections and so on. 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. Reproducibility is very important in order to allow the spread of knowledge.

Thereby, a simulator is an important tool for debugging and testing high-level algorithms without the need of dealing with real robot problems. Moreover, it allows the reproducibility of any proposed algorithm even if there is no robot available.

Hence, the goal of this work is the proposal of a 2D humanoid soccer simulator that allows the abstraction of real world problems during the development of high-level algorithms that are based on the Cross Architecture. Thus, the simulator's domain is the RoboCup Soccer Humanoid League.

There are already several kinds of robot simulators available, such as those depicted in Section III. However, the cognitive algorithms developed in the proposed simulator can be directly applied in the humanoid robots that use the Cross Architecture as their software architecture.

This work is structured as follows: Section II briefly describes the Cross Architecture. Section III depicts some usual robot simulators. Section IV shows the proposed simulator. In Section V a case study is presented and Section VI provides the conclusions and indicates avenues for extend this work.

II. Cross Architecture

Software architecture is essential to the development of successful applications in robotics, specially considering the complex architecture requested by humanoid robots. Software architecture can be described as a guidance for software design and development, acting as a high-level modeling, it represents the fundamental concepts, properties or restrictions of a system [2]. Following the software architecture concepts, the Cross Architecture was developed with focus in humanoid robot soccer players.

Most common paradigms for software architecture used in robotics are [3]: hierarchical, reactive [4] and hybrid [5]. The hierarchical paradigm is based in the sequential and orderly relation between the three primitives of robotics: Sense – Plan – Act. On the other hand, the reactive paradigm uses a vertical decomposition, where the robot has a Sense – Act type of organization with some primitive behaviors [3].

Due to the strong acceptance of reactive and hierarchical paradigms in robotics, a new paradigm, that combined these last two, was proposed. This new paradigm was named hybrid and it aims the creation of a system capable to plan as long as it senses and reacts [6].

The hybrid paradigm 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 [7], do not need to be planned or deliberated. When one of those situations occurs, the robot should simply sense and

978-1-5090-3656-1/16 $31.00 © 2016 IEEE
DOI 10.1109/LARS-SBR.2016.60

react, leaving complex behaviors, like deciding what to do or localizing itself, to the hierarchical structure.

The Cross Architecture is based on the hybrid paradigm. Due to the blackboard concept [8], that is a shared memory area that allows inter-process communication, the Cross Architecture allows an abstraction layer between the processes required for a humanoid robot soccer player. Additionally, this architecture also allows a humanoid robot to work properly without any dedicated sub-controller implemented in low-level hardware (e.g., a microcontroller board is not needed for controlling the motors). The Cross Architecture states that independent processes can communicate between each other in order to achieve an intelligent system. It is composed by 8 processes: Vision, Localization, Decision, Communication, Planning, Inertial Measurement Unit (IMU), Control and Management [1].

Figure 1 depicts how the processes are interconnected and how they access the blackboard (bkb) to exchange data among them, considering a robot n. For example, the Decision process, in order to reason about the next action, uses data from Vision, Localization and Communication. Then, it may set the Vision to continue searching for the ball or it may send to Planning a command to compute a different trajectory, which must send to Control a message to start a movement.

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 [7], [9] and [10].

III. Simulators

Robot simulators are commonly used for the development of algorithms without the need of having an actual robot. The usage of simulators allows the performance of a large number of experiments, without the risk of damaging the robots.

The most used robot simulators have some disadvantages. For example, the robot's behavior can differ between simulation and real world. This happens because the simulation model is commonly simplified and, hence, imperfect. So, the simulation might not reproduce the same results as the real world, even using a quite good physics engine [11].

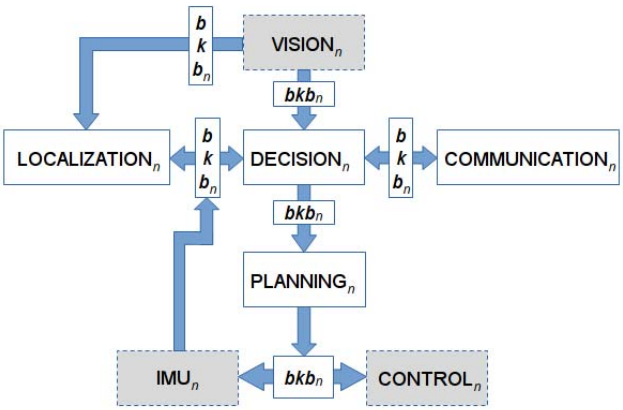


Fig. 1. The Cross Architecture. Adapted from [1]. Gray boxes are the simulated processes.

Beyond that, the 3D simulators like Gazebo [12], Webots [13], V-REP [14] and RoboCup Soccer 3D [15] often have high computational cost and they need to have the specific desired robot model, which is not always available by default. In the 2D simulators, such as the RoboCup Soccer 2D [15] and TauraSim [16], the great difficulty perceived was the integration of the Cross Architecture in these environments and the direct portability of the developed algorithms to real robots. The quoted simulators are briefly described hereafter.

A. Gazebo

Gazebo is an open source robotic simulation package, that uses the open source OGRE rendering engine, which produces high-quality graphics [17]. Gazebo offers the ability to simulate robots in indoor and outdoor environments. The environment can be made using simulated light and static objects. The physics engines it supports are: ODE, Bullet, Simbody, and DART (ODE is default engine). Gazebo is available for Mac and Linux [18] and it is currently being maintained by the company Open Source Robotics Foundation (OSRF), which is the same company that maintains the ROS.

B. Webots

Webots is a 3D simulation environment used to model, program and simulate mobile robots. Webots can runs natively on Windows, Mac and Linux, so the world files and API functions can easily be used in different operating systems. A simulation can have many robots, limited only by computer memory and CPU power, and each robot may run with its own specific controller code. The robot controllers can be programmed in C, C++, Java, Python and Matlab, or use a ROS node. The robot behavior can be tested in physically realistic worlds by using the Open Dynamics Engine (ODE), physics engine library [18].

C. V-Rep

V-REP (Virtual Robot Experimentation Platform) is available for Windows, Mac and Linux. The controllers can be programmed in C/C++, Python, Java, Lua, Matlab, Octave or Urbi. The physics engines are: Bullet Physics, ODE, Newton and Vortex Dynamics.

D. RoboCup Soccer 3D

In Robocup, 3D soccer simulation is performed in the RoboCup Simulated Soccer Server 3D (RCSSServer3D) since 2008, and runs on Linux, Windows and Mac [19]. RCSSServer3D is based on SimSpark. SimSpark uses the ODE library for simulating rigid body dynamics. The robot agents in the simulation are the Aldebaran NAO robot.

E. RoboCup Soccer 2D

The RoboCup Soccer Simulator 2D is the official simulator of RoboCup 2D simulation league. In this software, each agent connects to the server, simulating a soccer game and an independent monitor shows the status of the game.

F. TauraSim

TauraSim is a simulator developed by Taura Bots team for developing high-level strategies in RoboCup Soccer Humanoid League. This simulator works in two modules: the first one handles the 2D simulation and the second one is an interface for the robot's intelligence. The communication between the modules is made via socket in JSON format.

IV. RoboFEI-HT Simulator

RoboFEI-HT Simulator (Figure 2) is inspired by TauraSim, since it is also a 2D simulator made in Python with the support of the graphical library Pygame [20]. However, the proposed simulator is built upon the Cross Architecture, using the blackboard concept for communication between processes. The proposed simulator aims to simulate 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 runs in both platforms: real or simulated 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 communication between the robots is made via UDP, following the same procedure used by real robots.

RoboFEI-HT Simulator brings some contributions in relation to the other simulators, shown in Section III. The first one is the direct portability of the high-level algorithms from the simulation to a real robot that uses the Cross Architecture. Another advantage is the fact that the proposed simulator is free, open-source, and does not require a powerful computer to work, which guarantee the reproducibility of any proposed cognitive algorithm for humanoid robots.

Fig. 2. RoboFEI-HT Simulator.

A. 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 equation 1. The speed vector $\hat{v}_t$ is computed by applying the current orientation θ_t to the previous speed $\hat{v}_{t-1}$, as shown in equation 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 equation 3. The simulator runs 60 frames per second, however this value can be easily changed if needed.

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

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

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

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.

For the ball, friction is used for reducing its speed with time. Friction is represented by a factor f which, in each iteration, is applied to the speed $\hat{s}_{t-1}$, resulting in a new speed $\hat{s}_t$. Note that the factor f ranges from 0 to 1, as shown in equation 4. Then, the ball's position is updated following Equation 3.

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

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 goal poles 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.

B. 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 act 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.

C. Vision

The Vision process is the main robot sensor and it 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 degrees and the pan is limited to 270 degrees, following the rules of the RoboCup Humanoid League.

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 meters 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.

V. A CASE STUDY

Aiming the evaluation of the portability between the proposed simulator and a real robot, some high-level strategies, previously developed in the simulator, were applied in an actual robot.

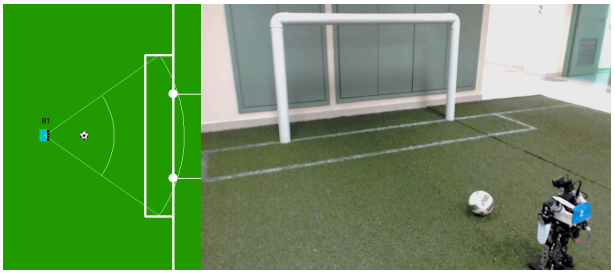
A. Software and Hardware Setup

A humanoid robot inspired on DARwIn-OP [21] was used in the experiments. This robot uses an Intel Core i5 1.66GHz computer with 8GB SDRAM, running Ubuntu 14.04 LTS. The same computer was also used in the simulated experiments.

For reproducibility reasons, the simulator proposed in this work along with the decision algorithms used during the experiments are available at http://fei.edu.br/~rbianchi/software.html, as well as the documentation that describes its installation and usage.

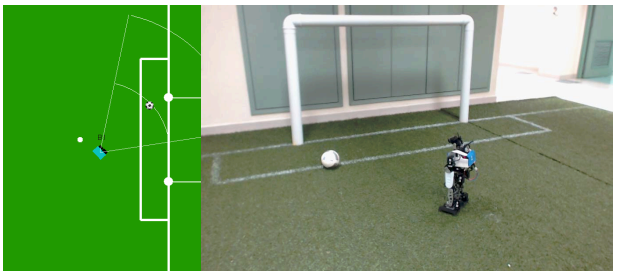
B. Experiments

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. The proposed simulator allows the scientific community to develop specific cognitive algorithms, such as strategies and robot localization, without dealing with physical robot problems.



(a) Simulation (b) Real robot

Fig. 3. First proposed scenario for both domains.



(a) Simulation (b) Real robot

Fig. 4. Second proposed scenario for both domains.

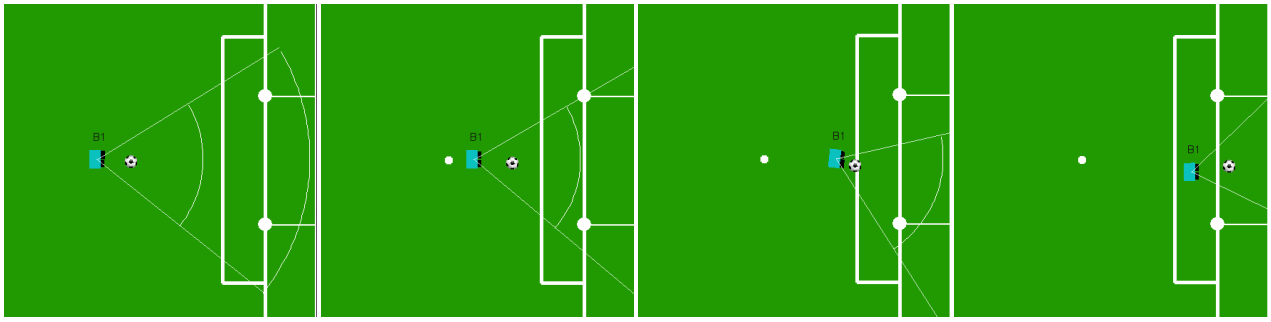
In order to validate the transference of the developed source code from the simulator to a real robot, the Decision process consisted of a basic approach, named naïve, in which the robot searches for the ball, walks toward it, aligns itself to the opponent's goal and kicks the ball.

This work focuses in the cognitive processes of the robot, therefore the analysis of the experiments were qualitatively made disregarding the physics of the simulator, aiming to compare the robot's behavior between simulated and real environments. Two scenarios were considered for the experiments. Figure 3 shows the first scenario, in which the ball is positioned at the penalty mark and the robot is positioned near the ball. Figure 4 presents the second scenario in which the ball is positioned to the left of the goal and the robot must walk toward it, aligns to the opponent's goal and kicks the ball. In order to analyze the portability of the algorithms between simulation and real world, the trajectory of the robot was stored and compared in both domains.

The results demonstrate that the actions performed by the real robot, in both experiments, are similar to the actions performed by the simulated robot, as shown in Figures 5 to 8, that describe the trajectories of the robot and the ball for each environment[1].

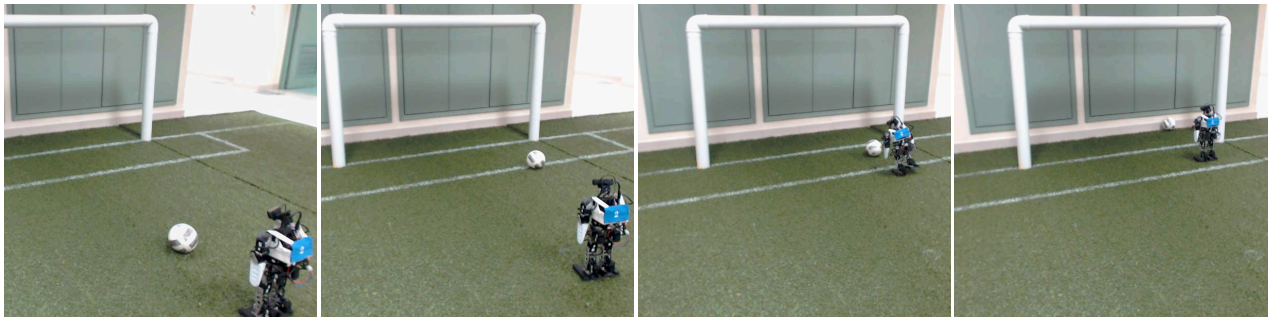
Figures 5 and 6 show the straight path performed by the robot in both environments, in which the robot is already oriented toward the opponent's goal, it walks forward and

[1]The video of the experiments is available at: https://youtu.be/W4RhSeu68Z4.



(a) (b) (c) (d)

Fig. 5. Simulation: path performed in the first scenario.



(a) (b) (c) (d)

Fig. 6. Real world: path performed in the first scenario.

kicks the ball. Figure 7 and 8 show the curved path, in which the robot walks toward the ball, turns left or right, according to its orientation, aligns to the opponent's goal and kicks the ball.

The 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) are very similar to the output of these same processes in real robots.

VI. Conclusion

This paper describes the RoboFEI-HT Simulator developed for the RoboCup Humanoid Soccer domain, which allows cognitive algorithms to be implemented, simulated and transferred to real humanoid robots.

The simulator has some advantages when compared to other existing simulators: the usage of the Cross Architecture, allowing the transference of reasoning, learning and localization algorithms directly to real robots that uses this architecture; and the fact that the software is open-source, allowing the reproducibility of any new proposed code, even when there is not a robot available.

The performed experiments showed that high-level algorithms developed in simulation can be extended to the robots without many changes. For instance, during the experiments, the same decision algorithm implemented in the simulator was extended to the real robot without many changes and the robot performed the same actions as running in simulation.

As future work, new cognitive actions such as reasoning, learning, and self-localization can be implemented and tested in order to evaluate the simulator and the portability of some other high-level Artificial Intelligence algorithms to real robots.

Acknowledgment

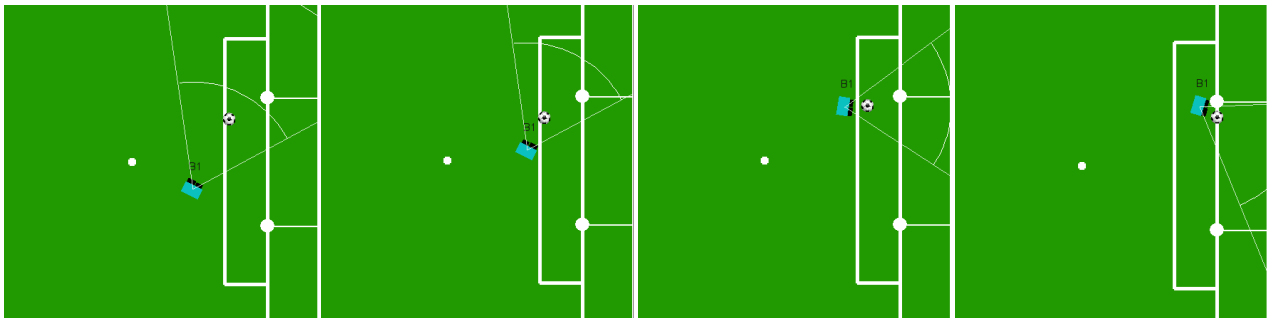
Danilo H. Perico, Thiago P. D. Homem, Isaac J. Silva, Claudio O. Vilão Jr. and Vinicius N. Ferreira acknowledge support from CAPES; Aislan C. Almeida acknowledges support from CNPq.

References

[1] D. H. Perico, I. J. Silva, C. O. Vilão Junior, T. P. D. Homem, R. C. Destro, F. Tonidandel, and R. A. C. Bianchi, *Robotics: Joint Conference on Robotics, LARS 2014, SBR 2014, Robocontrol 2014, São Carlos, Brazil, October 18-23, 2014. Revised Selected Papers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, ch. Newton: A High Level Control Humanoid Robot for the RoboCup Soccer KidSize League, pp. 53–73. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-48134-9_4

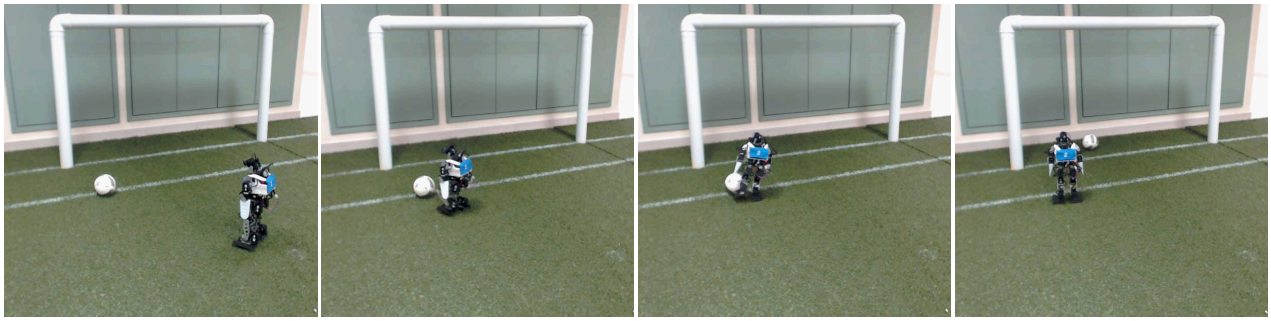
[2] C. Yang, P. Liang, and P. Avgeriou, "A systematic mapping study on the combination of software architecture and agile development," *Journal of Systems and Software*, vol. 111, pp. 157 – 184, 2016.

[3] R. Murphy, *Introduction to AI Robotics*. Cambridge, MA: MIT Press, 2000.



(a) (b) (c) (d)

Fig. 7. Simulation: path performed in the second scenario.



(a) (b) (c) (d)

Fig. 8. Real world: path performed in the second scenario.

[4] R. A. Brooks, "A robust layered control system for a mobile robot," *Robotics and Automation, IEEE Journal of*, vol. 2, no. 1, pp. 14–23, 1986.

[5] R. C. Arkin and T. Balch, "Aura: Principles and practice in review," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 9, pp. 175–189, 1997.

[6] R. C. Arkin, *Behavior-based robotics*. MIT press, 1998.

[7] I. J. Silva, D. H. Perico, T. P. D. Homem, C. O. Vilão, F. Tonidandel, and R. A. C. Bianchi, "Using reinforcement learning to improve the stability of a humanoid robot: Walking on sloped terrain," in *2015 12th Latin American Robotics Symposium and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR)*, Oct 2015, pp. 210–215.

[8] B. Hayes-Roth, "A blackboard architecture for control," *Artif. Intell.*, vol. 26, no. 3, pp. 251–321, Aug. 1985.

[9] C. O. Vilão, D. H. Perico, I. J. Silva, T. P. D. Homem, F. Tonidandel, and R. A. C. Bianchi, "A single camera vision system for a humanoid robot," in *Robotics: SBR-LARS Robotics Symposium and Robocontrol (SBR LARS Robocontrol), 2014 Joint Conference on*, Oct 2014, pp. 181–186.

[10] D. H. Perico, R. A. C. Bianchi, P. E. Santos, and R. L. de Mántaras, "Collaborative communication of qualitative spatial perceptions for multi-robot systems," in *Proc. 29th International Workshop on Qualitative Reasoning (IJCAI)*, New York, NY, USA, 2016, pp. 77–84.

[11] A. Farchy, S. Barrett, P. MacAlpine, and P. Stone, "Humanoid robots learning to walk faster: From the real world to simulation and back," in *Proceedings of the 2013 international conference on autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 39–46.

[12] "Gazebo - (robot simulation made easy)," http://gazebosim.org, 2016, last Accessed in: 19/05/2016.

[13] "Webots," http://www.cyberbotics.com/, 2016, last Accessed in: 19/05/2016.

[14] "Virtual robot experimentation platform," http://www.v-rep.eu/, 2016, last Accessed in: 19/05/2016.

[15] Robocup soccer simulation. RoboCup. [Online]. Available: http://wiki.robocup.org/wiki/Soccer_Simulation_League

[16] F. J. C. Montenegro. (2015) Taurasim. Taura Bots team, UFSM. [Online]. Available: https://gitlab.com/SplinterFM/TauraSim

[17] I. Afanasyev, A. Sagitov, and E. Magid, "Ros-based slam for a gazebo-simulated mobile robot in image-based 3d model of indoor environment," in *Advanced Concepts for Intelligent Vision Systems*. Springer, 2015, pp. 273–283.

[18] A. Staranowicz and G. L. Mariottini, "A survey and comparison of commercial and open-source robotic simulator software," in *Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments*. ACM, 2011, p. 56.

[19] Y. Xu and H. Vatankhah, "Simspark: An open source robot simulator developed by the robocup community," in *RoboCup 2013: Robot World Cup XVII*. Springer, 2013, pp. 632–639.

[20] Pygame. [Online]. Available: http://www.pygame.org/wiki/about

[21] I. Ha, Y. Tamura, H. Asama, J. Han, and D. W. Hong, "Development of open humanoid platform DARwIn-OP," in *SICE Annual Conference 2011*, 2011, pp. 2178–2181.