Página principal
Presentación
Comités
Comunicaciones
Programa
Patrocinadores
Contacto

Las Jornadas de Automática constituyen un punto de encuentro para la comunidad de Automática en España tanto en el ámbito de la investigación, la docencia o la empresa. Las Jornadas vienen realizándose anualmente desde el año 1977, promovidas por el Comité Español de Automática (CEA), y organizadas por distintas Universidades o Centros de Investigación de nuestro país. Este año 2011 celebrarán su XXXII edición en la Universidad de Sevilla, durante los días 7, 8 y 9 de septiembre en la Escuela Técnica Superior de Ingeniería, siendo ésta la segunda ocasión en la que se desarrollará en Sevilla.

LABORATORIO VIRTUAL DE EXPLORACIÓN DE ENTORNOS MEDIANTE SISTEMAS MULTIROBOT

M. Juliá, L. Payá, D. Valiente, L.M. Jiménez, D. Ubeda y O. Reinoso
Universidad Miguel Hernández de Elche, Dpto. de Ingeniería de Sistemas y Automática,
Avda. Universidad s/n, 03202, Elche (Alicante), España,
{mjulia, lpaya, dvaliente, luis.jimenez, ubeda, o.reinoso}@umh.es

Resumen

En este artículo se presenta una herramienta de software de simulación de robots móbiles. Dicha aplicación permite probar diferentes algoritmos de exploración y creación de mapas basados en marcas. La aplicación permite emplear distintos entornos de simulación y configurar múltiples parámetros de los robots y de los algoritmos de navegación, localización y creación de mapas soportados. La aplicación se encuentra disponible en Internet para ser descargada y utilizada libremente.

Palabras clave: robots móviles, exploración cooperativa, herramientas de simulación

1. INTRODUCCIÓN

En robótica móvil una tarea imprescindible es la de explorar un entorno desconocido creando un modelo o mapa del mismo que pueda ser empleado posteriormente para planificar trayectorias. Son muchos los algoritmos para dedicir las trayectorias que debe seguir el robot o equipo de robots móviles para explorar el entorno [1] [2] [3] [4] [5] [6] [7]. Al mismo tiempo, son muchos los algoritmos que permiten localizar y crear un mapa del entorno de forma simultánea [8] [9], proceso conocido como SLAM (Simultaneous Localization and Mapping).

A la hora de comparar dichos algoritmos es bastante complicado decidir cual se adapta mejor a un escenario concreto debido a la gran variedad de técnicas existentes. Por ello, se ha considerado necesario poner a prueba bajo unas mismas condiciones las técnicas más representativas para poder así evaluar y comparar dichas técnicas.

La aplicación que aquí presentamos incorpora un entorno de simulación de robots móviles, que permite trabajar con distintos escenarios reconfigurables así como variar distintas características de los robots empleados, como la odometría, sensores laser o cámaras de visión estereoscópica. Del mismo modo, los datos capturados, procedentes de los sensores de los robots simulados que se encuentran en el escenario seleccionado en simulador, se pasan a una arquitectura de control de los robots que realiza las tareas de SLAM así como de planificación de las trayectorias de exploración. Dicha arquitectura puede ser configurada permitiendo elegir entre distintos algoritmos de SLAM y de exploración, así como configurar distintos parámetros de los mismos.

2. TRABAJOS RELACIONADOS

Son numerosas las herramientas de simulación de robots móviles existentes. Un ejemplo es el simulador Stage [10]. Dicha herramienta permite simular robots móviles sobre un escenario simulado, obteniendo datos de los sensores del robot que pueden ser utilizados por el usuario en su propio algoritmo de control o de creación de mapas. Este tipo de simulador está centrado en proporcionar un entorno de simulación lo más abierto posible y similar a un robot móvil para poder probar los algoritmos de control sobre el mismo antes de probarlo en la plataforma real.

Sin embargo, comparar algoritmos de planificación de trayectorias para explorar un entorno desconocido resulta complicado. Hay herramientas para simular y comparar algoritmos de planificación de trayectorias como la presentada en [11], sin embargo, dicha herramienta asume un mapa ya conocido del entorno.

En cuanto a la creación de mapas el website *OpenSLAM.org* recoge diferentes algoritmos de SLAM, sin embargo, la gran mayoría de dichos algoritmos obvian el proceso de decidir las trayectorias que deben seguir los robots, trabajando directamente con los datos obtenidos sobre una trayectoria ya fijada.

Otros autores se han centrado en una arquitectura abierta que permita utilizar código sobre distintos robots móviles y simuladores incluyendo múltiples sensores como es el caso de ROS (Robot Operative System) [12]. Pese a la gran versatilidad de este tipo de arquitecturas, pocos algoritmos de exploración están disponibles. Además, para probar los algortimos requieren emplear un robot móvil real, lo cual introduce mucho ruido a la hora de comparar algoritmos o un simulador compatible, como

puede ser el propio Stage antes citado. Además, este tipo de arquitectura es bastante compleja de manejar.

3. DESCRIPCIÓN DE LA HERRAMIENTA

La herramienta que se ha diseñado permite simular y comparar de forma simple múltiples algoritmos de exploración. Está programada en C++ y funciona bajo Windows o Linux. A diferencia de otras arquitecturas y simuladores, la aplicación está pensada para ser fácilmente empleada, por ejemplo, como herramienta educativa, sin requerir ningún esfuerzo para ser usada. En general, está compuesta por un simulador, que permite simular los movimientos y los datos extraidos por los sensores de los robots en distintos escenarios que pueden ser fácilmente modificados. La arquitectura incluye un sistema de SLAM multirobot, que permite utilizar varias implementaciones como, por ejemplo, un EKF [8] o un filtro de partículas [9]. Dicho SLAM está basado en marcas que son adquiridas por medio de sensores de visión estereo. En este caso, a diferencia de otros simuladores, se ha simplificado la adquisición de datos. En lugar de generar una imagen virtual del entorno sobre la que extraer características, se trabaja directamente sobre coordenadas 3D de las marcas que se han configurado en el escenario virtual cargado en el simulador.

La figura 1 muestra la interfaz gráfica de la aplicación. A la izquierda, puede verse el escenario virtual introducido en el simulador determinado por un conjunto de obstáculos y marcas así como el grupo de robots móviles que se ha configurado. De esta forma, se pueden emplear distintos escenarios que son almacenados sobre un fichero de texto que define todas las propiedades del escenario.

La figura 2 muestra la ventana de configuración de las carácterísticas de los robots. En ella se puede configurar los parámetros de odometría, forma del robot, así como las caracteríscas de los sensores de rango y de visión a bordo del robot.

Del mismo modo, se han incluido en la aplicación dos algoritmos de SLAM: un EKF y un filtro de partículas, cuyos parámetros pueden también ser configurados. Además se puede elegir entre 7 algoritmos de planificiación de trayectorias para exploración que serán explicados en la sección 4. Los parámetros de dichos algoritmos también pueden ser ajustados mediante la ventana que se muestra en la figura 3.

En la parte derecha de la aplicación, como se puede observar en la figura 1 se muestra on-line el mapa global que están creando los robots, tanto

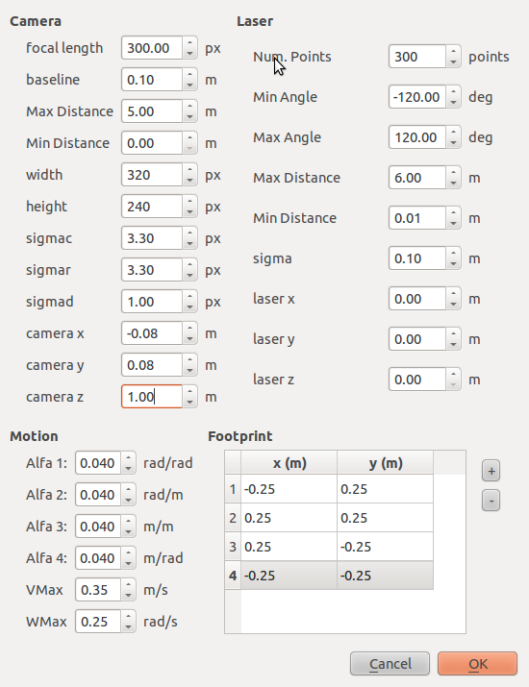

Figura 2: Pantalla de configuración de las características de los robots

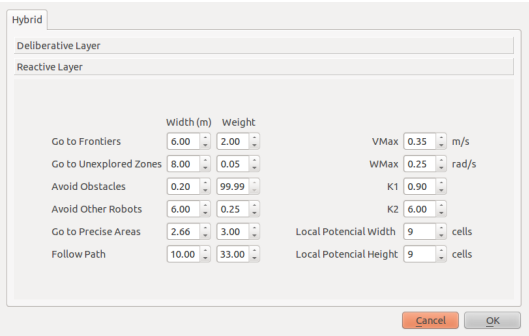

Figura 3: Pantalla de configuración de la estrategia de exploración

la posición de las marcas detectadas mediante el algoritmo de SLAM visual como el mapa de ocupación correspondiente atendiendo a las medidas realizadas por un sensor de rango.

Al final de cada experimento la aplicación devuelve el tiempo requerido por el equipo multi-robot para explorar el entorno así como el error cometido en los mapas creados.

La aplicación junto con un conjunto de escenarios de prueba se encuentra publicada en Internet y puede ser descargada desde la siguiente url: http://arvc.umh.es/mjulia/ExpApp.html

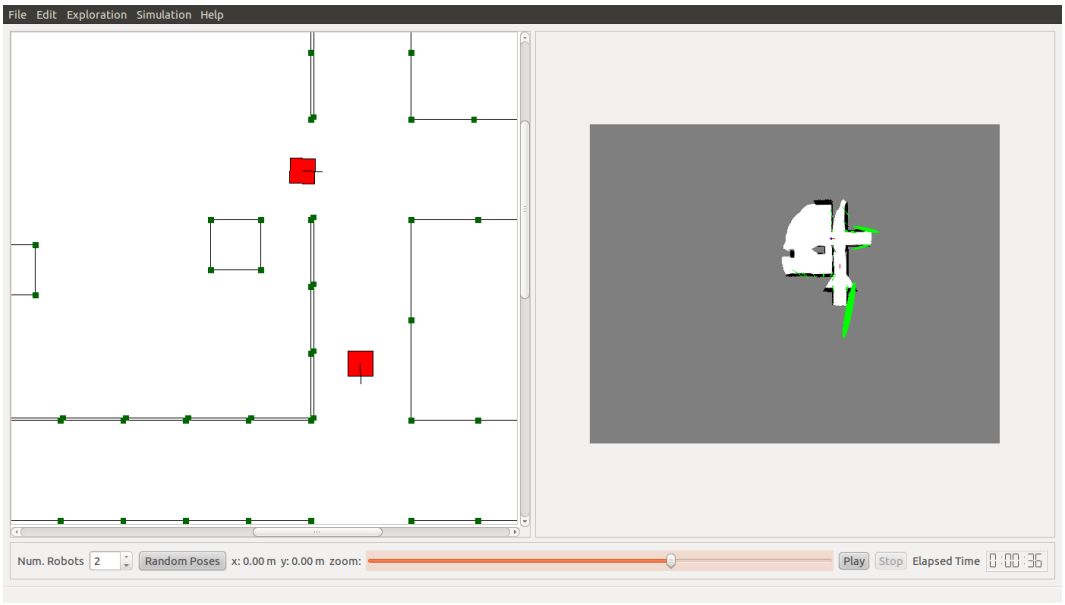

Figura 1: Interfaz gráfica de la aplicación

4. ALGORITMOS IMPLEMENTADOS

Los algoritmos de exploración cooperativa que se han incorporado en la aplicación son los siguientes:

- Frotera más cercana
- Coste-Utilidad
- Reactivo basado en comportamientos
- Coordinado
- Modelo de mercado
- Integrado
- Híbrido coordinado integrado

Los algoritmos que se han incluido en la interfaz han sido seleccionados entre la gran variedad de algortimos de exploración existentes en la literatura de forma que se encuentren presentes algoritmos de distintos tipos. En la tabla 1 se muestran los algoritmos clasificados. En ella se puede ver como se han seleccionado algoritmos que se centran tanto en la coordinación entre los robots para intentar realizar la tarea de la manera más rápida posible así como algoritmos que se centran en la integración de la planificación de trayectorias con el algoritmo de SLAM. Estos últimos tienen en cuenta la incertidumbre que tiene cada robot en su localización para buscar trayectorias que favorezcan o permitan mejorar la localización de modo que se obtengan mapas más precisos.

A continuación se detalla cada uno de los algoritmos.

4.1. Frontera más cercana

Esta técnica está basado en [1] y consiste en seleccionar el camino más corto a la frontera más cercana. De esta forma, la celda t^{NF} seleccionada por esta técnica será:

$$t^{NF} = \arg\min_{a \in F} L(a), \tag{1}$$

donde $L(a)$ representa la longitud del camino más corto para alcanzar dicha celda $a \equiv (a_i, a_j)$, y siendo F el subconjunto de celdas que se están evaluando, en este caso las celdas de frontera. Como se puede ver, este método solo tiene en cuenta el coste de llegar a la celda de frontera. Sin embargo, no considera cómo de útil es cada frontera ni tampoco incluye ningún mecanismo de coordinación. En este sentido, 2 robots podrían elegir la misma frontera si están en posiciones próximas entre sí.

4.2. Coste-Utilidad

Esta técnica introduce un término de ganáncia de información que mide la utilidad de alcanzar una celda dada. Al igual que la técnica anterior, esta técnica no considera ningún mecanismo de coordinación. El algoritmo está inspirado en [2] pero empleando un función coste-utilidad distinta. En este caso, se consideran las celdas de frontera como posibles destinos y el beneficio $B^{CU}(a)$ de alcanzar dicha celda a se evalua del siguiente modo:

$$B^{CU}(a) = U(a) - \lambda_{CU} C(a), \tag{2}$$

donde $U(a)$ es una función de utilidad, $C(a)$ es el

Tabla 1: Clasificación de los mtodos implementados

	No coordinados	Coordinados
No integrados	Frontera ms cercana Coste Utilidad	Coste Utilidad Coordinado Modelo de mercado Reactivo Basado en Comportamientos
Integrados	Coste Utilidad Integrado	Híbrido Coordinado Integrado

coste, y la constante λ_{CU} ajusta la importancia relavia entre ambos factores. La función de utilidad y la de coste son expresiones normalizadas entre 0 y 1 que se calculan de la siguiente forma:

$$U(a) = \frac{\text{Unex}(a, R_s)}{\pi R_s^2} \quad (3)$$

$$C(a) = \frac{L(a)}{\text{máx}_{b \in F}\, L(b)} \quad (4)$$

donde la función $\text{Unex}(a, R_s)$ es el resultado de contar el número de celdas no exploradas en el rango del sensor desde la celda a, siendo R_s el máximo rango del sensor expresando dicho valor en número de celdas.

Por lo tanto, la celda objetivo es elegida como aquella que maximiza la relación coste-utilidad expresada en la ecuación 2:

4.3. Reactivo basado en comportamientos

Basándose en [3], esta técnica combina tres comportamientos reactivos básicos para controlar la exploración. Estos comportamientos son *Ir a fronteras*, *Evitar obstáculos* y *Evitar otros robots*. Considerando estos comportamientos, se calcula un campo de potencial global como la suma ponderada del campo de potencial asociado a cada comportamiento:

$$P(a) = \lambda_f P_f(a) + \lambda_o P_o(a) + \lambda_r P_r(a) \quad (5)$$

siendo el campo de potencial asociado a cada comportamiento modelado como una suma de gausianas:

$$P_f(a) = -\sum_{b \in F} \exp\left(-\frac{d^2(a,b)}{2\sigma_f^2}\right), \quad (6)$$

$$P_o(a) = \sum_{b \in O} \exp\left(-\frac{d^2(a,b)}{2\sigma_o^2}\right), \quad (7)$$

$$P_r(a) = \sum_{b \in R} \exp\left(-\frac{d^2(a,b)}{2\sigma_r^2}\right), \quad (8)$$

donde $d(a, b)$ es la distancia euclídea entre las celdas a y b, F el subconjunto de celdas de frontera, O el subconjunto de celdas ocupadas por obstáculos, R el subconjunto de celdas ocupadas por otros robots, y donde σ_f, σ_o y σ_r son constantes ajustadas experimentalmente que modelan la anchura más apropiada asocidada a cada comportamiento junto con λ_f, λ_o y λ_r que modelan los pesos asociados a cada comportamiento. Cabe destacar que el comportamiento *Evitar otros robots* dispersa a los robots en el entorno con la finalidad de minimizar el tiempo de exploración.

El robot se dirigirá en la dirección opuesta al gradiente del campo de potencial global, esto es $-\nabla P(a)$. En caso de aparecer un mínimo local, se aplica la solución presentada en [3] consistente en planificar una ruta hasta la frontera más cercana. Cuando se alcanza dicha frontera se continúa con el modelo de comportamientos reactivo.

4.4. Coordinado

Esta es la versión coordinada de la técnica de ir a la frontera más cercana y está inspirada en [4]. En ella, la celda de frontera destino se selecciona como función de la distancia al posible destino y de la distancia de la misma a otras celdas de frontera seleccionadas previamente como destinos por otros robots:

$$B^C(a) = -C(a) - \lambda_C \sum_r f(a, r), \quad (9)$$

siendo $C(a)$ el mismo coste de la técnica coste-utilidad, λ_C una constante para ajustar la importancia relativa entre ambos factores y $f(a, r)$ una función que depende de la distancia a los destinos asignados a los otros robots que se evalúa de la siguiente forma:

$$f(a,r) = \begin{cases} d(a, t_r^C)/d_l, & if \quad d(a, t_r^C) < d_l \\ 0, & else \end{cases} \quad (10)$$

donde t_r^C es la celda destino asignada al robot r y d_l es un parámetro de diseño que modela el radio de influencia.

Finalmente, la celda destino se determina como la celda de frontera que maximiza la ecuación 9:

4.5. Modelo de mercado

Esta técnica está inspirada en [5]. El algoritmo emplea la misma función de beneficio del modelo de coste-utilidad. En esta técnica se propone un modelo de economía de mercado donde los robots negocian los destinos que tienen asignados siguiendo las siguientes reglas:

- Cada robot tiene su propia lista de celdas de destino.
- Cada cela destino tiene un beneficio asociado a la misma mediante la ecuación 2.
- Cada robot elige siempre el destino más beneficioso de su lista de celdas de destino.
- Cuando un robot llega a su destino actual busca nuevas fronteras, agrupadas en clusters. Si su lista no está vacía, dicho robot sacará a subasta dichas fronteras, en caso contrario las añadirá directamente a su lista.
- Para cada frontera sacada a subasta todos los robots hacen una oferta en función del beneficio que obtendría cada robot al visitar dicha frontera.
- Cuando un robot gana una frontera en subasta añade dicha frontera a su lista de destinos.
- Si una celda que estaba en la lista de destinos de un robot aparece en subasta otra vez y esta es ganada por otro robot diferente, la celda se quita de la lista del anterior ganador.
- Si una celda que fue añadida a una lista de destinos se comprueba posteriormente que tiene un beneficio excesivamente bajo se elimina de la lista de destinos.

4.6. Integrado

Esta técnica se basa en la propuesta presentada en [6]. En ella se integra el algoritmo de SLAM con la toma de decisión del próximo destino mediante la inclusión de un término de localizabilidad en la función de beneficio:

$$B^I(a) = U(a) - \lambda_{I_1} C(a) + \lambda_{I_2} \mathrm{Loc}(a). \qquad (11)$$

En este caso, la utilidad $U(a)$ y el coste $C(a)$ se evaluan del mismo modo que para la estrategia de coste-utilidad. Sin embargo, al añadir la función $\mathrm{Loc}(a)$ se tiene en cuenta también la localizabilidad del robot en el destino a que se analiza. Las constantes λ_{I_1} y λ_{I_2} ajustan el peso relativo de estos tres factores.

La función de utilidad $Loc(a)$ se evalua a partir de la incertidumbre esperada en la pose que se obtendría con las medidas adquiridas desde la posición destino a analizar tal y como es explicado en [6]. Este valor se normaliza por la incertidumbre actual del robot. Finalmente, la celda de destino se escoge como aquella celda de frontera que maximiza la ecuación 11.

4.7. Híbrido Integrado Coordinado

Esta técnica se basa en [7] y consiste en una arquitectura híbrida reactiva/deliberativa. La capa reactiva está formada por cinco comportamientos básicos, entre ellos están los tres empleados en el modelo reactivo basado en comportamientos de la sección 4.3 más otros dos comportamientos nuevos: *Ir a zonas no exploradas* e *Ir a poses precisas*. El comportamiento *ir a zonas no exploradas* se modela con el siguiente campo de potencial:

$$P_u(a) = -\sum_{b \in N} \exp\left(-\frac{d^2(a,b)}{2\sigma_u^2}\right), \qquad (12)$$

siendo N el subconjunto de celdas no exploradas y σ_u una constante apropiada para regular la anchura del potencial asociado a este comportamiento. Este comportamiento trata de considerar de forma reactiva la utilidad de dirigirse hacia una zona en función de la cantidad de celdas no exploradas que hay en dicha zona. Por otro lado, el comportamiento *Ir a poses precisas* atrae al robot hacia poses donde puede mejorar su localización:

$$P_{pp}(a) = -\sum_{b \in P_r} \exp\left(-\frac{d^2(a,b)}{2\sigma_{pp}^2}\right), \qquad (13)$$

siendo P_r un subconjunto de celdas visitadas por el robot anteriormente cuando el robot presentaba una buena localización y σ_{pp} una constante de anchura apropiada. Este comportamiento se emplea para integrar el SLAM en el algoritmo. La técnica empleada aquí es la de hacer regresar al robot a poses anteriores cuando la incertidumbre en la pose del robot es demasiado alta. En este sentido, se emplea un modelo de histéresis con dos umbrales de localización, el primer umbral delimita si la celda donde se encuentra el robot debe ser marcada como pose precisa, el segundo umbral indica cuando el robot ha alcanzado el máximo grado de incertidumbre deseado y debe regresar para reducir su incertidumbre por debajo del primer umbral.

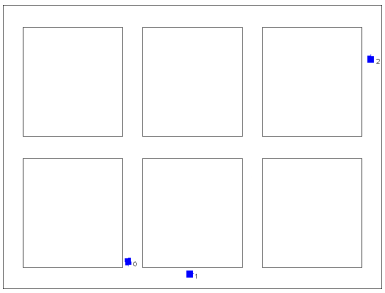

Figura 4: Escenario

Esta técnica además reduce la influecia de la capa reactiva a un área limitada hasta cierto rango en las proximidades del robots, además solo aquellas celdas visibles hasta dicho rango desde la posición del robot son empleadas. De esta forma se evita la aparición de mínimos locales.

La capa deliberativa controla los comportamientos que son activados en la capa reactiva. Así por ejemplo, cuando la localización es buena hará funcionar los comportamientos de *Ir a zonas no exploradas*, *Ir a fronteras*, *Evitar otros robots* y *Evitar obstáculos*. Si la incertidumbre crece en exceso, desactivará estos comportamientos de exploraciñon y activará solamente el de *Evitar obstáculos* junto con el de *ir a poses precisas*. Además esta capa es la encargada de analizar mediante la subdivisión del entorno mediante un árbol tal como es explicado en [7] la cantidad de celdas no exploradas que hay dentro del rango de acción de la capa reactiva. Si el número de celdas es bajo y considerando también las posiciones de los otros robots, la capa deliberativa puede decidir dirigir al robot (mediante un comportamiento de *ir a celda*) hacia una zona límite (o de puerta) entre la región de acción de la capa reactiva actual y otras regiones contiguas.

5. EJEMPLOS ILUSTRATIVOS

Como ejemplo de uso de la aplicación, se ha empleado esta herramienta para determinar que algoritmo de exploración es el más apropiado para explorar y crear un mapa del escenario de la figura 4 empleando un EKK-SLAM y variando el número de robots en el experimiento.

Tras varias simulaciones de cada algoritmo en dicho escenario para un grupo de uno a cinco robots, se han determinado los resultados que se muestran en las figuras 5 y 6. La figura 5 muestra el promedio de tiempo de exploración para cada algoritmo y número de robots. En ella se puede determinar que en general el algoritmo de la frontera más cercana es el más apropiado para explorar rápidamente el escenario de la figura 4, para los parámetros del robot configurados.

La figura 6 muestra el error medio cometido en el mapa de marcas visuales utilizando el filtro de Kalman extendido (EKF) como algoritmo de SLAM. En ella se puede ver como el algoritmo de arquitectura híbrida y el algoritmo integrado son en general los más apropiados para este escenario cuando lo que se desea es tener una alta calidad en el mapa obtenido.

6. CONCLUSIONES Y TRABAJOS FUTUROS

En este artículo se ha presentado una aplicación desarrollada para el estudio y comparación de las distintas técnicas existentes en robótica móvil de planificación de trayectorias para la exploración y mapeado de entornos desconocidos. Esta herramienta está disponible y permite simular dichos algoritmos en múltiples entornos virtuales cambiando los distintos parámetros tanto de los robots como de los propios algoritmos de exploración y SLAM.

Como trabajos futuros se pretende integrar la aplicación en una arquitectura abierta de control de robots, como por ejemplo ROS [12], para poder experimentar fácilmente todos los algoritmos con robots reales así como hacer más sencilla la incorporación de nuevas técnicas al simulador.

Agradecimientos

Este trabajo ha sido apoyado por el Gobierno de España (Ministerio de Ciencia e Innovación) a través del proyecto DPI2010-15308 Exploración Integrada de Entornos mediante Robots Cooperativos para la Creación de Mapas 3D Visuales y Topolgicos que puedan ser usados en Navegación con 6 Grados de Libertad.

Referencias

[1] B. Yamauchi, "Frontier-based exploration using multiple robots," in *Proc. of the Int. Conf. on Autonomous Agents*, 1998.

[2] H. H. González-Baños and J. C. Latombe, "Navigation strategies for exploring indoor environments," *Int. Journal of Robotics Research*, vol. 21, no. 10, 2002.

[3] H. Lau, "Behavioural approach for multi-robot exploration," in *Proc. of the Australasian Conf. on Robotics and Automation (ACRA'03)*, 2003.

[4] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun, "Collaborative multi-robot exploration," in *Proc. of the IEEE Int. Conf. on*

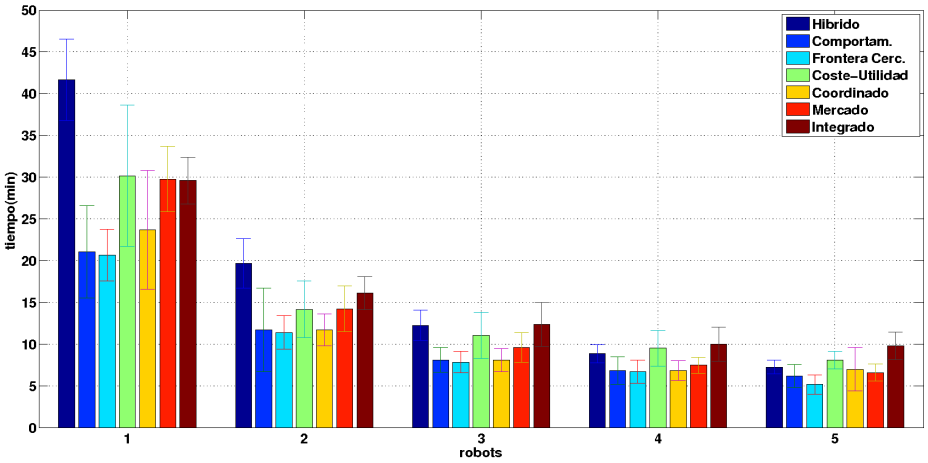

Figura 5: Tiempo de exploración

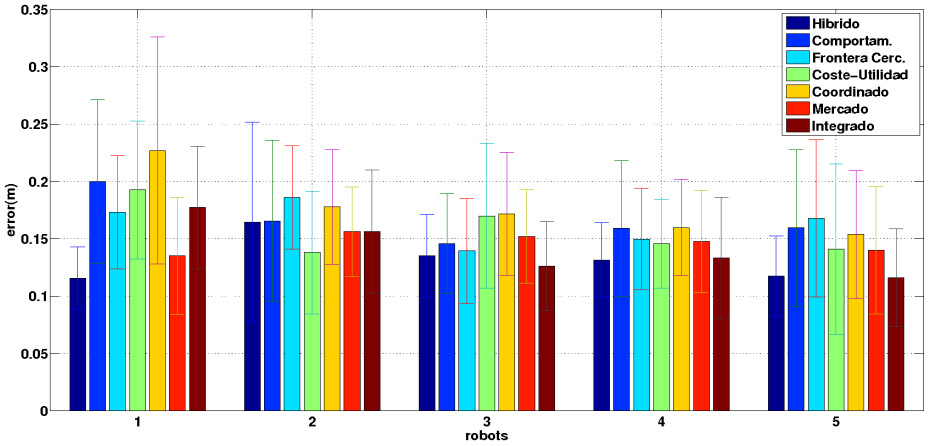

Figura 6: Error en el mapa de marcas visuales

Robotics and Automation (ICRA'00), vol. 1, pp. 476–481, 2000.

[5] R. Zlot, A. Stentz, M. B. Dias, and S. Thayer, "Multi-robot exploration controlled by a market economy," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA'02)*, 2002.

[6] A. Makarenko, S. Williams, F. Bourgoult, and F. Durrant-Whyte, "An experiment in integrated exploration," in *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems (IROS'02)*, 2002.

[7] M. Juliá, O. Reinoso, A. Gil, M. Ballesta, and L. Payá, "A hybrid solution to the multi-robot integrated exploration problem," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 4, pp. 473–486, 2010.

[8] L. Paz, J. Tardós, and J. Neira, "Divide and Conquer: EKF SLAM in $O(n)$," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1107–1120, 2008.

[9] A. Gil, O. Reinoso, M. Ballesta, M. Juliá, and L. Payá, "Multi-robot visual SLAM using a Rao-Blackwellized particle filter," *Robotics and Autonomous Systems*, vol. 58, no. 1, pp. 68–80, 2010.

[10] B. Gerkey, R. Vaughan, and A. Howard, "The Player/Stage Project: Tools for multi-robot and distributed sensor systems," in *Int. Conf. on Advanced Robotics (ICAR 2003)*, 2003.

[11] J. Guzmán, M. Berenguel, F. Rodríguez, and S. Dormido, "An interactive tool for mobile robot motion planning," *Robotics and Autonomous Systems*, vol. 56, no. 5, pp. 396 – 409, 2008.

[12] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.