Abstract— This paper discusses the pros and cons of using 3D simulators for testing the autonomous behavior of mobile robots in indoor environments. Major contribution of the paper is the discussion about which problems that can be faced using the simulator and those that cannot. We present the integration and calibration of a real non-commercial robot in a simulator, the characterization of the errors in sensing, navigation; and manipulation; and how these errors would impact in the real performance of the robot. The experimental support of the claims made in the paper has been developed using the gazebo simulator. RoCKIn competition rulebook defined the indoor restrictions.

Keywords - Robot simulation; indoor simulation; error calibration; simulator performance

I. INTRODUCTION

Simulators are used in robotics mainly to test the software controlling hardware platforms before their deployment in real commercial or research environments. It has been argued that simulator can speed up the development process, but it has not really well studied.

In this paper we will focused in research robots simulation, using robotic competitions as testing environments. Robotics research community has been using competitions (AAAI, RoboCup, DARPA Grand Challenge, or more recently RoCKIn) to foster research [1], and as a mean to evaluate the progress made [2]. Many competitions have provided simulators in order to make easier for new research groups to join their communities.

In this paper we will use “RoCKIn Challenge” environment to illustrate how simulators should be analyzed. RoCKIn [3] is a EU-FP7 project for fostering scientific progress in cognitive systems and robotics. It proposes two competitions: the @home and @work environments. We have chosen the RoCKIn@Home because is focused in robot benchmarking from bottom to top, defining different tests for each major subsystem of the robot (perception, manipulation, etc.) as well as for validating high level tasks.

RoCKIn provides a simulated environment for their competitions based on Webots [4] simulator. But we have chosen Gazebo [5], [6], [7] because it does not require commercial license, it is supported by the Open Source Robotics Foundation (OSRF) and its community, and we have previously defined our robot model using the Unified Robot Description Format (URDF) used by Gazebo.

The official simulated environment for RoCKIn competition and our version for gazebo simulator are presented in figure 1. The differences are only aesthetic: textures applied in floor, doors, and walls.

Motivation of this paper rose when moving from the simulator to the real robot. The performance of our software became greatly reduced and the same happened to many other teams. The reason is that simulators are not able to fully simulate their robots complexity. However, limited effort has been made to measure the accuracy of these simulators (they are simply characterized as “very realistic”), or to provide guides to researchers on how to move from simulators to real robots.

Craighead et al [8] concluded that it was no longer necessary to build robotic simulators from the ground up. There are many available simulators and game engines that can be used to simulate a robot with “high physical and functional fidelity”, but no evaluation of that fidelity is proposed in their survey.

Kramer and Scheutz [9] evaluated and compared nine different Robotic Development Environments (RDE) according to four domains: specification, platform support, infrastructure and implementation, but they did not evaluated specifically the simulators.

The work presented in this paper faces the evaluation in a different way. We will choose a simulator and will analyze different problems arising, and then we will try to generalize these problems in a survey that could help other researchers.

The survey made by Harris and Conrad [10] introduced another issue that we will face in this paper: “code developed in many of these simulators requires expensive hardware when porting to real robotics systems. The middleware that
is required to run on actual hardware is often too taxing for smaller, cheaper systems. There simply isn't a very good three-dimensional robotics simulator for custom robotic systems designed on a tight budget.”

More details about the robot we have designed for this competition can be found in the next section, as well as the details of the simulated environment. Section III presents a structured taxonomy of the problems arising when moving from simulated to real robots. Section IV discusses these problems, and finally section V presents some conclusions and further work.

II. SIMULATION ELEMENTS: ROBOT AND ENVIRONMENT

This section describes the two basic elements of any simulation: the simulated entity (the robot), and the environment where it is simulated. In our case we would like to analyze the accuracy of the simulator for three of the major modules of our robot: perception, navigation, and manipulation. We will not face other relevant modules, as for instance the speech recognition subsystem, but we think that the ones chosen are more than enough to validate our claims.

Our robot has two RGB-D sensors and one arm of 4 DoF mounted on a 120 cm tall chassis. In the base of the chassis there is a mandatory “button box” for emergency stop, and a range sensor module containing 5 ultrasonic sensors (operational range from 20 cm to 765 cm, and 1cm resolution). The computational power is provided by a laptop computer whose screen has been placed in the top of the robot. The robot is propelled by a two-wheel differential driving system and two auxiliary casters wheels.

This design is intended for a technically-optimized disposition of the hardware components of the robot, nor for an anthropomorphic or zoomorphic morphology classically pursued in other robots. The main reason for choosing this design is being able to adapt the robot (adjustable chassis) to dynamic environments. A zoomorphic design does not allow physical assistance or interaction in high positions.

The display could show a virtual face to give the robot an anthropomorphic look, but as it was added to provide augmented reality assistance [11] no faces are displayed.

All these elements have been built into a simulated model that can be inserted in the RoCKIn simulated environment as shown in figure 2.

The competition environment has been precisely defined by the RoCKIn committee. The dimensions of the real environment are superimposed over the simulated scenario that we have designed for Gazebo simulator and presented in figure 3. In this figure we show the environment without any furniture. It was generated with the building editor feature of Gazebo using the real values proposed by the organization.

III. EXPERIMENTAL SETUP

In order to test the differences between the real robot and the simulated one we made two experiments focused on perception and manipulation issues. Both experiments were performed in the real and simulated environments to test the behavior of the robot under the assumed same conditions.

These experiments were made using a GNU-Linux regular PC based on an Intel ® Core ™ i5-4200M CPU @ 2.50GHz, 6GiB RAM, and Nvidia GeForce GTX 765M. This computer runs the same control software in the simulated and real environment, and the simulator itself in those tests who need it.
A. Perception

In this section we are describing how we compared the visual perception systems. First, we characterize the real camera and the parameters used in the simulator. Then, we describe the off-the-self software library used in this test for object recognition. Finally, we analyze the application behavior and its performance during the task.

The real robot has two RGB-D cameras on-board for environment perception. The cameras field of view is 58°, 45° and 70° (horizontal, vertical, and diagonal). The camera allows a SXGA resolution 1280x1024 30 fps but we configured it to basic VGA 640x480. The physical dimensions of the camera are 18 x 3.5 x 5 cm. The camera resolution was the same in both the real and the simulated environment.

The software used in this test was FindObject developed by Mathieu Labbé [12]. This package provides different feature detectors and descriptors, such as SIFT, SURF, or FAST implemented using OpenCV. The program receives an image and extracts different features from the image (the number of features can be parametrized), the user then selects the cluster of features that represents an object. This cluster is used by the application to search similar clusters in every new frame.

The goal of the first test was to evaluate the object recognition capabilities of the robot. Two objects were modeled in the simulator and used in the experiments: a soda can and a beans can. We placed each one of them on a table at two different positions, first at 55 cm and then at 80 cm from the robot (figures 4 and 5 show the simulated robot and the real one). The textures applied in the simulator for both cans were based on the texture of the real ones, that was captured using a digital camera and then integrated in the Gazebo simulator.

We used the same software and the same parametrization for object detection in both cases. The parameters used were: contrastThreshold (0.04), edgethreshold (10), and nOctaveLayers(3), nFeatures (0), sigma (1.6).

B. Manipulation

In this section we are describing how we compared the manipulation capabilities. First we describe the hardware, and then the we explain how we measure the accuracy and smoothness of the simulated arm vs the real one.

The arm was a “do-it-yourself” (DIY solution). We use the commercial Bioloid educational kit that is based in Dynamixel AX-12 servos to build a basic arm. The arm has 4 DoF and it is mounted in the front part of the chassis. It also has a gripper. Its maximum achievable distance is 62.5 cm. in its working space.

In order to test the simulation we defined a choreography that moves the arm through five positions. The choreography is shown in figure 6 using the simulated robot. The positions were: the zero position (start point), two defined along the X-axis and two more in positive Z-axis and positive Y-axis. The robot arm reaches the maximum achievable distance in each one, and return to zero position. Figure 7 shows the arm reaching the positive Y-axis position.

![Figure 4. Find-Object Application running under Gazebo simulator conditions. The detection and recognition is performed from 55cm.](image)

![Figure 5. Object recognition in real world from 55cm.](image)

<table>
<thead>
<tr>
<th>55 cm Test</th>
<th>Total FPI</th>
<th>TFO</th>
<th>WTFO</th>
<th>OTFO</th>
<th>SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulator</td>
<td>85</td>
<td>70</td>
<td>29</td>
<td>21</td>
<td>9/10</td>
</tr>
<tr>
<td>Real</td>
<td>130</td>
<td>70</td>
<td>10</td>
<td>22</td>
<td>2/10</td>
</tr>
</tbody>
</table>

We measured several parameters of the object recognition performance. We carried out a set of 10 trials of 5 seconds each. TABLE I. shows the average measure for each of the following parameters:

- Total features per image (Total FPI)
- Total features which defines the object (TFO)
- If we have a detection, how many features fall within the TFO (WTFO)
- If we have a detection, how many features fall outside the TFO (OTFO)
- Success rate (SR) in each trial of 5 seconds.
The arm reaches the same positions in the real world and in the simulator, but the inertial behavior is different in the Gazebo simulator. The robot makes rough movements and even bounces during the tasks performance. In both cases we used the same speed.

We made five trials and found that the results were very similar. The only significant different was the time used: in the simulator each task took 19 seconds in average, using the real arm it took 21 seconds in average.

IV. DISCUSSION

Simulators offer advantages and disadvantages for testing robotics systems. We analyzed three concepts for evaluating the experience of using a simulated robot: the environment, the robot as hardware device, and the robot as software solution.

The “environment” is the set of constraints that should be applied into the virtual world. Some of these constrains are:

• The material properties: regular or anisotropic friction, rigid or soft objects.
• The ambient light: light (natural or artificial) change the object signature.
• Damping: how is the robot decreasing, the velocity.
• Bouncing of dynamic objects.

For analyzing the robot hardware and design we took into account that:

• Battery: cannot supply an infinite current.
• Sensors and Actuators: We can build perfect devices in simulators, but they do not exist in real world.
• Physical touch to robot buttons: which behavior is triggered, how we can press them.

• Design: we should design realistic models then we can spawn it in the virtual environment and build it trough rapid prototyping techniques.

Finally we analyzed the robot as a software using the results from the experimental tests:

• PC performance: in simulations to render the robot and the environment requires a computational cost and affects to other solutions behavior.
• The ROS solution uses extensively the local network, any latency problem decreases the solution performance. Encrypted networks decrease the system performance.

We can say that the results are similar in both environments, but the computer has to pay the computational costs. Under other circumstances the results can be different (old computer, extra algorithm, running…) for this reason we made a performance analysis of the overall system during all the tasks.

The outline of all conceptual issues and positive behaviors found during our research working with the simulator are presented in TABLE IV.

We also made a performance test on the PC to evaluate memory and CPU use. It consisted in several bash scripts to avoid overloading the system with any GUIs. The scripts recorded into a file at 1Hz the CPU and Memory status of the ten most expensive processes. The use of the CPU vary up to 400% because this PC has several cores and the hyperthreading was used.

For this particular test, the set-up and running were slightly different in the real and the simulated environments. In the simulated one we need to launch the Gazebo simulator which consists in two main components:

• Gazebo Server (gzserver): It is in charge of simulating the previously parsed environment description (sensors, physics, collisions, …)
• Gazebo Client (gzclient): It offers the GUI for visualizing the simulated environment by the Server.

Both components can be launched using a single command but for our purposes they are presented separately. TABLE II. summarizes the most expensive processes during the experiments.

find object was the most expensive process, using almost 50% of the CPU processing power. This process is always running because it is used by other behaviours of the system.

For instance, if we need to start a GUI application for managing or visualizing the robot as Rviz (one of the most common in ROS environment) we will need more CPU. This process runs from around 10% of CPU until the 100% it depends of data dump to the GUI (you can visualize cloud points, robot model, maps, etc). In our examples we manage for using Rviz with basic outputs: robot model and the position of a detected object.
The results during the simulator analysis are slightly different. They are presented in TABLE III. We can see that the overall consumption is higher than 200%. In this case we can see that if we need to launch the Rviz tool the system can reach easily the 300%, in this state if we have to launch more software solutions the CPU can reach the limits and the overall system can be collapsed. Figure 8 shows clearly the CPU behavior and the status during the experience.

V. CONCLUSIONS

We have presented the issues that must be taken into account when we move our research from simulated environments to the real world. The goal of the work described in this paper was to perform an analysis of environments competitions and the problems originated from the robotic simulation.

We made experimental tests that showed us that in simulator mode the CPU limits can be reached. We also exposed the common mistakes when the researcher is testing the robot only in simulated conditions and suddenly it is necessary to test in real environments.

There are still some features in other topics that we can evaluate as navigation under maps created using the simulator or grasping. Also the speech recognition system should be tested. To do this we can record the entire corpus through the real microphone and then we can use these files.

ACKNOWLEDGMENT

The authors would like to thank the Spanish Ministry of Economy and Competitiveness for the partial support to this work under grant DPI2013-40534-R.

REFERENCES


| TABLE II. FIVE EXPENSIVE PROCESSES IN CPU CONSUMPTION DURING THE REAL TEST. |
|----------------------------------|----------------------------------|----------------------------------|
| CPU(%) | MEM (%) | Process |
| 46.1 | 3.3 | find object 2d |
| 11.5 | 1.1 | nodelet manager |
| 10.8 | 0.3 | XnSensorServer /etc/openni |
| 5.8 | 1.2 | /usr/bin/X |
| 4.8 | 3.9 | compiz |
| 22.3 | 3.2 | /opt/ros/hydro/lib/rviz/rviz |

| TABLE III. FIVE EXPENSIVE PROCESSES IN CPU CONSUMPTION DURING THE SIMULATION TEST. |
|----------------------------------|----------------------------------|----------------------------------|
| CPU(%) | MEM (%) | Process |
| 122.0 | 5.0 | Gzserver |
| 58.0 | 4.2 | gzclient name:= gazebo gui |
| 14.1 | 4.5 | Find object 2d |
| 5.8 | 1.5 | /usr/bin/X |
| 4.6 | 3.9 | compiz |
| 57.7 | 3.2 | /opt/ros/hydro/lib/rviz/rviz |
TABLE IV. EXPERIENCE OUTLINE AFTER THE RESEARCH.

<table>
<thead>
<tr>
<th></th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Environment</strong></td>
<td>We can simulate furniture and its physics</td>
<td>We need to program (physics, errors, forces)</td>
</tr>
<tr>
<td></td>
<td>We don’t need to buy furniture</td>
<td>We lose the real feedback of some objects</td>
</tr>
<tr>
<td></td>
<td>We don’t need a huge lab</td>
<td>Real light conditions differs between competition places</td>
</tr>
<tr>
<td></td>
<td>We can simulate floor friction or external forces</td>
<td></td>
</tr>
<tr>
<td><strong>Robot Hardware and</strong></td>
<td>We can test whatever sensor with given specs</td>
<td>We avoid real aleatory errors</td>
</tr>
<tr>
<td><strong>Software</strong></td>
<td>We can model sensing errors</td>
<td>We need to program errors (e.g. Gaussian Models)</td>
</tr>
<tr>
<td></td>
<td>We can increase the speed in a test</td>
<td></td>
</tr>
<tr>
<td></td>
<td>We can change this computer without robot implications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sensors and actuators can’t be broken</td>
<td></td>
</tr>
<tr>
<td></td>
<td>No networks problems (loopback)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Improves the parallel development (HW and SW)</td>
<td></td>
</tr>
</tbody>
</table>

Figure 7. Choreography defined for testing the arm performance and movements smoothness. From left to right: zero position, X-axis, Y-axis, Y-axis, Z-axis and zero position.

Figure 8. CPU percentage consumption by top five process (and rviz) during the manipulation task in real (left) and simulated (right) environments.