https://doi.org/10.17979/spudc.9788497498081.0346

UWSim, un simulador submarino como herramienta educacional.

J. Pérez, D. Fornas, R. Marín , P.J. Sanz
Dep. de Ingeniería y Ciencia de los Computadores, Universidad Jaume I, Castellón de la Plana, España
{japerez, dfornas, rmarin, sanzp}@uji.es

Resumen

Debido a la creciente presencia de la robótica en los medios de comunicación se abren oportunidades a la educación utilizando este auge como factor motivador. En este trabajo se hace un análisis de las distintas formas de canalizar la motivación de los estudiantes para iniciar una carrera de futuro haciendo énfasis en las posibilidades de los simuladores. Además se propone un entorno de aprendizaje con un simulador con capacidades de benchmarking *como eje central del sistema destacando las ventajas de este tipo de herramientas en el autoaprendizaje. Por último se detallan mejoras al sistema propuesto con un simulador online integrando programación con blockly para un entorno aún más amigable.*

Palabras clave: Robótica educacional, simulador, UWSim, blockly, *benchmarking*.

1. Introducción

En la actualidad el interés por la robótica es cada vez mayor, a nivel mundial surgen revistas comerciales y programas de televisión sobre ello. No hay duda de que muchos niños, adolescentes e incluso adultos se sienten atraídos por este mundo. De hecho las ventas de juguetes y kits de construcción de robots están alcanzando niveles muy altos.

Es por esto que el creciente interés en el mundo de la robótica puede servir como motivador a la hora de enseñar. No solo es útil en el caso de clases ya relacionadas con el mundo de la robótica, también sirve para aprender habilidades sociales, de equipo, creatividad, ... Existen numerosos estudios sobre este tema como [1] [2] [3] [4] o [5].

A consecuencia de esto han surgido gran cantidad de competiciones robóticas orientadas a estudiantes de todas las edades. El programa FIRST consta de varias categorías desde 6 a 18 años como muestra la imagen 1. FIRST LEGO LEAGUE junior para niños de 6 a 10 años donde aprenden a diseñar y usar robots de LEGO.

Figura 1: Imagen de las distintas categorías del programa FIRST, de izquierda a derecha, arriba a abajo: FIRST LEGO league junior, FIRST Tech Challenge, FIRST LEGO league y FIRST Robotics Competition.

La FIRST LEGO League está enfocada a jóvenes de 9 a 14 años, compiten diseñando, construyendo y programando robots de LEGO mindstorms en diferentes desafíos. La FIRST Tech Challenge reúne a grupos de más de 10 adolescentes de 12 a 18 años que diseñan, construyen y operan soluciones robóticas para distintos juegos que motivan la creatividad.

La FIRST Robotics Competition (FRC) [6] es una competición para estudiantes de instituto de 14 a 18 años en la que equipos de 25 o más jóvenes trabajan juntos para desarrollar en seis semanas robots de tamaño industrial para jugar a un juego diferente cada año. Este año el reto era crear un equipo de robots para asediar el castillo del equipo contrario sorteando fosos y lanzando balones a las torres.

Otro ejemplo es la RoboCupJunior [7], versión de estudiantes de la RoboCup, iniciada en el año 2000 es una competición para estudiantes de hasta 19 años con tres disciplinas fútbol, rescate y baile.

A nivel nacional también existen otras como el CEABOT [8], concurso de robots humanoides organizado por el comité español de automática desde el 2006 para alumnos de grado y postgrado de las universidades españolas. En él compiten con robots humanoides en diferentes pruebas como ca-

rrera de obstáculos, subir escaleras, visión y las ya populares luchas de sumo.

El presente artículo se estructura de la siguiente manera: en la sección siguiente se resumen las ventajas de utilizar simuladores robóticos en la educación. A continuación se detalla las herramientas de simulación utilizadas para crear el entorno educativo propuesto. La sección 4 habla de UWSim como herramienta educativa y el ejemplo de uso. Finalmente se proponen unas líneas de trabajo futuro y unas conclusiones.

2. Simulación de robots para educación

Dado que el trabajo con robots reales en educación ofrece grandes ventajas en términos de motivación y potenciar futuras carreras técnicas es razonable pensar que la simulación también será atractiva.

A pesar de que la ausencia de robots físicos es una clara desventaja, la abstracción del hardware es interesante desde el punto de vista del profesor por varios motivos:

- Velocidad de diseño, test, 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 cero en un robot.
- Repetibilidad de los experimentos: En la vida real existen muchas variables que no podemos controlar, esto hace que un mismo código pueda funcionar sólo bajo determinadas condiciones. En un simulador esto no ocurre.
- 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. Por ello es más conveniente simularlos.

Sin embargo a parte de la pérdida de interés de los alumnos debido a la falta de una plataforma física, los simuladores abstraen de la impredicibilidad del mundo real. Aunque la simulación es cada vez más potente y añade patrones de ruido, eventos aleatorios, ... no es capaz de reflejar comportamientos inesperados que suceden en la realidad.

Es por ello que a menudo se afirma que la mejor solución es una combinación de ambos, un simulador para las fases iniciales y una plataforma robótica compartida por varios estudiantes. Para ello es necesario que el código desarrollado para uno pueda ser utilizado de forma transparente en el otro.

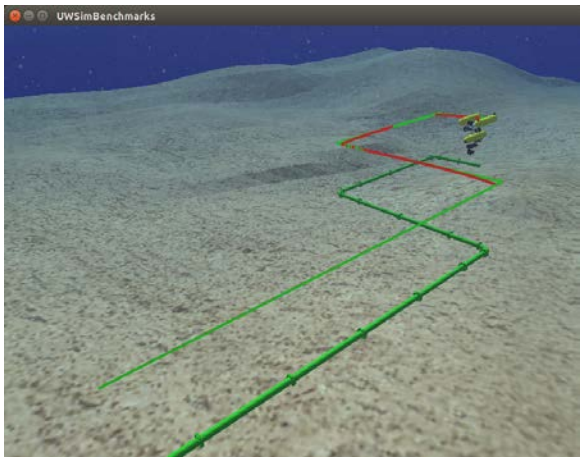

Figura 2: *Benchmark* de seguimiento de tuberías, la línea verde indica la trayectoria ideal y la roja la seguida por el vehículo.

De esta forma con un presupuesto reducido puede alcanzarse un compromiso entre las facilidades que proporcionan los simuladores y la motivación y aprendizaje extra de las plataformas robóticas.

Bajo esta premisa surgió la arquitectura player stage [9] pudiendo utilizar el mismo código desarrollado en simulación en el entorno real. Este simulador más tarde evolucionó en el conocido simulador Gazebo [10].

Ejemplo de ello es USARSim [11], el cual se está utilizando para iniciarse en la Robocup como herramienta de simulación. Una vez se está satisfecho con los resultados alcanzados en simulación se puede pasar a ejecutar en el robot real.

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

UWSim[1] [12] es una herramienta de software de código abierto para la visualización y la simulación de misiones robóticas submarinas y que permite conectarse con un módulo de *benchmarking* (ver Figs. 3 y 2). El software es capaz de visualizar escenarios virtuales bajo el agua que se pueden configurar utilizando software de modelado estándar y se puede conectar a programas de control externos mediante el uso de interfaces de ROS.

UWSim ha sido utilizado en diferentes proyectos financiados por la Comisión Europea (MORPH [13] y PANDORA [14]) con el fin de realizar *HIL (Hardware In the Loop)* y experimentos para reproducir misiones reales a partir de los registros o *logs* capturados durante la ejecución de las mis-

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

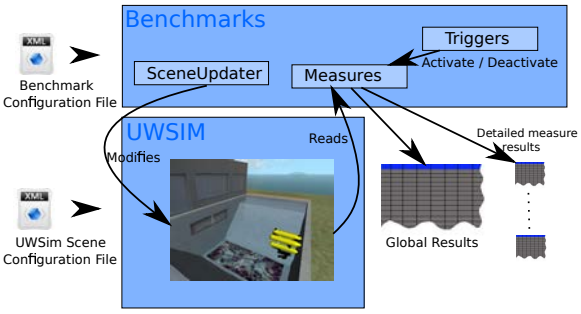

Figura 3: Diagrama de flujo del módulo de *benchmarking*: la configuración del *benchmark* se carga en el módulo y la escena en el simulador UWSim. A continuación, el sistema produce resultados que se guardan para un posterior análisis.

mas.

Recientemente, se ha desarrollado un módulo de *benchmarking* para UWSim [15]. Al igual que UWSim, este módulo utiliza ROS para interactuar con cualquier software externo. Utilizando esta comunicación el módulo permite evaluar un programa externo bajo ciertos criterios: *benchmarking*. Para ello utiliza diversas métricas configurables dependiendo del tipo de misión. Por ejemplo en el caso de búsqueda de un objeto, una métrica adecuada es la diferencia de posición del objeto en el simulador y la obtenida por el algoritmo a evaluar. Además en el caso de calibrar correctamente el entorno virtual del simulador con el simulado es posible evaluar algoritmos en entornos reales como en [16]. La información detallada sobre cómo configurar y ejecutar un *benchmark* en UWSim se puede encontrar en [17].

Además UWSim cuenta con una herramienta *online* [2] para la ejecución remota de *benchmarks* [18]. Mediante esta herramienta se asegura que los distintos algoritmos comparados se ejecutan en condiciones de igualdad de hardware, mismo procesador, gráfica... Además de la misma configuración de escena, objetos, sensores, ruido, condiciones iniciales... Ofreciendo unas condiciones de objetividad y replicabilidad óptimas para la comparación de distintas soluciones.

4. UWSim como herramienta educativa

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

- Motivación: Los simuladores robóticos motivan a los estudiantes debido a la aproximación a un problema real tal y como se demuestra en los estudios citados anteriormente.
- Aprendizaje progresivo: Es fácil controlar la curva de dificultad de los ejercicios pudiendo simplificarlos obteniendo información del simulador. Por ejemplo se puede utilizar la posición absoluta para desarrollar controladores, y más tarde extender el trabajo obteniendo la posición mediante visión.
- Competitividad: La autoevaluación mediante *benchmarking* de las soluciones desarrolladas fomenta la competitividad y por tanto el afán de superación.
- Autoaprendizaje: El hecho de poder evaluar de forma sencilla y automática cada cambio a la solución aplicada provoca que los alumnos sean capaces de aprender y experimentar de una forma más autónoma.
- 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.

Por todas estas ventajas que ofrece UWSim junto con el módulo de *benchmarking* 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 terminen desarrollando un código con el que el robot sea capaz de seguir la tubería autónomamente a una distancia especificada.

En este contexto se han creado tres escenarios con diferentes dificultades (ver figura 4) para que la curva de aprendizaje sea más asequible. El primero consiste en un tubería recta únicamente, el segundo contiene subidas y bajadas variando la profundidad de la tubería. Finalmente en el tercer escenario la tubería serpentea por el fondo submarino.

Además también se propone empezar con el problema cinemático, es decir únicamente asignando velocidades al vehículo asumiendo que las alcanzará. Posteriormente se pasará a resolverse el mismo problema de manera dinámica decidiendo la entrada a los distintos motores del robot simulado. Esto permite conseguir resultados de una manera

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

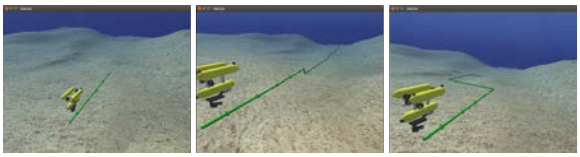

Figura 4: Escenas del entorno de seguimiento de tuberías, de izquierda a derecha escenarios básico, alturas y giros.

más rápida motivando al alumno a continuar con problemas más difíciles.

Por otra parte los ejercicios propuestos también se realizan de manera incremental, es decir parten de una solución sencilla de una parte específica para llegar a conseguir una solución completa al problema. En primer lugar se propone conseguir teleoperar el robot mediante órdenes de teclado, de esta forma el alumno puede familiarizarse con el entorno y el problema.

A continuación se propone utilizar como entrada waypoints de la tubería y una posición global del vehículo para aprender a controlar los movimientos del vehículo. Por último se utiliza una cámara simulada para detectar la tubería y navegar siguiendola. Este último se centra en el sistema de visión utilizando los conocimientos adquiridos en los anteriores.

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 2. Esto permite a los alumnos decidir si están en el camino correcto de una forma muy rápida antes de obtener la evaluación del mismo.

Finalmente para que el trabajo con toda esta arquitectura sea sencillo, se ofrece además de las instrucciones detalladas de instalación, una máquina virtual [3] con todo el software preinstalado de forma que el alumno puede ponerse a trabajar de forma inmediata.

4.1. Resultados

Para obtener los resultados de los ejercicios propuestos se propone utilizar el módulo de *benchmarking* del simulador. Este módulo mide automáticamente el desempeño del algoritmo a evaluar obteniendo la información del simulador. Dependiendo del objetivo del software que se esté utilizando se pueden configurar diferentes medidas.

En este caso se utiliza una medida de seguimiento de trayectorias típica de algoritmos de control: el

[3]Disponible en: https://goo.gl/dmYu9S

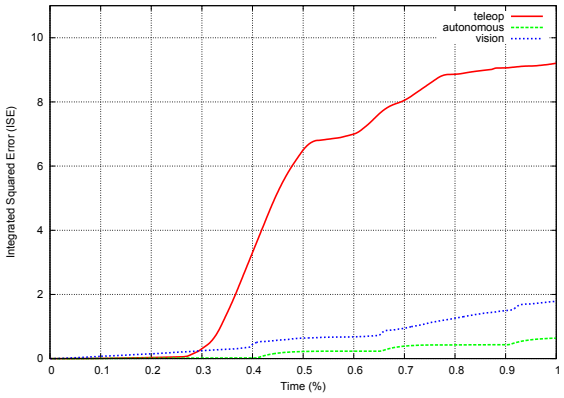

Figura 5: Resultados de algoritmos de seguimiento de tuberías: teleoperado, autónomo y guiado por visión.

error integrado cuadrático. En otras palabras se ha utilizado la suma de distancias a la trayectoria ideal, dos metros sobre la tubería, a lo largo del tiempo.

Esta medida es adecuada para este caso ya que aumenta rápidamente cuando la distancia del vehículo respecto a la trayectoria idónea incrementa, pero también penaliza el seguimiento demasiado lento de la trayectoria. En términos generales cuanto menor ISE mejor seguimiento de la trayectoria se ha realizado.

Además el módulo de benchmarking 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.

Por ejemplo en la figura 5 se muestra el resultado del algoritmo de seguimiento en el caso teleoperado, autónomo y guiado por visión para el escenario de giros. Para ser más fácilmente comparable la escala de tiempo ha sido ajustada para cada caso siendo esta un porcentaje, sin embargo la medida penaliza a los más lentos ya que es una integral del tiempo. El seguimiento autónomo es el más rápido 25 segundos, el teleoperado tarda 83 y el guiado por visión 112, pero en términos de error medio la teleoperación obtiene 0.10 metros mientras que autónomo es de 0.024 y guiado por visión 0.023.

Como se puede observar la medida ISE penaliza mucho la ejecución más lenta de visión que aunque obtiene un error medio menor acaba siendo peor que el autónomo. La teleoperación ofrece el peor resultado final. Esto es coherente con la percepción inicial del problema en el que la teleoperación es más lenta y menos precisa debido a que el piloto tiene peor tiempo de reacción y menor pre-

Figura 6: Interfaz de programación visual con Blockly para el simulador UWSim.

cisión. Por otra parte la visión acaba obteniendo peor resultado ya que el tiempo de procesado de imagen y la necesidad de adaptarse dinámicamente a la forma de la tubería en lugar de conocerla previamente afectan al tiempo de la misión.

5. Trabajo en progreso

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 submarino, el robot y su manipulador. Este entorno consiste en una interfaz de programación visual que genera un código Python, que se puede ejecutar en la plataforma online como se vé en la figura 6.

La programación visual se realiza con la herramienta Blockly [4], que permite utilizar tanto bloques genéricos (*if*, *while*, condiciones, variables...) como bloques personalizables, 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 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 virtuales del robot para que puedan ver su evolución en el tiempo. La integración con la arquitectura de benchmarks también se considera.

6. Conclusiones

Los simuladores son una herramienta recomendable 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 que mantengan el interés y la motivación de los estudiantes. Además el uso de herramientas que permitan evaluar el desempeño de los alumnos ayuda a la comprensión de los resultados y por tanto a un mejor aprendizaje.

Por ello, en el presente trabajo se muestra un entorno simulado para el seguimiento de tuberías pudiendo medir los resultados obtenidos. Además la dificultad de este entorno es gradual pudiendo segmentar el desarrollo consiguiendo resultados de manera progresiva y reduciendo por tanto la curva de dificultad.

Sin embargo aún existe margen de mejora, la posibilidad de incorporar un sistema online es muy prometedora. Además permite integrar herramientas de programación con bloques que facilitan la asimilación de conceptos.

Agradecimientos

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

Referencias

[1] J. Johnson, "Children, robotics, and education," *Artificial Life and Robotics*, vol. 7, no. 1-2, pp. 16–21, 2003.

[2] R. Goldman, A. Eguchi, and E. Sklar, "Using educational robotics to engage inner-city students with technology," in *Proceedings of the 6th international conference on Learning sciences*. International Society of the Learning Sciences, 2004, pp. 214–221.

[3] D. Alimisis, "Educational robotics: Open questions and new challenges," *Themes in Science and Technology Education*, vol. 6, no. 1, pp. 63–71, 2013.

[4] F. B. V. Benitti, "Exploring the educational potential of robotics in schools: A systematic review," *Computers & Education*, vol. 58, no. 3, pp. 978–988, 2012.

[5] A. Eguchi, "Educational robotics theories and practice: Tips for how to do it right," *Robotics: Concepts, Methodologies, Tools, and Applications: Concepts, Methodologies, Tools, and Applications*, p. 193, 2013.

[6] C. Haynes and J. Edwards, "First robotics competition [competitions]," *Robotics &* Au-

[4] Disponible en: https://developers.google.com/blockly/

tomation Magazine, IEEE, vol. 22, no. 1, pp. 8–10, 2015.

[7] A. Eguchi, "Robocupjunior for promoting stem education, 21st century skills, and technological advancement through robotics competition," *Robotics and Autonomous Systems*, vol. 75, pp. 692–699, 2016.

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

[9] B. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *Proceedings of the 11th international conference on advanced robotics*, vol. 1, 2003, pp. 317–323.

[10] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3. IEEE, 2004, pp. 2149–2154.

[11] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper, "Usarsim: a robot simulator for research and education," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 1400–1405.

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

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

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

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

[16] J. Perez, J. Sales, A. Penalver, D. Fornas, J. Javier Fernandez, J. C. Garcia, P. J. Sanz, R. Marin, and M. Prats, "Exploring 3-d reconstruction techniques: A benchmarking tool for underwater robotics," *Robotics & Automation Magazine, IEEE*, vol. 22, no. 3, pp. 85–95, 2015.

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

[18] J. Pérez, J. Sales, R. Marín, E. Cervera, and P. J. Sanz, "Configuración y ejecución de benchmarks de intervención robótica submarina en uwsim mediante herramientas web," in *XX Jornadas de Automática 2014*, 09/2014 2014.