Desplazamiento de un robot con localización y evasión de obstáculos por visión y ultrasonido

Mariana N. Ibarra Bonilla, Fernando J. Quiñones Novelo, Irma J. García Enríquez,

Juan M. Ramírez Cortés

Coordinación de Electrónica; Instituto Nacional de Astrofísica, Óptica y Electrónica;
Tonantzintla, Puebla, México.
Tel/fax: (222)247-0517, email: mibarra@inaoep.mx

Abstract— **Se describe la implementación de un sistema de navegación autónoma para una arquitectura móvil LEGO NXT capaz de localizar y evadir los obstáculos presentes en un ambiente controlado mediante el uso de un sensor de ultrasonido. El sistema incorpora la búsqueda de la ruta más corta en base al Algoritmo de Dijkstra y la alineación en el desplazamiento de la ruta basada en visión y lógica difusa. El sistema ha sido desarrollado en Matlab v.2007b utilizando una interfaz gráfica para simular la posición y movimiento del móvil, así como las cajas de herramientas de lógica difusa y de adquisición y tratamiento de imágenes. Se presentan resultados obtenidos.**

Keywords-móvil, localizar, evadir, visión, difuso.

I. INTRODUCTION

La necesidad de incrementar la autonomía en las aplicaciones robóticas ha motivado la creación y desarrollo de robots móviles. El propósito es limitar en todo lo posible la intervención humana. Para ello, el robot debe poseer la suficiente inteligencia para reaccionar y tomar las decisiones basándose en observaciones de su entorno, aún cuando éste sea completamente desconocido. La autonomía de un robot móvil se basa principalmente en el sistema de navegación autónoma. En éstos sistemas incluyen diversas tareas como: planificación, percepción y control. La planificación en los robots móviles, consiste generalmente en establecer la misión, la ruta y la evasión de obstáculos. El sistema de percepción tiene un triple objetivo: permitir una navegación segura mediante la detección y localización de los obstáculos y situaciones peligrosas en general, modelar el entorno construyendo un mapa o representación de este último, y conocer la posición precisa del vehículo.

El problema de la localización y mapeo simultáneo, mejor conocido en la literatura como SLAM debido a sus siglas en inglés (Simultaneous Localization and Mapping), pregunta si es posible para un robot móvil autónomo, el cual está colocado en una localización desconocida en un ambiente desconocido, poder construir incrementalmente un mapa consistente del ambiente mientras simultáneamente determina su localización dentro del mapa [1]. El problema del SLAM ha sido formulado y resuelto de diferentes formas. Actualmente, existen aplicaciones SLAM en una gran variedad de dominios interiores y exteriores, incluyendo lugares muy apartados [1], muy peligrosos [2] o simplemente en lugares donde el acceso humano resulta muy costoso. Para adquirir el modelo del ambiente de un robot, un sistema SLAM puede requerir el uso de diversos sensores, tales como, ultrasonido, sistemas laser, sonar, o cámaras de video. A la fecha, las investigaciones y trabajos científicos han utilizado nuevas técnicas de localización y mapeo para la solución del problema SLAM, comenzando con filtros Bayesianos y pasando luego a los Kalman, aplicado generalmente a ambientes reconstruidos con sensores laser [1,3]. Una vez obtenido el mapa del entorno, el objetivo se centra en su representación topológica y en los algoritmos de enrutamiento pertinentes a la aplicación deseada, como por ejemplo el algoritmo de Dijkstra para la determinación de la ruta más cercana [4]. Por otra parte, la aplicación del control difuso en los sistemas de navegación permite tomar decisiones en forma de razonamiento aproximado al ser humano de acuerdo a la información contenida en una base de conocimientos. De igual forma, diversos sistemas basados en localización global visual orientados a la navegación automática han sido reportados con buenos resultados [5]. Las técnicas de procesamiento de imágenes adquiridas a través de una o varias cámaras, utilizadas como elemento de sensado para cerrar el lazo de control [6,7,8], presentan alternativas muy interesantes en sistemas de navegación.

El presente trabajo presenta un sistema de navegación autónoma basado en la localización y evasión de colisiones en un ambiente controlado, mediante la utilización de un sensor de ultrasonido. El sistema incorpora la búsqueda de la ruta más corta en base al algoritmo de Dijkstra. Se propone el uso de la arquitectura móvil Lego NXT por incluir un sensor de ultrasonido y la conexión inalámbrica Bluetooth. El objetivo es implementar un algoritmo que consecuentemente, al detectar una colisión, encuentre la ruta más corta desde su localización a una meta. Durante el recorrido el robot realiza ajustes a su localización en la representación matricial del mapa por medio de una cámara web con operaciones de control difuso. La representación esquemática de este sistema se ilustra en la Figura 1 y el diagrama de flujo en la Figura 2.

ISBN: 978-607-00-1861-9

Fig. 1. Representación esquemática del sistema.

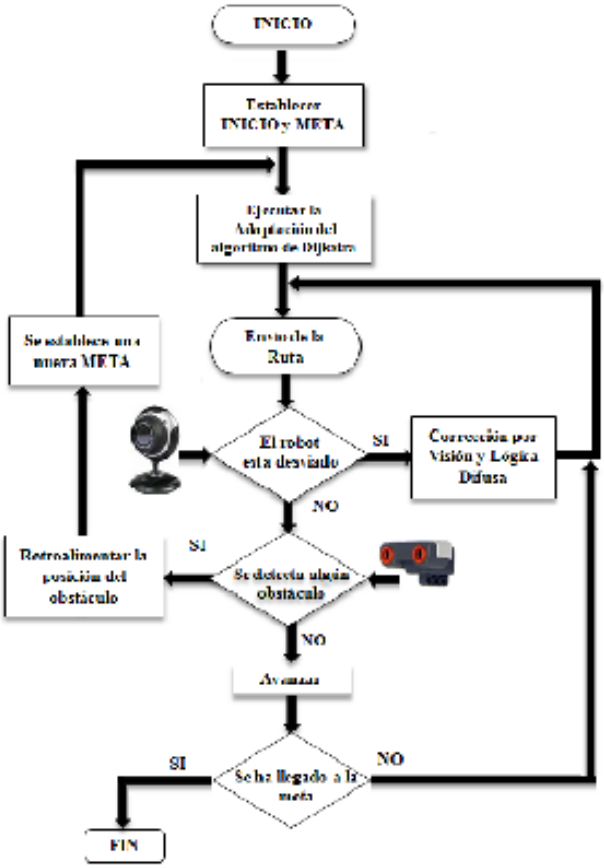


Fig. 2. Diagrama de Flujo del sistema

II. DESARROLLO EXPERIMENTAL

A. Descripción y dimensiones del sistema

El sistema se implementó en un ambiente controlado, el cual consiste de un tapete de 1m2 sobre el cual se ha dibujado una rejilla de cuadros de 20x20 cm. Dentro del ambiente se colocan obstáculos tridimensionales en forma de cubos. El robot es libre de moverse en cualquier dirección siempre y cuando no se salga del área delimitada por el tapete. La arquitectura Lego NXT es un robot tipo triciclo con dos grados de libertad, dos de sus ruedas son motorizadas y permiten hacer giros de hasta 360°. El robot posee un sistema de comunicación inalámbrica Bluetooth, el cual es capaz de conectarse con ordenadores o PC's y recibir órdenes directas de éstos. El sensor de ultrasonido le permite al robot ver y detectar los obstáculos, así como medir la distancia a la que se encuentra. El sensor puede detectar objetos que estén situados de 0 a 255 cm. de distancia, con una precisión de ± 3 cm. La Figura 3 ilustra las dimensiones del sistema.

B. Algoritmo de Enrutamiento

Como requerimiento inicial, el sistema demanda la posición del inicio y meta del robot. Una vez obtenida esta información, se requiere que la computadora genere y entregue en forma inalámbrica al robot móvil un algoritmo de enrutamiento para que sea capaz de recorrer la ruta más corta libre de colisiones. Existen diversos algoritmos de enrutamiento por la ruta más corta, sin embargo, se eligió el Algoritmo de Dijkstra por la facilidad de su aplicación, ya que requiere un bajo nivel de procesamiento [9]. El algoritmo funciona de la siguiente manera: iniciando desde un vértice origen *s*, todos los nodos se etiquetan con su distancia al vértice origen *d*[*v*]. El algoritmo utiliza la técnica de relajación; inicialmente no se conocen las rutas, pero a medida que avanza el algoritmo y se encuentran las demás, las etiquetas pueden cambiar, reflejando mejores rutas. En un inicio, todas las etiquetas son temporales. Una vez que se descubre que una etiqueta representa la ruta más corta posible del origen a ese nodo, se vuelve permanente y no cambia más [9]. El pseudocódigo del algoritmo de Dijkstra se describe en la Tabla I y la Figura 4 ilustra en forma resumida un ejemplo de funcionamiento, las ponderaciones en cada nodo representan las distancias correspondientes.

B. Algoritmo de Enrutamiento
Como requerimiento inicial, el sistema demanda la posición del inicio y meta del robot. Una vez obtenida esta información, se requiere que la computadora genere y entregue en forma inalámbrica al robot móvil un algoritmo de enrutamiento para que sea capaz de recorrer la ruta más corta libre de colisiones. Existen diversos algoritmos de enrutamiento por la ruta más corta, sin embargo, se eligió el Algoritmo de Dijkstra por la facilidad de su aplicación, ya que requiere un bajo nivel de procesamiento [9]. El algoritmo funciona de la siguiente manera: iniciando desde un vértice origen *s*, todos los nodos se etiquetan con su distancia al vértice origen *d*[*v*]. El algoritmo utiliza la técnica de relajación; inicialmente no se conocen las rutas, pero a medida que avanza el algoritmo y se encuentran las demás, las etiquetas pueden cambiar, reflejando mejores rutas. En un inicio, todas las etiquetas son temporales. Una vez que se descubre que una etiqueta representa la ruta más corta posible del origen a ese nodo, se vuelve permanente y no cambia más [9]. El pseudocódigo del algoritmo de Dijkstra se describe en la Tabla I y la Figura 4 ilustra en forma resumida un ejemplo de funcionamiento, las ponderaciones en cada nodo representan las distancias correspondientes.

TABLA I. ALGORITMO DE DIJKSTRA

Entrada: requiere un vértice origen s y una matriz de distancias d donde $d[i, j]$
Salida: Regresa la distancia $d[v]$ desde s a cualquier otro vértice $v \in V$.

```
1:  for cada vértice v ∈ V do
2:      d[v] = ∞
3:  end for
4:  d[s] = 0
5:  S = 0
6:  while existan nodos que no han sido permanentes, V \ S ≠ 0 do
7:      u ← argmin {d[v]}
8:      S ← S ∪ u
9:      for cada arista (u, v) do RELAJACIÓN
10:         if d[v] > d[u] + w(u, v) then
11:             d[v] = d[u] + w(u, v)
12:         endif
13:     end for
14: end while
```

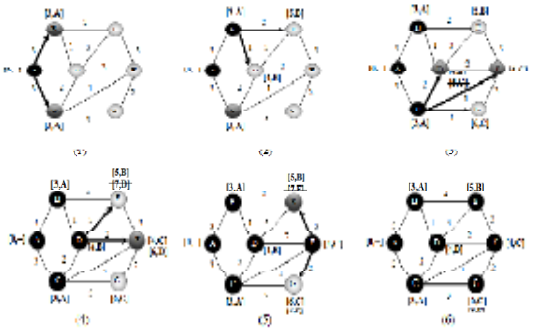
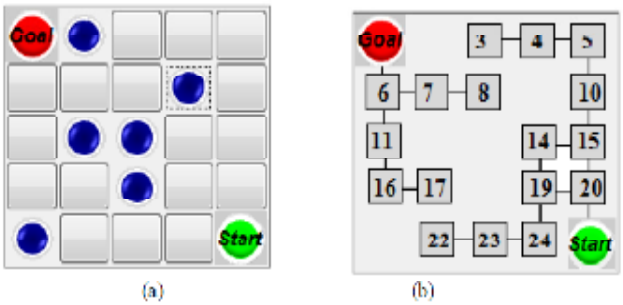


Fig. 4. Cálculo de la ruta más corta del nodo A a cualquier nodo contenido en el grafo mediante el algoritmo de Dijkstra

El ambiente virtual donde se ejecuta el algoritmo de Dijkstra consiste en una matriz de cuadros. Cada uno de éstos corresponde a un nodo. Dicha matriz cambia de dimensiones según la decisión del usuario. Éste último tiene la libertad de elegir la posición del nodo de INICIO y del nodo META. Cabe mencionar, que únicamente hay un nodo INICIO y un nodo META en todo el ambiente, por lo que no hay múltiples caminos. Esta representación gráfica del ambiente se realiza con la finalidad de verificar que los movimientos del robot sean correctos.

Para adaptar el Algoritmo de Dijkstra es necesario asociar la matriz de cuadros con un grafo. Tomando en cuenta las condiciones antes mencionadas, el grafo se genera extrayendo únicamente los nodos válidos eliminando los obstáculos, para posteriormente evaluar numéricamente las alternativas de recorrido, tal como lo establece el algoritmo.



(a) (b)

Fig. 5. (a) Esquema del ambiente gráfico desarrollado en Matlab. (b) Representación del grafo.

En la Figura 5 se puede apreciar el ambiente gráfico simulado y la estructura del grafo.

Como puede observarse en la Figura 6, el algoritmo se ejecuta con rutas ortogonales, cuatro alternativas de movimiento en cada nodo, y omnidireccionales con ocho alternativas por nodo; el robot es capaz de moverse usando cualquiera de las dos posibilidades en función de lo que especifique el usuario.

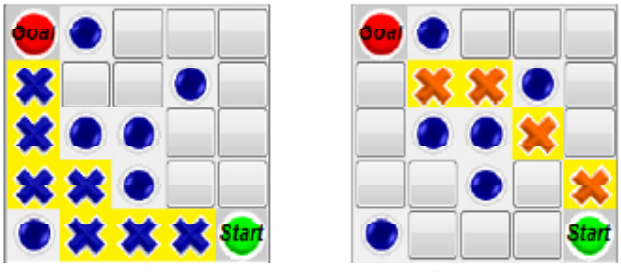
C. Envío de la ruta al robot móvil

Esta etapa consiste en lograr que el robot Lego NXT realice la ruta trazada por el algoritmo de Dijkstra. Para realizar tal actividad es necesario que la ruta sea traducida a instrucciones que el robot pueda interpretar. Para esta traducción se utiliza la caja de herramientas *"Lego Mindstorms NXT para Matlab y Simulink"* [13]. Dicha caja de herramientas contiene todo un set de funciones escritas en Matlab para ordenarle al robot realizar diferentes tareas, tales como: configuración de sensores, encendido y apagado de motores, lectura de la salida de los sensores, etc. mediante el uso de la conexión inalámbrica Bluetooth.

Cada movimiento del robot se asocia con una dirección cardinal. Como referencia se establece la condición inicial en la cual el robot Lego se encuentra mirando hacia la dirección norte. Por lo tanto, hacia la dirección Norte el robot se mueve en línea recta por 20cm. Las direcciones Este, Oeste y Sur involucran giros hacia la derecha y hacia la izquierda por 90°. Los motores están configurados por la misma velocidad pero sus direcciones son completamente opuestas. De la misma manera se ejecutan los movimientos para las direcciones Noreste, Sureste, Suroeste y Noroeste cambiando el giro por 45°.

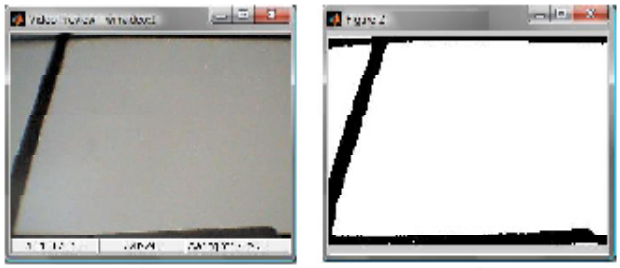
D. Corrección por Control Difuso y Adquisición de Imágenes.

Cuando el robot Lego se desplaza de un nodo hacia otro, existe la posibilidad de que su ruta se desvíe ligeramente; especialmente en los giros de 90°. Por esta razón es necesario incluir la etapa de corrección de ruta en la forma de un control de lazo cerrado con retroalimentación en base a información visual. Si se detecta la acotación del camino quiere decir que el robot ha realizado una desviación indeseable. La lógica difusa y el procesamiento de imágenes han demostrado resultados favorables en aplicaciones de



(a) (b)

Fig. 6. Ruta encontrado con el Algoritmo de Dijkstra. (a) Movimientos ortogonales. (b) Movimientos omnidireccionales.



(a) (b)

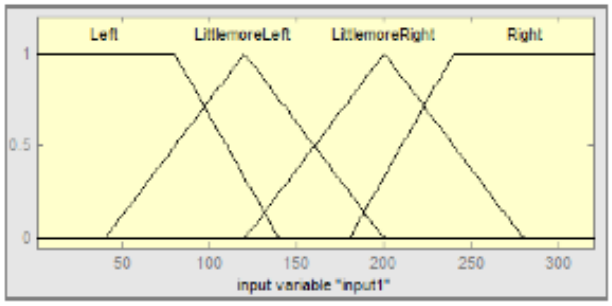
Fig. 7. Captura de la acotación del camino. (a) Imagen captada por la cámara web en tiempo real. (b) Imagen procesada en Matlab.

TABLA II

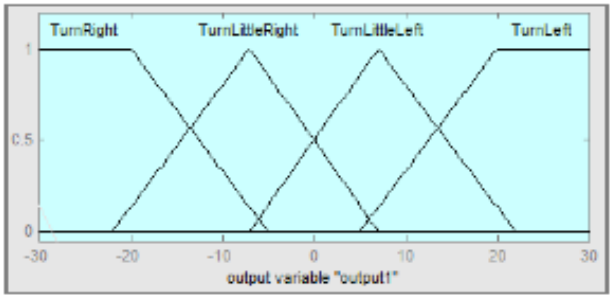
FUNCIONES DE MEMBRESÍA DEL CONTROLADOR DIFUSO

FUNCIONES DE MEMBRESÍA DE ENTRADA		
Variable	**Tipo**	**Partición (Pixeles)**
Left	Trapezoidal	[1 1 80 140]
LittlemoreLeft	Triangular	[40 120 200]
LittlemoreRight	Triangular	[120 200 280]
Right	Trapezoidal	[180 240 320 320]

FUNCIONES DE MEMBRESÍA DE SALIDA		
Variable	**Tipo**	**Partición (Grados)**
Turn Rigth	Trapezoidal	[-30 -30 -20 -5]
Turn LittleRight	Triangular	[-22 -7 7]
Turn LittleLeft	Triangular	[-7 7 22]
Turn Left	Trapezoidal	[5 20 30 30]



(a)



(b)

Fig. 8. Representación gráfica de las funciones de membresía de: (a) Entrada. (b) Salida.

detección y seguimiento de algún objeto [10, 11,12]. En la Figura 7 se ilustra la imagen detectada por la cámara web montada en el robot. El control difuso requerido para éste sistema es a una sola entrada y una sola salida. Éste consiste en tomar como dato de entrada, la información numérica correspondiente a la posición de la línea de acotamiento detectada por la cámara web, para posteriormente entregar como salida un nuevo dato numérico correspondiente al desplazamiento en grados en el movimiento del robot.

Las funciones de membresía introducidas a la entrada y salida del control difuso son: dos de tipo "trapezoidal" y dos de tipo "triangular". Dichas funciones de membresía se describen en la Tabla II y su representación gráfica se muestra en la Figura 8.

Las reglas de inferencia para el control son basadas en las reglas heurísticas IF (antecedente) THEN (consecuente). Para el sistema se implementaron cuatro reglas de inferencia [7]. Dado que se requiere que el desplazamiento del robot sea en dirección contraria a la posición de la línea de acotamiento detectada, las reglas son propuestas de tal manera que se cumpla con lo anterior. Las cuatro reglas de inferencia se describen en la Tabla III.

TABLA III

REGLAS DE INFERENCIA PARA EL CONTROLADOR DIFUSO

Regla	Entrada (IF)	Salida (THEN)
1	Left	Turn Little Right
2	LittlemoreLeft	TurnRight
3	LittlemoreRight	Turn Left
4	Right	TurnLittle Left

III. RESULTADOS

Se construyó físicamente un ambiente de 25 cuadros de longitud 20x20 cm, por lo que el robot se desplazó sobre un área total de 1m2. Se comprobó la funcionalidad del sistema desarrollado a través de varias pruebas con diversas disposiciones de obstáculos. La Figura 9 ilustra un ejemplo de la ruta trazada por el algoritmo de Dijkstra en su ejecución ortogonal, sobre el ambiente virtual. Si algún obstáculo es detectado por el sistema, pero no interfiere en la ruta, tal como es el caso de la Figura 9d, simplemente es plasmado sobre el ambiente virtual. La Tabla IV muestra un resumen de los resultados obtenidos por el sistema en cada una de sus etapas, así como en la incorporación de las mismas.

En todos los experimentos llevados a cabo, el algoritmo de Dijkstra estableció adecuadamente la ruta más corta, misma que fue plasmada sobre el ambiente virtual. La ruta fue completada con éxito en un 83% de las pruebas realizadas por el vehículo, tal como se ilustra en la Figura 10. Cabe mencionar, que dadas las dimensiones del entorno real, aquellas rutas que el robot recorrió en más de 3 minutos se consideraron fracasadas.

El robot logró desplazarse sobre el entorno y reportó la localización de los obstáculos detectados por el sensor de

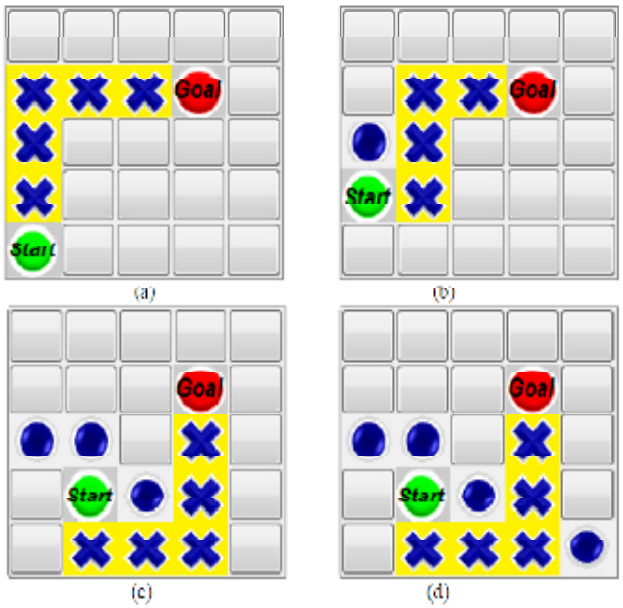


Fig. 9. Secuencia de rutas trazadas por el algoritmo de Dijkstra en la localización de obstáculos.

ultrasonido. Se detectó que los objetos colocados a distancias menores a los 6 cm. no son detectados por el sensor de ultrasonido, por lo que el rango real del sensor es de 6-255 cm. El sistema de control difuso controlado por visión le permitió al robot realizar los ajustes correspondientes, a efecto de mantener alineado al robot a lo largo de su recorrido. El desempeño del sistema es satisfactorio al reportar al menos un 80% de éxito en los experimentos realizados.

IV. CONCLUSIONES

Se presentó un sistema de navegación autónoma con orientación a aplicaciones de localización y mapeo simultáneo, implementado en su totalidad en Matlab, con la ayuda de las cajas de herramientas de lógica difusa y procesa de imágenes. Un aspecto central del sistema es la búsqueda de la ruta más cercana en un mapa de localización tipo rejilla, con la gráfica de nodos representada en forma matricial, a través del algoritmo de Dijkstra en un entorno pequeño con obstáculos. Las funciones desarrolladas en este

Fig. 10. Interfaz gráfica de usuario.

TABLA IV
RESUMEN DE RESULTADOS DEL SISTEMA DE NAVEGACIÓN

Etapa	TP	PE	PF	%E
1. Implementación del Algoritmo de Dijkstra.	50	50	0	100%
2. Recorrido de la ruta más corta con la alineación correspondiente	30	25	5	83%
3. Reporte de la localización de los obstáculos	20	19	2	90%
4. Incorporación de todas las etapas.	25	20	5	80%

TP. Total de pruebas PE. Pruebas Exitosas
PF. Pruebas Fallidas %E. Porcentaje de Éxito

sistema pretenden servir como base en cursos de robótica y control para permitir a los estudiantes transferir los conocimientos teóricos a experimentos prácticos, así como contar con una plataforma para la incorporación de diversas tareas en aplicaciones de navegación autónoma de robots, tales como la identificación de salidas en un espacio cerrado, o el estacionamiento automático de un vehículo. El sistema desarrollado permite también la experimentación con técnicas de localización y mapeo simultáneo, que podrían ejecutarse completamente en la computadora transfiriendo vía Bluetooth las señales de control para el desplazamiento del robot. Como trabajo futuro, se plantea la optimización de las funciones de membresía del controlador difuso mediante el uso de redes neuronales a través del modelo ANFIS incluido en Matlab.

V. AGRADECIMIENTOS

El primer autor agradece al CONACYT por el apoyo otorgado a través de la beca para estudios de Maestría con número 212443.

VI. REFERENCIAS

[1] G. Durrant-Whyte, T. Bailey, “Simultaneous localization and mapping (SLAM): Part I the essential algorithms”, *Robotics and Automation Magazine*, vol 3, No. 2, pp. 99-110, 2006.

[2] Thrun S., Whittaker S., et al . “Autonomous exploration and mapping of abandoned mines”, *IEEE International Conference on Robotics and Automation,* pp. 79- 91, Dec 2004.

[3] S. Holmes, G. Klein, D.W. Murray, “A Square Root Unscented Kalman Filter for visual monoSLAM”, *IEEE International Conference on Robotics and Automation, ICRA 2008,* pp. 3710-3716, Pasadena, Cal., USA, May 19-23, 2008.

[4] O. Booij, B. Terwijn, Z. Zivkovic, B. Krose, “Navigation using an appearance based topological map”, *IEEE International Conference on Robotics and Automation,* pp. 3927-3932, Rome, Italy, 10-14 April 2007.

[5] S. Achar, C.V. Jawahar, "Adaptation and learning for image based navigation", Sixth Indian Conference on Computer Vision, Graphics & Image Processing, ICVGIP '08 pp.103-110, Bhubaneswar, India, 16-19 Dec. 2008.
[6] Juan Manuel Ramírez, Pilar Gómez-Gil, Filiberto López Larios, "A Robot-vision System for Autonomous Vehicle Navigation with Fuzzy-logic Control using Lab-View", *Cuarto Congreso de Electrónica, Robótica y Mecánica Automotriz*, CERMA 2007, pp. 295-300, Cuernavaca, Morelos, 2007.
[7] I.J. García Enríquez, M. N. Ibarra Bonilla, J. M. Ramírez Cortés, P. Gómez Gil, "Seguimiento Autónomo de la Posición de un Objeto por Visión y Control Neuro-difuso en MATLAB", *6to. Congreso Internacional de Investigación en Ingeniería Eléctrica y Electrónica* (CIIIEE), Aguascalientes, México, Nov. 3-7, 2008.
[8] M. Usov "Vision Based Mobile Robot Navigation", M.Sc. Thesis, *University of Twente EEMCS / Electrical Engineering Control Engineering,* June, 2006.
[9] A. Koning, "Shortest path algorithms base on component hierarchies", *Ph.D. Thesis, University of Utrecht, department of Mathematics in cooperation with ORTEC*, 2007.
[10] J.E. Naranjo, et al., "Using Fuzzy Logic in automated Vehicle Control", *IEEE Intelligent Systems,* 2007, pp. 36-45.
[11] D. Stipanicev , M. Cecic. "Eye-Hand coordination based on fuzzy visión" in *Fuzzy System Conference IEEE International,* 1992, pp. 29-35.
[12] S. Jantz, K.L. Doty, "A Wireless enables Mobile Robot for Vision Research ", *Florida Conference on Recent Advances in Robotics,* Florida International University, May 25-26, 2006.
[13] http://www.mathworks.com/programs/mindstorms/