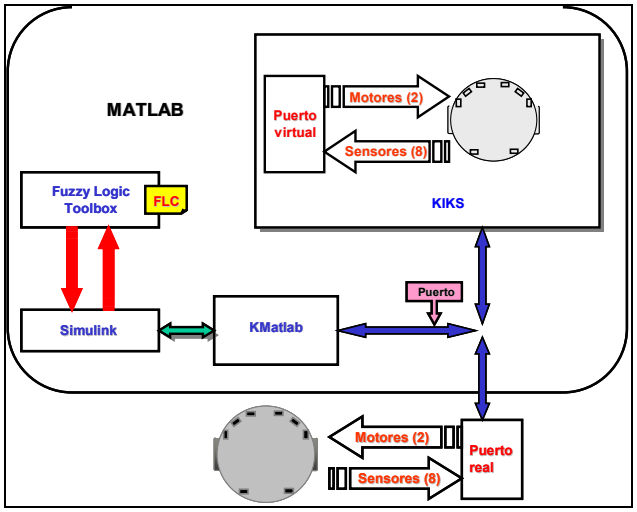
Plataforma para Aprendizaje Evolutivo de Reglas Fuzzy para el Guiado de Robots Móviles



❒ **Eric Vallejo, Dr.**
Docente del Dpto. de Ingenierías Eléctrica y Electrónica, Universidad del Norte, Barranquilla, Colombia.

Grupo de Investigación en Robótica y Sistemas Inteligentes de la Universidad del Norte.
Tel. 57+5+3509750
evallejo@uninorte.edu.co

❒ **Ginés Benet, Dr.**
DISCA Universidad Politécnica de Valencia, España.

Grupo de trabajo de sensorización y percepción en robótica móvil.
Tel. 34+96 3875775.
gbenet@disca.upv.es

RESUMEN

Se presenta la creación de una plataforma en software que incluye un algoritmo para evolucionar el aprendizaje de FLCs para robots móviles. Para ello se utilizan diversas herramientas de apoyo al diseño y simulación. Con base en SIMULINK y otras herramientas de MATLAB, se diseña e implementa la plataforma de trabajo enfocada en particular a un robot Khepera pero aplicable a otros robots que operen con estrategias reactivas-deliberativas. La navegación está basada comportamientos simples desarrollados como sistemas de inferencia fuzzy. Los controladores se generan automáticamente como una variante de los Sistemas Clasificadores. Los controladores se han probado tanto en simulación como con un robot real para verificar su comportamiento.

PALABRAS CLAVES

Aprendizaje de Máquina, Computación Evolutiva, Inteligencia Artificial, Sistemas Clasificadores.

ABSTRACT

In this work, the creation of a software platform that includes an algorithm to learn FLCs for mobile robots is presented. For it, diverse support tools to the design and simulation are used. With base in SIMULINK and other tools of MATLAB, the platform is designed and implemented focused to a particular robot, the Khepera, but applicable to other robots that operate with reactivate-deliberative strategies. The navigation is based on behaviours developed by means of fuzzy inference systems. The controllers are generated automatically as a variant of the Classifier Systems. The controllers have been proven as much in simulation as with a real robot to verify its behaviour.

KEYWORDS

Artificial Intelligence, Classifier Systems, Evolutionary Computing, Machine Learning.

1. INTRODUCCIÓN

La navegación autónoma de robots en entornos no estructurados constituye uno de los retos más importantes en el campo de la robótica móvil.
En este trabajo nos hemos orientado a mejorar y aplicar un algoritmo de aprendizaje sobre una plataforma de software, desarrollada especialmente para este fin, para generar controladores con inferencia fuzzy. Ella facilita el desarrollo de sistemas de control con bloques de fácil interconexión, y permite además la validación, operando directamente un robot simulado o uno real. Se utiliza un mini-robot muy conocido en robótica móvil, el Khepera. Por otro lado, se ha dispuesto de herramientas en software para diseño y simulación, proporcionadas tanto por los fabricantes del robot como por otras fuentes, además de MATLAB y algunas de sus *toolboxes*.
Este artículo se divide en cuatro partes: en la primera se presentan los aspectos necesarios para una adecuada conceptuación del trabajo. En la segunda parte se describen algunas de las herramientas utilizadas. La parte tercera muestra la realización experimental. En la última se ofrecen las conclusiones y propuestas para trabajo posterior.

2. CONTROL INTELIGENTE

2.1. Inteligencia artificial

El concepto de Inteligencia Artificial (AI) puede tener diferentes connotaciones, dependiendo de la fuente que se tome como referencia. Lo que sí es claro es que la AI tiene como uno de sus objetivos el estudio del comportamiento inteligente de las máquinas [8].

2.2 Controladores inteligentes

El término control inteligente se refiere tanto al enfoque del diseño de control como a la implantación de técnicas de AI [9]. El paradigma del control inteligente se enfoca a aproximaciones de inspiración biológica, aunque los controladores resultantes tal vez difieran drásticamente de ella y que a menudo parezcan simplemente controladores adaptables [1]. Los Sistemas Expertos, por ejemplo, tienen su predecesor en técnicas de control convencional con controladores basados en autómatas, redes de Petri y otros sistemas de eventos discretos [14]. Algunos sistemas basados en el conocimiento, encuentran aplicación en automatizar características de percepción, conocimiento y toma de decisiones propias de los operarios humanos. A las Redes Neuronales Artificiales (ANN) se les ha utilizado, por ejemplo, para aprender la forma de controlar un sistema "observando" las acciones que efectúa un operario. Los Algoritmos Genéticos (GA) se utilizan en el diseño asistido por computadora para "evolucionar" controladores bajo el principio de supervivencia.

2.3 Control de robots móviles

Las alternativas para el control y guiado de robots móviles en entornos no estructurados totalmente podrían reunirse en tres grandes grupos: paradigmas jerárquicos, reactivos e híbridos [10], como lo vemos en la figura 1.
En los sistemas jerárquicos la planificación incluye un modelado, y pueden entregar una solución óptima si el problema y el entorno están bien definidos. En los sistemas con control reactivo los sensores están directamente relacionados con las acciones de control. Estas máquinas constituyen los vehículos básicos de Braitenberg [6]. Las estructuras híbridas poseen una base reactiva en un nivel bajo y un bloque planificador en otro superior.

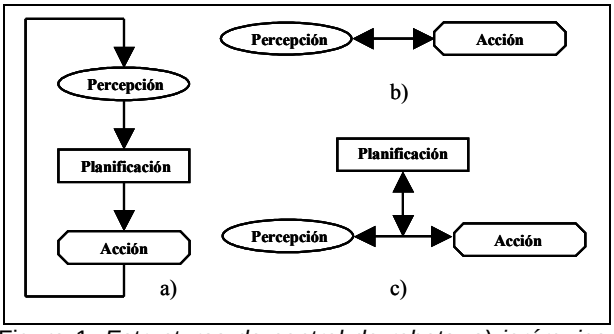


Figura 1. *Estructuras de control de robots: a) jerárquica, b) reactiva, c) híbrida.*

El concepto de comportamientos (behaviours) fue propuesto por Brooks [7] hace ya algunas décadas. Se trata de una arquitectura multicapas sobre la cual es posible instalar una gran variedad de métodos de control, tanto en la parte reactiva como en la activación ponderada de los comportamientos.
1) Control borroso en robótica: Difícilmente puede lograrse la representación sistemática de la ambigüedad y la vaguedad, propias del conocimiento común [11] [12]. La lógica borrosa permite representar la incertidumbre y la ambigüedad, por lo cual se le ha utilizado en sistemas inteligentes para transferir conocimiento común y experiencia. Por ejemplo, la inferencia

SI *(x* ***es*** *A)* ***Y*** *(y* ***es*** *B)* ***ENTONCES*** *(z* ***es*** *C)*

podría traducirse en un comportamiento elemental como *"Si hay un obstáculo al frente y hay espacio libre a la derecha, entonces girar a la derecha"*.

2) Aprendizaje: El aprendizaje elimina la programación de algoritmos específicos, creando inteligencia de forma muy cercana a los métodos biológicos.

3. SIMULACIÓN DE ROBOTS

Con los simuladores es posible diseñar y probar estrategias de control de manera rápida, corrigiendo acciones inadecuadas sin poner en riesgo al robot o al entorno.

Aquí nos referiremos a dos programas de libre distribución para la simulación de robots:

1) KMatLab: El KMatLab fue creado por el K-Team [15] para operar el Khepera desde MatLab a través de un puerto serie del ordenador. Se trata de un conjunto de librerías *.dll* para Windows para ejecutar las instrucciones de bajo nivel propias del Khepera. Por ejemplo, una instrucción de KMatLab tiene la forma

KSetSpeed(ref, left, right)

en la cual *ref* es la variable en MatLab que representa la dirección del puerto serie al que está conectado el robot y otros parámetros de la comunicación. *left* y *right* representan los valores de velocidad para los motores del robot. La instrucción del ejemplo es totalmente equivalente a la instrucción D del Khepera [15].

2) KIKS: KIKS (Kiks Is a Khepera Simulator) es un programa que permite interactuar con un robot Khepera simulado. KIKS es un simulador sobre KMatLab, aunque también permite la comunicación con un robot real [13]. Como ejemplo tomemos

ref = kiks_Kopen([port, baud, 1])

que le indica al sistema el puerto que se habilitará para el robot, la velocidad de comunicación y el tiempo muerto para la comunicación. Esta instrucción es equivalente en KMatLab a

ref = Kopen([port, baud, 1])

4. TRABAJO EXPERIMENTAL

4.1. Plataforma para diseño de controladores

En un trabajo publicado [16] se mostró la viabilidad de utilizar herramientas de apoyo al diseño de software (CASD), recurriendo a una plataforma de experimentación modular desarrollada con MatLab, Simulink y algunas Toolboxes. Sobre esta pueden generarse controladores con lógica borrosa (FLCs) y trasladarse al robot a través de un puerto de la PC. Si el puerto se define como virtual, las acciones del controlador irán al entorno simulado (KIKS). Si se dirigen las acciones a través de un puerto real, se tendrá control sobre el robot real. En la figura 2 se muestran las gráficas de los entornos simulado (KIKS) y real.

Dado que la Fuzzy Logic Toolbox de MatLab no proporciona algunos archivos necesarios para el proceso de aprendizaje adoptado, se adicionaron varias funciones para contar con parámetros los necesarios para el proceso. Se ve entonces la necesidad de modificar la plataforma mencionada, llevándola a lo que esquematizamos en la ilustración representativa de la primera página.

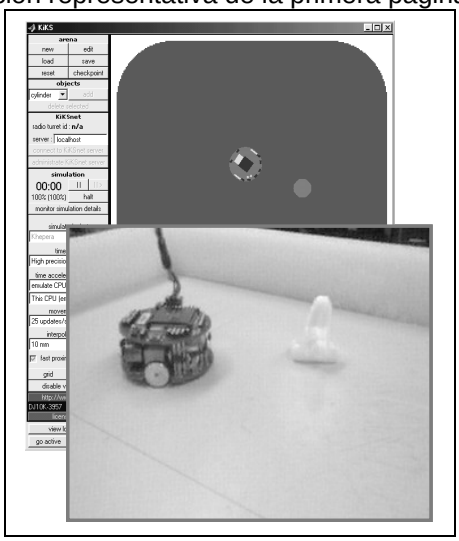


Figura 2. *Entornos simulado y real para robot Khepera.*

4.2. Aprendizaje evolutivo de reglas fuzzy (elf)

El sistema implementado se basa en el algoritmo ELF desarrollado por Bonarini [4], [5], un algoritmo *Q-Learning* adaptado para aprender controladores fuzzy (FLCs) en un robot móvil. Sobre la plataforma en MatLab descrita anteriormente se buscó la manera de hacer más eficiente tal algoritmo.

Algunas diferencias significativas se enfocan especialmente al uso de los algoritmos genéticos, funciones de desempeño (*PEF*) y actualizaciones de la población.

El sistema aprende la combinación de los antecedentes y los consecuentes de las reglas fuzzy para satisfacer los requerimientos especificados por la función de evaluación. La

evolución se efectúa básicamente con el operador de mutación. El controlador está compuesto por un conjunto de reglas cuyo número se ajusta dinámicamente por el sistema. Las funciones de membresía de los conjuntos borrosos del sistema son fijas. El agente puede iniciar con un conjunto de reglas, o sin ninguna regla. En síntesis, el método de aprendizaje es del tipo de modificación estructural y se corresponde con un *Sistema Clasificador* (Classifier System), con modificaciones para generar controladores para robots móviles.

1) El algoritmo ELF: Este algoritmo parte de contar con una población de reglas (ver figura 3). Sus características más relevantes son:

- La población completa de reglas se parte en sub-poblaciones, cuyos miembros comparten los mismos antecedentes (*etiquetas*).
- En cada sub-población se tienen reglas, *compitiendo* entre sí para proponer el mejor consecuente para la situación descrita por el antecedente.
- Todas las sub-poblaciones *cooperan* para producir la acción de control.

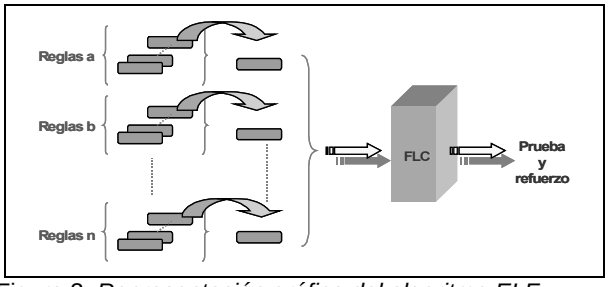


Figura 3. *Representación gráfica del algoritmo ELF*.

De manera simple podemos sintetizar lo expuesto con la gráfica 3. Hemos dividido la población de reglas en sub-poblaciones *a, b, ..., n* que reúnen las reglas que poseen el mismo antecedente y diferente consecuente. De cada grupo se toma de forma aleatoria una regla para generar un FLC que luego se prueba en el sistema. Posteriormente se valora la contribución de cada regla a las acciones para otorgarle el refuerzo correspondiente. De esta forma se consigue que todas las sub-poblaciones cooperen para la acción de control, mientras las reglas compiten entre sí. El ciclo de aprendizaje está compuesto por varios ensayos que a su vez se componen de episodios. Cada episodio está conformado por ciclos de control detección-acción.
La consolidación de una regla está dada por:

$$s_r = s_r(t-1) + (reinforcement(t) - s_r(t-1)) * \frac{currc_r}{pastc_r} \quad (1)$$

en donde s_r representa la solidez de una regla en el ciclo presente, $currc_r$ es un valor entre [0...1] que representa la contribución de la regla a las acciones dadas durante el episodio actual y $pastc_r$ es una medida de la contribución de la regla activada en episodios previos.
Con el fin valorar la contribución de las reglas activadas en episodios anteriores se aplica un refuerzo dado por

$$s_r = s_r(t-1) + (reinforcement(t) - s_r(t-1)) * \frac{currc_r}{pastc_r} * decay \quad (2)$$

estableciendo que hay una correlación entre un episodio y los pasados. Entonces

$$decay = correlation^n$$

Puesto que el número de reglas podría crecer exponencialmente causando una *"explosión"* de ellas, se recurre a una ecuación heurística que las mantiene en un número óptimo.

$$o\_c = \max\left(1, \left[\frac{Max\_reinf - (Max\_reinf * 0.1) - Max\_vote\_sp}{Max\_reinf * 0.1}\right]\right)$$

Max_reinf es el valor del refuerzo máximo y *Max_vote_sp* es el valor máximo obtenido hasta ahora por las reglas de la sub-población.

2) Propuesta: En el trabajo nos hemos enfocado a cuatro aspectos:

- Mayores capacidades de generalización.
- Mejor adaptación a entornos no estructurados.
- Menores limitaciones con elementos dinámicos.
- Mejora del manejo de mínimos locales.

Para ello, el trabajo teórico ha llevado a considerar algunos posibles elementos a experimentar. Uno de ellos es la posibilidad de colocar un valor de penalización en el proceso de aprendizaje, que en el caso de la misión de evitación de obstáculos podría tener la forma

$$pena = f\left(\sum_{i=0}^{5} s_i\right)$$

en donde S_i representa la lectura de los sensores. Las lecturas de los dos sensores frontales se fusionan.
En contraprestación a las buenas acciones de control, podría adicionarse el concepto de placer, que tendría la forma

$$placer = \gamma * f(m_l, m_r) + \beta$$

siendo m_l y m_r las acciones de los motores. γ y β son valores de ponderación. La ecuación de refuerzo para búsqueda de objetivos y evitación de obstáculos podría tener la forma

$$r(t) = g(dist\_inic - dist\_act) * f(dist\_obs) \quad (3)$$

en la que *dist_act* y *dist_inic* son, respectivamente, las distancias actual e inicial con respecto al objetivo y *f* es una función relacionada con la distancia a un obstáculo.

Las reglas fuzzy se representan como cromosomas y sus partes como genes con el fin de utilizar algoritmos genéticos para el aprendizaje. Los valores de cada gen no son estrictamente 0 ó 1, como en los algoritmos genéticos, sino que varían con el valor de membresía de cada variable. La representación de los cromosomas está dada por vectores que componen la matriz que conforma el controlador.

4.3. Pruebas y resultados

A continuación se presentan algunos resultados obtenidos con controladores para evitación de obstáculos y seguimiento de contornos.

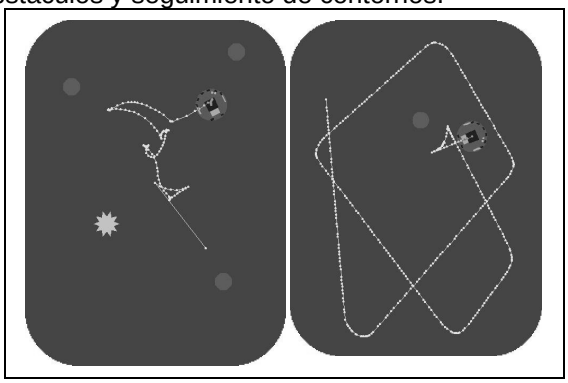


Figura 4. *Aprendizaje y comprobación del controlador.*

1) Evitación de obstáculos: Uno de los comportamientos más requeridos es la navegación con evitación de obstáculos. Se han logrado controladores que valoran positivamente el desplazamiento a gran velocidad en línea recta, evitando choques con los objetos del entorno. En la figura 4 se muestra el comportamiento del robot durante el aprendizaje y su accionar posterior con el controlador aprendido. La ecuación de refuerzo desarrollada para esta tarea es

$$r = \left(1 - \left(\frac{1}{80} * \left(\left|vel_l - vel_r\right| - \left(vel_l + vel_r\right) + \left|vel_l\right| + \left|vel_r\right|\right)\right)\right) * \frac{1}{e^{\sigma}}\left(e^{distn*\sigma} - 1\right)$$

2) Seguimiento de contornos: Otra tarea muy útil en robótica móvil es la de bordear objetos. En la figura 5 se presentan dos ejecuciones, en la primera no se ha colocado durante el entrenamiento la valoración a la trayectoria en línea recta. En la segunda se ha premiado esta conducta. La ecuación de refuerzo utilizada es

$$r = \left(1 - \left(\frac{1}{80} * \left(\left|vel_l - vel_r\right| - \left(vel_l + vel_r\right) + \left|vel_l\right| + \left|vel_r\right|\right)\right)\right) * \left(1 - e^{distn*\lambda}\right)$$

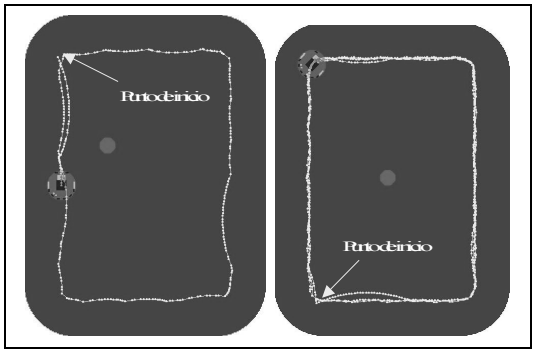


Figura 5. *Aprendizaje de seguimiento de contornos.*

5. CONCLUSIONES Y TRABAJO FUTURO

- La arquitectura del sistema de control es estratificada y modular, así que cada módulo tiene una complejidad baja y puede aprenderse en poco tiempo.
- Se estudió la forma de consolidar las reglas y se optó por soluciones con ecuaciones sencillas de fácil evaluación.
- Con estas estrategias, y la reducción del espacio de búsqueda, se mejora la robustez a datos con ruido, ayudados por la representación de los módulos de control en términos de reglas fuzzy.
- Los resultados obtenidos en el trabajo experimental muestran ser exitosos con sesiones inferiores a 500 ciclos para tareas simples como las mostradas, observando comportamientos adecuados en sesiones de 4 a 5 minutos en entrenamientos con el robot real.

Se pretende llevar las experiencias con el Khepera a otros robots, como YAIR [2], [3], robot móvil desarrollado por el grupo de *Sistemas de Tiempo Real* del Departamento de Informática de Sistemas y Computadores (DISCA) de la Universidad Politécnica de Valencia, y a TELÉMACO, robot del *Grupo de Investigación en Robótica y Sistemas Inteligentes* de la Universidad del Norte.
Con el apoyo de otras herramientas, se podrán obtener controladores para plataformas particulares, codificados en lenguajes de alto nivel. Se está trabajando en desarrollar controladores con capas superiores.

6. REFERENCIAS BIBLIOGRÁFICAS

[1] ATHANS, M. "Why I Despise Fuzzy Feedback Control" IEEE CDC/98 Debates, Tampa, Florida, (1998).

[2] BENET, G. SIMÓ, j. "Using Infrared Sensors for Distance Measurement In Mobile Robots". Robotics and Autonomous Systems, (2002) Vol.40 (p.255-266).

[3] BENET, G. SIMÓ, j. BLANES, F. MARTINEZ, M. "A Multisensor Robot Distributed Architecture" Editorial: Elsevier Science. Information Control Problems in Manufacturing 1998, ISBN 0-08-042928- 9, Páginas: 39 – 44.

[4] BONARINI, A. "Evolutionary learning of general fuzzy rules with biased evaluation functions: competition and cooperation". Proceedings. of the IEEE World congress on Computational Intelligence (WCCI) - Evolutionary Computation, IEEE Computer Press, Piscataway, NJ, 51-56.

[5] BONARINI, A. "Learning behaviors implemented as Fuzzy Logic Controllers for Autonomous Agents". Proceedings of the WEC2, University of Nagoya, Nagoya, J, 21-24.

[6] BRAITENBERG, V. (1984), Fourth printing (1994) "Vehicles", MIT press, Bradford Books, England.

[7] BROOKS, R. "A Robust Layered Control System for a Mobile Robot", A. I. Memo 864, MIT, Artificial Intelligence Laboratory (1985).

[8] BROOKS, R.. STEIN, L. A "Building Brains for Bodies", A. I. Memo 1439, MIT, Artificial Intelligence Laboratory (1993).

[9] DE ANDRÉS, T. "Homo cybersapiens: La inteligencia artificial y la humana". EUNSA, España, ISBN: 84-313-1982-8, 2002.

[10] GULLEY, N. "Fuzzy Logic Toolbox". MathWorks, 1995.

[11] MURPHY, R. "Introduction to AI Robotics", MIT press, Bradford Books, England, ISBN: 0-262-13383-0, 2000.

[12] NILSSON, N. "Inteligencia Artificial; una nueva síntesis", Ed. McGraw-Hill, España, ISBN: 84-481-2824-9, 2001.

[13] NILSSON, T. "KIKS User Guide", 2001. www.tstorm.se/projects/kiks/.

[14] PASSINO, ÖZGÜNER, K. Ü. "Intelligent Control: From Theory to Application", Guest Editors' Introduction to the IEEE Expert Special Track on Intelligent Control, 1996, Volume 11, No. 2, pp. 28-30.

[15] K-TEAM, "Khepera User Manual", 1999 www.k-team.com.

[16] VALLEJO, E. BLANES, F. BENET, G. "Herramientas CASD par el diseño de controladores inteligentes en robótica móvil" Editorial: Universidad de la Laguna, ISBN 84-699-8916-2, 2002. Páginas: 135 – 140.

AUTORES

Eric Vallejo R. Licenciado en Matemáticas y Física, Especialista en Computadores y Sistemas Digitales, Doctor en Automática e Informática Industrial (2004). Docente de la Universidad del Norte desde 1990 en las áreas de Circuitos y Sistemas digitales, Diseño Electrónico, Control y Robótica. Director del Grupo de Investigación en Robótica y Sistemas Inteligentes de la Universidad del Norte. evallejo@uninorte.edu.co

Ginés Benet Gilabert. Ingeniero Industrial; Doctor Ingeniero Industrial (1988). Es Profesor Titular de Universidad en el Departamento de Informática de Sistemas y Computadores de la U. Politécnica de Valencia. Imparte Tecnología de Computadores, Técnicas de Inspección y Mantenimiento e Instrumentación y Periféricos, en la Escuela Téc. Sup. De Informática Aplicada y en la Facultad de Informática. Ha dirigido 4 tesis doctorales y varios proyectos de investigación relacionados con la robótica móvil. gbenet@disca.upv.es