Comparative Analysis Between Gazebo and V-REP Robotic Simulators

Lucas Nogueira
School of Electrical and
Computer Engineering
Universidade de Campinas
Email: lucas_afonso@hotmail.com

***Abstract*—Robotic simulators are normally used in the design and testing of control algorithms for different platforms. But they can also be seen in a more broad context of Artificial Creatures. This a natural development of the cognitive paradigm proposed by Brooks. Related to this, in recent years the Robot Operating System (ROS) has emerged as a *de facto* standard for robot software architecture. I selected two of the best-known simulators and performed a comparison between them, evaluating the strengths and weaknesses for each one, and how well to these simulators interact with ROS, based on an experiment with a Pioneer 2DX robot.**

***Keywords*—robotic simulators, ROS, Gazebo, V-REP, cognitive science**

I. Introduction

Rodney Brooks proposed in his article 1990 "Elephants don't play chess" [1] that the "world is its best model" and that cognition should be studied in an embodied context, where the creature had temporal interactions with the world. This led to the development of cognitive robotics, where cognitive science theories are tested and validated on robotic platforms. More specifically, Brooks created the subsumption architecture that, in his words, resulted in "Intelligence without Representation" [2]- the title of another article of his.

It is in this context of robotics as an tool for cognitive science that I try to introduce two robotics simulators and a robotic framework that can be used together in order to develop, test and validate different approaches to cognition and it components, such as perception, memory, behavior and etc. The two simulators chosen for the comparison are Gazebo, an open-source solution maintained by Open Source Robotics Foundation, and V-REP and commercial - with an free educational version - solution provided by Coppelia Robotics. These two were chosen based on the result described in [3], where the authors did a survey on the use of robotic simulators by the community. They found that Gazebo was the most-known simulator and that V-REP was the best rated.

Robotic simulator build the bridge between artificial creatures theory and robotics. They provide an additional guarantee that the cognitive framework developed can be applied to real robots, with minor adjustments needed. This is can be very useful to the cognitive scientist that wants to put away the criticism of dealing with toy problems [1]. This paper aims to introduce the two simulators so that the prospective reader might choose among them the one the better suits his needs. In Section II, I present why is the Robot Operating System (ROS) of great importance in the evaluation of this simulators. In Section III, some details of both simulators are presented. In Section IV, I describe the experiment that was used to compare the simulators. In Section V, I write the results of such comparison. Finally, in Section VI, I present my conclusions.

II. ROS as a Cognitive Architecture

In this paper, I compare two simulators with the goal of introducing to the cognitive scientist these tools that might, and should, be used in the process of testing and validating cognitive theories, mainly within the Situated and Embodied Cognition and Dynamicist cognitive paradigms. I also argue here that the Robot Operating System (ROS), is another tool that should go hand-in-hand with the robotic simulators. ROS has many of the features that have been said to define a cognitive architecture. In this section, I point out these features.

Fig.1 shows an example of how ROS modules might be deployed in a simulation scenario. There are five robots running in parallel, represented by the "pioneer" boxes. In each of these boxes, it is possible to observe the sensor data (scan) entering the node that performs the control using an multi-layer perceptron, and the actuator commands (cmd_vel), that are produced by the MLP nodes. In this graph, it is also possible to see the Gazebo node, which provides the sensor readings and receives the commanded velocity. It is not difficult to recognize here an typical dynamicist cognitive cycle. Even more so because in dynamicism, the environment is considered a part of the cognition process, and in a ROS/Gazebo simulation, the environment is exactly another software component.

It is also possible to use Fig.1 to analyse Sun's definition of cognitive architectures, as can be found on his Desiderata [4]. In this work, he says:

> "A cognitive architecture provides a concrete framework for more detailed modeling of cognitive phenomena, through specifying essential structures, divisions of modules, relations between modules."

This definition highlights the importance of the modular construction of the overall structure of the cognitive phenomena. Indeed, ROS was designed by its creators to be "modular at a fine-grained scale" [5]. This can be observed in the figure, which shows, how different models interact together. In ROS, these modules are called nodes, and are able to function independently of each other. This allow for a modular construction of complex systems - much more complex than the ones showed in this paper. This complex systems can show all the standard modules in cognitive architecture, like learning, action selection and perception.

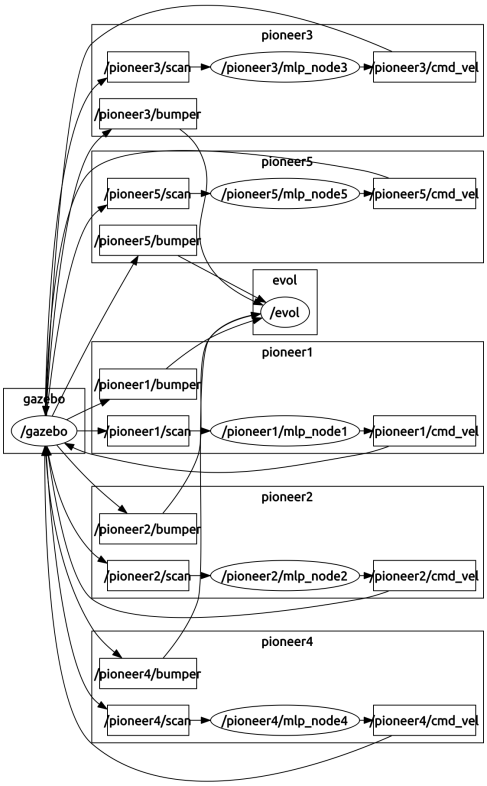

Fig. 1. ROS Nodes and Topics graph for the genetic algorithm assignment

In a sense, one can say that ROS is the cognitive architecture for the dynamicist cognitive scientist. It is a concrete framework that allows for the implementation of cognitive phenomena in robots. Also, when it comes to how it deal with crucial dichotomies like the ones proposed by [4], it will generally take the side of the dynamicist view. For example, ROS uses mostly subsymbolic processing - numeric but typed messages. Also, there is usually little formal definition of the modules in a high-level concept - like learning or action selection. There is not to say that these concepts are not present. It is just that they will usually be an emergent and/or implicit property of the system.

For these reasons, and the fact that ROS is quickly becoming a standard in robotics research and industry, I consider ROS integration one of the key criterion to use when comparing Gazebo and V-REP simulators. ROS is already proving itself in the robotics field - which is implicitly about artificial cognition - but the point that is being made here is that it can also be used for more explicit work in classical cognitive science.

III. The Simulators

A. V-REP

V-REP is a robotic simulators developed by Coppelia Robotics [6], which are based on Zurich, Switzerland. It is a commercial software, that can be obtained for free in its educational version. V-REP has support for Windows, Linux and Mac operating systems. It is possible to use 7 different programming languages with V-REP, the default language being Lua.

It also provides a lot more features, the following being a non-exhaustive list:

- Mesh editing.
- Interaction with the virtual environment during simulation.
- Physics Engines: ODE, Bullet and Vortex.

B. Gazebo

As stated in its website, Gazebo started as a project in the University of Southern California. Later on, it was integrated into ROS framework by John Hsu, who was a senior researcher at Willow Garage, ROS original maintainer. Since then, Gazebo has been maintained by Open Source Robotics Foundation, which is a Willow Garage spin-off, and the same maintainer that takes care of ROS. Gazebo is a completely open-source project, available to anyone under the Apache 2.0 license. Gazebo only runs on Linux, but support for Windows is planned to happen in next versions. A short list of features is presented:

- Cloud Simulation, the possibility to run the simulation on an online server.
- Physics Engines: ODE, Bullet, Simbody and Dart.

IV. The Experiment

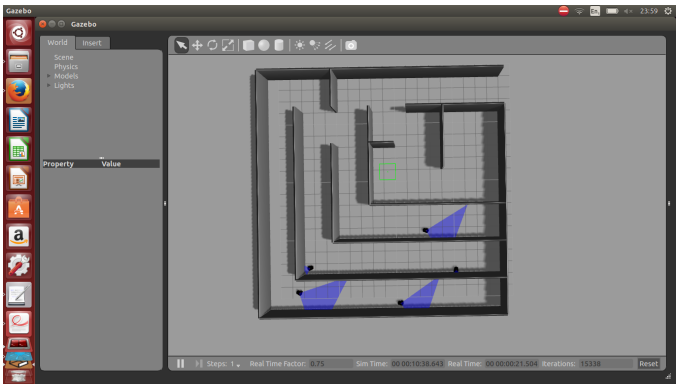

Fig. 2. Gazebo Simulation Interface

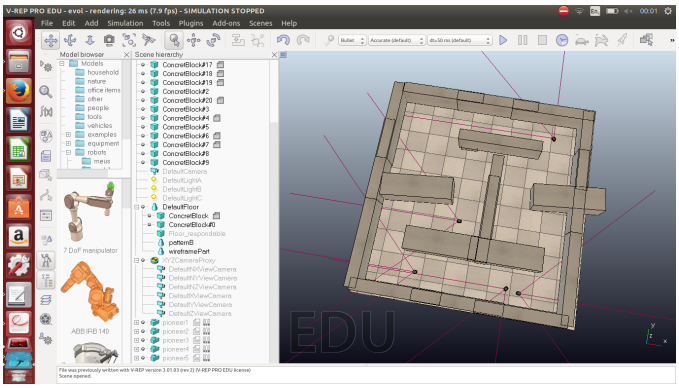

Fig. 3. V-REP Simulation Interface

In order to compare Gazebo and V-REP, I implemented the same experiment in both of them. The experiment was taken from an assignment of an Artificial Intelligence class taught by Professor Fernando Von Zuben [7]. In this assignment, the

student is asked to create an robot controller using both fuzzy logic and a multi-layer perceptron. In the original assignment, one has to actually design not only the control algorithms, but also the environment simulation. For this, the robot should be treated as a point in space moving in a labyrinth composed by walls in a 2d plane. The robot has three distance sensors - one facing forward, and the other two forming a 45 degree angle with the central sensor on each size. The forward velocity of the robot should remain constant in the simulation, and the control algorithms has to determine the best angular velocity for the robot, at each moment.

Instead of creating both the control system and the simulated world, I integrated ROS with V-REP and Gazebo, replacing the idea of a theoretical point robot with a virtual 3D robot model. The model chosen was the Pioneer 2DX, as it is a well-known robot and it was already in the model database of both simulators.

In the fuzzy logic part of the experiment, the control system has a fuzzy rule set made up of 36 propositions. Each rule is made of 3 antecedents - one for each proximity sensor - and one consequent - relating to the angular velocity. The central proximity sensor range is divided in 4 fuzzy granules, and the ones on the side have 3 granules each, thus making up for 36 possible combinations. The consequent has 4 granules. The purpose of the simulation is to determine the best consequent for each rule.

For the second part of the experiment, the control system used an Multi-Layer Perceptron, with 3 nodes in the middle-layer. The task was to find the best weights using and genetic algorithm. This was fundamentally different from the first part because it required multiple runs of the simulation using different set of weights. This implied in some level of programmatic control over the simulation environment, which was used as one of the criteria for the comparison - see next section.

As suggested in the assignment text, I used a population of 5 robots. Each one of them was positioned in a way that it tested a different movement challenge – namely hard/soft turn right/left, and going straight. These robots were controlled with different control systems running in parallel, but using the same set of weight at each time. These set of weights were used as the genotype in my genetic algorithms. The fitness used was the number of robots that hit the wall in each 10 second simulation run. So, if the fitness was 0, it meant that no robot had hit the wall, and that the weight set was successful in controlling the robots.

Both parts of the experiment were implemented in both simulators, thus allowing for a comparison between their features. They were performed on a Ubuntu 14.04 operating system, with ROS Indigo, Gazebo 2.2, and V-REP 3.1.3 PRO EDU, which is free for educational purposes. Even tough there's already a 4.0 version of Gazebo, the 2.2 is the one recommended to use alongside ROS Indigo. The comparison criteria chosen was the following:

ROS Integration
How easy it was to use the existing ROS framework with the simulator.

World Modelling
How easy it was to create world models like the labyrinth.

Robot Model Modifications
How easy it was to modify the robot model to add sensors and plugins.

Programmatic Control
How easy it was to control the simulation environmental using programming languages.

CPU Use
How much CPU power does the simulator needs.

V. Comparison

A. ROS Integration

Gazebo is the default simulator used in ROS framework. Although they're separated projects, there is a package for Gazebo in ROS official repository (*ros-indigo-gazebo-ros* [8]). This is a package kept by Gazebo maintainers themselves, the Open Source Robotics Foundation. It contains plugins that interface ROS and Gazebo. These plugins can be attached to objects in the simulator scene, and provide easy ROS communication methods, such as topics - both published and subscribed by Gazebo - and services. Packaging Gazebo as an ROS node also allows for it to be easily integrated into ROS default method for running large and complex systems, called launchfiles.

In the experiment, the Pioneer robot used a Differential Drive plugin, which listened for cmd_vel ROS messages and used it to apply appropriate commands to the robot's motors. Another plugin used was the one attached to the Hokuyo 2d laser scan model. It was used to publish a sensor_msgs/LaserScan message that contained the readings from the simulated sensor. This messages were then used by the control node.

V-REP doesn't have a native ROS node for it. This means that it is not yet possible to run it as a part of a ROS system in a single launchfile, but instead alongside it, in another Linux terminal. This is not much of a problem. On the other hand, V-REP does offer a default ROS plugin that can be used in V-REP Lua scripts for creating ROS publishers and subscribers. More useful than that, however, is the package created by the research group Lagadic from INRIA institute in France. This package, found in [9], tries to replicate the features of the *gazebo_ros* package. Likewise, it also provides a Differential Drive functionality on its manipulator handler plugin. For the proximity sensors, however, it was necessary to use V-REP Lua scripts to create publishers to transmit the sensor readings onto a ROS topic.

Gazebo and ROS have a strong relationship, being both open-source they can profit from each other developer's community. Therefore, in this criterion Gazebo has a clear advantage, mainly because even tough they are separate projects, they have a history in common, and Gazebo development cycle is planned from the start thinking about integration with ROS versions. Despite this proximity, V-REP isn't far behind. They offer an extensive API to access all of its functionality from any code, and they have integrated some ROS specific features like services and topics subscribing and publishing.

The key in this comparison is to realize that Gazebo and ROS already have a large base of community-developed plugins and code. V-REP, on the other hand, does provide the same capabilities, but doesn't have the same amount of ready-to-use components. For this reason, Gazebo is one step ahead on the matter of ROS integration.

B. World Modelling

In this criterion, I take into account how easy it is to create a world - that is, an environment scene in which the robots will move - from scratch on each simulator. The easiest way considered here is that of dragging and dropping - i.e, visual editing. Easy visualization of the scene graph and objects properties is also important.

V-REP offers all of the above natively. It comes with a lot of models that can be easily inserted in the scene. These models range from infrastructure objects like walls and doors, to furniture, and even terrain models. There's also an easy-to-use scene graph visualization, where all objects in the scene can be accessed and have all of its properties inspected and modified.

In this point, Gazebo is far behind. It does not offer many world modelling features out-of-the-box. It does provide a building editor which is very practical to design mazes and basic infrastructure. It also offers three simple geometric shapes to be inserted in the scene - a sphere, a cube and a cylinder. Lastly, Gazebo provides access to an online model database consisting of community developed models. This is one of its strong points, but this database is somewhat unorganized.

Editing models, however, is not possible inside Gazebo, for this, one has to use external 3d-modeling tools like Blender or Google Sketchup to draw the models, and then import them to Gazebo format. Gazebo uses a XML-based format called SDF. In order to use Gazebo in its full capacities, it is highly recommended that the user learns how does SDF works, and its relation to ROS URDF, which is a similar format used by ROS. V-REP has an URDF import tool bundled-in.

What I found in this criterion is that V-REP offers more user-friendly features for world modelling, that doesn't require any deep knowledge of XML. This is great for quick prototyping of simulations setups and even more complex use cases. If instead, one uses Gazebo, it will be necessary to dig deeper into SDF specifications in order to build any non-basic setup. After the learning curve has been mastered, however, it will be possible to create very complex simulations.

C. Robot Model Modifications

Much of the same analysis in the previous section also applies here. The reason for this is that there isn't much to tell a environment model - like a chair - from a robot model, except for the fact that the robot model will have actuators on some of its parts, and possibly sensors. So what we are really evaluating in this part is how easy it is to add sensors and actuators to a robot model.

As discussed before, in order to modify a model in Gazebo, it is necessary to use the SDF files. In my experiment, There was a need to integrate proximity sensors to a robot. I used Pioneer 2DX model available in Gazebo online database as a starting point. I also used a Hokuyo laser scan model from the same database. Even tough both models are already available, there is no way of combining them into a single model from inside Gazebo. This had to be done by editing the appropriate SDF files, which wasn't a trivial task at first.

In the sensing part, the Hokuyo model came with a ROS plugin that published the readings onto a ROS topic. This plugin - and Gazebo/ROS plugins in general - is quite practical. Due to the nature of the assignment, I had to read only three distances, spaced 45 degree between them. This options were easily available inside the SDF definition of the plugin. In the actuation part, the Pioneer model came with a Differential Drive plugin. This plugin reads ROS cmd_vel topics - the default topic for commanded velocities - and calculates how much torque needs to be applied to the robots wheels.

In V-REP, there is no need to use XML-based files, although there are tools to import such models. I also used an existing model of a Pioneer 2dx that is available by default on V-REP local model database. V-REP provides a variety of different sensors to be readily inserted in any existing model. One of them, the proximity sensor, was used here. Three of them were placed on top of the Pioneer robot, according to assignment specifications. All of this was done from inside V-REP. Later, the associated Lua scripts were modified to add ROS publishers that published the sensor's readings to ROS topics. This only took two lines of code. For the actuators, the vrep_ros_bridge ROS package was used. This package searches the scene graph for a set of objects that can be controlled, and creates the ROS plugins needed to control them. In this case, the plugin created controls the Robots wheels, in much the same way as the Differential Drive plugin for Gazebo does.

Again, the advantage of V-REP is it user-friendly features. To be able to modify an model from inside the simulator is very useful and practical. It is true that one of the tools used, the vrep_ros_bridge, is not maintained by Coppelia Robotics itself. But still, that is exactly what should happen more in V-REP. Gazebo, on the other hand, does not provide an simple graphical way of editing model, and one has to edit text files. And most of its sensors are solutions that were developed by the developer's community, like the Hokuyo model. This results in a lot of very useful plugins readily available, something that V-REP somewhat lacks.

D. Programmatic Control

On the second part of the experiment, the simulation was a part of a genetic algorithm, and was used to assess the fitness of each set of weights in a neural network. For this, it was necessary to be able to control the state of the simulation from another piece of code. The access and control over the simulation environments as a usable code library is what is considered here the criterion of programmatic control.

On both Gazebo and V-REP, all of the programmatic control that was needed for the experiment was available as ROS services provided by the ROS nodes associated with each simulator. This services were: start and stop the simulation, get the pose of the robot models in the scene and set the pose of the models. In Gazebo there was a need for getting and setting the pose of the robots because there was a problem with the service that reset the simulation, in which the differential drive plugins stopped working. So instead I used the services that stopped and start the physics, and reset the position of the robots in each fitness assessment. This was not needed in V-REP because the reset simulation service also resetted the position of the robots.

ROS Services are easily called via a ROS node code. The library used is part of ROS, and doesn't belong to any of the simulators specifically. This show that ROS can be used to abstract low-level implementation details, thus facilitating the

work of the user. However, if needed, both Gazebo and V-REP do offer an code API capable of completely controlling its internal simulation variables. The documentation pertaining to each can be found in [10] and [11]. Therefore, it is fair to say that in this criterion there is nothing to distinguish between Gazebo and V-REP.

E. CPU usage

In this section we compare how much CPU power does each simulator requires. The computer used in this experiment was a P Pavilion dv6 Notebook PC with an 8GB RAM memory and a Quad-Core AMD A8-3510MX APU with Radeon HD Graphics. I used the second part of the assignment, where there are multiple simulation iterations, with 5 robots in each, to compare V-REP and Gazebo. It was chosen because it was the most hardware-demanding setup available. Each simulation setup consisted of a ROS master node process, 5 ROS control nodes processes for each robot, one ROS node process for the genetic algorithm node, and 1 process for the simulator. The system monitoring tool *glances* was used.

On the V-REP simulation, all the process except for the simulator maintained a constant CPU usage. Each ROS control node used 6,3% of one CPU. The genetic algorithm node took only 0.7% and the ROS Master node used 130% - consider here that 100% corresponds to one CPU core, the computer used has 4 cores. The simulator process varied its CPU usage due to the fact that it was regularly having its simulation beign switched on and off. When the simulation was off, the CPU usage was about 32%, and when the simulation was on this number became approximately 105%.

The existing process in the Gazebo simulation are all the same, except for Gazebo itself, which is divided between an server process and a client. In this configuration, there was an increase in the CPU usage of the ROS control nodes to 12.8%. The ROS Master kept a constant level of 90% - smaller than in the previous scenario. The genetic algorithm process also took about 10% of CPU, which is considerably more than with V-REP. But the biggest difference was with Gazebo itself. The server part consumed 120% of CPU power, constantly, while the client part - which is the user interface, consumed around 27%.

From this, we gather that, in total, the V-REP simulation configuration used about 267% of CPU power, in its most demanding stage. The Gazebo setup took 311% of CPU power constantly. This shows that Gazebo is more hardware demanding. This result is in accordance with what is stated in [**?**]. It also interesting to point out that while V-REP simulation was divided between active and non-active stages, the Gazebo simulation was constantly active. This is probably due to the fact that Gazebo responds faster to the ROS services used to control the simulation. So, this might account for Gazebo higher values of CPU usage.

VI. Conclusion

In this paper, I tried to introduce to the prospective cognitive scientist tools that are currently being used in the robotics industry that could have a place in cognitive research. This claim is build upon the dynamicist paradigm of cognition, as it is only natural for robotics.

I argued that ROS, a software framework and middleware for developing robotic applications can be seen as a cognitive architecture because it is a concrete framework that implements cognitive phenomena, and has a structure composed of different modules that may implement different mental functions such as learning, memory, perception and action selection. However, most of this functions are implemented in an implicit and distributed fashion using subsymbolic (numeric) representations, which is in accordance to the dynamicist point-of-view.

I also compared V-REP and Gazebo robotic simulators using a basic experiment in robot control using fuzzy logic and evolutionary robotics. However basic, these experiments allowed for a comparison of the features offered by both simulators. The conclusion reached was that V-REP is a more intuitive and user-friendly simulator, and packs more features. Gazebo is more integrated into ROS framework and is an open-source solution which means it allows for complete control over the simulator. But it needs a number of external tools to match up with V-REP functionalities. Also, Gazebo is more hardware-demanding than V-REP. So, the cognitive scientist should have a better chance of implementing and validating their cognitive theories using V-REP than Gazebo, specially if he - or she - does not have an strong background in robotics.

References

[1] R. A. Brooks, "Elephants don't play chess," *Robotics and Autonomous Systems*, vol. 6, pp. 3–15, 1990.

[2] R. Brooks, "Intelligence without representation," *Artificial Intelligence*, vol. 47, pp. 139–159, 1991.

[3] S. Ivaldi, V. Padois, and F. Nori, "Tools for dynamics simulation of robots: a survey based on user feedback," *CoRR*, vol. abs/1402.7050, 2014. [Online]. Available: http://arxiv.org/abs/1402.7050

[4] R. Sun, "Desiderata for cognitive architectures," *Philosophical Psychology*, vol. 17, pp. 341–373, 2004.

[5] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[6] C. Robotics. (2014) V-rep - virtual robot experimentation platform. [Online]. Available: http://www.coppeliarobotics.com/

[7] F. J. Von Zuben. (2014) Ea072 - exercicios conceituais e computacionais 2 (ecc2). [Online]. Available: http://tinyurl.com/vonzubenecc2

[8] N. C. D. Hsu, John; Koenig. (2014) gazebo_ros_pkgs - ros wiki. [Online]. Available: http://wiki.ros.org/gazebo_ros_pkgs

[9] G. Spica, Riccardo ; Claudio. (2014) vrep_ros_bridge - ros wiki. [Online]. Available: http://wiki.ros.org/vrep_ros_bridge

[10] C. Robotics. (2014) Remote api functions (c/c++). [Online]. Available: http://tinyurl.com/vrepApi

[11] O. S. R. Foundation. (2014) Gazebo apis. [Online]. Available: http://gazebosim.org/api.html