Exploring 3D Reconstruction Techniques within Autonomous Underwater Manipulation Tasks

Javier Pérez, Jorge Sales, Antonio Peñalver, David Fornas, J. Javier Fernández, Juan C. García, Pedro J. Sanz, Raúl Marín and Mario Prats

***Abstract*—Nowadays, when the results of research in the field of robotics are presented to the scientific community the same question is asked repeatedly: are the results really reproducible? Regarding benchmarking issues, some technological areas, where complex mechatronic devices such as robots have a central role are, in general, very far from other research areas like physics or chemistry, to name but a few, where reproducibility is always mandatory. Leaving aside mechatronic complexities, the comparison between two different algorithms in the same conditions is influenced by the experimental validation scenario. In underwater environments, the difficulties for benchmarking characterization increase substantially. This is especially true when the testbed is the sea where uncertainty is really high. It is the aim of this work to present a software tool which enables a comparison between two different algorithms to be made when these algorithms are being used to solve the same problem in water tank conditions. This is a preliminary stage before the final validation on the seabed. The evaluated algorithms fall into the 3D image reconstruction context, as a prior step to their autonomous manipulation. Performance results are presented for both simulation and real water tank conditions.**

***Index Terms*—benchmarking; underwater interventions; open source simulator; 3D reconstruction.**

I. Introduction

CONCERNING benchmarking in robotics, a lot of effort has been made over the last few years. For instance, the EURON (EURopean RObotics research Network) has been very active in this context [1], and has recognized as a key area, the interaction of a robotic manipulation system with its environment. Indeed some recent European projects, like FP7-BRICS (Best Practice in Robotics), significantly contributed to this specific subject [2], promoting the interoperability of hardware and software components and building a software repository of best practice robotics algorithms [3]. Moreover, following previous research in this field [4], it is clear that: "In the domain of robotics research, it is extremely difficult not only to compare results from different approaches, but also to assess the quality of the research. This is especially true if one wishes to evaluate the performance of intelligent robot systems interacting with the real world." There are many definitions for the term "benchmark", but we can use a very simple one stated in the work mentioned above, that is defined as a standardized problem or test that serves as a basis for evaluation or comparison.

It is the aim of this work to present a benchmarking tool, in the underwater intervention context, so that a suitable comparison between different algorithms but with the same goals can be made, in the same context and with the same robotic platform. For a better understanding, a pair of algorithms will be compared, highlighting the main facilities available through this tool. Moreover, the same algorithms will be tested and compared in simulation and in real water tank conditions.

The rest of this article is organized as follows: Section II introduces the need for benchmarking in the context of underwater intervention systems; Section III makes a review of related benchmarking suites and toolkits; Section IV describes the UWSim (UnderWater SIMulator) simulation tool and its benchmarking capabilities; Section V explains both the benchmarking module for UWSim and the physical benchmarking platform used for the experiments; Section VI outlines the experiments specification; Section VII analyses the results, and finally, conclusions and further work are given in Section VIII.

II. Benchmarking for Underwater Intervention Systems

The use of underwater robots is becoming evermore widespread because technical advances make them increasingly useful. Some examples can be found in the oil and gas industry (e.g. operating submerged infraestructures), search and recovery missions (e.g. recovering a crashed airplane black-box), deep water archeology or scientific missions. Usually, the robots used in these missions are Remotely Operated Vehicles (ROV), which are costly both in financial and logistic terms. In addition, these robots use a master/slave architecture and pats all the responsibility on the pilot, who in turn suffers from cognitive fatigue and stress. The evolution of this kind of robot to the new Intervention Autonomous Underwater Vehicle (I-AUV), removes the human from the control loop (and therefore, the problems related with the pilot) and increases the intervention capabilities being able to increase robot precision and intervention time by avoiding huge delays caused by the difficulties in communication.

Experimentation with underwater robots is normally very difficult due to the wide range of resources required. For instance, a water tank deep enough for the systems to be tested, is normally needed and this implies significant space and maintenance costs. Another possibility is the access to open environments such as lakes or the sea, but this normally

Javier Pérez, Jorge Sales, Antonio Peñalver, David Fornas, J. Javier Fernández, Juan C. García, Pedro J. Sanz, Raúl Marín and Mario Prats are with the Computer Science and Engineering Department, University of Jaume-I, Castellón, Spain. e-mail: {japerez, salesj, penalvea, dfornas, fernandj, garciaju, sanzp, rmarin}@uji.es, marioprats@gmail.com.

involves high costs and requires special logistics. In addition, the nature of the underwater environment makes it very difficult for researchers (operating on the surface) to observe the evolution of the running system. As a consequence, experimental validation of these systems is very laborious. In order to facilitate the development of underwater robots, it is of utmost importance to develop suitable simulators that make it possible (i) to develop and benchmark the systems before they are deployed, and (ii) supervise a real underwater task where the developers do not have a direct view of the system.

Usually, experimental validation in underwater intervention system takes place in the sea where there are many changing parameters such as underwater currents, bad visibility... that can be hardly modelled and replicated. These uncertainties are the main issue when comparing and replicating results from different studies. The use of an automated comparison system in simulation and controlled environments helps to establish an objective benchmarking methodology.

There are previous simulators for underwater applications, which mainly have remained obsolete or are being used for very specific purposes. In [5] and [6], a review of virtual simulators for autonomous underwater vehicles (AUVs) can be found. Nevertheless, the majority of the reviewed simulators have not been designed as open source, which makes it difficult to improve and enhance the capabilities of the simulator. Other simulators, such as ROVSim, VMAX or DeepWorks, have been designed to train ROV pilots, which is not the objective of our research.

Underwater manipulation using I-AUV allows the design of new applications such as the one studied at the FP7 TRIDENT project [7], where a black-box from the seabed was autonomously recovered. To accomplish this, the use of the UWSim Underwater Simulator [8], in continuous development was crucial, for testing, integration and also benchmarking.

III. Review of related Benchmarking Suites and Toolkits

In recent 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 [9]. This suite is a software environment for comparative evaluation of grasping and dexterous manipulation using the OpenGrasp toolkit. It also provides a web-service that administers available benchmarks scenarios, models and benchmarking scores.

Another interesting benchmarking suite in the field of grasping is VisGrab [10] (a benchmark for Vision-Based Grasping), which provides tools to evaluate vision-based grasp-generation methods.

Motion planners, trajectory tracking and path planning have been very active research fields around benchmark metrics and benchmarking suites. In [11], authors describe a generic infrastructure for benchmarking motion planners. This infrastructure makes it possible 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) [12] MoveIt! integration.

Rawseeds [13], is a project focused precisely on benchmarking in robotics, although its 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. Unfortunately this project lacks of an automated comparison system.

Finally, there have been proposals of web-based benchmarking suites such as [14] where authors propose an interesting test-bed internet-based architecture for benchmarking of visual servoing techniques allowing users to upload their algorithms.

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

UWSim[1] is an open source software tool for visualization and simulation of underwater robotic missions that offers benchmarking capabilities through a specific module. 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 ROS interfaces. UWSim is currently used in different ongoing projects funded by European Commission (MORPH[2] and PANDORA[3]) in order to perform HIL (Hardware in the Loop) experiments and to reproduce and supervise real missions from the captured logs.

The main objectives in the simulator development are: it can be easy to be integrated with existing architectures; to be general, modular and easily extendible; support for underwater manipulators; and as realistic as possible. From a technical point of view, the simulator has been implemented in C++ and makes use of the OpenSceneGraph (OSG), ROS and osgOcean libraries.

The UWSim is divided into different modules (see Figure 1): there is a Core module in charge of loading the main scene and its simulated robots; an Interfaces module that provides communication with external architectures through ROS; a Dynamics module that implements underwater vehicle dynamics. This module has been designed as a generic dynamics module for underwater vehicles but users can replace it with a more accurate one, if needed using ROS interfaces or even use the real process as input; a Physics module that manages the contacts between objects in the scene; osgOcean, in charge of rendering the ocean surface and special effects; the GUI module, that provides support for visualization and windowing toolkits; and the UIAL and benchmarking modules that will be explained later.

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

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

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

The scene is defined with a XML *(eXtensible Markup Language)* file, which is loaded in a scene graph by OSG, getting access to the nodes easily (i.e. visualization effects, virtual cameras, etc.). UWSim includes, by default, some scenarios (the swimming pool facilities at CIRS[4] and a shipwreck scenario), an I-AUV (the Girona 500 robot) and two different underwater robotic arms (Light-weight ARM5E [15] and a Mitsubishi PA10 Arm).

As mentioned before, robots and scenarios can be created with any modeling software (e.g. Blender). Nowadays, some sensors (e.g. simulated position, lasers, measure distances to obstacles, etc.) and virtual cameras can be attached to the robots. Dynamics using a state-space model and physics using Bullet engine are supported. Other interesting features are the widgets, which are small windows that can be placed inside the scene to show specific data to the user and the multi-resolution terrain compatibility, allowing the user to load complex meshes with multi-resolution textures generated externally from bathymetry and imagery.

The simulator is in continuous development. Some of the recently added features consist of: new ROS versions and OSG libraries compatibility; multibeam sensor simulation; texture projector to simulate structured light; ROS TF publishing; force sensor integration with the physics engine; Dantzig physics solver to improve robotics and manipulation capabilities; and visualization improvements (trajectory trails, point clouds, etc).

The work in progress in the UIAL module can be divided into different aspects: improving the information to be shown to the user, reducing the data depending on the mission and context; integration with an immersive system, where the user would get the feeling of being inside the robot; adding a natural gesture control interface to control robot, improving traditional ways (e.g. leapmotion); and implementing an abstract layer, which will manage all these improvements and will make it possible to integrate every component within most of the current architecture.

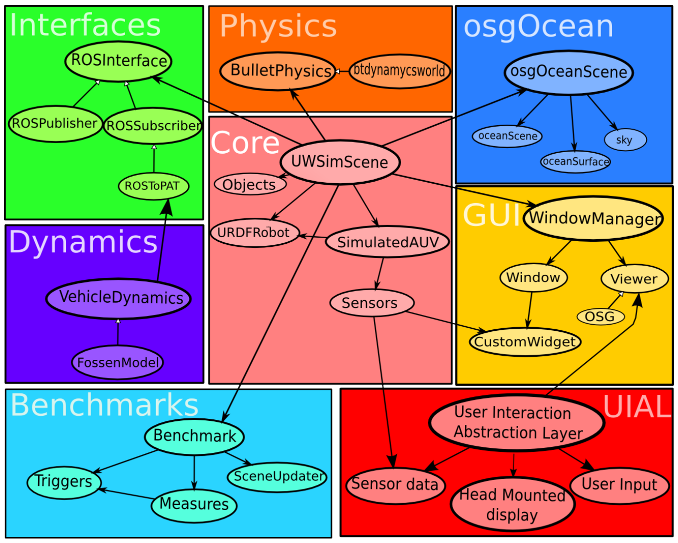


Fig. 1. UWSim modules diagram and its interconnections: Core, Interfaces, Physics, Dynamics, osgOcean, Graphical User Interface (GUI), User Interface Abstraction Layer (UIAL) and Benchmarks.

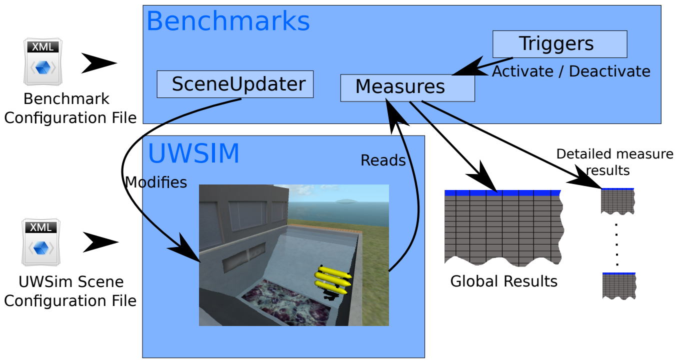


Fig. 2. *Benchmarking* module flow diagram: a benchmark configuration is loaded into the benchmark module, and a scene is loaded into the simulator. Then, the benchmark module produces some results that can be logged for posterior analysis.

V. Benchmarking Platform Description

A. The benchmarking module for UWSim

Recently, a *benchmarking* module for UWSim has been developed [16]. 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 online[5].

For the development of the module, two important objectives were taken into account. The first one was to be transparent to the user, in other words, that it does not require major modifications to the algorithm to be evaluated. The other objective of the module was that it must be adaptable to all kinds of tasks in the underwater robotics field.

Benchmarks are defined in XML files. Each file will define which measures are going to be used and how they will be evaluated. This allows the creation of standard *benchmarks* defined in a document to evaluate different aspects of underwater robotic algorithms, being able to compare algorithms from different origins. Each of these *benchmarks* will be associated with one or more UWSim scene configuration files, being the results of the *benchmark* dependent on the predefined scene. Consequently creating a new benchmark experiment is as simple as editing a configuration file. The whole process is depicted in Figure 2.

The benchmark configuration options are basically made up from three kinds of entities: measures, triggers and scene updaters. These entities have been created in a modular way, thus users can extend them and create new functionality easily. Measures can be chosen from a wide variety of already implemented measures such as position error, elapsed time, distance

[4]Underwater Robotics Research Center, Universitat de Girona, Spain

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

travelled, path following error, reconstruction 3D, etc. Some of these measures are split in different parts that will be shown in the final results, for instance position error is formed by X error Y error and Z error. Setting this parameters will make the benchmark to measure each of the configured option using the ground truth from UWSim or external sources via ROS depending on the availability and configuration options. In the case of positioning errors, ground truth is taken from UWSim and path following requires a path to follow configured via ROS input.

These measures are activated or deactivated depending on events configured through triggers. These events allow users to measure results in an easier way, for instance start (or stop) measuring when a vehicle reaches a position, a message is received in ROS or the vehicle moves. A case where this might be used is to measure collisions only when the vehicle is navigating and stop when the manipulation starts, when the hand should collide with the manipulated object but it is not a bad result.

Finally, scene updaters modify the simulated environment and restart the measurement being able to start a series of experiments. Possible scene updaters are underwater current updater, ambient light updater, camera noise updater, etc. This feature is useful to create automated tests that can check the influence of environmental parameters helping to create more robust algorithms.

Once the benchmarking has finished, caused by a stop trigger event, and all the scene configurations in the scene updater have been tested, results are written into output files. These output files are disaggregated for each scene configuration containing results for each measure and global results that can combine multiple measure results. For instance, a travel efficiency benchmark could use two measures: distance travelled and battery consumption and a global result of $distance/battery$. Additionally, each measure can be configured to log its result at regular intervals in order to see its evolution over time and not only the final result for each scene configuration. As a result, for each logged measure, the benchmark will generate a different output file containing the variation among the measured results.

B. The physical benchmarking system

In order to be able to evaluate the compared algorithms and validate the simulated results in a real platform, a physical benchmarking platform has been used (see Figure 3). It consists of the following elements, taking into account the main elements in the virtual scene used in UWSim:

- Water tank: dimensions 2.0 m (width) x 2.0 m (length) x 1.5 m (height).
- 4 D.O.F. ECA-CSIP Light-weight ARM 5E manipulator [15].
- Floating structure (underwater vehicle prototype) to hold the arm. In this work the floating structure has been fixed to the water tank.
- Bowtech DIVECAM-550C-AL COLOUR camera.
- Tritech SeaStripe Laser Line Projector (MKIII).
- Videre stereo camera.

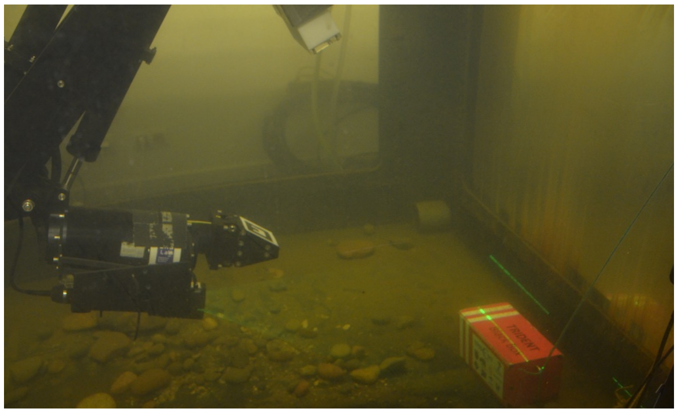


Fig. 3. Physical benchmarking system: water tank, Light-weight ARM5E manipulator, stereo camera, laser stripe projector and black-box mockup.

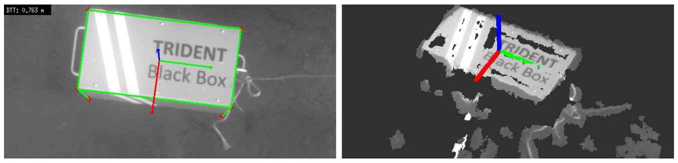


Fig. 4. Camera image with manually initialized corners and estimated box pose (left). Ground truth box pose with the point cloud obtained with the stereo camera (right).

- Black-box mockup size 140 mm (width) x 300 mm (length) x 160 mm (height).

In order to obtain the object pose ground truth, a pose estimation method has been used to compute the relative position of the target object (black-box mock-up) with respect to the camera, taking into account that the arm is firmly fixed to the water tank. As dimensions of the object are well known, the box corners can be used to estimate its pose. In the case of using a different object, easily recognizable points could be used instead of corners. While it is possible to detect them automatically, it has been judged that the manual initialization by the user is less error prone and best suited to get an accurate ground truth for the benchmarking system.

First of all, the user clicks on the visible corners on the box (in this approach six corners were visible). Then, after matching the obtained 3D points with the real object using the camera parameters, the object pose estimation is obtained using the ViSP library. In this case the frame is placed in the center of the top face of the box (see Figure 4). As there are several methods that can be used to obtain the estimation, all of them are used to estimate the pose and the one that minimizes the estimation error is selected.

This ground truth, however, is not perfect as small errors appear because of the limited camera resolution (actual pixel size), user accuracy and camera calibration. Nevertheless, the resulting error is small enough to allow the object position to be considered as a suitable ground truth so that the metrics described in this paper can be used. In fact, the camera calibration accuracy affects rectification and undistortion in these cameras, thus obviously introducing some shared error in this ground truth position and at the same time in the

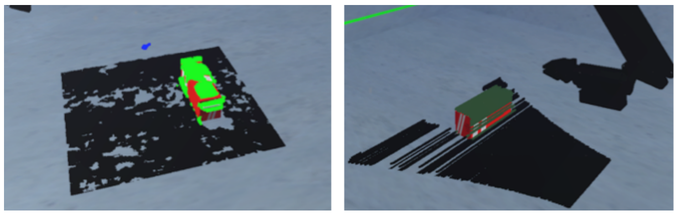


Fig. 5. Stereo (left) and laser stripe (right) reconstructions comparison overlayed on ground truth on simulated environment. Black points are filtered as ground, blue are considered outliers and green points represent reconstructed object

reconstruction processes.

VI. Experiments Specification

To test the benchmarking platform, two reconstruction algorithms are presented: stereo reconstruction using a stereo camera (see Figure 5 (left)) and laser stripe segmentation (see Figure 5 (right)). These approaches are used to obtain a point cloud from the scene, as a consequence, the object can then be processed. Both algorithms are exactly the same whether used in a simulation or in a real setup.

A. Stereo Reconstruction description

The aim of the stereo reconstruction is to obtain 3D reconstruction in the form of a dense point cloud, where each image pixel is used in order to obtain a 3D point instead of computing them for certain features only (sparse reconstruction). A good reconstruction can be obtained only if the camera parameters are properly estimated. The parameters are computed with camera calibration tools and a calibration checkerboard.

In runtime, images from left and right side are undistorted and rectified using the aforementioned camera parameters, so that their scanlines align for fast stereo processing. Once the images are aligned, a local dense stereo correspondence algorithm can be applied. In this case, OpenCV block matching algorithm [17] implemented in a ROS package is fast enough for most robotic applications, while only needing previous parameter tuning. With the chosen algorithm, both disparity images and dense point clouds can be obtained. This method estimates the corresponding pixel on the right image for every pixel on the left image, thus comparing each pixel to a block on the other image. The displacement between the two pixel is used to determine the 3D point coordinates based on the camera geometry computed in the calibration step.

B. Laser Reconstruction description

Before the system (see Figure 3) is able to perform a reconstruction, it is necessary to calibrate it. So, with the aid of a marker placed in the gripper of the arm, the transformation between the camera and the end-effector (${}^{c}\mathbf{M}_{e}$), is calculated [18]. Then, using the Direct Kinematics of the arm, the relationship between the base of the arm and the end-effector is obtained (${}^{b}\mathbf{M}_{e}$). Thus, using these two matrixes the transformation between the base of the arm and the camera is easy to calculate: ${}^{c}\mathbf{M}_{b} = {}^{c}\mathbf{M}_{e} * ({}^{b}\mathbf{M}_{e})^{-1}$

The next parameter that has to be obtained is the relationship between the laser and the end-effector (${}^{b}\mathbf{M}_{e}$). The camera installed in the vehicle is a stereo camera so, even though just one lens is used for the laser reconstruction, for this step of the calibration the two lenses are used. Using the stereo camera, the 3D position of the pixels projected by the laser are obtained by triangulation. With those 3D points, The RANSAC algorithm is used to determine the planar parameters of the laser plane [19] (${}^{c}\mathbf{M}_{l}$). These parameters are referenced to the stereo camera using the previously obtained transformation between the camera and the end-effector (${}^{c}\mathbf{M}_{e}$), it is possible to reference the plane of the laser respect to it: ${}^{l}\mathbf{M}_{e} = ({}^{c}\mathbf{M}_{l})^{-1} * {}^{c}\mathbf{M}_{e}$

Concerning the reconstruction, the floor is scanned by moving the elbow joint of the manipulator at a constant velocity between two predefined joint positions. At the same time, the camera captures images of the scene with the laser projected on it. For each image, a laser peak detector algorithm is used to segment the laser stripe from the rest of the image. This algorithm discards the pixels that are out of a predefined threshold of huge, saturation and value. Then, thanks to the laser pattern is a straight line and the camera is placed parallel to that line, there is only a point illuminated by the centroid of the laser at each column of the image. As a consequence, for each column of the image, the pixel with the highest intensity is selected and the center of masses algorithm is applied to this pixel and the five pixels above and below it to obtain, with subpixel accuracy, the position illuminated by the centroid of the laser.

Finally, the segmented laser stripe is triangulated to obtain its 3D position [20]. In order to triangulate each selected pixel, it is necessary to know the relationship between the camera and the laser (${}^{l}\mathbf{M}_{c}$) when the image is captured. So, when each image is taken, the values of the joints in this moment are also read. Using these values and the Direct Kinematics of the arm, the transformation between the end-effector and the base of the arm (${}^{b}\mathbf{M}_{e}$) is calculated. Finally, using this relation and the ones obtained in the calibration, the desired transformation can be easily calculated: ${}^{c}\mathbf{M}_{l} = {}^{c}\mathbf{M}_{b} * {}^{b}\mathbf{M}_{e} * ({}^{l}\mathbf{M}_{e})^{-1}$

C. Benchmarking metrics

These methods are compared in a simulated and a real environment, using the proposed benchmarking architecture, and taking into account 4 metrics measured using a high fidelity model of the object to be reconstructed as ground truth. Benchmarking module takes this object model as ground truth and a configuration file to get the position of the object with which the results can then be calculated. These four metrics have been introduced in [21], a work about reconstruction metrics, as quality measures of 3D models in order to find:

- Mean error: Average distance from every 3D reconstruction inlier point to the nearest point in the object surface.
- Standard deviation: Standard deviation for the previous error. A high value in this deviation means misalignment in the reconstruction, due to bad calibration.
- Coverage: Surface percentage that is nearer than a precision threshold to a 3D reconstruction point. It mea-

sures the percentage of the target that is correctly reconstructed. The threshold should be chosen depending on the experimental setup. It is not an inlier measure, instead of measuring the percentage of points near the target measures the percentage of the target that has a reconstructed point nearer than a threshold. For instance a perfect reconstruction of 3 faces of a box would return 50% instead of 100% that would get an inlier metric.
- Outliers: The percentage of the reconstruction that it is further than a threshold from the target.

In order to measure the object reconstruction only, reconstruction points that are part of the ground, such as the object that is lying on it, and outliers are filtered and do not count for the previous described measures.

Besides mathematical results, in this case, the benchmarking module is able to overlay the 3D reconstructed point cloud on the simulated 3D scene to get a visual result of the reconstruction using UWSim as visualization engine. Furthermore, 3D points are colored to show outliers, filtered ground points and object points. This is of great value not only to show results but to debug reconstruction algorithms.s

VII. Results

The following subsections describe and analyze the results of the benchmarking process. The evaluation of the proposed algorithms has been performed in both, simulation and real environments, using the aforementioned measures. This results are easily replicable because the software used, including simulator, benchmarking platform and algorithms are open source. Furthermore metrics, the ground truth acquisition and experimental setup have been described with enough details to allow other studies to be compared with the following results.

A. Simulated results

In the simulation experimentation, benchmarking capabilities have been exploited to test both reconstruction techniques under different conditions of light and noise. These conditions try to simulate the complex and adverse conditions in the underwater environment. Theoretically, a laser should not be affected by low illumination conditions as it produces its own light, but it may be more difficult to detect on brighter scenes. The presence of noise should cause a poorer performance on both methods.

The algorithms have been tested in conditions where the amount of light varies, ranging from 0 to 1.0 ratios where 0 means total darkness and 1.0 is the correct illumination (default values in UWSim), as can be seen on Figure 6. Besides that, gaussian noise has been added to the camera output from 0.00% standard deviation to 0.10% in an additive manner on RGB channels through the UWSim configuration. Noise effect can be appreciated on Figure 7.

Coverage results for laser reconstruction and stereo vision can be seen on Figure 8. As expected, stereo vision reconstruction needs some light to achieve a good reconstruction while laser is nearly immune to light variation and even decreases its performance in conditions where the light is bright. Regarding noise, stereo vision is again more sensitive to noise, especially

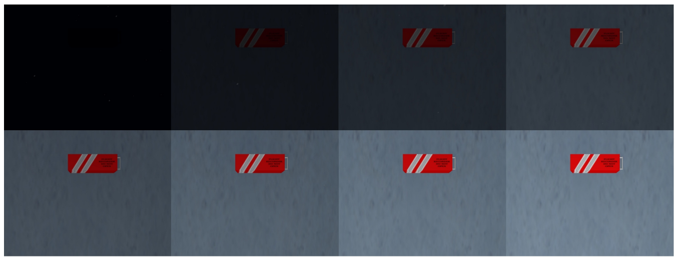


Fig. 6. From left to right, top to bottom increasing light conditions on virtual camera.

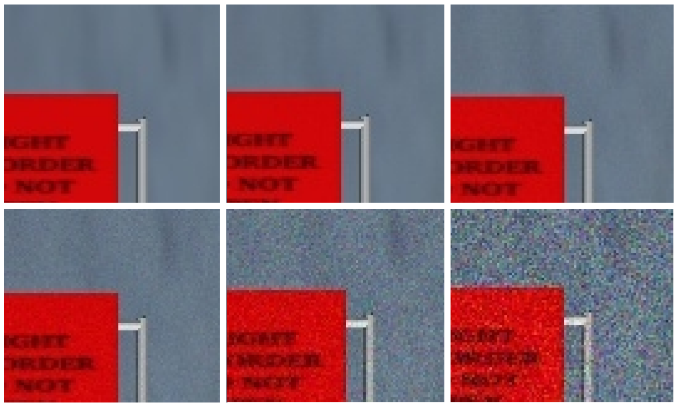
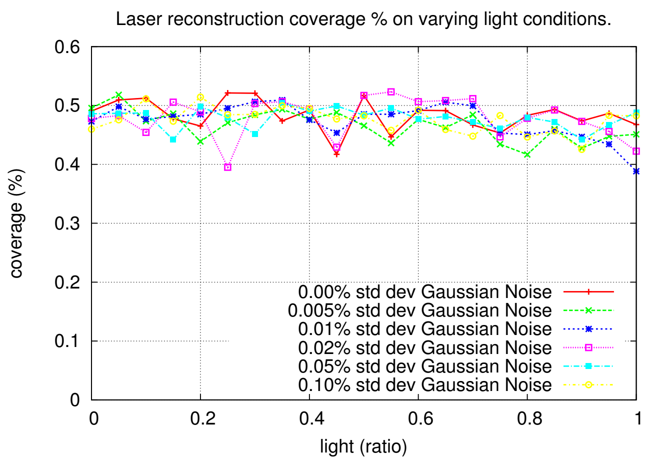


Fig. 7. From left to right, top to bottom increasing noise conditions on virtual camera.

in lower visibility conditions, and laser shows no noticeable differences between different noises on coverage. In absolute terms, the laser is able to reconstruct 50% of the object in almost every situation and stereo vision reconstructs 40% of the object in good light conditions. So it can be concluded that laser is clearly better for the tested environment.

About the mean error and standard deviation, both algorithms show similar results. In the case of mean error, due to the similar setup, both algorithms achieve 0.004 meters, which is a good value given the experimental setup. As the alignment is perfect on simulation, the standard deviation is negligible.

Finally, the outliers results are depicted on Figure 9 for both, the laser and stereo vision cases. As results show, stereo vision generates more outliers in the absence of light, while the laser reconstruction produces higher number of outliers in the presence of light. In both cases higher gaussian noise means a higher number of outliers. In the case of stereo vision, noise and the absence of light makes it more difficult to match the pixels on both cameras, and a large number of outliers appear. On the other hand, in strong light conditions it is more difficult to find the laser light on the camera, and mistaken detections cause a high number of outliers. It is also remarkable that there is a non-negligible number of outliers in the stereo vision results for 0.00% standard deviation noise. These outliers are caused by small floating particles simulated in UWSim, which are correctly detected by the system although they are not part of the object. The 3D stereo reconstruction is not able to find these outliers when the noise is higher and its impact decreases as more parts of the object are correctly reconstructed.



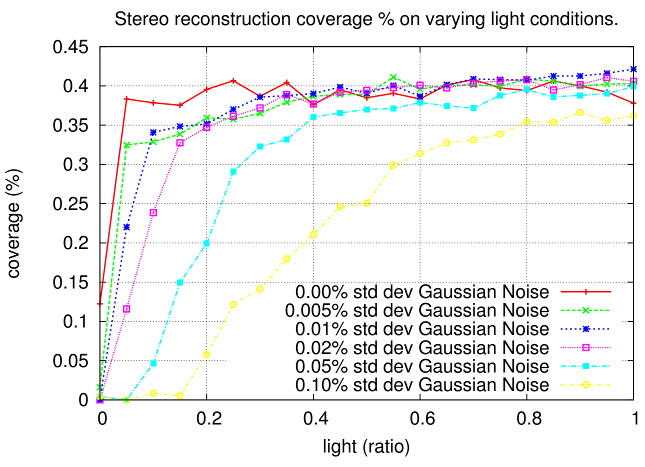
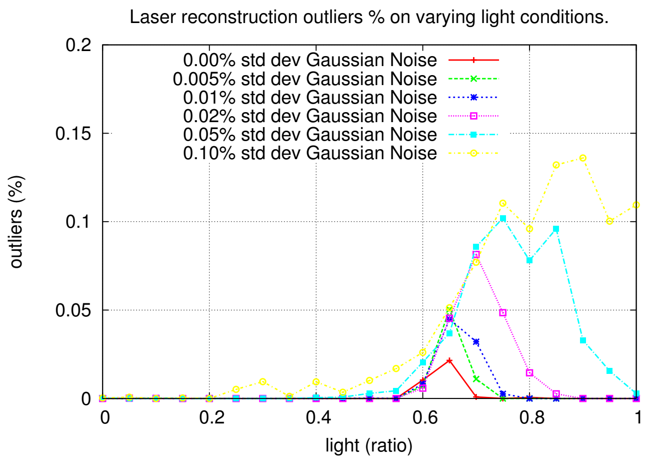


Fig. 8. Coverage results on varying light conditions for different gaussian noise on camera for Laser (left) and Stereo camera (right).



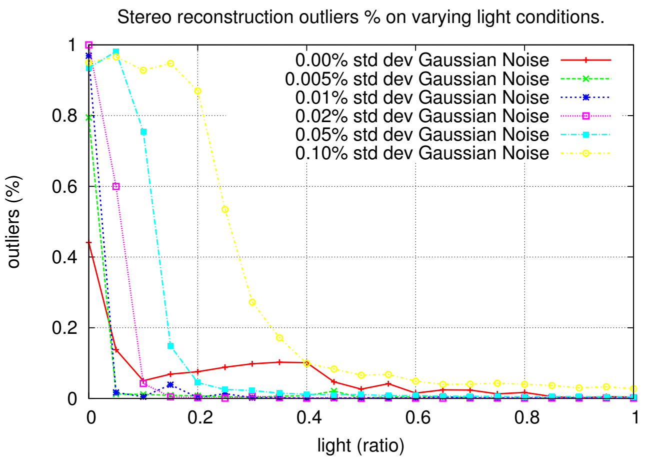


Fig. 9. Outliers results on varying light conditions for different gaussian noise on camera for Laser (left) and Stereo camera (right).

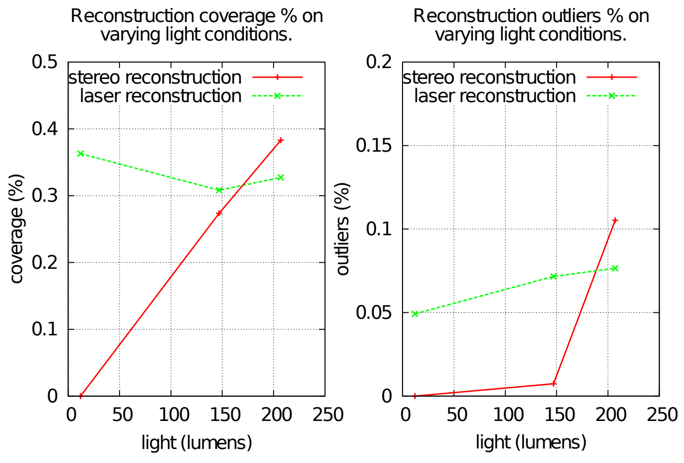


Fig. 10. Reconstruction coverage (left) and outliers results (right) for Stereo camera and Laser on real scenario for different light conditions.

B. Real results

In the experiment conducted under real conditions, both systems have been tested under three different light conditions shown on Figure 11, in order to replicate the simulation results. As the illumination of the environment is a key characteristic in this experiment, a lux meter was used to assure the replicability of the experiment. The lux meter was placed in a flat surface as near as possible to the black box. The values obtained for the testing scenarios were 12, 147 and 207 lumens.

The system has been calibrated in such a way so that the benchmarking module for UWSim can be used to measure real results in the same way as it is used to measure simulated results. In order to do this, the input data must be configured to be taken from the real world instead of a virtual scene and use the calibration method mentioned above in order to acquire the ground truth.

The 3D point clouds reconstructed are then evaluated by the benchmarking platform as can be seen in Figure 12, where real point clouds are displayed on UWSim while being processed. In the images, black dots are filtered as ground points, green

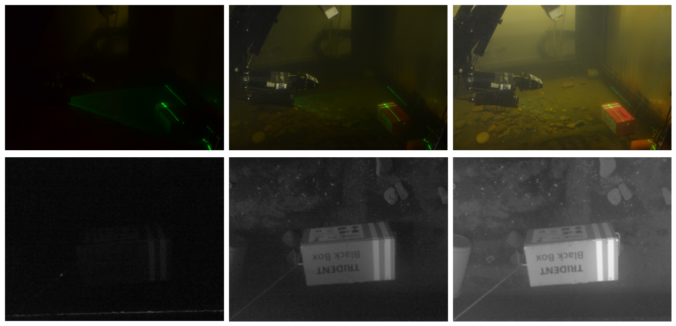


Fig. 11. From left to right: low, medium and high light conditions. From top to bottom: external view and camera view.

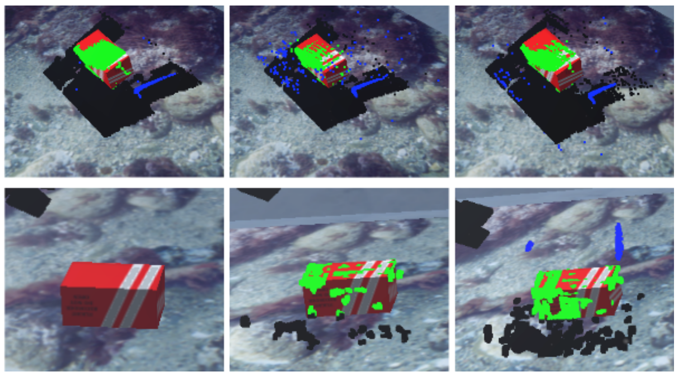


Fig. 12. From left to right: low, medium and high light conditions. From top to bottom: real laser point cloud reconstruction and real stereo point cloud reconstruction overlayed on UWSim.

points are considered object points and blue points are labeled as outliers. As happened in simulation, laser reconstruction works better in low light environments while stereo needs some light to work properly.

A further analysis of the visual results shows that although laser reconstructions looks better, there is a misalignment on the pointcloud. 3D laser reconstructions are slightly rotated with respect to the ground truth target due to small errors on camera to laser projector calibration. In the case of stereo reconstruction the ground reconstruction was very poor due to the absence of texture on it.

Results for coverage are depicted on Figure 10 (left). Laser results are slightly worse than simulated ones. In this case, laser achieves around 32%-38% while in simulation it reached 50%. Although laser works better in dark situations, it is highly resistant to light changes. On the other hand, stereo reconstruction is completely dependent on light conditions, achieving a 38% of coverage in good light environments. These results support the ones obtained in simulation where both algorithms behave in a similar way.

Regarding mean errors, laser and stereo have similar mean errors, around 0.008 meters. This result is much greater than in simulation due to the added ground truth estimation error. The standard deviation, though, is greater than the one obtained in simulation, around 0.005 meters in stereo and 0.008 meters in the case of laser reconstructions, it is small enough to conclude that the tested algorithms reached a good alignment and the ground truth estimation was fairly good. This shows the same small misalignment as visual output in the case of laser reconstruction.

The outliers results depicted on Figure 10 (right) show that both algorithms increase the number of outliers as light increases. In this case, stereo reconstruction shows a 0% on outliers in the absence of light because is not able to obtain any points. Although in the visual output laser reconstructions seemed to show a higher number of outliers in terms of percent it is compensated to the higher amount of reconstruction points that means they could be filtered easily.

VIII. Conclusion

In this work, a benchmarking process is presented to allow easy objective comparison and replication of the results of two 3D object reconstruction algorithms as a prior step to manipulation in simulated and real scenarios. This process involves a benchmarking platform developed to evaluate software using a simulator as ground truth for the evaluation. The presented results show the potential of the benchmarking techniques making it possible to obtain measurable results in simulated scenarios, just as in real situations, helping to decide which approach is better in each situation so that the design of the system can be improved at an early stage. Results replicability is assured as the simulator, benchmarking platform and the algorithms used to test it, are offered as open source. Furthermore key parameters such as light conditions have been measured in order to provide sufficient information for the experiment to be replicable. As a work in progress, an online benchmarking platform is being actively developed, to avoid software installation and make algorithm evaluation and comparison faster and more user friendly.

Acknowledgment

This work was partly supported by Spanish Ministry under grant DPI2011-27977-C03 (TRITON Project) and DPI2014-57746-C3 (MERBOTS Project), by Universitat Jaume I grants PID2010-12, PREDOC/2012/47 and PREDOC/2013/46, and by Generalitat Valenciana grant ACIF/2014/298.

References

[1] EURON, “Survey and inventory of current efforts in comparative robotics research,” in *EURON - European Robotics Search Network*. [Online]. Available: http://www.robot.uji.es/EURON/en/index.htm
[2] W. Nowak, A. Zakharov, S. Blumenthal, and E. Prassler, “Benchmarks for mobile manipulation and robust obstacle avoidance and navigation,” in *Deliverable D3.1 from FP7-BRICS Project (Best Practice in Robotics)*, Apr 2010. [Online]. Available: http://www.best-of-robotics.org/home
[3] R. Bischoff, T. Guhl, E. Prassler, W. Nowak, G. Kraetzschmar, H. Bruyninckx, P. Soetens, M. Haegele, A. Pott, P. Breedveld, J. Broenink, D. Brugali, and N. Tomatis, “BRICS - Best practice in robotics,” in *Proc. IFR Int. Symp. Robotics*, Jun 2010, pp. 968–975.
[4] DEXMART, “Specification of benchmarks,” in *Deliverable D6.1 from FP7-DEXMART Project (DEXterous and autonomous dual-arm/hand robotic manipulation with sMART sensory-motor skills: A bridge from natural to artificial cognition*, Jan 2009. [Online]. Available: http://www.dexmart.eu
[5] O. Matsebe, C. Kumile, and N. Tlale, “A review of virtual simulators for autonomous underwater vehicles (AUVs),” in *Proc. IFAC Workshop Navigation, Guidance and Control of Underwater Vehicles*, Killaloe, Ireland, Apr. 2008.

[6] J. Craighead, R. Murphy, J. Burke, and B. Goldiez, "A survey of commercial open source unmanned vehicle simulators," in *Proc. IEEE Int. Conf. Robotics Automation*, Rome, Italy, Apr. 2007, pp. 852–857.

[7] 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 *Proc. MTS/IEEE OCEANS'13 Int. Conf.*, San Diego, CA, 2013.

[8] M. Prats, J. Pérez, J. Fernández, and P. Sanz, "An open source tool for simulation and supervision of underwater intervention missions," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Vilamoura, Algarve, Portugal, Oct 2012, pp. 2577–2582.

[9] 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 *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Sep 2011, pp. 1761–1767.

[10] 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

[11] B. Cohen, I. Sucan, and S. Chitta, "A generic infrastructure for benchmarking motion planners," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Vilamoura, Algarve, Portugal, Oct 2012, pp. 589–595.

[12] 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.

[13] 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.

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

[15] J. Fernández, M. Prats, P. J. Sanz, J. García, R. Marín, M. Robinson, D. Ribas, and P. Ridao, "Grasping for the seabed: Developing a new underwater robot arm for shallow-water intervention," *Robotics Automation Magazine, IEEE*, vol. 20, no. 4, pp. 121–130, 2013.

[16] 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 *Proc. 11th Int. Conf. Informatics in Control, Automation and Robotics (ICINCO)*, 2013.

[17] K. Konolige, "Small vision systems: Hardware and implementation," in *Robotics Research*, Y. Shirai and S. Hirose, Eds. Springer London, 1998, pp. 203–212.

[18] A. Peñalver, J. Pérez, J. J. Fernández, J. Sales, P. J. Sanz, J. C. García, D. Fornas, and R. Marín, "Autonomous intervention on an underwater panel mockup by using visually-guided manipulation techniques," in *Proc. 19th World Congress of the International Federation of Automatic Control (IFAC)*, Cape Town, Africa, Aug. 2014, pp. 5151–5156. [Online]. Available: http://dx.doi.org/10.3182/20140824-6-ZA-1003.02545

[19] G. Inglis, C. Smart, I. Vaughn, and C. Roman, "A pipeline for structured light bathymetric mapping," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Vilamoura, Algarve, Portugal, Oct 2012, pp. 4425–4432.

[20] M. Prats, J. Fernández, and P. Sanz, "Combining template tracking and laser peak detection for 3D reconstruction and grasping in underwater environments," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Vilamoura, Algarve, Portugal, Oct 2012, pp. 106–112.

[21] S. Oude Elberink and G. Vosselman, "Quality analysis on 3D building models reconstructed from airborne laser scanning data," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 66, no. 2, pp. 157–165, 2011.