https://doi.org/10.17979/spudc.9788497498081.0610

SIMULACIÓN DE KHEPERA IV EN V-REP

Ernesto Fabregas[1], Emmanuel Peralta[2], Gonzalo Farias[2], Héctor Vargas[2], Sebastián Dormido[1]
[1] Departamento de Informática y Automática. Universidad Nacional de Educación a Distancia
Juan del Rosal, 16, 28040, Madrid, Spain, e-mail: efabregas@bec.uned.es
[2] Pontificia Universidad Católica de Valparaíso. Av. Brasil, 2147, Valparaíso, Chile

Resumen

En los últimos años la robótica ha tenido un gran impacto en los entornos educativos a todos los niveles. En estos ámbitos los simuladores han jugado un papel fundamental en el uso de la Robótica en el proceso de enseñanza - aprendizaje. El presente artículo describe el diseño, la implementación y las pruebas de un modelo del robot Khepera IV *para incorporarlo al simulador* V-REP*. El objetivo fundamental es obtener una herramienta lista para ser usada en enseñanza de la Robótica en el Área la de Ingeniería de Control.*

Palabras clave: Khepera IV, Simulador V-REP, Enseñanza de Ingeniería de Control.

1 INTRODUCCIÓN

Hace algunos años los robots eran máquinas grandes y caras, que a menudo tenían que estar conectadas a ordenadores por medio de cables para poder funcionar. Esto mantuvo a los robots alejados de las aulas durante mucho tiempo. En la actualidad, la incorporación de la robótica a las aulas se ha incrementado a todos los niveles.

La robótica es una combinación de varias disciplinas como son: matemática, física, mecánica, cinemática, electrónica, programación, sistemas de control, etc. Esto hace a la robótica una herramienta muy versátil desde el punto de vista pedagógico.

Los robots son el perfecto ejemplo de un sistema de control desde el punto de vista de la enseñanza del Ingeniería de Control. En un robot se pueden incluir varios bucles de control que incluyen el controlador, sensores y actuadores.

En este ámbito, el uso de simuladores ha tenido un gran impacto en el proceso de enseñanza - aprendizaje, debido a que la experimentación virtual ofrece a los estudiantes la posibilidad de observar rápidamente los resultados de sus experimentos en un ambiente interactivo, atractivo y sin dañar los robots.

En el mercado existen varios simuladores que pueden ser usados gratuitamente con fines educativos. Algunos de los más conocidos y utilizados son: *Gazeebo* [3], *ARGoS* [14], *Webots* [8] y *V-REP* [9]. Estos simuladores tienen incorporados muchos modelos de robots reales muy conocidos como por ejemplo: Khepera III y Pioneer P3-DX, que son a su vez muy utilizados en muchas universidades alrededor de todo el mundo.

El trabajo está dividido como se describe a continuación: La Sección 2 presenta las principales características de los simuladores de robótica más usados de los existentes actualmente en el mercado. La Sección 3 presenta la implementación de la librería del *Khepera IV* en *V-REP*. La Sección 4 muestra algunos experimentos con la Librería. Finalmente la Sección 5 presenta las conclusiones y trabajos futuros.

2 SIMULACIONES DE ROBÓTICA MÓVIL

En esa sección se describen las características de los principales simuladores de robótica existentes en el mercado, incluyendo el *V-REP* que es en el que se basa esta investigación.

2.1 Simuladores de Robótica

En la actualidad existe una gran variedad de simuladores de robótica. La mayoría de ellos son de pago, pero algunos poneen licencias gratuitas para su uso con fines educativos. A continuación se presentan y describen las principales características de los más significativos [15].

ARGoS es un simulador para experimentos con varios robots (*multi-robots*). Ha sido diseñado para simular comportamientos de grandes enjambres de robots eficientemente en tiempo real. Es un simulador modular paralelo que permite al usuario presonalizar sus características en un ambiente virtual muy atractivo. Su característica más distintiva es que el ambiente de simulación en 3D puede ser dividido en regiones y cada una de estas regiones puede ser asignda a un procesador diferente, lo que hace que funcione muy efi-

cientemente a pesar de simular muchos modelos al mismo tiempo. Todos sus componentes son "*plug-ings*" (modelos de robots, sensores, actuadores, motores, etc.). Los usuarios pueden extender o sobreescribir estos "*plug-ins*" para crear nuevos modelos de robots e implementar sus simulaciones. *ARGoS* está disponible solamente para *Linux* y *MacOS* debido a las limitaciones que impone el sistema de la DLLs de *Windows*.

Webots es un entorno de desarrollo integrado diseñado para modelar y simular robots. Este simulador posee una amplia lista de sensores y actuadores que permiten a los usuarios diseñar e implementar sus propios robots en un ambiente en 3D. Los contrladores pueden ser programados dentro del propio entorno o en otros entornos e importados a emphWebots. Una vez programandos, lo controladores pueden ser exportados a algunos de los robots comerciales más conocidos del mercado. Lo que popsibilita probar dichos controladores en los robots reales. Este simulador es sin dudas, uno de los más usados en las universidades de todo el mundo para la enseñanza de Ingeniería de Control. Su código fuente ha sido mantenido continuamente durante los últimos 19 años. Actualmente está disponible para *Linux*, *MacOS* y *Windows*.

RFCSIM Es un simulador en 2D desarrollado en nuestro Departamento con fines educativos. Ha sido desarrollado en EJS (Easy Java Simulations) y su prósito fundamental es el desarrollo de experimentos con robóts móviles diferenciales. Permite implementar experimentos basados en control de posición de un robot móvil o control de formaciones de varios robots. Incluye además la implementación de varios algoritmos de evitación de obstáculos. Está disponible en todos los sistemas operativos que soporten la Máquina Virtual de Java [6], [7].

2.2 *V-REP*

El *V-REP* ("*Virtual Robot Experimentation Platform*") es un simulador escalable que permite la implementación de simulaciones en 3D. Esta herramienta fué creada en 2010 y desde su creación ha experimentado un rápido crecimiento, a tal punto que hoy es sin dudas, el simulador más usado en cursos de robótica en universidades de todo el mundo.

V-REP tiene un entorno de desarrollo integrado (IDE) basado en una estructura de control distribuida. En la que cada objeto o modelo puede ser individualmente controlado por medio de su propio código fuente. Los controladores pueden ser programados en varios lenguajes: *C/C++*, *Python*, *Java*, *Lua*, *Matlab*, *Octave* or *Urbi* [9]. El simulador tiene incorporados un gran número de modelos de robots, sensores, actuadores, etc. Con ellos se pueden diseñar y construir robots personalizados, que a su vez pueden interacutar directamente en tiempo de ejecución con la larga colección de ejemplos de robots que posee *V-REP*.

Una caracterísitica importante de este simulador es puede importar modelos gráficos diseñados en 3D (*.stl). Sobre estos modelos gráficos se pueden incorporar los sensores y actuadores de *V-REP*, de modo que se pueden construir robots totalemnte presonalizados, como es el caso del presente trabajo.

3 MODELO DEL *KHEPERA IV*

En esta Sección se presenta el robot en cuestión y se describe el desarrollo de la librería para incorporar dicho robot al simulador *V-REP*.

El robot *Khepera IV* es el último producto de la compañía Suiza KTEAM [9]; ha sido desarrollado con propósitos docentes. Posee comunicación inalámbrica por *bluetooth* y *WI-FI* y sensores de ultrasonidos, infrarrojos, acelrómetro, giróscopo y una cámara a bordo. El robot puede ser representado por un modelo diferencial con ruedas accionadas por dos motores independientes. La figura 1 muestra dicho robot.

Figura 1: Robot *Khepera IV*

3.1 Modelo del Robot Khepera IV

Todos los componentes visuales del robot (la carcasa, las ruedas, la base, etc...) se modelaron utilizando Autodesk Inventor [1], se importaron a *V-REP* y se enseamblaron. Posteriormente se acoplaron y configuraron los sensores y la cámara usando los elementos de *V-REP*. La Figura 2 muestra parte de este proceso.

Para probar el modelo desarrollado se implementó el experimento de control de posición del robot. Este experimento consiste en que el robot alcance el punto de destino (T_p), conociendo en todo momento su posición actual (P) y el punto al que desea llegar [12].

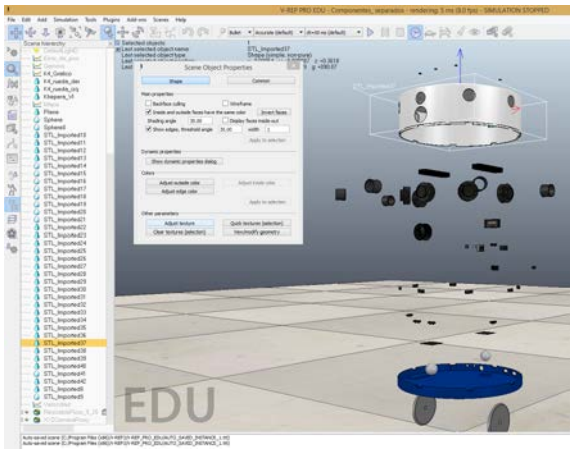

Figura 2: Componentes del modelo del robot

En este experimento se debe diseñar la ley de control que reciba como entradas la distancia (d) y el ángulo (θ) al que se encuentra el punto de destino y devuelva como salidas las velocidades angular (ω) y lineal (ν) del robot para alcanzar dicho punto [13], [4]. La Figura 3 muestra el diagrama en bloques de este experimento. Como se puede apreciar la distancia y el ángulo al que se encuentra el punto de destino también se calcula en el algoritmo de control, pero en el diagrama se ha separado en un bloque independiente para que pueda observarse con mayor claridad.

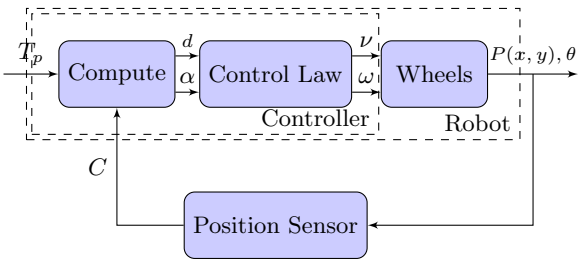

Figura 3: Diagrama en bloques del control de posición

En este caso, para probar el modelo se ha implementado la ley de control [16]. Las Ecuaciones 1 y 2 se ejecutan en el Bloque *Compute*.

$$d = \sqrt{(y_p - y_r)^2 + (x_p - x_r)^2} \tag{1}$$

$$\alpha = \tan^{-1}\left(\frac{y_p - y_r}{x_p - x_r}\right) \tag{2}$$

Mientras que las Ecuaciones 3 y 4 representan el cálculo de la ley de control que se ejecuta en el Bloque *Control Law*

$$\nu = \begin{cases} \nu_{max} & if\ |d| > k_r \\ d\left(\frac{\nu_{max}}{k_r}\right) & if\ |d| \leq k_r \end{cases} \tag{3}$$

$$\omega = \omega_{max} sin\,(\alpha - \theta) \tag{4}$$

En el caso de la velocidad lineal (ν), como se puede apreciar el robot alcanza la velocidad lineal máxima (ν_{max}) cuando está se encuentra muy alejado del punto de destino (si $d > k_r$). Mientras que cuando se encuentre cerca del punto de destino (si $d < k_r$), la velocidad lineal será inversamente proporcional a dicha distancia, para que el robot vaya frenando según se acerca al punto de destino.

En el caso de la velocidad angular (ω), su valor dependerá del seno del error de ángulo (diferencia entre la orientación del robot (θ) y el ángulo al que se encuente el punto (α). Por lo que la velocidad angular será máxima (ω_{max}) cuando la diferncia de ángulos sea $\pm 90°$. Lo que implicará que el robot debe priorizar el giro en detrimento de la velocidad lineal. El siguiente segmento de código muestra la implementación de dicha ley de control en *V-REP*.

```
1  pos = simGetObjectPosition(body,-1)
2  target = simGetObjectPosition(Target,-1)
3  robotOr = simGetObjectOrientation(body,-1)
4  while(d<0.01) do
5    gamma = robotOr[3]
6    Xr = pos[1]
7    Yr = pos[2]
8    Xp = target[1]
9    Yp = target[2]
10   d = math.sqrt(((Xp-Xr)^2)+((Yp-Yr)^2))
11   alpha = math.atan2(Yp-Yr,Xp-Xr)
12   W = (Wmax*math.sin(alpha-theta))
13   if (d>Kr) then
14       V=Vmax
15   else
16       V=d*(Vmax/Kr)
17   end
18   Vr = (2*V+W*L)/2
19   Vl = (2*V-W*L)/2
20   simSetJointTargetVelocity(leftMotor,Vl)
21   simSetJointTargetVelocity(rightMotor,Vr)
22 end
```

La posición del robot (X_r;Y_r), su orientación (θ) y las coordenadas del punto de destino $P(X_p;Y_p)$ se obtienen en las líneas de la 1 a la 8. Mientras que en las líneas 10 y 11 se calculan la distancia (d) y el ángulo (α) a dicho punto. La velocidad angular (ω) se calcula en la línea 12 y la velocidad lineal (ν) se calcula en las líneas de la 12 a la 17. Las líneas de la 18 a la 21 calculan las velocidades de los motores izquierdo y derecho del modelo diferencial y las envían a los motores.

Este experimento es relativamente sencillo y es la base del resto de los que se realizan con este tipo de robot. Porque al tener controlada la posición de robot, se puede pasar entonces a implementar algoritmos más complejos como por ejemplo: evitación de obstáculos, seguimiento de trayectorias y control de formaciones.

4 PROBLEMAS DE CONTROL

En esta Sección se muestran los resultados de las implementaciones de algunos problemas típicos de control, usando el modelo desarrollado.

4.1 Experimento multi-robots

Para probar la integración del modelo desarrollado para el *Khepera IV* se ha implementado un experimento de control multi-robots. Para ello se ha incorporó al experimento del control de posición, un Cuadrirrotor que trae incorporado el *V-REP*. El objetivo de este experimento es que el Cuadrirrotor siga al nuestro robot.

El modelo del Cuadrirrotor tiene implementado su propio controlador de posición. En este caso las coordenadas de su punto de destino tiene coordenadas en los 3 ejes. Para que el Cuadrirrotor pueda seguir al *Khepera* las coordenadas (x,y) del punto de destino del DRON se cambian por las coordenadas (x,y) de la posición actual del robot. De esta forma el DRON puede seguir al robot con una altura constante mientras éste intenta alcanzar su punto de destino.

La Figura 4 muestra al robot *Khepera* intentando alcanzar su punto de destino (representado por la semiesfera roja). El Cuadirotor "persigue" al robot desde el aire a una altura constante. En la parte inferior izquierda de la imagen se observa el vídeo obtenido de la cámara de abordo del robot. Mientras que en la parte inferior derecha se observa la imagen obtenida de la cámara que tiene el DRON apuntando al suelo.

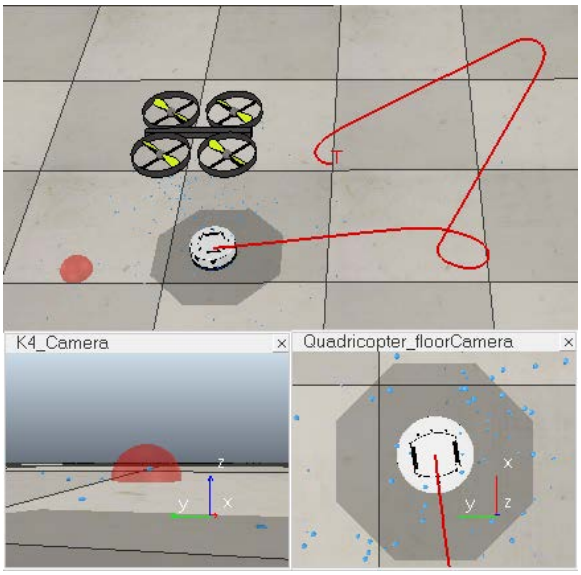

Figura 4: Experimento multi-robots

4.2 Evitación de obstáculos

Este experimento ha sido ampliamente estudiado y para implementarlo se han usado diferentes técnicas como en [11] y [2]. El aspecto más importante en este experimento es el tipo de sensores y sobre todo su distribución geométrica a bordo del robot. Porque de ello depende en gran medida la capacidad que tendrá el robot para poder "ver", según el tamaño y la morfología de los obstáculos. En este sentido la ubicación de los sensores y su alcance de detección pueden constituir una limitante importante para el algoritmo que se quiera implementar.

El robot *Khepera IV* tiene 8 sensores de ultrasonidos con 30 cm de alcance y distribuidos a su alrededor cada 45°. Debido a esta configuración se implementó el algoritmpo de Braitenberg ([17]) que calcula directamente las velocidades angular y lineal en función de los obstáculos detectados.

La Figura 5 muestra el resultado de la simulación para este experimento. Los obstáculos se muestran en colores negro y amarillo. Los conos de color verde representa los sensores de ultrasonidos que no detectan obstáculos y los conos de color rojo los que sí. En la parte inferior derecha de la imagen se observa la imagen obtenida de la cámara de a bordo del robot.

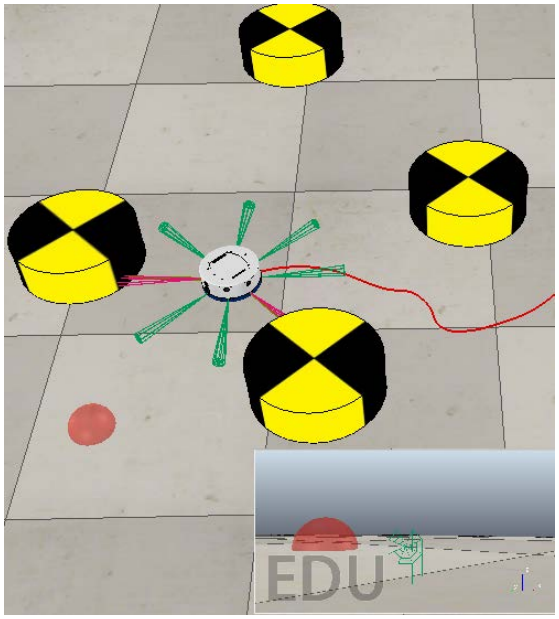

Figura 5: Experimento de evitación de obstáculos

Como se puede apreciar, el robot intenta alcanzar el punto de destino (representado por la semiesfera roja) mientras que evita los obstáculos que encuentra en su camino. Describiendo una trayectoria suave (representada en rojo).

4.3 Seguimiento de Trayectorias

Este experimento ha sido también muy estudiado con propósitos pedagógicos [10]. El mismo consiste en que el robot siga una trayectoria que se le pasa como referencia. La trayectoria se divide en una secuencia de puntos (P_n, $P_{n+1}...P_m$) que el robot debe ir alcanzando (uno tras otro), sin tener en cuenta el tiempo en que alcanza cada uno de ellos. Para ello, se implementa el experimento de control de posición descrito anteriormente y ejecutarlo para cada punto que reciba. En este experimento el robot solamente tiene en cuenta el punto de destino actual (P_n), por lo que la orientación con que alcance dicho punto, será determinante para alcanzar el siguiente punto debidamente sin desviarse de la trajectoria. La Figura 6 muestra el resultado de este experimento para una trayectoria de *Lissajous*. Como se puede apreciar, el robot sigue la trayectoria correctamente sin desviarse.

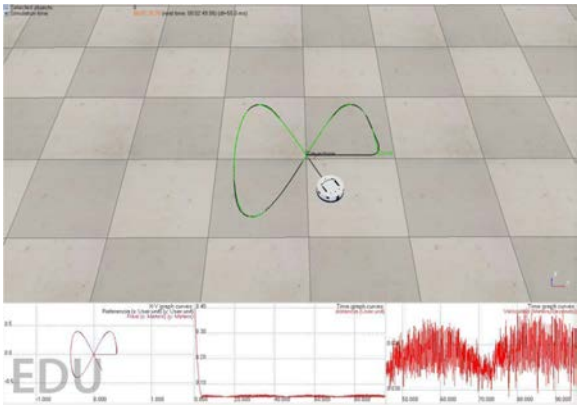

Figura 6: Experimento de seguimiento de trayectorias

4.4 Seguimiento de Rutas

Este experimento puede ser considerado como el paso siguiente del experimento anterior [5]. Igualmente el robot debe seguir una trayectoria que se le pasa como referencia, pero en este caso sí debe tener en cuenta el tiempo en el que alcanza cada uno de ellos. La trayectoria se divide en una secuencia de puntos, que el robot debe ir alcanzando. En este caso, las velocidades se calculan usando el tiempo, el punto de destino actual (P_n) y el siguiente punto de destino (P_{n+1}). La Figura 7 muestra una comparación entre estos dos experimentos. La trayectoria se muestra en color negro y está dividida en los puntos (P_1, P_2, P_3 y P_4). La línea discontínua roja representa el experimento de seguimiento de trajectorias y la línea negra el de seguimiento de ruta.

Este experimento se puede considerar una mejora del caso anterior debido a que al tener en cuenta el siguiente punto de destino, el robot alcanzará el actual punto de destino con una orientación apropiada (hacia el siguiente), para no desviarse de la trayectoria. La Figura 8 muestra los resultados de este experimento para el caso de una trayectoria cirucular. Como se puede apreciar el robot sigue correctamente la trayectoria teniendo en cuenta el tiempo.

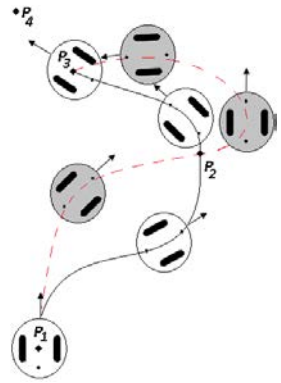

Figura 7: Seguimiento de trayectorias vs. seguimiento de rutas

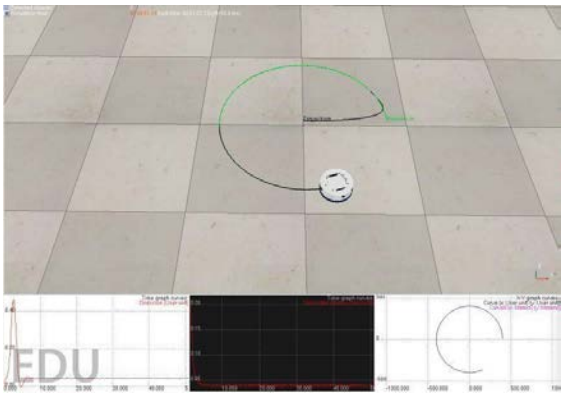

Figura 8: Experimento de seguimiento de rutas

5 Conclusiones

En este artículo se presenta el modelo del robot *Khepera IV* para el Simulador *V-REP*. Este robot es el más reciente diseño de la compañía Suiza K-Team y el *V-REP* es el simulador de robótica más utilizado en ambientes educativos.

El modelo incluye todos los componentes del robot real (motores, ruedas, sensores, cámara, etc...). Por lo que con este modelo se pueden diseñar y probar experimentos que después pueden trasladarse al robot real.

Para demostrar el funcionamiento del modelo se implementaron y probron un conjunto de experimentos típicos: control de posición, evitación

de obstáculos, seguimiento de trayecotorias y seguimiento de rutas. Además se demostró la integración del modelo del robot con otros existentes en el simulador (modelo del Cuadrirrotor).

Con el modelo desarrollado se pueden implementar diferentes algoritmos de control de posición de robots móviles y experimentos multi-robots. Esta herramienta puede ser utilizada en cursos de Ingeniería de Control para enseñar conceptos básicos de robótica móvil.

En el futuro cercano planeamos desarrollar experimentos de control de sistemas multi-agentes. Por medio de la incorporación de más robots *Khepera* a las simulaciones y hacerlos interactuar con otros modelos existentes. Por ejemplo en el experimento multi-robot presentado, utilizar las imagenes de la cámara del DRON para posicionarlo encima del robot o para que detecte obstáculos y le transmita sus posiciones al robot.

Referencias

[1] Autodesk Inc. Autodesk Inventor, 2015.

[2] J. Borenstein and Y. Koren. The vector field histogram-fast obstacle avoidance for mobile robots. *Robotics and Automation, IEEE Transactions on*, 7(3):278–288, Jun 1991.

[3] C., V., R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, and M. Dorigo. ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence*, 6(4):271–295, 2012.

[4] D. Chwa, S. Hong, and B. Song. Robust posture stabilization of wheeled mobile robots in polar coordinates. In *The 17th International Symposium on Mathematical Theory of Networks and Systems*, volume 39, pages 343–348, 2006.

[5] P. Encarnacao and A. Pascoal. 3D path following for autonomous underwater vehicle. In *Proc. 39th IEEE Conference on Decision and Control*, 2000.

[6] E. Fabregas, G. Farias, S. Dormido-Canto, and S. Dormido. RFCSIM simulador interactivo de robótica móvil para control de formación con evitación de obstáculos. In *XVI Congreso Latinoamericano de Control Automático, Cancún, Quintana Roo, México*, 2014.

[7] E. Fabregas, G. Farias, S. Dormido-Canto, M. Guinaldo, J. Sánchez, and S. Dormido Bencomo. Platform for teaching mobile robotics. *Journal of Intelligent & Robotic Systems*, 81(1):131–143, 2016.

[8] L. Guyot, N. Heiniger, O. Michel, and F. Rohrer. Teaching robotics with an open curriculum based on the e-puck robot, simulations and competitions. In *Proceedings of the 2nd International Conference on Robotics in Education. Vienna, Austria*, 2011.

[9] K-Team. V-REP user manual, 2015.

[10] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi. A stable tracking control method for an autonomous mobile robot. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pages 384–389 vol.1, May 1990.

[11] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90–98, 1986.

[12] F. Kühne, W. F. Lages, and J. M. Gomes da Silva Jr. Point stabilization of mobile robots with nonlinear model predictive control. In *Mechatronics and Automation, 2005 IEEE International Conference*, volume 3, pages 1163–1168. IEEE, 2005.

[13] P. KyuCheol, Ch. Hakyoung, and L. Jang Gyu. Point stabilization of mobile robots via state space exact feedback linearization. *Robotics and Computer-Integrated Manufacturing*, 16(5):353–363, 2000.

[14] C. Pinciroli, V. Trianni, and R. O'Grady. ARGoS: A modular, multi-engine simulator for heterogeneous swarm robotics. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 5027–5034, Sept 2011.

[15] A. Staranowicz and G. L. Mariottini. A survey and comparison of commercial and open-source robotic simulator software. In *Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments*, page 56. ACM, 2011.

[16] V. J. G. Villela, R. Parkin, M. L. Parra, J. M. González, M. J. G. Liho, and H. Way. A wheeled mobile robot with obstacle avoidance capability. *Tecnología Y Desarrollo*, 1(5):159–166, 2004.

[17] Xiaoyu Yang, Rajnikant V Patel, and Mehrdad Moallem. A fuzzy braitenberg navigation strategy for differential drive mobile robots. *Journal of Intelligent and Robotic Systems*, 47(2):101–124, 2006.

Agradecimientos

Este trabajo ha sido financiado por el Ministerio de Economía y Competitividad de España bajo el Proyecto de Ivestigación DPI2012-31303.