Revista Iberoamericana de Automática e Informática Industrial

Revista Iberoamericana de Automática e Informática Industrial 15 (2018) 70-78

www.revista-riai.org

# UWSim, un Simulador Submarino Conectado a la Nube como Herramienta Educacional

Javier Pérez[a,*], David Fornas[a], Raúl Marín[a], Pedro J. Sanz[a]

[a]*Departamento de Ingeniería y Ciencia de los Computadores, Universidad Jaume I, Av. de Vicent Sos Baynat 12071 Castellón de la Plana, España.*

**Resumen**

La creciente demanda social de nuevas aplicaciones de la robótica, desde robots domésticos a coches autónomos, confirma la conveniencia de utilizar dicha tecnología como factor motivante en el contexto educacional. Así, el presente trabajo analiza cómo canalizar esta motivación hacia fines productivos, poniendo el énfasis en las posibilidades que ofrecen los simuladores de robots submarinos. En particular, se propone un entorno de aprendizaje en la nube con un simulador capaz de evaluar al alumno como eje central del sistema. Utilizando este tipo de herramientas tan solo es necesario un dispositivo capaz de acceder a Internet a través de un navegador para alcanzar una cantidad virtualmente ilimitada de recursos. Como caso de estudio, se detallan las mejoras implementadas, en una aplicación de seguimiento de tuberías submarinas, creando un entorno de comparación en la nube que permite a los alumnos competir por obtener el mejor resultado posible. Finalmente, es importante destacar que se aporta una primera experiencia de aplicación en un contexto de enseñanza real de la herramienta propuesta, demostrándose la viabilidad e idoneidad de la misma para el aprendizaje de robótica y ROS.

*Palabras Clave:*
Robótica, Educación, Benchmarking, Vehiculos autónomos, Sistemas marinos, Ayudas educacionales, Telerobótica

**UWSim, an underwater robotic simulator on the cloud as educational tool**

**Abstract**

Due to the introduction of robotic applications in the modern society, such as service robots or self-driving cars, it is possible to use this trend as motivating factor in the learning process of robotics. Several possibilities about how to use this motivation to increase learning rate are analysed, focusing on underwater robotic simulators. Moreover, a cloud learning environment able to evaluate the students with a robotic simulator is proposed as key element of the system. These kinds of tools can be used with just an Internet-capable system through a web browser, reaching a virtually unlimited amount of resources. The implemented features are used in a underwater pipe following application, creating a comparison environment on the cloud that immerse students in a competition to reach the best possible result. Finally, a first experience in a real educational environment using the proposed tool is detailed, demonstrating the viability and suitability of the proposed tool.

*Keywords:* Robotics, Education, Benchmark examples, Autonomous vehicles, Marine systems, Educational aids, Telerobotics

## 1. Introducción

La simulación de robots altamente precisa es un proceso computacionalmente caro. Requiere un ordenador de altas prestaciones con una tarjeta gráfica para simular el entorno en un tiempo razonable, y más aún si es a tiempo real. Además, la instalación y configuración de todas las librerías necesarias puede desanimar a muchos posibles usuarios, sobre todo si están usando un sistema operativo o dispositivo diferente al recomendado.

Esto es especialmente cierto si se pretende realizar múltiples experimentos para probar muchas configuraciones, situa-

*Autor en correspondencia: japerez@uji.es
**To cite this article:** J. Pérez, D. Fornas, R. Marín, Pedro J. Sanz. 2018. UWSim, an underwater robotic simulator on the cloud as educational tool. Revista Iberoamericana de Automática e Informática Industrial 15 (2018) 70-78. https://doi.org/10.4995/riai.2017.8827

ciones o posibles soluciones. En ese caso, el tiempo necesario se multiplica con las combinaciones de opciones a probar. Por otra parte, es recomendable que el ordenador de simulación esté totalmente dedicado a ello para no afectar a los resultados, de forma que puede quedar ocupado durante muchas horas, o incluso días, si se quiere asegurar una comparación justa de los resultados.

Por último, el uso de la tecnología portable en la sociedad está creando la necesidad de trabajar con dispositivos cada vez mas pequeños como mini portátiles, tablets e incluso teléfonos inteligentes. A pesar de que estos dispositivos no son adecuados para simulación, ya que no suelen contar con tarjetas gráficas dedicadas o suficientes capacidades de computación, estos son comúnmente utilizados. Sin embargo, todos estos dispositivos tienen una ventaja a su favor: la conexión a Internet, con capacidad para alcanzar un cantidad de recursos virtualmente infinita. Por ello, es posible trabajar mientras se está en casa, viajando, en una oficina diferente o casi en cualquier lugar.

Teniendo en cuenta estos hechos, parece evidente que el siguiente paso para los simuladores de robots es proporcionar servicios a través de internet. De esta forma los usuarios son capaces de trabajar desde cualquier dispositivo, en cualquier lugar, sin que se vea afectado el rendimiento del simulador. Para conseguir esto es necesario contar con un servidor de simulación de altas prestaciones, que llevará a cabo el proceso, y una forma de acceder a él. Este tipo de capacidades suelen denominarse como "trabajar en la nube".

Además, un servicio de estas características puede servir para atacar otro problema inherente al mundo de la robótica, la caracterización y comparación de resultados o *benchmarking*. En el caso de la robótica, los resultados son altamente dependientes de la plataforma específica y en muchas ocasiones los artículos publicados no incluyen suficientes detalles para replicar los resultados en otra plataforma. Sin embargo, un servicio que permita ejecutar y almacenar los resultados en la nube de forma que cualquiera pueda compararse e incluir su propio algoritmo a la comparación puede ayudar para un análisis objetivo del resultado.

Desde el punto de vista educativo, este servicio puede considerarse como un aula en internet, que permite enviar la solución a un ejercicio y automáticamente se recibe la puntuación del mismo. Esta respuesta inmediata se puede además dividir en apartados para que el alumno sepa en que falla y sea capaz de aprender y mejorar el resultado sin la intervención directa del profesor.

Por todos estos motivos, en este trabajo se presenta una herramienta que permite simulación online, evaluación y comparación automática de resultados. En detalle, la principal aportación del artículo es la extensión de las capacidades del simulador de robots submarinos UWSim, ver figura 1, para permitir la comparación y evaluación automática de resultados a través de una plataforma online. Además, la utilidad de esta herramienta se muestra en la aplicación a un contexto de enseñanza real, donde los alumnos la utilizan para aprender a teleoperar, navegar y controlar por visión un robot submarino simulado para inspección de tuberías.

El artículo está organizado de la siguiente forma: en la siguiente sección se hace un estado del arte de los servicios de simulación robótica existentes teniendo en cuenta sus posibilidades de evaluación automática y online. En la sección 3 se presenta la plataforma de simulación utilizada en la que se basa el artículo: UWSim. En la siguiente sección se describe la extensión de las capacidades en la nube del simulador. A continuación, en la sección 5, se presenta un caso de uso en el ámbito de la educación centrado en el seguimiento de tuberías. Por último, se aportan ideas de trabajo futuro y conclusiones.

Figura 1: Captura del simulador de robots submarino UWSim.

## 2. Estado del arte

En el ámbito de la robótica existen gran cantidad de simuladores, aunque en el campo de robótica submarina las posibilidades se reducen, en Craighead et al. (2007), Matsebe et al. (2008) y Cook et al. (2014) pueden encontrarse análisis de los mismos. Sin embargo son pocos los que incorporan servicios en la nube que permitan simular de manera remota, aunque es una tendencia creciente. Y si además de simulación submarina online se requieren capacidades de evaluación y comparación automáticas, como la herramienta que se presenta, no existe ninguno del que tengan conocimiento los autores. A continuación, se describen alternativas que cumplen parcialmente con los requisitos, haciendo especial énfasis en las capacidades de evaluacion automática online similares a las presentadas en el artículo.

Una de ellas es *the construct sim*, representada en la figura 2 y presentada en Tellez (2017), capaz de hacer funcionar los simuladores Gazebo y Webots desde cualquier dispositivo sin ningún tipo de instalación. Este es el proyecto más maduro, tan solo requiere un navegador para empezar a interactuar con las simulaciones de robots. Proporciona una gran variedad de robots y entornos con los que empezar además de sus completos webinarios y tutoriales. Sin embargo, este servicio no es gratuito, el precio a pagar va en función de las horas de simulación que se necesitan y las características de CPU y GPU requeridas. Además carece de un sistema de evaluación automático, y las capacidades de simulación de robots submarinos son experimentales.

En Pavin et al. (2015) se propone un simulador de vehículos submarinos autónomos en la nube. En este caso la aplicación se centra en navegación y control ofreciendo pocos sensores y capacidades de simulación muy limitadas. El sistema es capaz de funcionar en cualquier dispositivo a través de una interfaz web reconfigurable. La principal dificultad es que no hay mucha información sobre él, ya que no está disponible para la comunidad científica, imposibilitando su adaptación a cualquier otra aplicación.

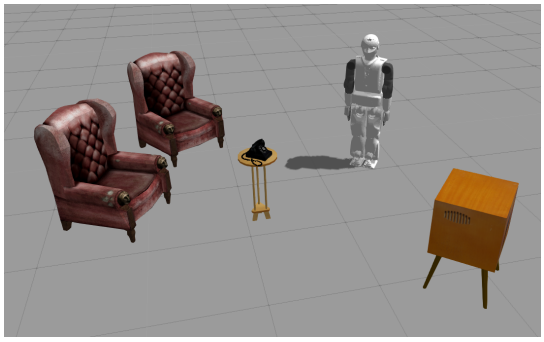

Figura 2: Captura del simulador en la nube the construct sim, completamente funcional a través de un navegador.

Los mantenedores de Gazebo, *Open Source Robotics Foundation* (OSRF), han desarrollado su propia herramienta para crear y controlar simulaciones en la nube. CloudSim, descrito en Foundation (2015), permite lanzar simulaciones en una nube de máquinas de simulación localizadas en internet, dando acceso al usuario para monitorizar e incluso teleoperar el robot. A pesar de que el servicio no se puede utilizar de manera pública, se ofrece el *software* para crear un servicio propio con el simulador Gazebo. Al igual que con *the construct sim* la herramienta carece de opciones de evaluación y comparación automática y la simulación de robots submarinos es limitada.

Otro servicio de simulación de robots disponible por internet es el descrito en Center (2015), RoboBlockly, desarrollado por *UC Davis C-STEM Center*. Este recurso está centrado en enseñar matemáticas a jóvenes estudiantes a través de la robótica. Está basado en robots de Lego y la librería Blockly[1] de Google, la cual genera ejecutables en C++ a partir de bloques visuales en una interfaz web. Este código puede ejecutarse luego tanto en el simulador como en el robot real. A pesar de que esta herramienta comparte la aplicación con la presentada en este artículo, carece de un sistema de comparación y el simulador utilizado es muy básico y poco atractivo para los alumnos.

Otras alternativas como The Robot Programming Network (RPN), presentado en Cervera et al. (2016), ofrece acceso a herramientas educacionales a través de la web en forma de cursos. Está organizado en lecciones de moodle que introducen diferentes simuladores y conceptos a los estudiantes. Dentro de estos cursos se puede directamente crear y probar código en un diálogo integrado. Sin embargo esta herramienta no cuenta por el momento con ningún simulador de robots propio, ni con sistemas de comparación automáticos.

En el marco de la docencia preuniversitaria en España existe EducaControLaboV, presentado en Cerezo y Sastrón (2015). Este laboratorio virtual se centra en diferentes casos de uso para alumnos de Educación Secundaria Obligatoria E.S.O y bachillerato. En cada caso de uso se pretende enseñar conceptos específicos como sistemas mecánicos, control de vehículos o movimiento de brazos robóticos industriales. Aunque la herramienta muestra su utilidad en la docencia preuniversitaria no está pensada para simulaciones complejas o de propósito general como la aquí descrita.

En el contexto de simuladores comerciales de vehículos teleoperados, *Remote Operated Vehicles (ROV)*, también están adaptándose a estos cambios ofreciendo herramientas de simulación web. ROVSim, disponible en LLC (2006), tiene una alternativa web para su simulador capaz de funcionar en un navegador bajo Windows o Mac. Sin embargo, aún le falta compatibilidad en otras plataformas que son muy comunes como Android. Aunque aún le faltan características como evaluacion automática, es un indicador de hacia qué tipo de alternativas se está moviendo el mercado.

En cuanto a la evaluación automática, sin contar con simulación, también existen trabajos previos como la presentada en Gómez-Estern et al. (2010) para mejorar la eficiencia en clases masificadas y poder aumentar el número de prácticas y pruebas. Este tipo de sistemas suelen ser eficaces en materias con una fuerte base teórica, como en el caso de este trabajo: teoría de sistemas y fundamentos de la informática. Pero suele ser más complicado de establecer en campos como la robótica, donde apenas existen trabajos de estas características. Por ello, en este trabajo se propone el uso de métricas que midan el desempeño para una tarea en función del resultado obtenido mediante *benchmarking*.

En la linea de evaluación con métricas existen trabajos como Blasco et al. (2012). En él se describe la simulación y evaluación del concurso en ingeniaria de control 2012 organizado por el comité español de automática. En él se evaluaba el seguimiento de trayectorias de un vehiculo cuatrirrotor mediante los errores a lo largo del seguimiento y el tiempo invertido. En este caso, la herramienta no cuenta con características online ni de simulación.

Otra aproximación a la evaluación de algoritmos para robots en el ámbito de la educación son las competiciones de robots como la FIRST Robotics Competition (FRC) Haynes y Edwards (2015), la RoboCupJunior Eguchi (2016) o a nivel nacional el CEABOT García et al. (2011) organizado por el comité español de automática. Aunque estas competiciones comparten algunas de las caracteristicas buscadas, obviando la simulación, como la evaluación de resultados obviando la simulación, requieren una inversión de dinero para comprar y mantener los robots. Esto en el contexto de robótica submarina es aún mas complicado dada la necesidad de unas instalaciones que permitan el desarrollo de la competición, haciendo en muchos casos inviable el uso de robots reales.

Ninguna de las opciones analizadas ofrece las características que estamos buscando, un simulador de robots, preferiblemente submarinos, accesible por interfaz web y con capacidades de evaluación y comparación automáticas. Por ello, en este artículo se propone una extensión del simulador submarino UWSim para poder además de acceder *online* y ejecutar simulaciones de manera remota, evaluar automáticamente los algoritmos en función de su propósito comparándolos en un entorno que asegura la igualdad de condiciones. Para demostrar la efectividad de la herramienta se describe el uso en un contexto de educación.

[1] Disponible en: https://developers.google.com/blockly/
[2] Disponible en: http://www.irs.uji.es/uwsim

## 3. UWSim: una herramienta de simulación 3D para *benchmarking*

UWSim[2], inicialmente presentado en Prats et al. (2012), es un simulador de robots submarinos de código abierto que permite tanto visualizar una reconstrucción de una intervención real como simular misiones. Está desarrollado en C++ y utiliza OpenSceneGraph (OSG) Osfield et al. (2004) como motor de visualización, osgOcean descrito en Bale (2012) para el océano y el motor de física Bullet presentado en Coumans (2012). OSG es una librería de código abierto para aplicaciones de gráficos en 3D que se ha usado en diferentes ámbitos como simulación, videojuegos, realidad virtual, visualización científica y modelado. Está basado en OpenGL y funciona en la gran mayoría de sistemas operativos incluyendo Microsoft Windows, Mac OS X, Linux e incluso Android. Haciendo uso de esta librería, osgOcean, producto de una iniciativa financiada por la Unión Europea llamada proyecto VENUS, se encarga de implementar una visualización realista del océano. Finalmente, Bullet es un motor de física que simula la detección de colisiones y dinámica. Ha sido usado tanto en videojuegos como en películas animadas y es el motor de física de múltiples simuladores de robots.

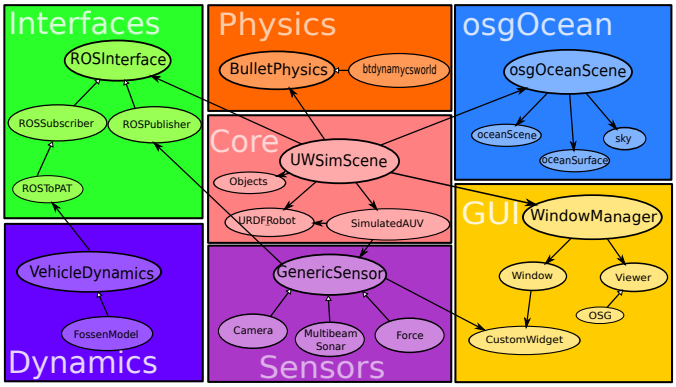


Figura 3: Arquitectura del simulador submarino UWSim.

UWSim utiliza las librerías anteriormente mencionadas y añade funcionalidad para fácilmente utilizar robots submarinos, sensores e interfaces para manejarlos mediante mensajes estándar de ROS, *Robot Operating System*. Inicialmente presentado en Quigley et al. (2009), ROS es un conjunto de librerías que se ha convertido en un estándar de facto para la comunidad robótica y ofrece comunicación entre programas mediante paso de mensajes estándar. En la figura 3 se muestra la arquitectura de UWSim y sus principales componentes. Básicamente se compone de un núcleo encargado de cargar la escena principal y los robots. El módulo de sensores es el responsable de la simulación las cámaras, sonares, etc. La categoría de interfaces gestiona la comunicación con otras arquitecturas. La parte de dinámica simula la física de los vehículos bajo el agua. El módulo de física modela los contactos entre objetos de la escena. osgOcean se encarga de renderizar la superficie y la visión bajo el agua del mar. Por último, la interfaz de usuario (GUI) soporta la visualización y maneja las ventanas.

UWSim ha sido utilizado en diferentes proyectos financiados por la Comisión Europea como MORPH, Kalwa et al. (2012), y PANDORA, Lane et al. (2012), con el fin de realizar *HIL (Hardware In the Loop)*, experimentos mezclando hardware real y simulado, y reproducir misiones reales a partir de los registros o *logs* capturados durante la ejecución de las mismas.

Además del simulador UWSim, se hace uso de un módulo de *benchmarking* que permite evaluar código en función de la misión, presentado por Pérez et al. (2013). Por ejemplo, es capaz de medir el error de detección de un objeto. Es decir, puntúa algoritmos de robótica que actúan sobre el simulador. Para la comunicación con *software* externo utiliza interfaces de ROS. Utilizando esta comunicación lee los resultados de un programa y los compara con el resultado esperado, *ground truth*, del simulador. En función de unas métricas configurables dependiendo del objetivo de la intervención, evalúa el resultado de la misma. Además, en el caso de calibrar correctamente el entorno virtual del simulador con el real, es posible evaluar algoritmos en entornos reales como se describe en Perez et al. (2015).

El funcionamiento del módulo puede verse en la figura 4, recibe como entrada las métricas, activadores y actualizadores y proporciona los resultados de la intervención. Las métricas definen qué se debe valorar en un algoritmo, por ejemplo el tiempo de intervención, el error de detección, las colisiones, etc. Los activadores especifican cuándo se activan las distintas medidas. Es posible que ciertas medidas no tengan sentido en ciertos momentos, por ejemplo las colisiones al agarrar un objeto deben ignorarse entre el objeto a coger y la mano robótica. Por último, los actualizadores de escena describen cómo evoluciona el entorno a lo largo del experimento. Con ellos es posible decidir si las condiciones de visibilidad empeorarán o la luz de la escena en cada momento con el objetivo de evaluar diferentes condiciones. El módulo lanzará automáticamente la simulación con las condiciones configuradas y producirá como salida múltiples ficheros con los resultados de las métricas descritas.

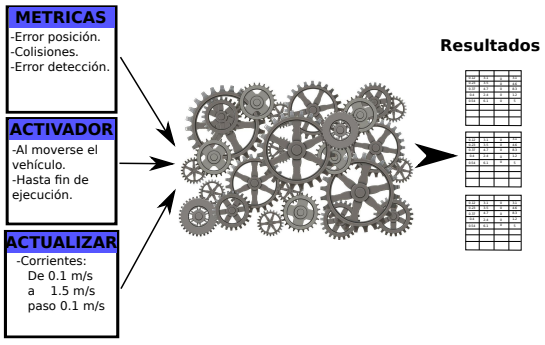


Figura 4: Diagrama de flujo del módulo de *benchmarking*.

En el caso de no poder realizar simulaciones de manera nativa, UWSim dispone de un sistema de ejecución remota[3]. Esta herramienta, presentada en Pérez et al. (2014), crea un servicio accesible a través de una interfaz web para lanzar ejecuciones y recuperar los resultados de las mismas. En la figura 5 se puede ver el funcionamiento de este servicio de ejecución. La comunicación se realiza mediante *rosbridge* a través de la li-

[3]Disponible en: http://robotprogramming.uji.es/UWSim/config

brería *roslibjs*. Esta librería permite el uso de mensajes de ROS estándar en WebSockets que pueden ser directamente enviados a través de la web. En el caso del streaming de video se utiliza *mjpeg_server*, una librería especializada en comunicación de vídeo a través de HTTP usando mensajes de imágen de ROS como fuente. De esta forma el simulador no requiere cambios ya que todo lo que necesita son interfaces de ROS ya existentes, y por tanto es más sencillo de mantener.

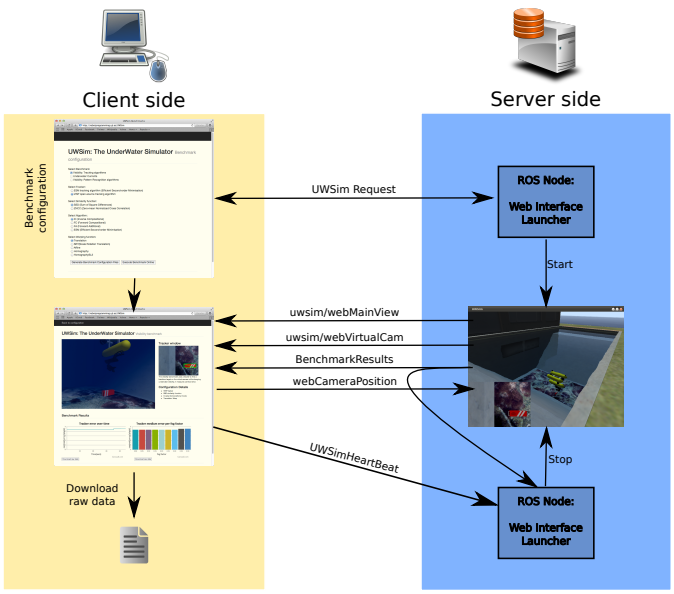


Figura 5: Funcionamiento del servicio de ejecucion de simulaciones en la nube.

El usuario del servicio inicia la ejecución configurando las opciones en una interfaz que permite elegir diferentes tipos de experimentos y configuraciones. Una vez el usuario inicia la ejecución se envía una petición al servidor de simulación que una vez que comprueba su validez lanza la instancia del simulador correspondiente y envía la visualización. El cliente recibe estos datos y puede interactuar navegando por la escena usando el ratón, tal como haría en una ejecución local, y visualizar los resultados del experimento en gráficas interactivas. Por último, el servidor monitoriza la conexión del cliente y del experimento para abortar la simulación cuando el cliente abandone el servicio o la ejecución finalice y así liberar recursos de simulación para otros posibles usuarios.

De esta forma las capacidades de UWSim son accesibles no solo de forma local, sino a través de Internet. Sin embargo presenta una importante carencia para ser de utilidad a la hora de comparar los resultados: la accesibilidad y disponibilidad de los mismos. Es por ello que este artículo se centra en el desarrollo de herramientas automáticas que permiten tratar la salida del simulador automáticamente y hacer accesibles los resultados para compararse con otros usuarios.

## 4. Simulación en la nube

Para hacer esto posible, la arquitectura ha sido ampliada para crear un espacio compartido donde poder publicar los resultados y comparar con otros usuarios del servicio automáticamente. Esta arquitectura permite procesar y subir resultados a la nube automáticamente que pueden ser consultados a través de una interfaz de resultados, *ScoreCard*, con gráficos y tablas relevantes del experimento. Además de esto, es posible reproducir cualquier experimento almacenado en la nube a modo de video en el simulador. Esto permite ver la ejecución de otra persona sin almacenar código ni información sensible del algoritmo empleado para comparar de una manera más subjetiva.

El funcionamiento del servicio, que puede verse en la figura 6, está formado por tres partes principales: el simulador UWSim, un espacio en la nube intermedio y la interfaz de resultados. El simulador UWSim, detallado previamente, es el encargado de generar los resultados utilizando el módulo de *benchmarking* que serán almacenados en la nube.

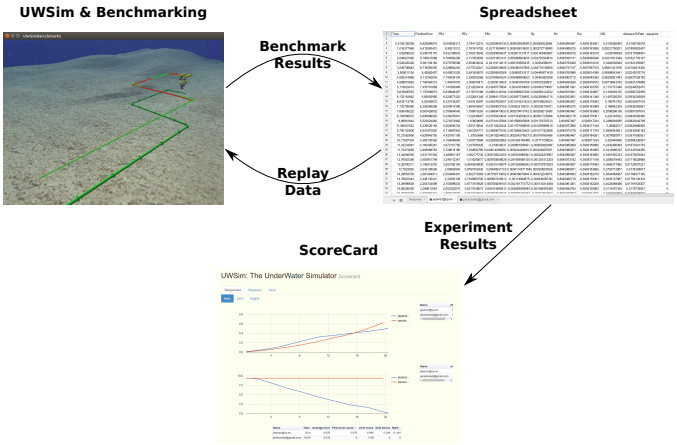


Figura 6: Diagrama de funcionamiento del servicio de *benchmarking* en la nube.

El almacenamiento en la nube está resuelto utilizando hojas de cálculo, *spreadsheets*, de Google a modo de base de datos. Se ha escogido esta opción porque aunque son hojas de cálculo compartidas *online* comunes, permiten introducir y consultar datos de manera remota a través de una API, en este caso la API de Python y de javaScript. Además, la accesibilidad de los datos está asegurada al utilizar un servicio ampliamente soportado en cualquier plataforma que muchos usuarios ya utilizan, pudiendo reutilizar estas cuentas de usuario sin necesidad de crear una nueva.

Teniendo en cuenta esto, se ha desarrollado un módulo que permite utilizar los datos recogidos como *benchmarking* e introducirlos automáticamente en la hoja de cálculo con un formato adecuado para después consultarlos a través de la API. Para ello el módulo de *benchmarking* necesita configurarse con un usuario que identificará al autor de los resultados. Desde el punto de visto del usuario es lo único que se requiere, el módulo de *benchmarking* se encarga automáticamente de enviar los datos a la hoja de cálculo asegurándose de no sobrecargar la comunicación y almacenarlos adecuadamente.

Aunque la hoja de cálculo está compartida y visible para cualquiera con el link correcto, por seguridad los datos en ella están protegidos. Cada vez que un usuario sube sus resultados se protegen automáticamente contra edición de forma que solo el autor puede modificarlos. De esta forma la hoja de cálculo se divide en hojas, que actuarían como las distintas tablas de una base de datos, de las que el autor es el propietario asegurando que los resultados almacenados son de su propiedad y nadie ha podido modificarlos.

Por último, la interfaz web de resultados sirve para observar de una manera gráfica y resumida la evaluación del experimento. Se trata de una web en HTML y *JavaScript* que puede alojarse en cualquier servidor e incluso ejecutarla localmente. Utiliza la librería *Google Charts* para consultar la información de la hoja de cálculo, como si de una base de datos se tratase, y producir resultados visuales. El diseño y tipo de información mostrada depende del tipo de experimento, pero hay un gran abanico de posibilidades como gráficos de líneas, circulares, barras, tablas, etc.

Aparte de esto, el módulo desarrollado también permite visualizar la ejecución de cualquier usuario consultando los resultados almacenados en la nube. Es posible por tanto, observar el movimiento de un vehículo simulado por la escena tal y como lo simuló otro usuario. Sin embargo, no se almacena el código ni detalles, únicamente posiciones e información necesaria para la visualización posterior. Esto es una herramienta potente ya que permite comparar y detectar los puntos débiles y fuertes de cada algoritmo comparado. Para llevarlo a cabo tan solo es necesario especificar el nombre del usuario y el sistema descargará automaticamente los datos necesarios y lanzará la reconstrucción de la simulación.

Todas las herramientas desarrolladas son automáticas, por lo que para crear un nuevo caso de uso tan solo es necesario configurarlas adecuadamente. En concreto, es necesario diseñar los escenarios y tareas para el problema a resolver, definir las métricas adecuadas para evaluar el problema y las gráficas resultantes para mostrar los resultados. Esto puede hacerse simplemente editando ficheros XML sencillos, sin necesidad de programar (a menos que se requiera un sensor no soportado por el simulador u otros cambios mayores), haciendo fácil la reutilización de la herramienta para todo tipo de problemas.

## 5. Caso de uso: seguimiento de tuberías

Las herramientas anteriormente presentadas posibilitan crear un entorno muy adecuado para la robótica educativa. La unión de simulación de robots y evaluación mediante *benchmarking* es muy interesante. Las ventajas de trabajar en un entorno simulado como el presentado anteriormente son múltiples desde la perspectiva de la educación como son:

- **Velocidad de diseño, pruebas, *debug***: Es mucho más rápido programar y comprobar el correcto funcionamiento en un simulador que en un robot. De hecho es raro comenzar el desarrollo de una aplicación robótica directamente sobre un sistema real. Esto es debido a la repetibilidad de los experimentos. En el mundo real existen muchas variables que no se pueden controlar, mientras que en simulación el experimento puede repetirse exactamente en las mismas condiciones.
- **Disponibilidad**: En ocasiones los robots, sensores o instrumentos necesarios para la educación no están disponibles en suficiente cantidad, se averían fácilmente o son demasiado caros como en el caso de la robótica submarina.
- **Aprendizaje continuo**: Es fácil controlar la curva de dificultad de los ejercicios pudiendo simplificarlos obteniendo información del simulador. Por ejemplo en un problema de navegación se pueden desactivar las perturbaciones hasta que el alumno se sienta seguro y activarlas más tarde para aprender a controlarlas.
- **Competitividad**: La posibilidad de compararse al instante con los compañeros hace que los alumnos se motiven para obtener el mejor resultado fomentando la creatividad.
- **Formación específica**: De igual forma que se puede controlar la curva de dificultad omitiendo partes de la solución completa, pueden omitirse partes específicas para centrar el esfuerzo en un punto concreto. Por ejemplo, desactivando la física del robot para centrarse en aprendizaje de algoritmos de visión. Otro ejemplo es obtener posiciones exactas de objetos para aprender algoritmos de agarres.

Es por ello que el contexto utilizado para la validación de la herramienta son los alumnos del master EMARO+[4] (Master Europeo en Robótica avanzada). A modo de introducción a la robótica submarina, se ha creado un entorno para la teleoperación y control de un robot submarino inspeccionando tuberías en busca de averías o fugas. El objetivo es que los alumnos sean capaces de controlar el robot siguiendo la trayectoria de las tuberías de forma que la tubería se recorra completa y manteniendo la visión de la misma dentro de la cámara simulada.

En este entorno se han creado tres escenarios con diferentes dificultades, mostradas en la figura 7, para que la curva de aprendizaje sea más asequible. El primero consiste en un tubería recta únicamente, el segundo incluye subidas y bajadas variando la profundidad de la tubería. Finalmente en el tercer escenario la tubería serpentea por el fondo marino.

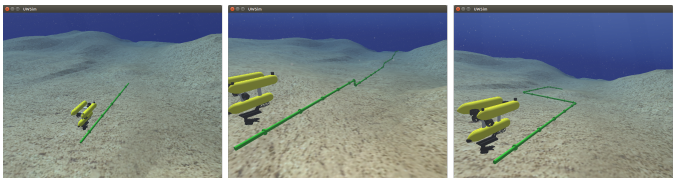

Figura 7: Escenas del entorno de seguimiento de tuberías, de izquierda a derecha escenarios básico, alturas y giros.

Además, también se propone empezar con el problema cinemático, es decir únicamente asignando velocidades al vehículo asumiendo que las alcanzará. Posteriormente se pasará a resolver el mismo problema de manera dinámica, decidiendo la entrada a los distintos motores del robot simulado. Esto introduce el problema al alumno de una manera progresiva, de forma que al ir consiguiendo resultados gradualmente no se desmotive, permitiendo resolver problemas de mayor dificultad.

Por otra parte la resolución del problema se plantea en tres etapas, de forma que cada una de ellas permite concentrarse en un elemento del mismo. En primer lugar los alumnos deben

[4]Mas detalles en: http://emaro.irccyn.ec-nantes.fr/

aprender a controlar el vehículo teleoperandolo, con ello se pretende que aprendan a enviar y leer mensajes con la plataforma robótica ROS.

A continuación se les proporciona una lista de puntos por la que el vehículo debe pasar para hacer la inspección de la tubería. Con ella se debe hacer una navegación autónoma lo más precisa posible. De esta forma los alumnos pueden concentrarse en el control puro abstrayéndose de la visión u otros problemas derivados. Finalmente se enfrentan al problema completo añadiendo un sistema de visión. Utilizando el control diseñado anteriormente, el reto es generar una trayectoria adecuada utilizando la cámara disponible. Dada la complejidad del sistema completo guiado por visión y con la dinámica del agua y la brevedad del curso, este último ejercicio únicamente se plantea como opcional.

Para tener una referencia visual de lo que está ocurriendo mientras se ejecuta el código unas líneas muestran la trayectoria ideal y la que está siguiendo el vehículo tal y como se ve en la figura 8. La línea verde muestra la trayectoria ideal, mientras que la roja es la que está siguiendo el vehículo. Esto permite a los alumnos decidir si están en el camino correcto de una forma muy rápida antes de obtener la evaluación del mismo.

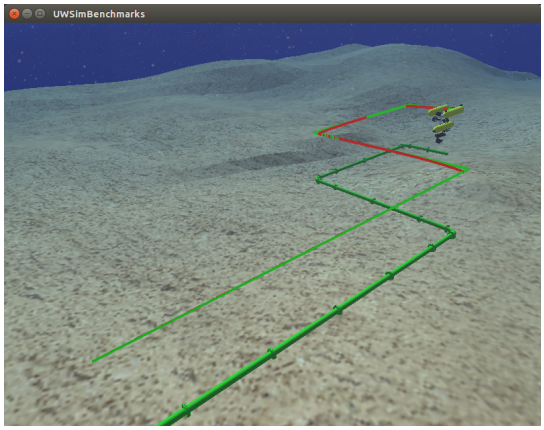


Figura 8: Captura de seguimiento de una tubería siendo evaluado. La línea verde muestra la trayectoria ideal y la roja la seguida

Finalmente, para que el trabajo con toda esta arquitectura sea rápido, se ofrece, además de las instrucciones detalladas de instalación, una máquina virtual[5] con todo el software de forma que el alumno puede ponerse a trabajar de forma inmediata.

### 5.1. Resultados

A través de la plataforma desarrollada el desempeño de las soluciones alcanzadas puede ser automáticamente evaluado conforme a una métrica. En este caso se utiliza una medida de seguimiento de trayectorias típica de algoritmos de control, el error integrado cuadrático (ISE), y el error final. La medida ISE es la suma de distancias a la trayectoria ideal, la cual es dos metros sobre la tubería, a lo largo del tiempo. Esta medida es inversamente proporcional a la calidad del seguimiento, ya que tiene en cuenta tanto el tiempo empleado como la precisión. Aumenta rápidamente cuando la distancia del vehículo respecto a la trayectoria idónea incrementa, pero también penaliza el seguimiento demasiado lento de la trayectoria.

Por otra parte es necesario saber si se ha alcanzado el final de la tubería. Para ello se calcula la distancia de la posición final del vehículo respecto al final de la tubería. De esta forma se puede detectar si el vehículo ha perdido la tubería a mitad de seguimiento o ha detectado el final erróneamente. Teniendo en cuenta estas medidas la evaluación final se calcula con la ecuación 1.

$$puntuacion = \left(1 - error^2\right) * \left(0{,}1 - \frac{errormedio^2}{0{,}1}\right) + \frac{tiempo - ref}{100} \quad (1)$$

Donde *error* es el error final, *errormedio* es la media de error a lo largo del seguimiento obtenida a partir de ISE, *tiempo* es lo que ha costado realizar la intervención y $ref$ es la referencia de tiempo de cada escenario calculada a partir de la distancia recorrida, giros y cambios de altura. El primer término de la ecuación evalúa la posición final del vehículo penalizando aquellos casos en los que el vehículo está lejos del objetivo. El segundo término evalúa el error medio a lo largo de la trayectoria. El último término es un bonus que beneficia los seguimientos rápidos y penaliza los lentos en función de la complejidad del recorrido de la tubería. Por ejemplo, en el caso de la tubería recta el tiempo objetivo es bajo en comparación con la que tiene giros.

Además, el *software* permite observar la evolución de esta medida a lo largo del tiempo y representarla en forma de gráfica. De esta forma se puede ver si ha ido creciendo constantemente o hay algún punto problemático durante el seguimiento, como los giros.

Para mostrar a los alumnos el resultado final que se espera, los resultados de una solución al problema se incluyen en la comparación [6]. De esta forma se puede consultar cómo se mueve el vehículo si se implementan correctamente los conceptos explicados y sirve de motivación para obtener mejores resultados.

Los resultados están disponibles en una página web [7] que permite que los alumnos puedan consultar sus resultados en cualquier momento. Como se puede ver en la figura 9 las puntuaciones están disponibles y motivan a mejorar el algoritmo desarollado para superar el resultado del resto de compañeros y el profesor.

En esta figura se muestran las gráficas interactivas de resultados de ISE en la parte superior y error final en la inferior. Además se incluye una tabla interactiva con el resumen de resultados, el cual detalla las puntuaciones intermedias de cada apartado como son error medio, error final y bonus de tiempo y la final(*mark*). Los resultados mostrados corresponden al ejercicio de seguimiento de puntos sin visión para el escenario básico. El resto de resultados pueden consultarse en la página web.

[5] Disponible en: https://goo.gl/Qm6K8j
[6] Disponible en https://goo.gl/VreM8a
[7] http://www.irs.uji.es/uwsim/files/benchmarks/pipefollowing.html


| Name | Time | Average error | Final error score | error score | time bonus | Mark |
|---|---|---|---|---|---|---|
| japerez@uji.es | 11.869 | 0.02 | 0.997 | 0.996 | 0.131 | 11.243 |
| al07 | 9.702 | 0.081 | 0.958 | 0.934 | 0.153 | 10.477 |
| al04 | 18.344 | 0.099 | 0.999 | 0.902 | 0.067 | 9.674 |
| al05 | 21.434 | 0.025 | 0.923 | 0.994 | 0.036 | 9.533 |
| al06 | 11.235 | 0.107 | 0.756 | 0.885 | 0.138 | 8.068 |
| al01 | 45.553 | 0.083 | 0.996 | 0.931 | -0.206 | 7.217 |
| al03 | 61.219 | 0.035 | 0.937 | 0.987 | -0.362 | 5.635 |
| al02 | 8.685 | 0.227 | 0.773 | 0.485 | 0.163 | 5.375 |
| al08 | 12.26 | 0.301 | 0.54 | 0.093 | 0.127 | 1.775 |

Figura 9: Resultados de seguimiento de tuberías dinámico mediante waypoints para el escenario básico utilizando la simulación en la nube.

Con estos datos se puede observar los diferentes perfiles de alumnos del master EMARO+ donde se aplicó la plataforma de aprendizaje. Los alumnos de este master provienen de diferentes ingenierias y dependiendo de qué ingeniería provienen han utilizado diferentes estrategias para seguir la tubería. Por un lado, los alumnos “al01” y “al03”, con un perfil más cercano al control automático preciso, han diseñado controladores más exactos pero que necesitaban mas tiempo para completar el recorrido. Mientras que los alumnos “al02” y “al08”, con menos conocimientos sobre controladores, han decidido primar la velocidad sin dar mucha importancia a la precisión.

Aunque ambas estrategias son válidas y resuelven el problema, olvidan una parte importante del mismo y son penalizadas por la medida de error. En el primer caso el tiempo penaliza en gran medida el resultado final y en el segundo es el error medio el que empeora el resultado final. Es por esto que la marca final se queda en un rango intermedio en estos casos.

Los mejores resultados son los que adoptan estrategias equilibradas que no descuidan la precisión ni la velocidad como el caso de los alumnos “al07”, “al06”, “al05” y “al04”. Es por ello que se puede concluir que la métrica evalua el seguimiento de la tubería adecuadamente, ya que puntua mejor las estrategias que se buscan.

Por último, es importante destacar que aunque con diferentes resultados, todos los alumnos completaron con éxito las tareas requeridas. Conceptos complejos, como controladores dinámicos en cascada, surgieron naturalmente usando un ejemplo de uso y las herramientas adecuadas. Además, tres de los ocho alumnos decidieron intentar la versión dinámica del problema de visión, la cual no entraba dentro de lo requerido, y consiguieron completar una solución a un problema complejo dentro de la robótica submarina.

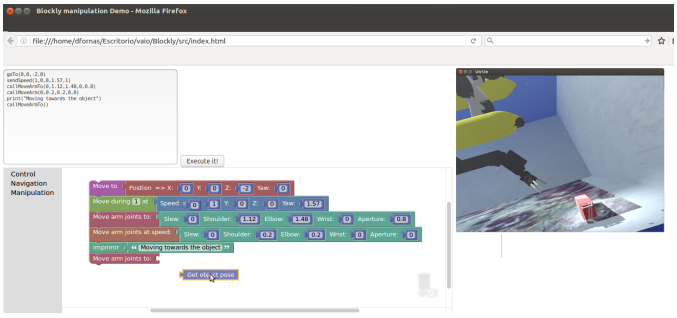


Figura 10: Interfaz de programación visual con Blockly para el simulador UWSim.

## 6. Trabajo futuro

Con el objetivo de aumentar las posibilidades de aprendizaje del simulador, se está desarrollando un entorno de programación visual que facilite la familiarización con el simulador. Este entorno consiste en una interfaz de programación visual que genera un código Python, que se puede ejecutar en la plataforma *online* como se puede ver en la figura 10.

La programación visual se realiza con la herramienta Blockly, que permite utilizar tanto bloques genéricos (*if*, *while*, condiciones, variables...) como bloques personalizables mediante *JavaScript*, mover el robot a una posición por ejemplo, para obtener código ejecutable en un lenguaje de programación, en este caso Python.

Además de la plataforma *online* se está desarrollando una arquitectura con una serie de servicios de ROS para comunicar el código *Python* generado con la interfaz del vehículo y el manipulador ya disponibles en C++. Con esta herramienta, se pueden desarrollar ejercicios para que los alumnos prueben las capacidades del simulador.

Finalmente, otra posibilidad de mejora consiste en mostrar información de los sensores del robot para que puedan ver su evolución en el tiempo. Esto junto con la arquitectura presentada, configura un entorno de aprendizaje con muchas ventajas.

## 7. Conclusiones

Si bien la demanda por la robótica crece cada día, no es menos cierto que aparecen algunos impedimentos que lastran su impacto social como la complejidad inherente a dichos sistemas y los costes asociados para según que aplicaciones.

Para facilitar el progreso de estas tecnologías a nivel social parece indiscutible el rol que debiera jugar en la educación de nuestros jóvenes, la mejor comprensión de los sistemas robóticos.

Una clara alternativa que resulta atractiva a la par que económica son los entornos de simulación. Se puede afirmar que los simuladores son herramientas recomendables para el aprendizaje ya que incrementan la velocidad de diseño, repetibilidad de los experimentos y disponibilidad de recursos. Además pueden ser usados de manera complementaria a plataformas físicas de manera que mantengan el interés y la motivación de los estudiantes. Por otra parte, el uso de herramientas que permitan evaluar el progreso de los alumnos ayuda a la

comprensión de los resultados, facilitando la conexión deseable entre enseñanza y aprendizaje.

Lo anterior se torna más interesante si se añade la disponibilidad *online* de la simulación. Los simuladores robóticos ejecutados localmente requieren un ordenador de altas prestaciones para obtener resultados satisfactorios. Sin embargo, la posibilidad de ejecución remota abre las puertas a la simulación desde cualquier dispositivo o lugar. Esto unido a la comparación de resultados en la nube conforma un entorno ideal para una herramienta educacional, ya que un robot real similar al utilizado en la simulación es muy caro y en consecuencia no está disponible para los alumnos.

Como caso de estudio, en el presente trabajo se muestra un entorno simulado para el seguimiento de tuberías pudiendo cuantificar los resultados obtenidos. Además la dificultad de este entorno es gradual, pudiendo segmentar el desarrollo y consiguiendo resultados de manera progresiva, reduciendo por tanto la curva de aprendizaje. Además, como la herramienta es accesible *online*, se pueden consultar los resultados de manera remota y comparar con otros alumnos o incluso con el profesor. De esta forma se posibilita e impulsa un mecanismo competitivo con el objetivo de alcanzar el mejor resultado posible, motivando el aprendizaje de nuevas estrategias que permitan la superación constante sobre los resultados iniciales.

## Agradecimientos

Este trabajo ha sido parcialmente financiado por el Ministerio de Economía y competitividad, código de proyecto DPI2014-57746-C3 (proyecto MERBOTS), por la Generalitat Valenciana GVA, con el código de proyecto PROMETEO/2016/066 y por la Universidad Jaume I,proyecto MASUMIA, becas PREDOC/2012/47 y PREDOC/2013/46.

## Referencias

Bale, K., 2012. osgocean.

Blasco, X., García-Nieto, S., Reynoso-Meza, G., 2012. Control autónomo del seguimiento de trayectorias de un vehículo cuatrirrotor. simulación y evaluación de propuestas. Revista Iberoamericana de Automática e Informática Industrial RIAI 9 (2), 194–199.

Center, U. D. C.-S., 2015. Roboblockly.
URL: http://roboblockly.ucdavis.edu/

Cerezo, F., Sastrón, F., 2015. Laboratorios virtuales y docencia de la automática en la formación tecnológica de base de alumnos preuniversitarios. Revista Iberoamericana de Automática e Informática Industrial RIAI 12 (4), 419–431.

Cervera, E., Martinet, P., Marin, R., Moughlbay, A. A., del Pobil, A. P., Alemany, J., Esteller, R., Casañ, G., 2016. The robot programming network. Journal of Intelligent & Robotic Systems 81 (1), 77–95.
URL: http://dx.doi.org/10.1007/s10846-015-0201-7
DOI: 10.1007/s10846-015-0201-7

Cook, D., Vardy, A., Lewis, R., 2014. A survey of auv and robot simulators for multi-vehicle operations. En: 2014 IEEE/OES Autonomous Underwater Vehicles (AUV). IEEE, pp. 1–8.

Coumans, E., 2012. Bullet physics engine.

Craighead, J., Murphy, R., Burke, J., Goldiez, B., 2007. A survey of commercial & open source unmanned vehicle simulators. En: Proceedings 2007 IEEE International Conference on Robotics and Automation. IEEE, pp. 852–857.

Eguchi, A., 2016. Robocupjunior for promoting stem education, 21st century skills, and technological advancement through robotics competition. Robotics and Autonomous Systems 75, 692–699.

Foundation, O. S. R., 2015. Cloudsim.

García, J. C., Sanz, P. J., Cervera, E., 11/2011 2011. Using humanoids for teaching robotics and artificial intelligence issues. the uji case study. En: III Workshop de robótica: robótica experimental. Sevilla (Spain).

Gómez-Estern, F., López-Martínez, M., de la Peña, D. M., 2010. Sistema de evaluación automática víaweb en asignaturas prácticas de ingeniería. Revista Iberoamericana de Automática e Informática Industrial RIAI 7 (3), 111–119.

Haynes, C., Edwards, J., 2015. First robotics competition [competitions]. Robotics & Automation Magazine, IEEE 22 (1), 8–10.

Kalwa, J., Pascoal, A., Ridao, P., Birk, A., Eichhorn, M., Brignone, L., Caccia, M., Alvez, J., Santos, R., 2012. The european r&d-project morph: Marine robotic systems of self-organizing, logically linked physical nodes. IFAC Proceedings Volumes 45 (5), 349–354.

Lane, D. M., Maurelli, F., Kormushev, P., Carreras, M., Fox, M., Kyriakopoulos, K., 2012. Persistent autonomy: the challenges of the pandora project. IFAC Proceedings Volumes 45 (27), 268–273.

LLC, M. S., 2006. Rovsim.

Matsebe, O., Kumile, C., Tlale, N., 2008. A review of virtual simulators for autonomous underwater vehicles (auvs). IFAC Proceedings Volumes 41 (1), 31–37.

Osfield, R., Burns, D., et al., 2004. Open scene graph.

Pavin, A., Inzartsev, A., Eliseenko, G., Lebedko, O., Panin, M., 2015. A reconfigurable web-based simulation environment for auv. En: OCEANS 2015-MTS/IEEE Washington. IEEE, pp. 1–7.

Pérez, J., Sales, J., Marín, R., Cervera, E., Sanz, P. J., 09/2014 2014. Configuración y ejecución de benchmarks de intervención robótica submarina en uwsim mediante herramientas web. En: XX Jornadas de Automática 2014.

Perez, J., Sales, J., Penalver, A., Fornas, D., Javier Fernandez, J., Garcia, J. C., Sanz, P. J., Marin, R., Prats, M., 2015. Exploring 3-d reconstruction techniques: A benchmarking tool for underwater robotics. Robotics & Automation Magazine, IEEE 22 (3), 85–95.

Pérez, J., Sales, J., Prats, M., Martí, J. V., Fornas, D., Marín, R., Sanz, P. J., 2013. The underwater simulator uwsim-benchmarking capabilities on autonomous grasping. En: ICINCO (2). pp. 369–376.

Prats, M., Pérez, J., Fernández, J., Sanz, P., 2012. An open source tool for simulation and supervision of underwater intervention missions. En: Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on. pp. 2577–2582.
DOI: 10.1109/IROS.2012.6385788

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A. Y., 2009. Ros: an open-source robot operating system. En: ICRA workshop on open source software. Vol. 3. Kobe, p. 5.

Tellez, R., 2017. A thousand robots for each student: Using cloud robot simulations to teach robotics. En: Robotics in Education. Springer, pp. 143–155.