Revista Iberoamericana de Automática e Informática Industrial

Revista Iberoamericana de Automática e Informática Industrial 15 (2018) 404-415

www.revista-riai.org

Entorno Docente Universitario para la Programación de Robots

J.M. Cañas[a,*], A. Martín[a], E. Perdices[a], F. Rivas[a], R. Calvo[b]

[a]*Laboratorio de Robótica, Universidad Rey Juan Carlos, Fuenlabrada 28943 Madrid, España.*
[b]*IMDEA Networks Institute, Av. Mar Mediterráneo, 22, 28918 Leganés, España.*

Resumen

Se presenta un entorno docente de robótica universitaria orientado al aprendizaje práctico. Es software libre, multiplataforma (Linux, Windows, MacOS) y hace énfasis en la programación de la inteligencia de los robots. Consta de una colección de prácticas variadas, cercanas a las aplicaciones robóticas que están llegando a la sociedad recientemente (coches autónomos, drones, aspiradoras...). Utiliza el simulador Gazebo como referencia y Python como lenguaje. Para cada práctica se ha programado una aplicación académica que realiza tareas auxiliares como la interfaz gráfica, la conexión con sensores y actuadores concretos, la temporización, etc. y aloja al código del estudiante, que así se concentra en los algoritmos de percepción y control. Cada aplicación está formada por una parte específica preparada, que queda oculta, y el código del estudiante, que simplemente rellena un sencillo fichero plantilla con la lógica del robot. Se ha utilizado con éxito en varios cursos de grado, de máster y cursos de introducción a la robótica.

Palabras Clave:
Programación de robots, Enseñanza, Herramientas software, Entornos de programación.

Academic framework for teaching robot programming at university

Abstract

This paper presents a framework for teaching robotics at engineering university degrees in a practical way. It is open source, multiplatform (Linux, Windows, MacOS) and emphasizes the programming of the robot intelligence. It consists of a collection of exercises of several types, which are similar to the new robotic applications (autonomous vehicles, drones, vacuum cleaners...). It uses the Gazebo simulator and the Python programming language. For each exercise an academic application has been created which performs all the auxiliary tasks like graphical interface, connection to the sensors and actuators, timing of the code, etc. and hosts the student's code, who can focus on the perception and control algorithms. Each academic application combines an infrastructure part, which it is provided and remains hidden, and the student part, who simply fills a form file with the robot logic for such exercise. It has been successfully used in several subjects at engineering degree, master and robotics introductory courses.

Keywords: Robot programming, Teaching, Software tools, Programming environments.

1. Introducción

Hoy día hay cada vez más aplicaciones robóticas para el público masivo. Más allá de las clásicas aplicaciones en entornos industriales y ensamblado de vehículos se utilizan robots, por ejemplo, en el envasado de alimentos o la gestión de almacenes. Las aspiradoras robóticas han supuesto un éxito sin precedentes resolviendo una necesidad del mercado doméstico con robots autónomos. También los coches incorporan cada vez más tecnología robótica, como el aparcamiento automático o asistentes de conducción autónoma. Los grandes fabricantes de automoción están empujando estos nuevos avances, tienen prototipos avanzados de coches autónomos y empresas de software como Google o Apple se han posicionado muy bien. Igualmente las aplicaciones de drones, de robots aéreos, están en pleno crecimiento.

*Autor para correspondencia: josemaria.plaza@urjc.es
To cite this article: J.M. Cañas, A. Martín, E. Perdices, F. Rivas, R. Calvo. 2018. Academic framework for teaching robot programming at university. Revista Iberoamericana de Automática e Informática Industrial 15, 404-415. https://doi.org/10.4995/riai.2018.8962

Correos electrónicos: josemaria.plaza@urjc.es (J.M. Cañas), almartinflorido@gmail.com (A. Martín), eperdices@gsyc.urjc.es (E. Perdices), franciscomiguel.rivas@urjc.es (F. Rivas), roberto.calvo@imdea.org (R. Calvo)

Uno de los factores que permiten a los robots desplegar inteligencia y lidiar con situaciones reales ofreciendo robustez similar a la humana es su software, su programación. Debajo de todas las aplicaciones robóticas que están llegando al mercado de masas hay un software en el que reside gran parte de la inteligencia de los robots. Típicamente este software para robots tiene varias capas (drivers, middleware y aplicaciones) y presenta unos requisitos específicos distintos del software para otros campos: funcionamiento en tiempo real, robustez, carácter distribuido, hardware heterogéneo. Además, se suelen utilizar interfaces gráficos principalmente para depurar, pero pocas veces en tiempo de ejecución (sistemas empotrados). En los últimos años se han incorporado a los robots ordenadores personales, micros de bajo coste y sistemas operativos generalistas (como Linux), lo que ha facilitado el uso de herramientas estándar de desarrollo.

Esta pujanza creciente de la tecnología robótica recomienda formar profesionales en este sector, que incluso lleven más allá las fronteras actuales y ayuden a crear nuevas aplicaciones robóticas que sirvan a las personas. La robótica es un campo transversal donde concurren muchas tecnologías: electrónica, mecánica, informática, telecomunicaciones... Actualmente la formación en robótica aparece en secundaria y se realiza fundamentalmente en la universidad, con titulaciones de grado y postgrados específicos.

En secundaria la educación en robótica está cobrando una importancia creciente. Tiene poder de motivación en los estudiantes y eso permite acercar la tecnología a los niños, usando la robótica como herramienta para exponerles conceptos básicos de ciencias, tecnología, ingeniería y matemáticas de manera lúdica. Como ejemplo en esta línea, la Comunidad de Madrid (Decreto 48/2015) ha introducido recientemente la asignatura *Tecnología, programación y robótica* en el currículum oficial de Educación Secundaria Obligatoria.

En la praxis de la enseñanza robótica en educación secundaria son frecuentes plataformas como los robots LEGO (RCX, NXT, Evo, WeDo) y placas con procesadores Arduino a los que se conectan sensores de bajo coste y servos. Se enseña el funcionamiento básico de sensores, actuadores y los rudimentos de la programación. Se usan lenguajes sencillos que facilitan su programación a niños, como los lenguajes gráficos RCX-code y recientemente Scratch o Blockly (Jiménez et al., 2010; Cerezo and Sastrón, 2015).

Dentro de la enseñanza universitaria tradicionalmente se imparten asignaturas de robótica en las escuelas de ingeniería (industrial, electrónica...) y en las de informática (Touretzky, 2013; Soto et al., 2006; Aliane, 2011), tanto en grado como en postgrado. En España las titulaciones más afines y con más solera son los grados en "Electrónica Industrial y Automática" o en "Ingeniería Electrónica, Robótica y Mecatrónica" y similares, ofertados por varias universidades. Recientemente está apareciendo algún grado con mayor énfasis en el software y la inteligencia artificial como el de "Ingeniería Robótica" en la U.Alicante. En postgrado, en los últimos años se han creado varios másteres exitosos en robótica. En el plano internacional, prestigiosas asociaciones como la ACM (Association for Computing and Machinery) y la IEEE-CS (IEEE Computer Society) contemplan la robótica como una de las áreas de conocimiento fundamentales en los estudios de informática e ingeniería, especialmente en el área de sistemas inteligentes (Berenguel et al., 2016). Las universidades americanas más punteras en tecnología (Carnegie Mellon University, Stanford, MIT, Georgia Institute of Technology, etc.) incluyen programas de grado y postgrado con contenidos robóticos, relacionados también con áreas afines como la visión computacional y la inteligencia artificial.

Dado su caracter transversal, la robótica se puede ver desde muchas perspectivas. Una de ellas es verla como la conjunción de sensores, actuadores y software inteligente entre medias. Siguiendo este enfoque, el objetivo de este trabajo es aumentar la formación y herramientas docentes disponibles en robótica. En concreto se presenta un nuevo entorno académico de enseñanza universitaria en robótica, tanto para grado como para postgrado, llamado `JdeRobot-Academy`. Este entorno docente pone el énfasis en el software para robots como portador principal de su inteligencia, especialmente en que los estudiantes programen los algoritmos de percepción, planificación y control habituales en los robots.

Este entorno está orientado inicialmente a cursos universitarios de 12-14 semanas, que ponen el foco en los algoritmos más que en el *middleware*. De hecho, la idea es ocultar el *middleware* en la medida de lo posible y simplemente usarlo para conectar el software del estudiante con los sensores y actuadores del robot. Es ideal para cursos de robótica en informática o en cursos de introducción a robótica. Se ha utilizado con éxito en la asignatura "Robótica" del grado en Ingeniería Telemática, en la asignatura "Visión en robótica" del Máster de Visión Artificial y en varios cursos de introducción a la robótica y los drones, todos ellos en la Universidad Rey Juan Carlos. También se ha empleado en las dos primeras ediciones del campeonato de programación de robots PROGRAM-A-ROBOT [1].

En la segunda sección se ofrece una panorámica de la enseñanza de robótica en la universidad y se repasan algunas herramientas docentes presentes en la comunidad. En la tercera se explica el diseño general del entorno docente propuesto y varias de las decisiones de organización que han guiado su construcción. En la cuarta se describen cinco prácticas concretas ilustrativas incluidas en el entorno que cubren varios aspectos del temario habitual de las asignaturas de robótica. En la quinta se presentan algunos resultados de los cursos en los que se ha usado el entorno y una valoración de los estudiantes. Se finaliza el artículo recopilando las conclusiones principales.

2. Plataformas robóticas en docencia universitaria

En los contenidos de las asignaturas de robótica en la universidad se pueden apreciar dos tendencias: por un lado las asignaturas más orientadas a brazos robotizados y manipuladores (Aliane, 2011; Mateo and Andújar, 2012, 2011; López-Nicolás et al., 2009, 2014; Jara et al., 2013; Gil et al., 2015) y por otro las más orientadas a robótica móvil (Fabregas et al., 2016; Detry et al., 2014; Guzmán et al., 2008; Guyot et al.,

[1] http://jderobot.org/Campeonato-programacion-de-robots

2011; Soto et al., 2006; Thrun, 2006). En las primeras se suelen abordar técnicas de control, cinemática inversa, directa, cálculo de trayectorias, etc.. En las segundas se suelen explicar técnicas de navegación local y global, control en posición esquivando obstáculos, percepción, autolocalización, etc.. Esta dicotomía de contenidos se refleja también en las plataformas robóticas que se emplean en las prácticas.

Ambos perfiles de asignaturas robóticas tienen un marcado caracter práctico. La interactividad de los alumnos con los robots facilita el aprendizaje y asentamiento de los conceptos teóricos, los algoritmos y técnicas (Jara et al., 2011). Las prácticas fomentan el paradigma de *aprender haciendo*, de aprendizaje activo. Suelen realizarse con alguna plataforma robótica concreta, utilizando alguna herramienta o entorno docente en el que se programa el comportamiento del robot en cierto lenguaje. Son frecuentes en la literatura los laboratorios virtuales (Jara et al., 2011, 2013; Fabregas et al., 2016) y también los remotos.

Uno de los entornos docentes más empleados para las prácticas es MATLAB, con su propio lenguaje, a veces acompañado del paquete Simulink. Por ejemplo, Gil et al. (Gil et al., 2015) ofrecen la toolbox de MATLAB ARTE (*A Robotics Toolbox for Education*), orientada a manipulación y la han usado en sus cursos incluyendo visualización 3D y programación de los brazos con un lenguaje industrial. Aliane (Aliane, 2011) utiliza MATLAB y Simulink para ofrecer a sus alumnos prácticas sobre cálculo de trayectorias y control de un manipulador SCARA. Gonzalez et al (González et al., 2015) han desarrollado la *Robot Motion Toolbox* que permite prácticas de planificación, navegación y control de un robot móvil, incluyendo un simulador propio. Corke (Corke, 1996, 2015) desarrolló la *Robotics Toolbox* y permite programar robots móviles, brazos y robots con visión (Corke, 2011).

Algunos entornos docentes no abordan la programación y se centran en el diseño, el modelado y la configuración de un brazo robótico, enfocándose en la cinemática, dinámica del robot y ofreciendo interfaces gráficos que ayudan a ver el efecto de la configuración elegida. Por ejemplo, en la U.Huelva han desarrollado 3D-RAS (*3D Robotic Arm Simulator*) (Mateo and Andújar, 2011, 2012) orientada al análisis cinemático de brazos robótizados de hasta 5 grados de libertad . En la U. Zaragoza se han desarrollado dos herramientas para la docencia con robots manipuladores: RobotScene y el simulador SGRobot (López-Nicolás et al., 2009, 2014). Entre las dos cubren prácticas de modelado, cinemática y configuración así como de su programación en un lenguaje similar al comercial VAL-II.

Otro entorno para brazos robóticos es RobUALab (Jara et al., 2013, 2011), programado en Java y con tecnología web, que permite la teleoperación, experimentación y programación de un manipulador. Proporciona un laboratorio remoto con un visor 3D vía web que se conecta a un brazo real Scorbot ER-IX ubicado en la U.Alicante. También incluye un laboratorio virtual con una celda industrial construida con realidad aumentada, que incluye un brazo simulado equivalente y obstáculos según se desee. Con ellos los alumnos pueden probar distintas configuraciones, evaluar pares, trayectorias tanto cartesianas como de articulaciones, y programar el brazo con listas de comandos.

Una plataforma docente para robots móviles es *Mobile Robot Interactive Tool* (Guzmán et al., 2008). Es un entorno bidimensional y está hecho con SysQuake (similar a MATLAB). Permite probar y ajustar parámetros de las diferentes soluciones de navegación integradas en la herramienta (*Visibility Graph, Generalized Voronoi Diagram, Wavefront, Bug, Visbug,* etc.) y de la cinemática del robot. Usan el robot Tritton o el PeopleBot para seguir una trayectoria. Este entorno y también MATLAB se utilizan en los cursos organizados por Berenguel et al (Berenguel et al., 2016), que ofrecen un perfil mixto cubriendo tanto manipuladores (Scorbot-ER Plus V) como robots móviles (LEGO, PeopleBot).

Fábregas et al. presentaron recientemente el entorno docente *Robots Formation Control Platform* (Fabregas et al., 2016) orientado al control de grupos de robots móviles y su movimiento en formaciones. Más que enfocarse en la programación del comportamiento permite seleccionar alguno de los algoritmos de control ya programados dentro del entorno (VFH, VFH+, VFH*), parametrizarlos de diferente manera, decidir si hay obstáculos o no, la formación deseada o el modelo cinemático de los robots y ver las trayectorias resultantes. Ofrecen tanto un laboratorio remoto con robots reales Moway en las instalaciones de la UNED como un laboratorio virtual con un simulador propio, llamado RFC-SIM, y robots simulados equivalentes.

Guyot et al. (Guyot et al., 2011) plantean un entorno docente completo con el simulador Webots para un robot móvil de pequeño tamaño, el e-puck. El entorno incluye prácticas de varios niveles, desde el estudiante novato, intermedio, avanzado, hasta el experto. Se usa el lenguaje C para algunas de las prácticas concretas desarrolladas en el entorno. En los niveles más altos las dos prácticas incluidas tienen un planteamiento competitivo en el que utilizar tecnologías robóticas variadas: el escenario de un laberinto y el RobotStadium (con el humanoide Nao simulado).

Un entorno software docente interesante para robótica en titulaciones de informática es Tekkotsu (Touretzky, 2013). Inicialmente diseñado para el robot Aibo, se ha hecho multiplataforma e incorpora varias bibliotecas propias con funcionalidad ya programada (manejo de información tridimensional, navegación...). Su planteamiento hace énfasis en la integración, en comprender código existente más que en desarrollar una nueva versión básica de algún algoritmo. Propone prácticas realistas, relativamente complejas, donde el alumno ha de escribir sólo una parte y el resto se le dan resueltas. El argumento subyacente es que este planteamiento es muy parecido a la realidad de la profesión, donde tendrá que realizar algún módulo del sistema complejo que es el robot.

TRS (Detry et al., 2014) es un entorno docente robótico de software libre para postgrado. Está basado en el simulador V-REP, que es multiplataforma y se ha usado en un curso de robótica en la universidad de Liège y en Aalborg Universitet København. Hace énfasis en una instalación rápida y sencilla. Permite a los estudiantes la programación de algoritmos de control, navegación o manipulación en Python o código MATLAB.

Más allá de la impartición clásica a través de lecciones teóricas y prácticas, otro enfoque docente interesante para las asignaturas de perfil robótico es el aprendizaje por proyectos. En ellos se suele proponer uno o varios retos (construir o programar un robot) y formar equipos de estudiantes que trabajan en las funcionalidades necesarias resolviendo los problemas

que aparecen. Dos casos de éxito de este enfoque son la creación del robot Xavier (Simmons et al., 2000) y el robot Stanley (Thrun, 2006) que ganó el campeonato DARPA Grand Challenge en el 2005.

Por otro lado, la digitalización está cambiando cómo el conocimiento se produce, se consume y se enseña. Explorando nuevas posibilidades educativas las universidades ofrecen cada vez más cursos masivos abiertos *en línea* (MOOC). En particular, universidades como Stanford, MIT, Harvard lideran desde 2011 iniciativas en esta dirección. Esto les permite llegar a un público masivo, ganar visibilidad y captar talento. El campo de la robótica no es ajeno a este movimiento y cada vez hay más cursos robóticos de este estilo (Corke et al., 2016), como el curso *Artificial Intelligence for Robotics*[2] de la U.Stanford (Udacity, Sebastian Thrun), el curso *Autonomous Navigation for Flying Robots*[3] de Technical University of Munich o el curso *Autonomous Mobile Robots*[4] del ETH de Zurich.

3. Diseño del entorno docente con JdeRobot

El entorno docente está formado por una colección de prácticas independientes que siguen el diseño de la Figura 1. Cada práctica plantea un problema robótico concreto (típicamente un comportamiento autónomo) y el estudiante ha de programar la inteligencia del robot que lo resuelve. Este entorno surge como evolución de las experiencias docentes con un entorno previo (Cañas et al., 2014) en clases de robótica en un grado y un master de ingeniería.

Se han elegido unas prácticas atractivas, acordes con aplicaciones que recientemente están llegando a la sociedad: coches autónomos, aspiradoras robóticas, drones, etc.. La idea de esta elección es que el alumno pueda relacionarlas con aplicaciones de la vida real, en las que es más fácil y motivante ver su utilidad real directa.

Cada práctica tiene tres componentes que se muestran en la Figura 1. Primero, en la capa inferior está el robot que se quiere realice alguna tarea en cierto entorno, ya sea simulado o real. Segundo, en la capa intermedia se tienen los drivers respectivos que dan acceso software a los sensores y actuadores del robot. Y tercero, en la capa superior está la *aplicación académica* que analiza los datos sensoriales y toma decisiones de actuación, planificando si es necesario. Esta aplicación consta de una funcionalidad auxiliar ya programada y del código del propio alumno. Programar robots no es sencillo y conviene dosificar la complejidad al estudiante, no exponerle a toda ella de golpe. Por ello cada práctica tiene su *aplicación académica* correspondiente que resuelve varios aspectos auxiliares pero no el núcleo del algoritmo, el cual debe ser programado por el estudiante. Tanto el robot como los drivers pueden reutilizarse en diferentes prácticas.

El resto de esta sección describe los principios subyacentes, las ventajas de esta organización en tres niveles, y varias de las decisiones de diseño tomadas en cada uno de ellos.

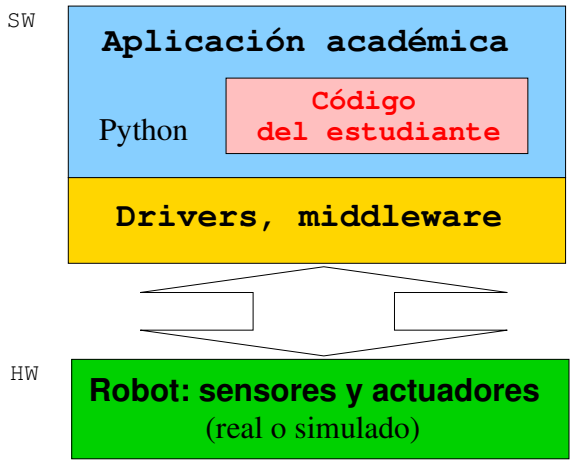

Figura 1: Diseño de una práctica robótica.

3.1. Robots reales y simulados soportados

Una de las primeras decisiones de diseño fue no centrar las prácticas en un robot concreto, sino elegir plataformas variadas: drones, robot TurtleBot con ruedas, coches, humanoides... con idea de poder diseñar prácticas que cubran diferentes aspectos de la robótica sin la limitación de un robot particular. Esto contrasta con otros entornos docentes centrados en un modelo concreto de robot (Guyot et al., 2011).

La Figura 2 muestra algunos de los robots soportados por la versión actual del entorno docente. Hay robots de interiores con ruedas y tracción diferencial, como el robot TurtleBot (tanto en real como en simulador), o de exteriores como un coche Fórmula-1 y un taxi simulados. También se incluye soporte para drones como el ArDrone de Parrot, real y simulado. Igualmente ofrece la posibilidad de prácticas con sensores como cámaras, láseres, GPS, cámaras RGBD (Kinect, Xtion), etc..

Los simuladores son herramientas frecuentes en robótica, tanto en investigación como en docencia (Cañas et al., 2009). Algunas de sus ventajas en educación son:

- permiten aprender robótica con robots complejos y potentes sin disponer del hardware. O si se tienen robots reales, posibilitan cursos de muchos alumnos, incluso más numerosos que los robots disponibles. Por ejemplo son ideales para los cursos masivos en línea.
- en el simulador las caídas no duelen, de modo que los fallos de programación por parte de los alumnos no dañan ningún hardware real.
- es posible dar por resueltos algunos problemas para que el alumno se enfoque en otros. Por ejemplo el simulador proporciona la posición absoluta verdadera del robot en todo momento y se puede asumir que tiene una autolocalización perfecta si así interesa.
- permiten comparativas justas. El código de todos los estudiantes puede ser ejecutado exactamente en la misma máquina.

[2] https://www.udacity.com/course/artificial-intelligence-for-robotics–cs373
[3] https://www.edx.org/course/autonomous-navigation-flying-robots-tumx-autonavx-0
[4] https://www.edx.org/course/autonomous-mobile-robots-ethx-amrx-1

- posibilitan la automatización de pruebas, elaborar correctores automáticos de las prácticas. Estos correctores se conectan al mundo simulado donde está ejecutando el robot programado por el estudiante y toman datos que permiten evaluar objetivamente la calidad de su comportamiento. Por ejemplo, en una práctica de un coche autónomo aparcando solo, mide automáticamente el tiempo que tarda en aparcar, cuán alineado está con la acera, distancia delantera y trasera a los coches ya aparcados, etc..

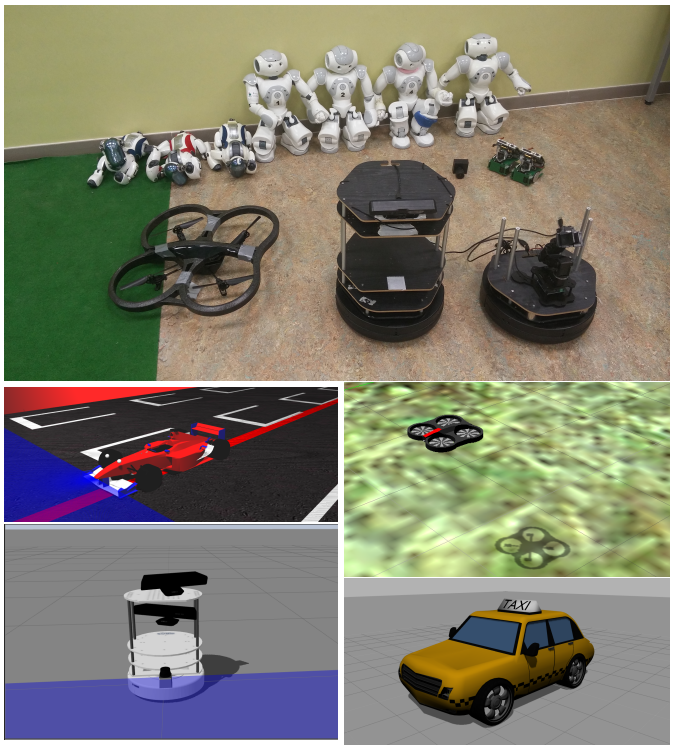

Figura 2: Algunos robots soportados por el entorno docente *JdeRobot-Academy*.

La experiencia con plataformas reales es académicamente muy valiosa por sí misma, aunque tenga algunos incovenientes prácticos. Desde un punto de vista docente parece recomendable un balance entre prácticas con robots reales y con robots simulados, que recoja ventajas de ambas aproximaciones. El diseño software elegido para este entorno docente permite ese balance entre prácticas con robots simulados y con robots reales (cuando están disponibles). Permite conectar la misma aplicación académica a un robot físico o a su homólogo simulado sólo con cambios en su configuración. Basta que el interfaz que ofrecen el driver del robot real y el del robot simulado sean equivalentes, la aplicación se conectará a ellos indistintamente sin que el estudiante tenga que cambiar su código (Figura 3). Esta flexibilidad permite a los alumnos madurar su código en el simulador y luego aterrizarlo sin drama en el robot real. En la práctica suelen ser necesarios pequeños cambios en el código fuente para que funcione óptimamente sobre hardware real, pero más bien ajustes que estructurales.

Hay muchos simuladores disponibles: VREP, Webots, etc.. Todas las prácticas del entorno docente presentadas en este trabajo se han preparado para que funcionen con el simulador Gazebo, que se ha elegido como referencia. Es software libre, de alta calidad, mantenido por la Open Source Robotics Foundation y tiene gran aceptación en la comunidad robótica internacional. Fue elegido también por el DARPA para su último campeonato internacional *Darpa Robotics Challenge*, quién invirtió mucha financiación en su desarrollo.

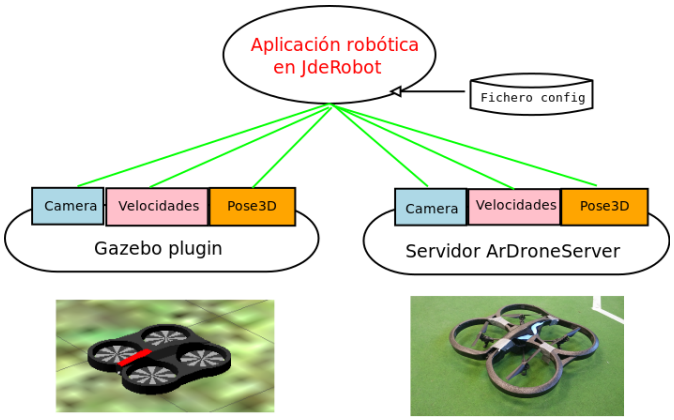

Figura 3: Aplicación conectable a un robot real o al homólogo simulado.

La arquitectura típica de una práctica con simulador se muestra en la Figura 4. El simulador Gazebo materializa el comportamiento del robot físico en un escenario emulado. Los drivers, tal y como está construido Gazebo, son *plugins* que se ejecutan dentro del propio simulador y ofrecen acceso a sensores y actuadores de los robots simulados. Para que Gazebo inicie el mundo simulado de una práctica hay lanzarlo con un fichero de configuración que determina el escenario, los robots involucrados, etc.. Ofrece un visor local a través del cual observar la evolución de mundo simulado, el comportamiento del robot programado por el alumno e incluso interactuar con la escena (añadiendo algún elemento en tiempo de ejecución, detener, relanzar la simulación, etc..). Igualmente ofrece un visor web gracias al cual también se puede ver e interactuar con el mundo simulado desde un navegador web.

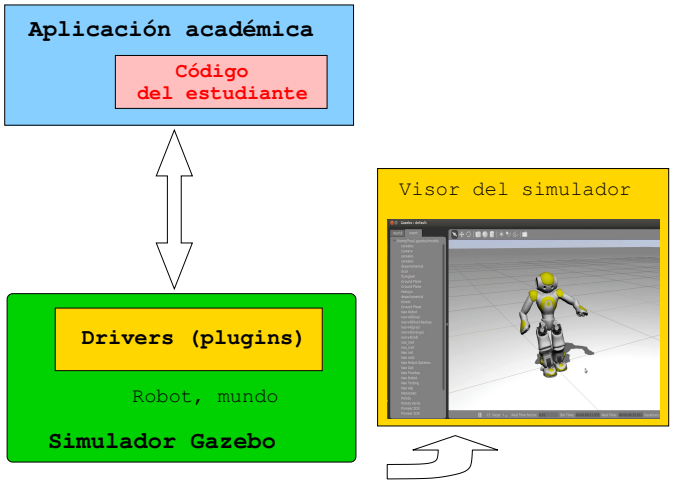

Figura 4: Diseño típico de una práctica con Gazebo y visor local del mundo simulado.

3.2. *Drivers y middleware*

En los últimos años han aparecido numerosos entornos (*middlewares*) que facilitan la programación de aplicaciones

para robots; por ejemplo ROS, OROCOS, YARP, etc.. Suelen proporcionar drivers de varios robots, dar una arquitectura software concreta a las aplicaciones e incluir varias utilidades. Ofrecen también una manera concreta de llegar a los sensores y actuadores. Son una herramienta muy útil. La tendencia dominante es que sean orientados a componentes y distribuidos.

Como *middleware* se ha elegido JdeRobot [5], que es orientado a componentes distribuidos y utiliza ICE para resolver las comunicaciones entre ellos. Es multilenguaje, multimáquina y permite a un mismo componente conectarse indistintamente al robot real o al robot simulado simplemente cambiando su configuración. Los interfaces de sensores y actuadores son exactamente los mismos para el robot real que para el simulado (Figura 3).

Los drivers reales en JdeRobot son componentes y las aplicaciones también, son uno o un conjunto de ellos interoperando entre sí. Tanto los drivers como las aplicaciones necesitan ficheros de configuración oportuna. Algunas de las utilidades a disposición de los estudiantes en este *middleware* son un sintonizador de filtros de color, un calibrador de cámaras y un generador de autómatas de estado finito.

3.3. Una aplicación académica por práctica

Para cada práctica se crea una *aplicación académica* que contiene una parte ya programada y otra parte que debe rellenar el alumno. La parte preparada es específica, resuelve tareas auxiliares e incluye un único fichero plantilla para que el estudiante introduzca ahí su código. Esta reducción de los ficheros que el estudiante ha de tocar a uno solo simplifica la recogida de prácticas en cursos numerosos.

Hay muchos lenguajes de programación posibles para que el alumno desarrolle su parte. Se han mencionado entornos docentes que usan Java o MATLAB, y gran parte de la investigación en robótica se materializa en C++ por su velocidad y potencia expresiva. Para el entorno propuesto se ha elegido Python. Suaviza la curva de entrada que tiene C++ manteniendo la potencia de la programación orientada a objetos. Python ya ha sido utilizado en la enseñanza robótica en varias ocasiones (Joseph, 2015; Blank et al., 2006, 2004).

El entorno docente `JdeRobot-Academy` fomenta el uso de bibliotecas estándares en el campo, como OpenCV para procesamiento de imágenes o PCL para manejo de datos RGBD 3D (aunque desarrollada en C++ tiene extensiones de Python).

La parte ya programada de la aplicación académica hace de contenedor donde se incorpora el código del estudiante y ofrece varios aportes:

1. resuelve el interfaz gráfico (GUI), necesario para depurar
2. proporciona un interfaz de programación (API) local y sencillo para sensores y actuadores
3. proporciona un esqueleto temporal
4. incluye una plantilla para que el alumno rellene con su código

El GUI permite mostrar datos sensoriales o incluso de procesamientos parciales adecuados para cada práctica. Por ejemplo, puede incluir código para mostrar si un filtro de color está funcionando bien sobre la imagen de la cámara del robot. O mostrar en el mundo del robot el camino seleccionado por el algoritmo planificador. Al dárselo resuelto, el alumno no se distrae programando funcionalidades gráficas, pero se beneficia de ellas para depurar.

La *aplicación académica* oculta la complejidad y los detalles del *middleware* robótico usado. Se conecta con los sensores y actuadores del robot empleando el *middleware* oportuno, pero ofrece un sencillo API local en Python para el alumno. De este modo, el estudiante simplemente invoca métodos de Python para actualizar los datos recogidos por los sensores y para enviar comandos a los actuadores, que funcionan tanto si están conectados a la misma máquina como si están accesibles por red.

El esqueleto temporal más usado es un bucle continuo de iteraciones, cada una de ellas con cuatro pasos: recoger datos sensoriales, procesarlos, decidir actuación y enviar órdenes a los actuadores. Este motor temporal es muy adecuado para materializar comportamientos reactivos. La *aplicación académica* ofrece una función `Callback` que ha de rellenar el estudiante y que se invoca desde el motor temporal a un ritmo controlado. La frecuencia nominal de este motor se fija a unos 10 o 15Hz, según la práctica. Este motor tiene la ventaja de no consumir toda la CPU disponible: si la CPU es rápida ejecuta iteraciones al ritmo nominal fijado dejando tiempo de cómputo para otras aplicaciones. Además, tiene una degradación suave: si el ordenador donde se ejecuta es lento o está muy cargado entonces ejecuta a la máxima frecuencia posible en esas condiciones, aunque se quede por debajo de la frecuencia nominal.

La *aplicación académica* se ejecuta con un fichero de configuración como parámetro en el cual se especifica la configuración de los drivers o el simulador empleados en cada práctica.

3.4. Distribución e instalación

El entorno presentado es software libre, descargable desde GitHub[6]. Está abierto a colaboraciones, extensiones, modificaciones y uso en todo o en parte.

Como proyecto de código abierto, si la instalación es compleja se reduce el número de usuarios reales. Además, que la instalación del entorno sea fácil es importante para que el estudiante no se despiste con la herramienta ni con el *middleware*. Para que sea fácil de instalar se han preparado paquetes binarios para Ubuntu y para Debian que ya incluyen los drivers, el simulador y la *aplicación académica* de cada una de las prácticas elaboradas. El entorno es simple de usar, hay recetas de ejecución de cada una de las prácticas.

Inicialmente el entorno nació en Linux, pero se ha programado la versión multiplataforma para ampliar el abanico de potenciales usuarios. Ahora funciona correctamente tanto en Linux como en MS-Windows y MacOS. Las *aplicaciones académicas* son en Python, lo que las hace multimáquina (sólo hay que tener disponible en cada sistema una versión de Python compatible). En la última versión se ha creado un contenedor ligero Docker de modo que el simulador se ejecuta dentro de ese contenedor sin mostrar la interfaz gráfica (Figura 5), tanto en MS-Windows como en MacOS. En este caso se usa cualquier

[5] http://jderobot.org
[6] https://github.com/JdeRobot/Academy

navegador web como visor del simulador gracias a la infraestructura `GzWeb-server` que ofrece Gazebo, y tanto la *aplicación académica* como el navegador web ejecutan en nativo.

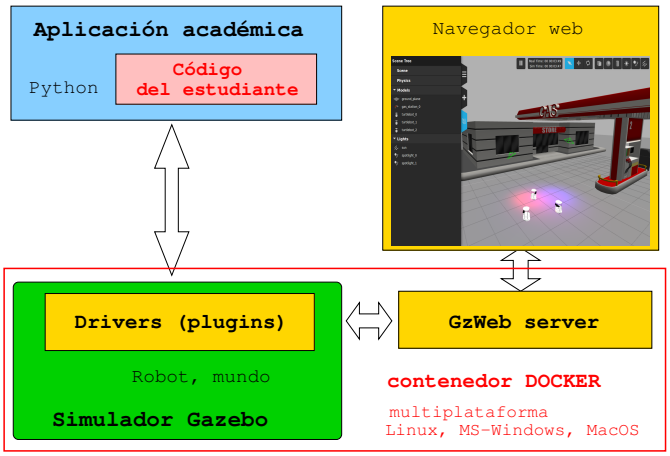

Figura 5: Entorno de prácticas en MS-Windows utilizando contenedor Docker.

El conjunto de prácticas del entorno docente se presenta como una colección de prácticas independientes. Cada una de ellas es autocontenida: tiene su *aplicación académica* específica y sus respectivos ficheros de configuración del simulador o del robot. Al diseñarlas como independientes y autocontenidas el conjunto es fácilmente extensible.

4. Prácticas

La colección de prácticas disponibles está en la web[7] y aborda diferentes problemas de robótica, control, automática y visión artificial. Actualmente hay 15 prácticas ya desarrolladas, según muestra la Tabla 1, y la colección es ampliable. Por ejemplo, una práctica aborda el aparcamiento automático de un coche autónomo, otra la negociación de un cruce. Una práctica aborda el comportamiento choca-gira en un robot móvil empleando un autómata finito de estados, otra la programación de una aspiradora robótica para limpiar una casa. Del repertorio actual se pueden extraer grupos temáticamente coherentes para utilizarlos en cursos específicos: prácticas de visión, prácticas de coches autónomos, prácticas de drones, etc.. En esta sección se detallan cinco de ellas a modo ilustrativo, describiendo el robot y escenario que maneja cada una de ellas, la tarea robótica a resolver y la *aplicación académica* desarrollada. Todas ellas tienen forma de un componente de JdeRobot.

4.1. *Drones: persecución*

Esta práctica consiste en el juego del ratón y el gato entre dos drones. El estudiante tiene que programar al drone gato para que persiga y se mantenga cerca del drone ratón. En el mundo simulado el drone ratón está pintado de rojo y se mueve autónomamente. La parte izquierda de la Figura 6 muestra el mundo en Gazebo, libre de obstáculos, con el ratón y el gato. Para esta práctica se han preparado un conjunto de ratones de varios niveles. Desde los más sencillos, que son lentos y predecibles en su desplazamiento, hasta los más rápidos y nerviosos en su movimiento.

Tabla 1: Colección de prácticas disponibles en JdeRobot-Academy.

`Prácticas de robots móviles`
Comportamiento choca-gira
Aspiradora Roomba sin autolocalización
Sigue línea
`Prácticas de coches autónomos`
Fórmula1: Navegación local con VFF
TeleTaxi: Navegación global con GPP
Coche negociando un cruce
Coche aparcando solo
`Prácticas de drones`
Navegación por posición
Persecución de objeto terrestre
Seguimiento de una carretera
Juego del gato-ratón
Salir de laberinto mediante pistas visuales
Búsqueda de víctimas dentro de un perímetro
`Prácticas de visión artificial`
Filtro de color
Reconstrucción 3D desde par estéreo

El drone gato está equipado con dos cámaras, una cenital y otra frontal, inclinómetros y GPS. Tiene cuatro rotores, pero se manejan con un interfaz de nivel intermedio que acepta órdenes de velocidad de avance o retroceso, de elevación o descenso, de desplazamiento lateral y velocidades de giro. Este robot aéreo es similar al ArDrone2 de Parrot. El API local ofrece sencillos métodos en Python para acceder a estos sensores y actuadores. Por ejemplo el método `sensor.getImage()` para recoger imágenes de la cámara, el método `pose.getPose3D()` para recoger la información conjunta de los dos inclinómetros y GPS (como un vector XYZ y un cuaternión para la orientación), los métodos `extra.takeoff()` y `extra.land()` para despegar y aterrizar, y el método `cmdvel.sendCMDVel()` para enviar órdenes de movimiento o de giro al drone.

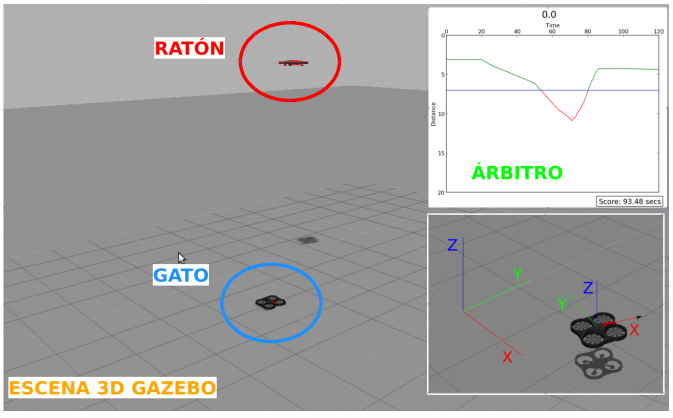

Figura 6: Un drone persiguiendo a otro.

[7] http://jderobot.org/JdeRobot-Academy

El estudiante ha de programar la percepción visual necesaria, por ejemplo filtros de color, filtros morfológicos o la segmentación del otro drone en imagen. En cuanto a la toma de decisiones la solución típica realiza un control basado en casos, incluyendo el caso de búsqueda automática si se pierde al ratón y el caso normal de seguimiento basado en visión, donde funciona satisfactoriamente un control reactivo PID que acelera cuando el ratón se hace pequeño, sube cuando el ratón se eleva en la imagen, etc..

Esta práctica incluye un componente que automatiza la evaluación objetiva. Es un corrector automático que mide en todo momento la distancia espacial entre los dos drones: puntúa el número de segundos que la distancia está por debajo de cierto umbral. La parte superior derecha de la Figura 6 muestra el interfaz gráfico de este árbitro automático que dibuja la evolución temporal de esa distancia, con el cero en el nivel superior. La prueba consiste en varias mangas de dos minutos con los sucesivos ratones. La puntuación es el número de segundos que el gato programado por el estudiante está cerca (a menos de 3 metros) del ratón.

4.2. *Control visual: sigue líneas*

El objetivo de este ejercicio es que un robot móvil TurtleBot siga la línea roja en un circuito en el menor tiempo posible. El estudiante ha de programar el algoritmo que extrae la información necesaria desde los píxeles de una cámara a bordo del robot y ordena a los motores el movimiento adecuado. La Figura 7 muestra en su parte inferior el circuito en Gazebo con una línea roja pintada y el homólogo con el robot real, con una línea blanca.

El robot TurtleBot tiene dos ruedas motrices, con tracción diferencial, a las que se ordenan comandos a través del interfaz intermedio de movimiento: velocidad de avance V y velocidad de giro W. Está equipado con una cámara, la del portátil a bordo en el robot real.

En su parte perceptiva, esta práctica da pie a filtros de color y procesamientos ad-hoc que extraen alguna medida de si hay línea en la imagen, si el robot está en curva o en recta, si está desviado hacia derecha o izquierda y en qué medida. Una solución típica consiste en procesar cuatro o cinco líneas de cada fotograma y filtrar por color sólo en esas líneas, midiendo ahí la desviación respecto de la situación en la que el robot está centrado sobre la línea. En su parte de actuación, esta práctica ilustra muy bien la utilidad de un control PID. Se puede probar a ajustar los valores de sus constantes K_p, K_d y K_i. Se ve también la utilidad de programar el control basado en imágenes, con ciertos procesamientos intermedios adhoc, en vez de señales sensoriales unidimensionales. En el interfaz gráfico de esta aplicación académica se facilita la visualización del resultado de los procesamientos realizados sobre la imagen, como muestra la parte superior de la Figura 7.

El criterio de evaluación más simple es el tiempo por vuelta, siempre que el robot no se salga excesivamente de la línea. Otros criterios son medir el error en la trayectoria respecto de la ideal (penalizando así las oscilaciones) o ver si pierde muchas veces la línea y si la recupera. Ambos se miden también indirectamente con el tiempo por vuelta.

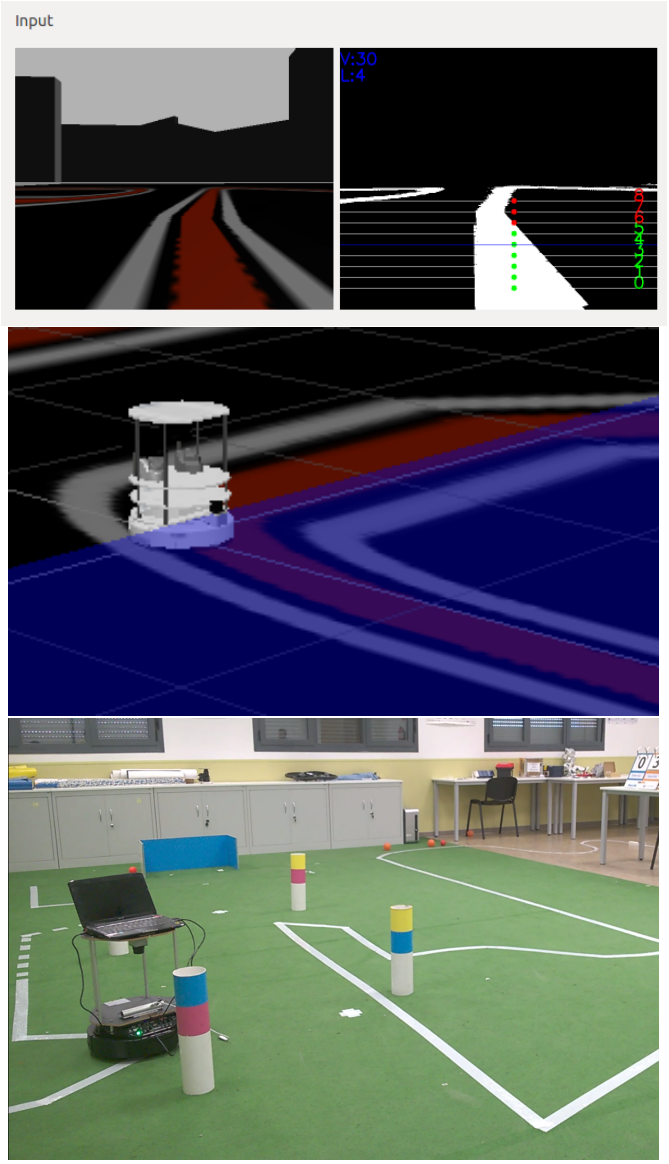

Figura 7: Robot TurtleBot siguiendo una línea de color en un circuito.

4.3. *Fórmula 1: navegación local*

Esta práctica consiste en programar un coche de Fórmula-1 para que complete una vuelta a un circuito de carreras esquivando los obstáculos que le aparezcan por el camino, otros coches. Se ha creado el circuito de carreras que se muestra en la Figura 8, con su parrilla de salida, otros coches que hacen de obstáculos, varias curvas y rectas, etc.. Además, se proporcionan las coordenadas de una secuencia de puntos que caen dentro del circuito para que hagan de destinos locales para el coche. Si se alcanzan todos esos puntos intermedios en secuencia se completa una vuelta entera a todo el circuito.

El robot es un coche de Fórmula-1 que cuenta entre sus sensores con odometría, GPS y un sensor láser. En cuanto al movimiento tiene un volante y dos ruedas motrices, pero admite un interfaz intermedio de órdenes: velocidad de avance V y velocidad de giro W. El robot tiene inercia y no consigue instantáneamente las velocidades comandadas por el software. Se ha creado el modelo de Fórmula-1 en el simulador Gazebo, con su apariencia realista y sus sensores empotrados.

En este ejercicio los estudiantes programan un algoritmo de navegación local, por ejemplo el de fuerzas virtuales VFF. La visualización ofrecida por esta *aplicación académica*, como muestra la Figura 8, incluye los datos instantáneos del laser, el destino local y las fuerzas vectoriales repulsiva, atractiva y total calculadas por el algoritmo, todo ello en coordenadas relativas al coche. Esto permite el ajuste de varios parámetros de las fuerzas y su combinación lineal.

La evaluación de esta práctica tiene en cuenta el tiempo en completar la vuelta y la seguridad con la que ha navegado, si se ha chocado o no, etc.

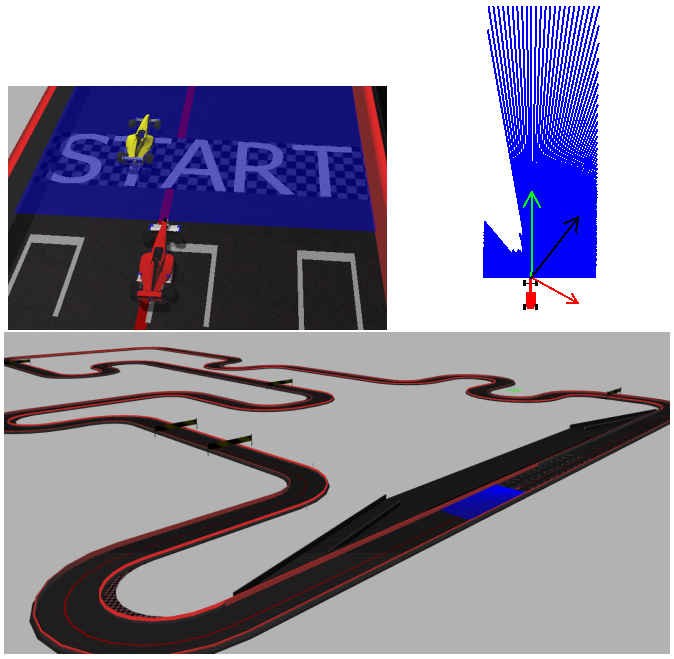

Figura 8: Un fórmula-1 esquivando a otros coches.

4.4. *TeleTaxi: navegación global*

El objetivo de este ejercicio es que un coche autónomo pueda navegar de un punto a otro cualquiera de una ciudad de la que se le proporciona el mapa. Esto requiere programar un algoritmo de planificación de rutas y otro de pilotaje basado en posición. En la Figura 9 se observa al coche autónomo, un taxi, en una ciudad simplificada, con sus calles y edificios.

El taxi está equipado con un sensor GPS que entrega en todo momento una estimación de su posición dentro de la ciudad. En cuanto a actuadores tiene el ya comentado interfaz intermedio de locomoción basado en velocidad de avance V y velocidad de giro W. Se ha creado el modelo de coche utilitario para Gazebo con su apariencia realista y sus sensores empotrados (Figura 2), homólogos a los empleados en coches autónomos reales como los de Google o Tesla.

En esta práctica el estudiante ha de programar el algoritmo de navegación global GPP para planificar su camino desde la posición actual hasta una marcada por el usuario en el interfaz gráfico picando en la posición deseada. El GUI de esta *aplicación académica* (Figura 9) muestra el mapa 2D de la ciudad, la ruta ideal calculada y permite visualizar también el valor del campo generado por el código del alumno, que es producto de esta técnica. Una vez calculada la ruta, el código incluye además el algoritmo de pilotaje que tiene en cuenta la posición instantánea y el campo generado para comandar velocidades al robot y que éste navegue hacia el destino.

El cálculo del campo se encuadra en el esqueleto temporal iterativo como la primera iteración, sin duración acotada, y el pilotaje encaja sin problema como un control reactivo basado en iteraciones.

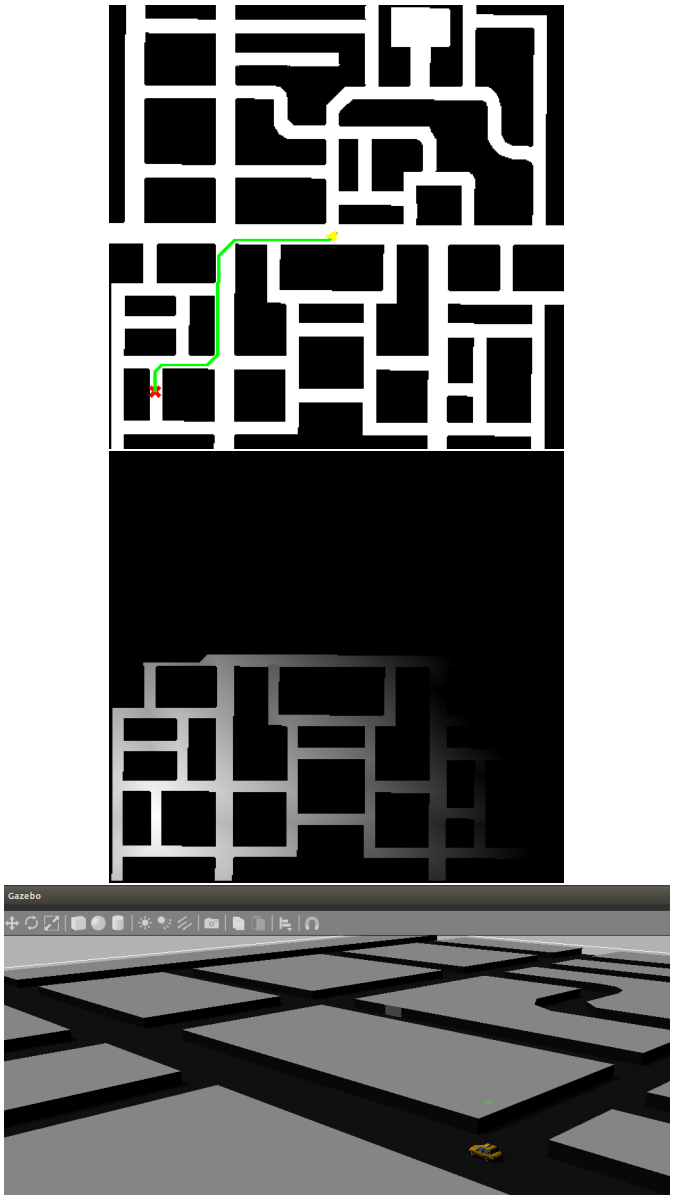

Figura 9: Un coche planificando y siguiendo rutas.

4.5. *Visión: reconstrucción 3D*

El objetivo en esta práctica es que un robot Pioneer equipado con un par estéreo de cámaras sea capaz de reconstruir en 3D la realidad que tiene delante. Con este algoritmo perceptivo se pueden estimar distancias a obstáculos empleando sólo cámaras y tenerlas en cuenta para navegar, por ejemplo. Las cámaras están calibradas y sus parámetros ópticos se proporcionan en un fichero. Se sitúa al robot delante de varios objetos con mucha textura en un mundo simulado, tal y como muestra la parte superior de la Figura 10.

El estudiante ha de programar un algoritmo de reconstrucción 3D clásico, con sus tres pasos: (1) detección de puntos

de interés en las dos imágenes, (2) emparejamiento de píxeles homólogos entre ambas cámaras y (3) triangulación espacial para calcular el punto tridimensional que origina cada pareja de píxeles homólogos en sendas cámaras. La reconstrucción no es densa y para la detección de puntos de interés suele bastar un sencillo filtro de bordes. Para el emparejamiento se suelen utilizar técnicas de correlación entre parches vecinos y maximización del parecido. También se emplean algunas restricciones que aceleran el cálculo, como la epipolar. Para la triangulación se utilizan geometría estéreo y retroproyecciones.

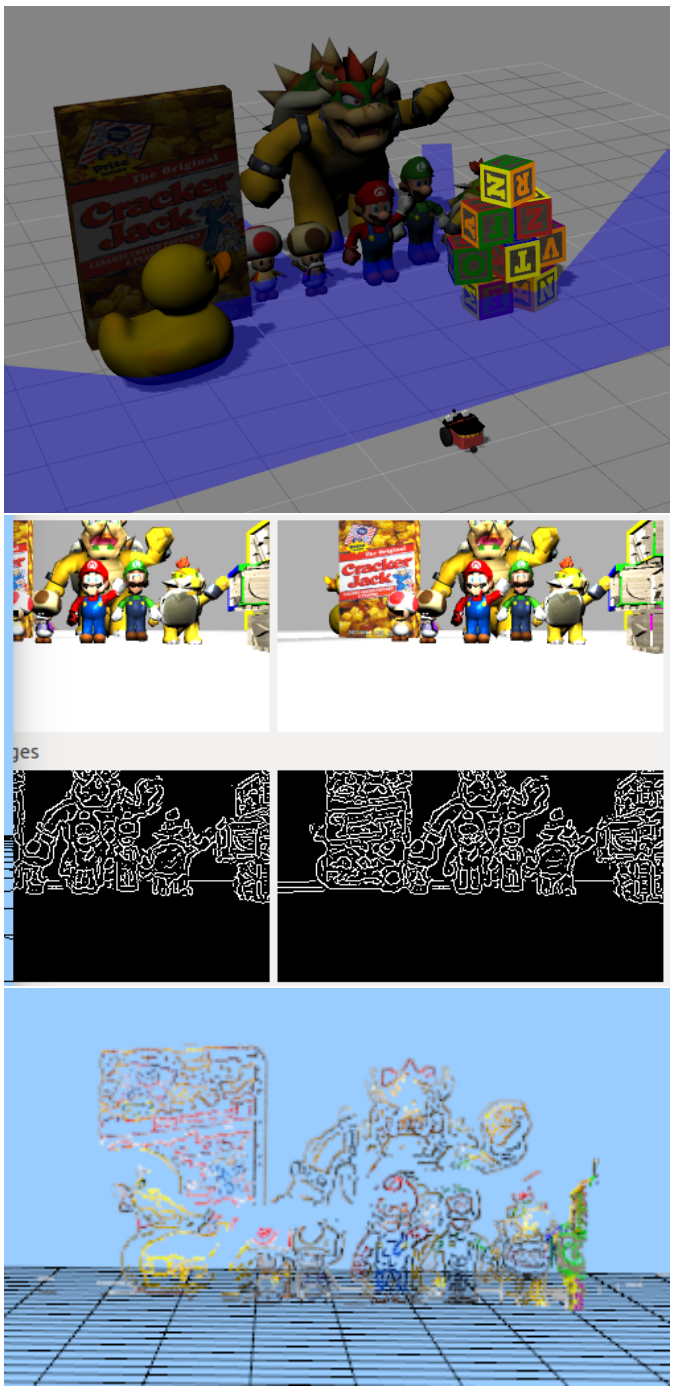

Figura 10: Reconstrucción 3D desde un par estéreo de cámaras.

El GUI de esta *aplicación académica* (Figura 10) muestra los puntos de interés de la imagen, los que realmente se llevan a 3D. También incluye un visor 3D donde se van pintando los puntos 3D calculados. Además, facilita la depuración de los tres pasos por separado al poder llevar a 3D sólo el píxel donde clique el alumno con el ratón o realizar su emparejamiento a mano.

La evaluación de esta práctica suele tener en cuenta la calidad de la reconstrucción tridimensional, su parecido con la realidad y el tiempo que tarda el código del alumno en conseguirla.

5. Resultados de implantación

El entorno docente `JdeRobot-Academy` se ha empleado con éxito en el Grado de Telemática (asignatura *Robótica*), en el Máster de Visión Artificial de la URJC (asignatura *Visión en robótica*) en cursos de 13 semanas. También se ha utilizado en varios cursos cortos de 3 o 5 semanas de introducción a la robótica y a la programación de drones.

En todos ellos se combinaba un 50 % de horas de contenidos teóricos con un 50 % de prácticas, cuya realización se ha planteado individual. La evaluación incluía un examen oral por cada práctica en el que cada alumno mostraba su solución funcionando y respondía a preguntas sobre el código. Además, los alumnos tenían que describir sus desarrollos en un blog incluyendo texto, imágenes y videos. Esto ayudaba a que compartieran ideas y competieran entre sí.

En el master se han empleado dos prácticas: la de control visual sigue-líneas y la de reconstrucción 3D. En el grado se han empleado cinco prácticas: el comportamiento chocagira, sigue-líneas, gato-ratón con drones, navegacion local de Fórmula1 y teletaxi. En las primeras prácticas se ofreció a los alumnos con mejores resultados en simulación probarlas y afinarlas con los (pocos) robots reales disponibles en el laboratorio.

Lo han empleado más de 120 estudiantes. Las encuestas entre ellos muestran una opinión muy satisfactoria del entorno y de las prácticas, con un 80 % de valoraciones iguales o por encima de 8 puntos sobre 10, según muestra la Figura 11. También revelan que la valoración de la instalación es positiva pero debe simplificarse aún más.

Además, la práctica del gato-ratón con drones se ha empleado satisfactoriamente en dos ediciones del campeonato de programación de robots PROGRAM-A-ROBOT[8], la última de ellas dentro de las Jornadas Nacionales de Robótica. Aquí el corrector automático desarrollado permitió un desarrollo ágil del campeonato entregando una puntuación objetiva e instantánea de la solución de cada participante.

6. Conclusiones

Se ha presentado el entorno `JdeRobot-Academy` para la docencia universitaria en robótica. Está compuesto por un conjunto de prácticas independientes sobre drones, coches autónomos, robots móviles y visión artificial. Es un entorno de software libre dentro del proyecto JdeRobot. Se ha descrito la última

[8] http://jderobot.org/Campeonato-programacion-de-robots

versión, detallando cinco prácticas representativas de las quince disponibles actualmente.

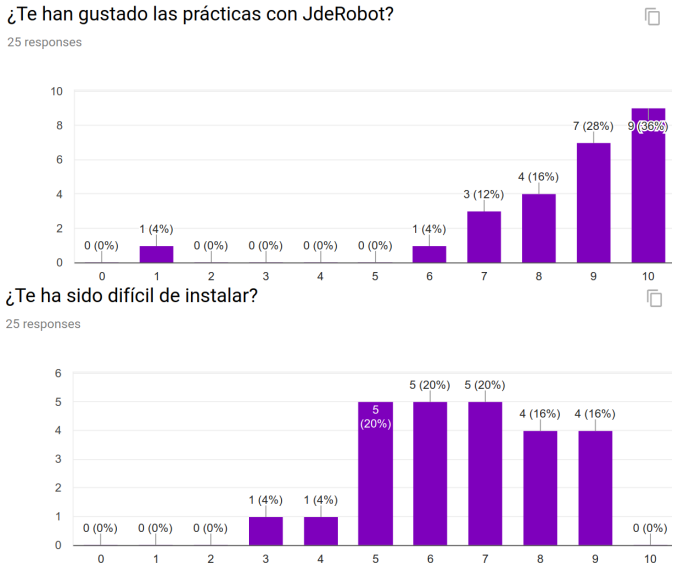

Figura 11: Encuestas de valoración de `JdeRobot-Academy`.

El entorno está diseñado con varios principios como punto de partida. Primero, el software robótico es complejo y conviene dosificar esa complejidad al alumno, especialmente al que se introduce por primera vez en robótica. Para ocultar esa complejidad se ha diseñado una *aplicación académica* por cada una de las prácticas. Las aplicaciones académicas ofrecen un interfaz sencillo al estudiante para acceder a los dispositivos, envolviendo y ocultando los detalles del *middleware*. Resuelven muchas tareas auxiliares como el interfaz gráfico específico para cada práctica, su motor temporal o el acceso real a sensores y actuadores. Cada *aplicación académica* proporciona una plantilla donde el estudiante empotra su código para cada práctica. De este modo, el alumno se puede concentrar más en el algoritmo de percepción o control.

Esta perspectiva hace que las prácticas pongan más énfasis en los algoritmos y su programación que en el *middleware*. El *middleware* es una herramienta valiosa, pero en este entorno simplemente se usa de manera invisible para los estudiantes. Tanto algoritmos como *middleware* son interesantes y útiles, pero en cursos de un cuatrimestre no da tiempo a contar ambas materias con suficiente profundidad. No obstante, al emplear este entorno con estudiantes de niveles crecientes (graduados, máster, doctorado) se les pueden ir descubriendo más partes y detalles de la infraestructura subyacente, por ejemplo, enseñarles más el *middleware*.

Segundo, los simuladores aportan muchas ventajas a la docencia en robótica, aunque la experiencia con robots reales es valiosa igualmente. Como simulador de referencia se ha elegido Gazebo, generalista, estándar en la comunidad y hecho por expertos en simulación. Al utilizar un simulador robótico de propósito general el abanico de prácticas implementables es enorme, mayor que el de otros entornos educativos centrados en un tipo de robot nada más. Por ejemplo, puede servir de base para prácticas heterogéneas de navegación, de manipulación, de visión computacional, etc.

Tercero, el lenguaje no puede ser excesivamente complejo para que no se convierta en el foco, sino en un instrumento más para aprender los conceptos robóticos. En el entorno diseñado el estudiante programa sus prácticas en Python, que se ha elegido por su simplicidad y potencia expresiva.

Cuarto, el planteamiento de las prácticas como juego competitivo (*gamificación*) las hace más atractivas y divertidas. En nuestra experiencia docente y de organización de campeonatos de robots (Cañas et al., 2007) esto aumenta la motivación de los estudiantes y hace que aprendan más. Por ello, varias prácticas del entorno se han presentado de modo competitivo: qué dron está más tiempo cerca de otro, cuánto tarda el Fórmula-1 programado en dar la vuelta completa al circuito, etc.. Para facilitar este planteamiento competitivo se han desarrollado correctores automáticos (en simulador) que entregan instantáneamente una puntuación objetiva.

Quinto, un punto práctico importante es la fácil instalación del entorno (*plug & play*) para que el alumno no pierda tiempo configurándolo. Se han elaborado paquetes binarios (para Linux Debian y para Ubuntu Linux) de `JdeRobot-Academy` con todo el entorno preparado y configurado que simplifican enormemente la instalación . También se ha preparado una versión multiplataforma que puede ejecutarse en máquinas Microsoft Windows o MacOS.

Este entorno docente se ha empleado con éxito en los últimos años en el Grado de Telemática, en el Máster de Visión Artificial de la URJC y en cursos cortos con más de 120 estudiantes. En todos ellos las encuestas de los alumnos reflejan una opinión satisfactoria sobre esta nueva herramienta.

Se está trabajando en varias líneas futuras para hacer crecer la plataforma docente. Primero, desarrollar nuevas prácticas: con un brazo robótico, con una carrera de Fórmula-1 en la que compiten cuatro coches autónomos a la vez. Segundo, utilizar ROS como *middleware* para acercarnos al estándar de facto que ya se emplea en muchos centros educativos y está muy extendido en la comunidad internacional investigadora en robótica. Tercero, difundir este entorno educativo de software libre, invitar a disfrutarlo y usarlo a más profesores. En este sentido la tercera edición del campeonato de programación de robots PROGRAM-A-ROBOT utilizará como prueba competitiva una de las prácticas del entorno.

Agradecimientos

Este trabajo ha sido financiado parcialmente por la Comunidad de Madrid a través del proyecto RoboCity2030-III (S2013/MIT-2748) y por el Ministerio de Economía y Competitividad de España a través del proyecto RETOGAR (TIN2016-76515-R). Los autores agradecen a Google la financiación al proyecto JdeRobot en sus convocatorias *Google Summer of Code* de 2015 y 2017.

Referencias

Aliane, N., 2011. Teaching fundamentals of robotics to computer scientists. Computer Applications in Engineering Education 19, 615–620. DOI: 10.1002/cae.20342

Berenguel, M., Rodríguez, F., J.C. Moreno, J. G., González, R., 2016. Tools and methodologies for teaching robotics in computer science & engineering studies. Computer Applications in Engineering Education 24(2), 202–214. DOI: 10.1002/cae.21698

Blank, D., Kumar, D., Meeden, L., Yanco, H., 2004. Pyro: A python-based versatile programming environment for teaching robotics. Journal of Educational Resources in Computing.

Blank, D., Kumar, D., Meeden, L., Yanco, H., 2006. The pyro toolkit for ai and robotics. AI Magzine 27(1).

Cañas, J., Agüero, C., Barrera, P., Matellán, V., Morales, R., 2007. Robótica Móvil y Programación en Educación Secundaria, Robocampeones 2007. ISBN: 978-84-690-9401-3.

Cañas, J., Cazorla, M., Matellán, V., 2009. Uso de simuladores en docencia de robótica móvil. IEEE Revista Iberoamericana de Tecnologías del Aprendizaje 4(4), 268–277.
DOI: ISSN: 1932-8540

Cañas, J., Martín, L., Vega, J., 2014. Innovating in robotics education with gazebo simulator and jderobot framework. In: Proceedings of XXII Congreso Universitario de Innovación Educativa en Enseñanzas Técnicas, CUIEET-2014. pp. 1483–1496.

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.
DOI: 10.1016/j.riai.2015.04.005

Corke, P., 1996. A robotics toolbox for matlab. IEEE Robotics and Automation Magazine 3, 24–32.

Corke, P., 2011. Robotics, Vision and Control: Fundamental Algorithms in MATLAB (Springer Tracts in Advanced Robotics). Springer.

Corke, P., 2015. Robotics, toolbox for matlab (release 9)). http://www.petercorke.com/, accessed: 2016-09-15.

Corke, P., Greener, E., Philip, R., jun 2016. An innovative educational change. IEEE Robotics & Automation Magazine, 81–89.
DOI: 10.1109/MRA.2016.2548779

Detry, R., Corke, P., Freese, M., 2014. TRS: An open-source recipe for teaching/learning robotics with a simulator. http://ulgrobotics.github.io/trs, accessed: 2016-09-15.

Fabregas, E., Farias, G., Dormido-Canto, S., Guinaldo, M., Sánchez, J., Dormido, S., 2016. Platform for teaching mobile robotics. J. Intell. Robot Syst. 81, 131–143.
DOI: 10.1007/s10846-015-0229-8

Gil, A., Reinoso, O., Marín, J., Paya, L., Ruiz, J., 2015. Development and deployment of a new robotics toolbox for education. Computer Applications in Engineering Education 23, 443–454.
DOI: 10.1002/cae.21615

González, R., Mahulea, C., Kloetzer, M., 2015. A matlab-based interactive simulator for mobile robotics. In: IEEE CASE'2015: Int. Conf. on Autom. Science and Engineering.

GTRob, C., 2008. Libro blanco de la robótica: De la investigación al desarrollo tecnológico y aplicaciones futuras. CEA - GTRob, España.

Guyot, L., Heiniger, N., Michel, O., Rohrer, F., 2011. Teaching robotics with an open curriculum based on the e-puck robot, simulations and competitions. In: Stelzer, R., Jafarmadar, K. (Eds.), Proceedings of 2nd International Conference on Robotics in Education (RiE 2011). INNOC - Austrian Society for Innovative Computer Sciences, pp. 53–58.

Guzmán, J., Berenguel, M., Rodríguez, F., Dormido, S., 2008. An interactive tool for mobile robot motion planning. Robotics and Autonomous Systems 56, 396–409.

Jara, C., Candelas, F., Pomares, J., Torres, F., 2013. Java software platform for the development of advanced robotic virtual laboratories. Computer Applications in Engineering Education 21, 14–30.
DOI: 10.1002/cae.20542

Jara, C. A., Candelas, F. A., Puente, S., Torres, F., 2011. Hands-on experiences of undergraduate students in automatics and robotics using a virtual and remote laboratory. Computers & Education 57 (4), 2451–2461.

Jiménez, E., Bravo, E., Bacca, E., 2010. Tool for experimenting with concepts of mobile robotics as applied to children education. IEEE Trans. Education 53 (1), 88–95.

Joseph, L., 2015. Learning Robotics using Python. Packt Publishing, UK.

López-Nicolás, G., Romeo, A., Guerrero, J., june 2009. Simulation tools for active learning in robot control and programming. In: Proceedings of 20th EAEEIE Annual Conference: Innovation in Education for Electrical and Information Engineering. IEEE.

López-Nicolás, G., Romeo, A., Guerrero, J., 2014. Active learning in robotics based on simulation tools. Computer Applications in Engineering Education 22, 509–515.
DOI: 10.1002/cae.20576

Mateo, T., Andújar, J., 2011. 3D-RAS: A new educational simulation tool for kinematics analysis of anthropomorphic robotic arms. Int. J. Eng. Educ. 27 (2), 225–237.

Mateo, T., Andújar, J., december 2012. Simulation tool for teaching and learning 3d kinematics workspaces of serial robotic arms with up to 5-dof. Comput. Appl. Eng. Educ. 20 (4), 750–761.

Simmons, R., Fernandez, J., Goodwin, R., Koenig, S., O'Sullivan, J., June 2000. Lessons learned from xavier. IEEE Robotics and Automation Magazine 7 (2), 33–39.

Soto, A., Espinace, P., Mitnik, R., 2006. A mobile robotics course for undergraduate students in computer science. In: Proceedings of IEEE 3rd Latin American Robotics Symposium, LARS'06. IEEE, pp. 187–192.

Thrun, S., 2006. Teaching challenge. IEEE Robotics and Automation Magazine 13 (4).

Touretzky, D., 2013. Robotics for computer scientists: what's the big idea? Computer Science Education 23(4), 349–367.
DOI: 0.1080/08993408.2013.847226