Online Tool for Benchmarking of Simulated Intervention Autonomous Underwater Vehicles: Evaluating Position Controllers in Changing Underwater Currents

Javier Pérez, Jorge Sales, Raúl Marín, and Pedro J. Sanz, *Senior Member, IEEE*
Computer Science and Engineering Department
University of Jaume-I
Castellón, Spain
e-mail: {japerez, salesj, rmarin, sanzp}@uji.es

***Abstract*—Benchmarking is nowadays an issue on robotic research platforms, due to the fact that it is not easy to reproduce previous experiments and knowing in detail in which real conditions other algorithms have been performed. Having a web-based tool to configure and execute benchmarks opens the door to new opportunities as the design of virtual tele-laboratories that permit the implementation of new algorithms using specific and detailed constraints. This is fundamental for designing benchmarks that allow the experiments to be made in a more scientific manner, taking into account that these experiments should be able to be reproduced again by other people under the same circumstances. In the context of underwater interventions with semi-autonomous robots, the situation gets even more interesting, specially those performed on real sea scenarios, which are expensive, and difficult to perform and reproduce. This paper presents the recent advances in the online configuration tool for benchmarking, a tool that is continuously being improved in our laboratory. Our last contribution focuses on evaluating position controllers for changing underwater currents and the possibility for the user to upload its own controllers to the benchmarking tool to get online performance results.**

Keywords–benchmarking; underwater interventions; datasets; open source simulator; online simulator.

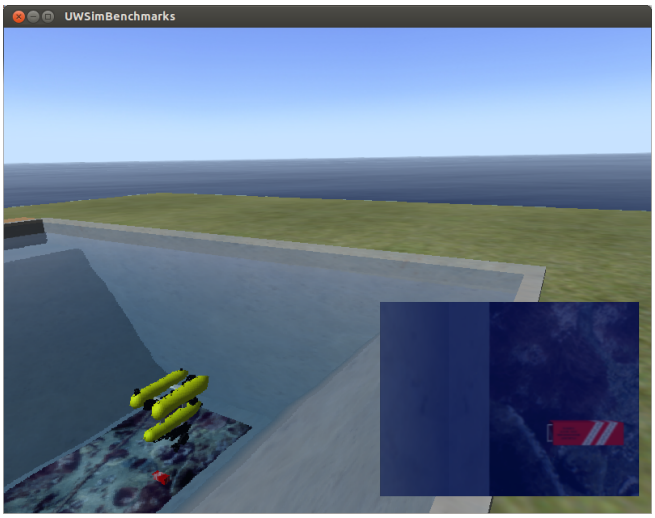

Figure 1. *Benchmark* being executed in UWSim underwater simulator: Girona500 I-AUV in water pool conditions (main scene) and visual tracker (lower right corner virtual camera) for changing visibility conditions in the scene.

I. Introduction

Underwater manipulation using I-AUV (Autonomous Underwater Vehicles for Intervention) allows the design of new applications such as the one studied at the FP7 TRIDENT project [1], where a black box from the sea bed was autonomously recovered. To accomplish this, the use of the UWSim (Underwater Simulator [2]) has been crucial, for testing, integration and also benchmarking (see Fig. 1).

There are previous simulators for underwater applications, which mainly have remained obsolete or are being used for very specific purposes [3] [4]. Moreover, the majority of the reviewed simulators has not been designed as open source, which makes difficult to improve and enhance the capabilities of the simulator. Moreover, there are other commercial simulators such as ROVSim [5], DeepWorks [6] or ROVolution [7]. However, they have been designed to train ROV pilots, which is not the objective of our research.

On the other hand, the use of simulators to define specific *benchmarks* for underwater interventions allows having a platform to better compare, under the same conditions, the efficiency of two different algorithms. Several definitions of benchmarks have been proposed, but we will use the one stated at [8]: "adds numerical evaluation of results (performance metrics) as key element. The main aspects are repeatability, independency, and unambiguity". Some previous works on *benchmarking* have been performed in other simulators such as [9] in Stage, [10] in USARSim and [11] in Webots. However, this work describes both the development of specific benchmarks and an online platform for comparing algorithms. Moreover, in the context of underwater robotics have not been provided any work of this kind before, for the best of our knowledge.

In this paper, an online platform for the design of configurable *benchmarks* is presented, that makes use of the UWSim simulator. In particular, a specific benchmark configuration to evaluate position controllers in changing underwater currents is described. The controller uses the reference input from visual tracking algorithms. The web-based user interface allows the configuration of the controller

constants, and also uploading algorithms to be executed on the platform. Moreover, different visual tracking algorithms and parameters can be selected for each benchmarking experiment.

II. Review of related benchmarking suites & toolkits

In the last years, several benchmarking suites have been developed in the field of robotics. Many of them focus purely on a specific sub-field of robotic research but, to the best of the authors' knowledge, none of them is focused on autonomous underwater vehicles.

In the grasping field, several suites have been presented such as the OpenGrasp Benchmarking suite [12]. This suite is a software environment for comparative evaluation of grasping and dexterous manipulation using OpenGrasp toolkit. It also provides a web-service that administers available benchmarks scenarios, models and benchmarking scores. The main drawback of this suite is the evaluated software needs to be written in an OpenRAVE (Open Robotics Automation Virtual Environment) plugin or it requires a glue layer.

Another interesting benchmarking suite in the field of grasping is VisGrab [13]. VisGrab is a Benchmark for Vision-Based Grasping. This suite provides a software able to evaluate vision-based grasp-generation methods. In this case website only provides results publishing.

Motion planners, trajectory tracking and path planning has been a very active research field around benchmark metrics and benchmarking suites. In [14], authors describe a generic infrastructure for benchmarking motion planners. This infrastructure allows to compare different planners with a set of measures. The key point of the contribution is the easy to compare design due to ROS (Robot Operating System) [15] MoveIt! integration.

Rawseeds [16], is a project focused precisely on benchmarking in robotics, although it's global nature has been widely used for SLAM, localization, and mapping. The Rawseeds project aim is to build benchmarking tools for robotic systems through the publication of a comprehensive, high-quality benchmarking toolkit composed of, datasets with associated ground truth, benchmark problems based on datasets and benchmark solutions for the problems. Rawseeds' Benchmarking Toolkit is mainly targeted at the problems of localization, mapping and SLAM in robotics but its use is not limited to them. Unfortunately this project lacks of an automated comparison system.

Finally there have been proposals of web-based benchmarking suites such as [17] where authors propose an interesting test-bed architecture for benchmarking of visual servoing techniques. In this work, authors suggest the use of an Internet-based architecture that allows users to upload their algorithms to be tested. The proposed system architecture of an Internet robotic system allows the upload of a Matlab code to control a robotic arm/hand by means of visual servoing techniques.

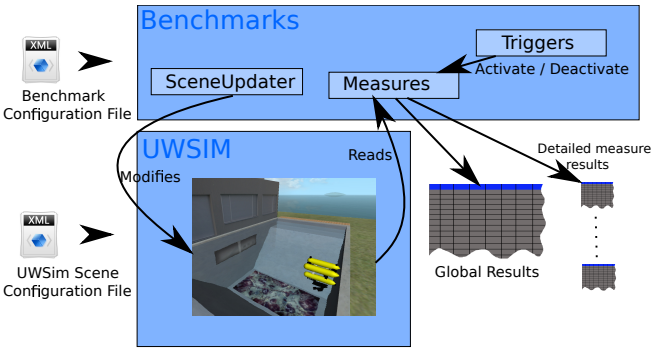

Figure 2. *Benchmarking* module flow diagram: a benchmark configuration is loaded into the benchmark module, and a scene is loaded into the simulator. Then, the benchmark module produces some results that can be logged for posterior analysis.

III. UWSim: A 3D Simulation Tool for Benchmarking and HRI

UWSim[1] [18] is an open source software tool for visualization and simulation of underwater robotic missions (see Figs. 1 and 2). The software is able to visualize underwater virtual scenarios that can be configured using standard modeling software and can be connected to external control programs by using the ROS interfaces. UWSim is currently used in different ongoing projects funded by European Commission (MORPH [19] and PANDORA [20]) in order to perform HIL (Hardware in the Loop) experiments and to reproduce real missions from the captured logs.

Recently, a *benchmarking* module for UWSim has been developed [21]. Like UWSim, this module uses ROS to interact with other external software. The ROS interface permits users to evaluate an external program which can communicate both with the simulator (which can send commands to perform a task) and with the *benchmarking* module (which can send the results or data needed for evaluation). Detailed information on how to configure and run a *benchmark* in UWSim can be found in [22].

IV. The benchmarking online excecution

A. The configuration and visualization web server

The benchmarking execution architecture can be seen in Fig. 3. The online server[2] *(robotprogramming.uji.es)* listens to web requests and serves the pages. Moreover, it also acts as an intermediary with the simulation server *(arkadia.act.uji.es)*, which is inaccessible from the outside (as it pertains to the internal network servers in the domain *uji.es* and is protected by a *firewall*), redirecting to it the calls from `rosbridge` and `mjpeg_server`. This way,

[1]Available online: http://www.irs.uji.es/uwsim
[2]Available online: http://robotprogramming.uji.es

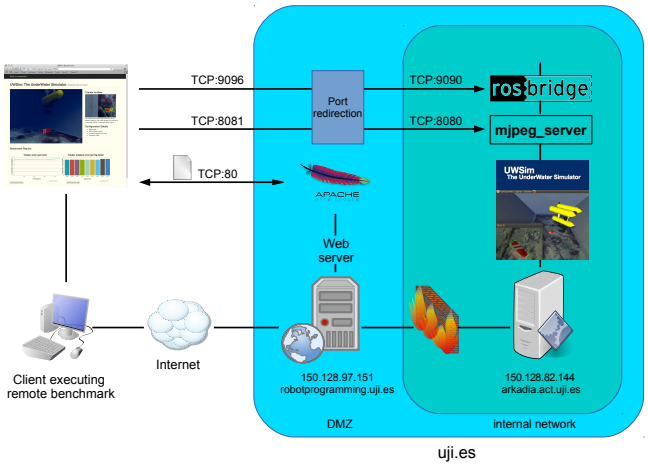

Figure 3. Interaction diagram of the configuration web service and benchmark execution.

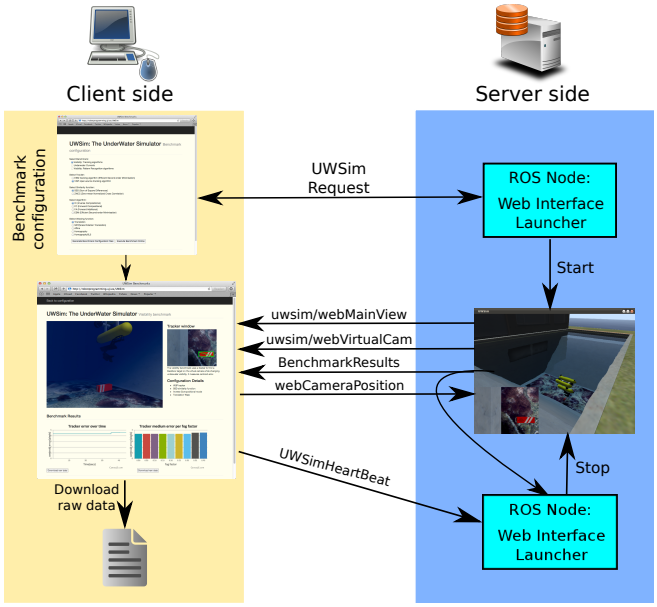

Figure 4. Networking diagram of the configuration web service and benchmark execution.

the interaction with the simulation server is transparent to the client, which only makes calls through the JavaScript interface for `rosbridge` and receives streaming video using `mjpeg_server` and results through `rosjslib`.

B. The benchmarking execution service

The functionality of the web execution service can be seen in Fig. 4. First, the user selects the desired configuration options through a form. This form allows the user to choose among different benchmarks and algorithms to evaluate. At the time of writing, visibility and current benchmarks are already functional and available online. The system is also prepared to execute pattern recognition benchmarks, but this option is not yet fully operational. Moreover, in the case of the currents benchmark, the user can provide the source code of the controller to be executed, simulated with data and evaluated by the system. In the future, the user will be able to choose if the benchmarking results and the algorithms can be published on the website, to be compared with the algorithms from other users.

Once the user has configured the desired options, the benchmark is executed online. To do that, the `/uwsimRequest` service that is called, makes use of `rosbridge`, which is a message that contains a dictionary of keys and values in the form of two vectors. Those vectors store the configuration options selected by the user, and are sent to the `webInterfaceLauncher` ROS node located in the simulation server that is responsible for executing the request. Similarly, the user can upload its own code in the request, which will be stored on the server until the execution. In this case, the code provided by the user will replace the one already existing on the server by default. Right now, it is only possible to execute code written in *Python* but it is already planned to allow execution of code written in *C++*, *Matlab* or *Simulink*.

V. The Benchmarking Web Interface

The benchmarking web interface (see Fig. 5), allows the user to configure and execute benchmarks remotely. In this interface, users can execute and monitor the benchmark execution, being able to see in real time the results that the simulator is producing. It is also possible to download raw results at any time for further processing.

A. Benchmark Configuration Screen

Besides executing benchmarks completely locally editing configuration files, see [22], it is also possible to use the benchmark configuration web tool [23]. In Fig. 5, a currents benchmark is being configured. Configuration options include the choice between multiple position controllers and trackers and many parameters such as Kp (proportional gain) and Ki (integral gain) constants for controllers or similarity functions and warps for trackers. After users have configured the experiment they are able to run the simulation online.

B. Benchmark Execution Screen

While the benchmark is being executed, users can monitor it, using the main feedback, the secondary feedback, and the results provided by the web interface. The main feedback displays the simulator view. Users can interact with it by using the mouse, and can move through the virtual scene while the benchmark is running, as they would do in the local simulator (see Fig. 6).

The secondary feedback is placed at the right part of the main simulator screen. In this feedback the most relevant information about the benchmark being executed is displayed.

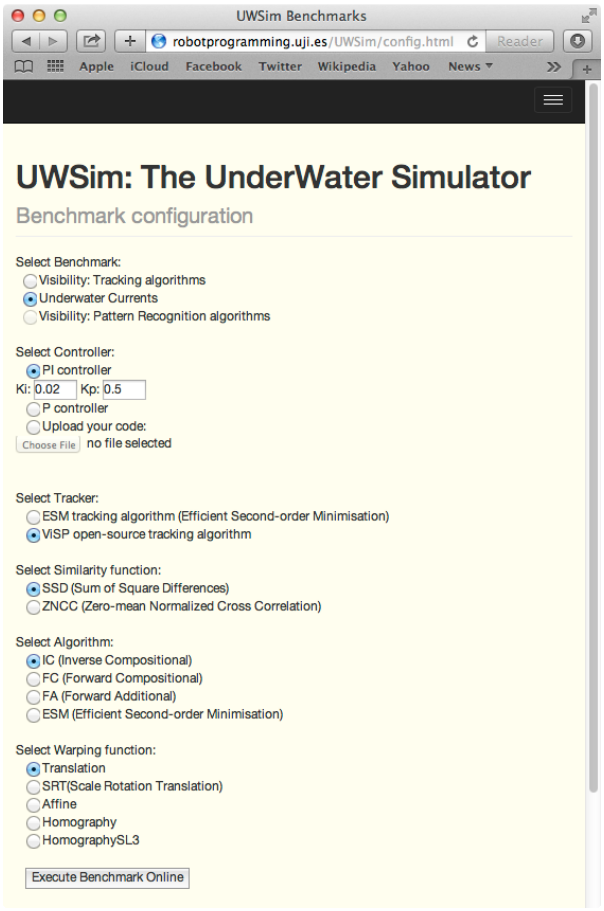

Figure 5. Benchmarking configuration web interface for UWSim: underwater currents benchmark configuration example.

In the case of the currents benchmark, the camera used by the vehicle to position itself respect to the target is shown. As in the visibility case, a green box displays where the tracker thinks the target is. Furthermore, the chosen configuration options are shown on the screen. As this section is reserved for the benchmark output, it will show the most relevant information depending on the executed benchmark.

Finally, the results zone shows real time charts created by `canvasJS`. `canvasJS` is a powerful tool that allows data charting on real time using HTML5, assuring compatibility with different devices such as tablets, smartphones, etc. In the case of the underwater currents benchmark, it shows charts using the distance from the vehicle to target position, and splitted charts using X axis error and Y axis error. Depending on the selected benchmark, most relevant charts are automatically displayed. Nevertheless, in future releases, this output will be available to be choosen by the users. In addition, raw data can be downloaded at any time in text format by pressing the *"Download raw data"* button. This feature allows to download the complete results data for further analysis.

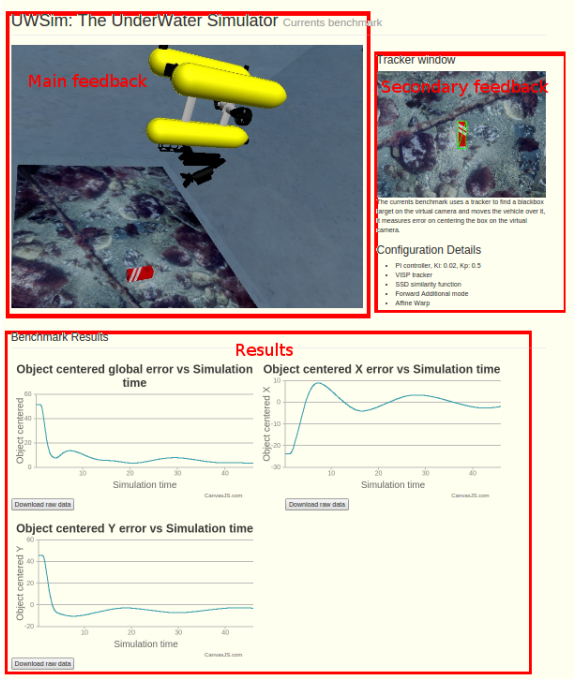

Figure 6. Benchmark execution web interface for UWSim: underwater currents benchmark execution, showing main feedback, secondary feedback and results sections.

VI. Use case: Evaluating Position Controllers for Changing Underwater Currents

Besides the example shown on previous work [24], where a visibility tracker was introduced, the web-based benchmarking tool has been extended now to a more realistic use case. The goal is to maintain the vehicle (Girona500) in a relative position from a target (blackbox) so it's able to start a manipulation intervention. In this case the software architecture created to solve the problem consists of 4 pieces:

- Tracker: The tracker is the software which starts the action finding the target on a camera and publishing its position on the camera's image. In this experiment we make use of two different trackers: ViSP [25] and ESM [26] with different configurable options available to choose.
- VisualStationKeeper: It is the program that takes the tracker's position and converts it to the world's distances (meters instead of pixels) and decides where the vehicle needs to move, in order to be in the best position to start a manipulation action, publishing the error between the vehicle position and goal position. In this case, it just tries to keep the vehicle on the top of the object.
- Position controller: This controller is in charge of deciding a velocity to reach the goal assigned by the visual station keeper. The web-based benchmark tool offers a P and PI controller, being able to change their

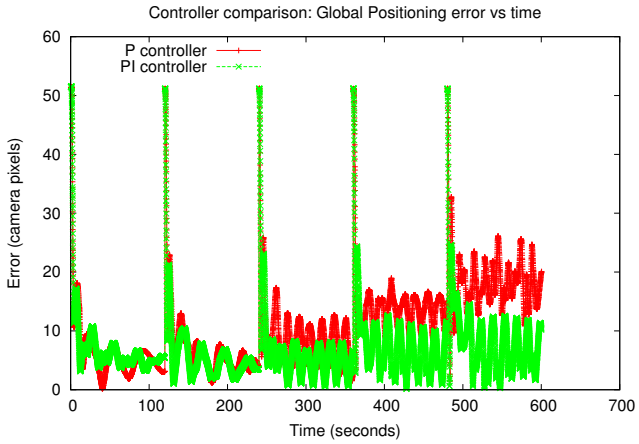

Figure 7. Controllers comparison global error (distance to target position) using Ki=0.02, Kp=0.5 on PI and Kp=0.5 on P controller.

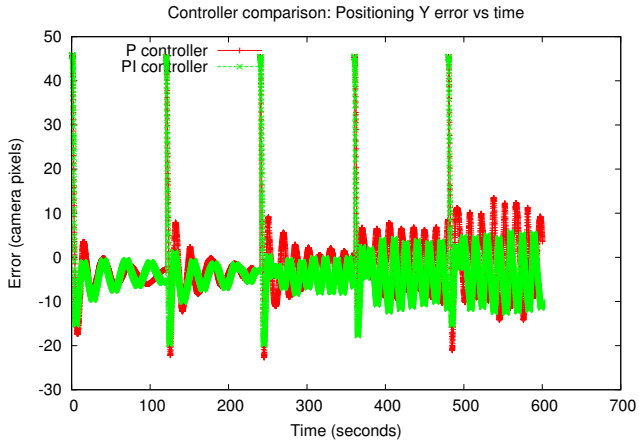

Figure 8. Controllers comparison on Y axis using Ki=0.02, Kp=0.5 on PI and Kp=0.5 on P controller.

constants, and it also provides an interface to upload *Python* code to substitute the default controllers.

- Velocity controller: This program reads the velocity reference from position controller and the vehicles velocity measured from a DVL (Doppler Velocity Log) sensor and decides the effort needed on each thruster of the vehicle. The software provided in this case is a proportional controller.

In order to be able to do multiple experiments at once, every program receives on a ROS topic notifications when the experiment is restarted. In this use case, every 120 seconds, the benchmark is restarted to increase perturbations, that is, underwater currents velocity. Starting from no current at all it tests from 0.1m/s to 0.4m/s in 0.1m/s steps.

Using this architecture, users can develop, test, compare and improve their position controllers faster and easier than using traditional methods, although further real experiments are needed to achieve a fully working code.

A. *Current Benchmarking Execution Results*

The system has been tested with two different controllers. As can be seen on Fig. 7, the P controller is not able to reach a 0 error when current force increases. It is possible to see this on the graph, as every 120 seconds, the system is restarted using a higher current force. The error is able to achieve depends on the current force. On the other hand, the PI controller is able to reduce this error despite underwater currents, as it was expected. Nevertheless, it's sinusoidal behaviour increases due to the sinusoidal nature of currents, which PI controller is not completely able to compensate.

It is also possible to compare errors on X and Y axis only. As underwater currents have its sinusoidal component on Y axis the expected results are an increasing sinusoidal error on it. This results are shown on Fig. 8. In the graph, both controllers are centered on error=-5 pixels instead of

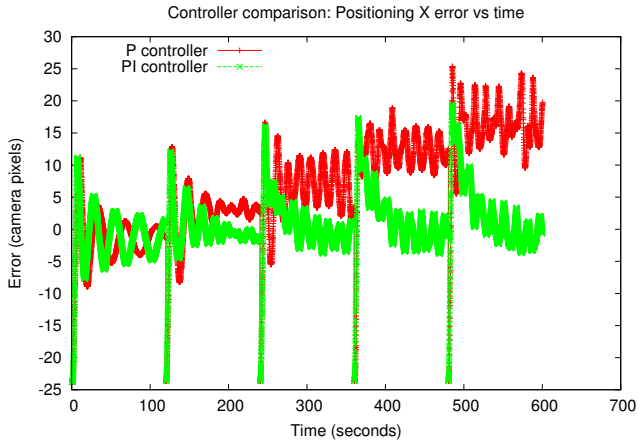

Figure 9. Controllers comparison on X axis using Ki=0.02, Kp=0.5 on PI and Kp=0.5 on P controller.

zero. This is caused by small errors on the tracker software which is guiding the vehicle to this position instead of zero. As can be seen, PI controller gets slightly better results, but it is not able to completely neutralize sinusoidal currents.

On the X axis, an increasing error is expected for P controller while PI controller should be able to drive the error to zero independently from underwater currents. This behaviour is depicted on Fig. 9. Although in stronger underwater currents, PI controller needs around 40 seconds to reach a zero error, it is able to do it, while P controller stabilizes around 18 pixels error. Results when current force is low or there is no current at all, are similar for P and PI controllers.

Besides comparing two different controllers, it is also possible to test different trackers and compare control results depending on the tracker used. Most trackers do not present big differences, but in general, the PI controller helps the

tracking task due to its smoother movements. Translation warp is not suitable for this experiment because it restricts target movements on Z, scale transformations, and after some seconds is not able to find the target.

VII. Conclusion

In this paper, recent advances on the online benchmarking configuration tool for the UWSim (Underwater Simulator) software have been presented, which permits the design of specific experiments on autonomous underwater interventions. More specifically, the simulator allows the integration, in a unique platform, of the data acquired from the sensors in a real submarine intervention and define a dataset, in order to allow further experiments to work on the same scenario, permitting a better understanding of the results provided by previous experiments. The presented benchmarks focus on the effect of underwater currents in the vehicle position controllers. Using this tool, the end user can also test its own developed controllers, obtaining detailed results of its performance and being able to download the resulting data for further analysis. As a work in progress, the online user interface is being improved, so that it will be possible for the user to launch the benchmarks experiments on the server side, sending software to evaluate and compare using different languages (C, C++, Python, MATLAB, Simulink), and downloading or seeing the results online, without the need to locally install any software.

Acknowledgment

This work was partly supported by Spanish Ministry of Research and Innovation DPI2011-27977-C03 (TRITON Project), by Foundation Caixa Castelló-Bancaixa, Universitat Jaume I grant PI.1B2011-17 and University Jaume I grant PREDOC/2012/47.

References

[1] P. J. Sanz, P. Ridao, G. Oliver, G. Casalino, Y. Petillot, C. Silvestre, C. Melchiorri, and A. Turetta, "TRIDENT: An european project targeted to increase the autonomy levels for underwater intervention missions," in *OCEANS'13 MTS/IEEE conference. Proceedings*, San Diego, CA, 2013.

[2] M. Prats, J. Pérez, J. Fernández, and P. Sanz, "An open source tool for simulation and supervision of underwater intervention missions," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, oct. 2012, pp. 2577 –2582.

[3] J. Craighead, R. Murphy, J. Burke, and B. Goldiez, "A survey of commercial open source unmanned vehicle simulators," in *Robotics and Automation, 2007 IEEE International Conference on*, april 2007, pp. 852 –857.

[4] O. Matsebe, C. Kumile, and N. Tlale, "A review of virtual simulators for autonomous underwater vehicles (auvs)," *NGCUV, Killaloe, Ireland*, 2008.

[5] Marine Simulation, "Marine Simulation ROVsim," Available online: http://marinesimulation.com.

[6] Fugro General Robotics Ltd., "Deepworks," Available online: http://www.fugrogrl.com/software/.

[7] GRL, "ROVolution, GRL (General Robotics Limited). ROV pilot training and mission planning simulator."

[8] R. Dillman, "KA 1.10 Benchmarks for Robotics Research," University of Karlsruhe, Tech. Rep., 2004.

[9] D. Calisi, L. Iocchi, and D. Nardi, "A unified benchmark framework for autonomous mobile robots and vehicles motion algorithms (movema benchmarks)," in *Workshop on experimental methodology and benchmarking in robotics research (RSS 2008)*, 2008.

[10] B. Taylor, S. Balakirsky, E. Messina, and R. Quinn, "Analysis and benchmarking of a whegs robot in usarsim," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, sept. 2008, pp. 3896 –3901.

[11] O. Michel and F. Rohrer, "The rat's life benchmark: competing cognitive robots," in *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, ser. PerMIS'08. New York, NY, USA: ACM, 2008, pp. 43–49. [Online]. Available: http://doi.acm.org/10.1145/1774674.1774682

[12] S. Ulbrich, D. Kappler, T. Asfour, N. Vahrenkamp, A. Bierbaum, M. Przybylski, and R. Dillmann, "The opengrasp benchmarking suite: An environment for the comparative analysis of grasping and dexterous manipulation," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, Sept 2011, pp. 1761–1767.

[13] G. Kootstra, M. Popović, J. Jørgensen, D. Kragic, H. Petersen, and N. Krüger, "Visgrab: A benchmark for vision-based grasping," *Paladyn*, vol. 3, no. 2, pp. 54–62, 2012. [Online]. Available: http://dx.doi.org/10.2478/s13230-012-0020-5

[14] B. Cohen, I. Sucan, and S. Chitta, "A generic infrastructure for benchmarking motion planners," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, Oct 2012, pp. 589–595.

[15] 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.

[16] G. Fontana, M. Matteucci, and D. G. Sorrenti, "Rawseeds: Building a benchmarking toolkit for autonomous robotics," in *Methods and Experimental Techniques in Computer Engineering*, ser. SpringerBriefs in Applied Sciences and Technology, F. Amigoni and V. Schiaffonati, Eds. Springer International Publishing, 2014.

[17] R. Esteller-Curto, A. del Pobil, E. Cervera, and R. Marin, "A testbed internet based architecture proposal for benchmarking of visual servoing techniques," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, July 2012, pp. 864–867.

[18] M. Prats, J. Pérez, J. Fernández, and P. Sanz, "An open source tool for simulation and supervision of underwater intervention missions," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 2012, pp. 2577–2582.

[19] FP7-MORPH, "Marine Robotic System of Self-Organizing, Logically Linked Physical Nodes (MORPH)," http://morph-project.eu/.

[20] FP7-PANDORA, "Persistent Autonomy through learNing, aDaptation, Observation and Re-plAnning (PANDORA)," http://persistentautonomy.com/.

[21] J. Pérez, J. Sales, M. Prats, J. V. Martí, D. Fornas, R. Marín, and P. J. Sanz, "The underwater simulator UWSim: Benchmarking capabilities on autonomous grasping," in *11th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, 2013.

[22] UWSim-Benchmarks. (2014) The UWSim Benchmarks Workspace. Available online: http://sites.google.com/a/uji.es/uwsim-benchmarks.

[23] ——. (2014) UWSim: The UnderWater Simulator online. Available online: http://robotprogramming.uji.es/UWSim/config.html.

[24] J. Perez, J. Sales, R. Marin, and P. Sanz, "Web-based configuration tool for benchmarking of simulated intervention autonomous underwater vehicles," in *Autonomous Robot Systems and Competitions (ICARSC), 2014 IEEE International Conference on*, May 2014, pp. 279–284.

[25] E. Marchand, "ViSP: a software environment for eye-in-hand visual servoing," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 4, 1999, pp. 3224–3229 vol.4.

[26] E. Malis, "Improving vision-based control using efficient second-order minimization techniques," in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 2, 26-may 1, 2004, pp. 1843 – 1848 Vol.2.