Navegación autónoma de un robot guiado por visión con operaciones básicas de localización y mapeo en un ambiente controlado

Mariana N. Ibarra Bonilla, Juan M. Ramírez Cortés, Alejandro Díaz Méndez, Jorge Martínez Carballido, Rogerio Enríquez-Caldera, Irma J. García Enríquez

1. Coordinación de Electrónica; Instituto Nacional de Astrofísica, Óptica y Electrónica; Tonantzintla, Puebla, México.

***Resumen* — Se presenta la implementación de un sistema de visión por computadora y control difuso orientado a la navegación autónoma con algunas capacidades básicas de localización y mapeo simultáneo, de un robot móvil LEGO NXT. Así mismo, se describe el algoritmo de navegación implementado para el desempeño de diversas tareas, tales como la búsqueda de la ruta más cercana en base al algoritmo de Dijkstra. La arquitectura móvil se encuentra equipada con una cámara web y un sensor de ultrasonido para la percepción del ambiente y de los obstáculos presentes en el mismo. 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 preliminares obtenidos.**

Palabras clave – **visión, control, difuso, localización, mapeo.**

I. Introducción

El proceso de posicionamiento y la generación simultánea de un mapa del entorno es formalmente conocido como SLAM por las siglas en inglés (Simultaneous localization and mapping). Localización y mapeo simultáneo, SLAM, es una técnica usada en robots y vehículos autónomos para construir un mapa dentro de un ambiente desconocido, aún cuando controlado en muchos casos, manteniendo registro de su posición en todo momento. A la fecha se han estudiado aplicaciones SLAM en una gran variedad de dominios, incluyendo interiores, exteriores, aéreos o subacuáticos [1]. Un sistema SLAM puede requerir el uso de diversos sensores, tales como, ultrasonido, sistemas laser, sonar, o cámaras de video. En la actualidad han sido abordados diversos enfoques para la solución del problema de SLAM, especialmente desde el punto de vista probabilístico con un enfoque Bayesiano, con soluciones que involucran técnicas del tipo filtrado Kalman, aplicado generalmente a ambientes reconstruidos con sensores laser [1, 2]. De igual forma, diversos sistemas basados en localización global visual con una o varias cámaras, orientados a la navegación automática han sido reportados con buenos resultados [3].

Una vez obtenido el mapa del entorno, el objetivo se centra en su representación topológica y en los algoritmos de navegación pertinentes a la aplicación deseada, como por ejemplo el algoritmo de Dijkstra para la determinación de la ruta más cercana [4]. Un sistema de navegación autónoma consiste básicamente de un vehículo que no requiere de un operador para navegar y cumplir con sus diversas tareas. Dicho sistema debe contar con alguna heurística de control, la cual permita tomar las decisiones adecuadas para su desplazamiento automático y las demás tareas para las que haya sido concebido.

Fig. 1. Representación esquemática del sistema.

En la actualidad existen un gran número de aplicaciones donde los vehículos autónomos son empleados con éxito: exploración espacial, manejo de explosivos, agricultura, transporte automático en áreas urbanas, sistemas de seguridad, así como exploración en sitios de riesgo para el ser humano [5,6,7]. Por consecuencia, existe una gran variedad de estudios y aplicaciones relacionadas a la robótica móvil [8,9,10,11]. Se han logrado múltiples métodos para lograr que un robot se mueva en un entorno y cumpla con diversos propósitos [9,11,13], sin embargo, la complejidad de la navegación de un robot móvil depende principalmente del tipo de arquitectura de control elegida.

En los últimos años, la aplicación del control difuso en los sistemas de navegación se ha ido incrementando [12,13,14]. La razón es que éste último permite tomar decisiones en forma de razonamiento aproximado al ser humano de acuerdo a la información contenida en una base de conocimientos. Por su parte, 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 [13,14], presentan alternativas muy interesantes en sistemas SLAM y de navegación

En este trabajo se presenta un sistema de navegación autónomo basado en control difuso y visión por computadora, para un escenario pequeño mapeado hacia una rejilla, y su posterior representación matricial en forma

de nodos. El sistema incorpora la búsqueda de la ruta más cercana 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 para logar que dicha arquitectura móvil cumpla con el propósito de encontrar la ruta más corta libre de colisiones, desde un inicio hasta una meta. Durante el recorrido el robot realiza ajustes a su localización en la representación matricial del mapa por medio de operaciones de control difuso. La representación esquemática de este sistema se ilustra en la Figura 1 y sus dimensiones reales se detallan en la Figura 2 y la Tabla I.

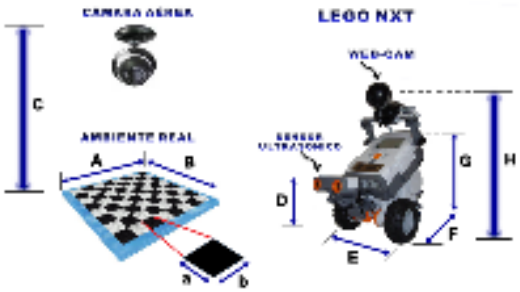


Fig. 2. Descripción de las dimensiones del ambiente real

TABLA I
DIMENSIONES REALES DEL AMBIENTE

Variable	Descripción	Dimensión
A	Lado A del Entorno	1.6 m.
B	Lado B del Entorno	1.6 m.
a	Lado *a* del cuadro	20 cm.
B	Lado *b* del cuadro	20 cm.
C	Altura de la Cámara Aérea	3 m.
D	Altura del Sensor Ultrasónico	7 cm.
E	Ancho del Lego NXT	15 cm.
F	Largo del Lego NXT	18 cm.
G	Altura del Lego NXT	16 cm.
H	Altura al lente de la Cámara	19 cm.

El objetivo final de este proyecto es contar con una plataforma didáctica de desarrollo que permita posteriormente experimentar con diversas técnicas de posicionamiento, generación simultánea y navegación automática, contando para tal efecto con diversos tipos de sensores.

II. DESARROLLO EXPERIMENTAL

A. Implementación del Algoritmo de Enrutamiento

Una vez que el sistema obtiene información visual relacionada con el entorno, localización del robot, obstáculos, inicio y fin del recorrido, 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 diversas razones. En primer lugar, las numerosas e importantes aplicaciones existentes del mismo hacen de él uno de los más populares en la ciencia computacional, aumentando así la importancia de su aprendizaje; y por otra parte, la facilidad de su aplicación, ya que requiere un bajo nivel de procesamiento [15].

En el algoritmo de Dijkstra, cada nodo se etiqueta *(a, b)* con su distancia al nodo de origen *(a)* y con el nodo procedente *(b)*. La etiqueta del nodo inicio es [0,–], que indica que éste nodo no tiene predecesor. 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. La figura 3 ilustra en forma resumida un ejemplo de funcionamiento del algoritmo, donde las ponderaciones en cada nodo representan las distancias correspondientes.

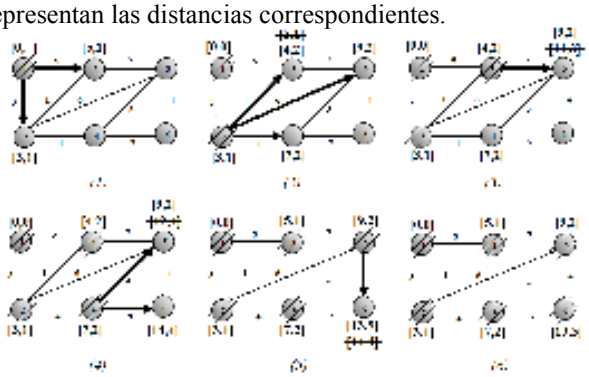
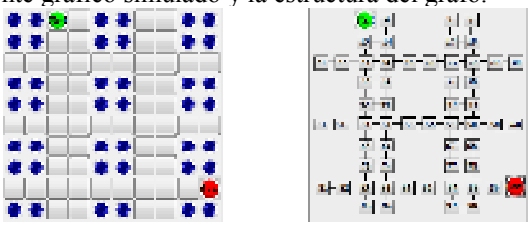


Fig. 3. Cálculo de la ruta más corta del nodo 1 a cualquier nodo contenido en el grafo mediante el algoritmo de Dijkstra.

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, así como también el número y posición de los obstáculos. 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

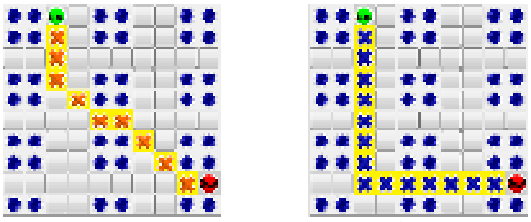
establece el algoritmo. En la Figura 4 se puede apreciar el ambiente gráfico simulado y la estructura del grafo.



(a) (b)

Fig. 4. (a) Esquema del ambiente gráfico desarrollado en Matlab. (b) Representación del grafo.

Como puede observarse en la Figura 5, 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.



(a) (b)

Fig. 5. Ruta encontrada con el Algoritmo de Dijkstra. (a) Movimientos omnidireccionales. (b) Movimientos ortogonales.

B. Captura del ambiente real

El sistema desarrollado le ofrece al usuario dos opciones. En la primera, que puede ser utilizada como una etapa de prueba, el usuario especifica manualmente el número y posición de obstáculos. En la segunda opción, el sistema opera en forma automática, a través de la captura de la información visual del entorno real en el que se desplazará el robot dentro de un entorno controlado. La imagen adquirida mediante una cámara web es importada hacia el ambiente de trabajo de Matlab por medio de la funciones de adquisición de imágenes. Con fines experimentales, se realizó un modelo a escala (1:20 cm) del entorno real, de tal manera que una cámara web con una resolución de 320x240 a una distancia de 25 cm. logre captar toda la imagen, tal como se muestra en la Figura 6a.

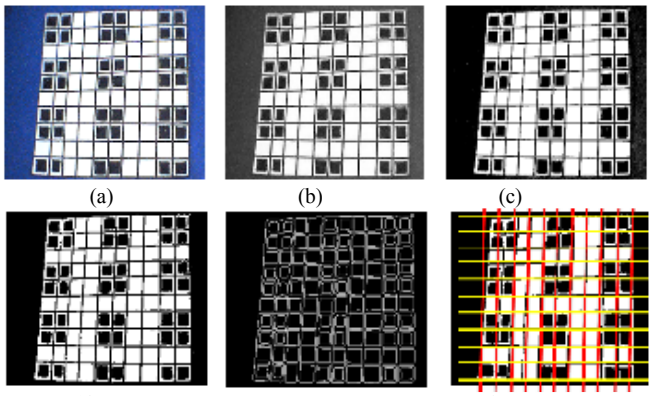
Con el objetivo de eliminar el área que no corresponde al entorno, además de extraer la posición y número de los obstáculos presentes en el mismo, se lleva a cabo una serie de operaciones de procesamiento sobre la señal de video capturada. La Tabla II presenta en pseudocódigo el algoritmo utilizado y la Figura 6 los resultados obtenidos.

C. Envío de la ruta al robot móvil

Esta etapa consiste en lograr que el robot Lego NXT realice la ruta libre de colisiones trazada por el algoritmo de Dijkstra. Para realizar tal actividad es necesario que la ruta sea traducida a instrucciones que el robot pueda interpretar.

TABLA II
PSEUDOCÓDIGO DE LA CAPTURA DEL AMBIENTE REAL

```
1.  Captura de un cuadro de la imagen.
2.  Conversión a escala de grises.
3.  Mejora del contraste de la imagen y eliminación del ruido.
4.  Binarización de la imagen.
5.  Detección de bordes.
6.  Delimitación de fronteras del ambiente dentro de la imagen.
7.  Segmentación del ambiente en una rejilla.
8.  Ciclo i = 1 : Último Nodo
9.      Si  moda(segmento(i)) = 0
10.         Entonces Nodo (i) = Obstáculo
11.     Sino
12.         Nodo (i) = Nodo libre
13.     Fin
14. Fin Ciclo
```



(a) (b) (c)

(d) (e) (f)

Fig. 6. Procesamiento de la imagen escalada. (a) Captura del ambiente real. (b) Escala de grises. (c) Eliminación de ruido. (d) Binarización. (e) Detección de bordes. (f) Segmentación en una rejilla.

Para esta traducción se utiliza la caja de herramientas *"Lego Mindstorms NXT para Matlab y Simulink"* [16]. 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, por lo que se implementaron ocho rutinas básicas descritas en la Tabla III. La velocidad de los motores es la misma en todos los casos, solo cambia el tiempo de ejecución y el sentido de giro; el movimiento en línea recta es por 20 cm. Como referencia se establece la condición

inicial en la cual el robot Lego se encuentra mirando hacia la dirección norte.

TABLA III
RUTINAS DE MOVIMIENTO PARA EL ROBOT LEGO NXT

Hacia la dirección:	Giro	Movimiento de los motores
Norte (North)	0°	Derecho ← Línea recta Izquierdo ← Línea recta
Noreste (North east)	-45°	Derecho ← Avanza hacia atrás Izquierdo ← Avanza hacia adelante
Este (East)	-90°	Derecho ← Avanza hacia atrás Izquierdo ← Avanza hacia adelante
Sureste (South east)	-130°	Derecho ← Avanza hacia atrás Izquierdo ← Avanza hacia adelante
Sur (South)	180°	Derecho ← Avanza hacia adelante Izquierdo ← Avanza hacia atrás
Suroeste (South west)	135°	Derecho ← Avanza hacia adelante Izquierdo ← Avanza hacia atrás
Oeste (West)	90°	Derecho ← Avanza hacia adelante Izquierdo ← Avanza hacia atrás
Noroeste (North west)	45°	Derecho ← Avanza hacia adelante Izquierdo ← Avanza hacia atrás

D. Corrección por Control Difuso y Adquisición de Imágenes.

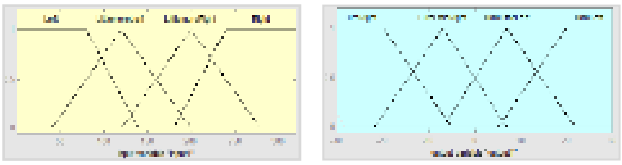
Cuando el robot Lego se desplaza de un nodo hacia otro, existe la posibilidad de que su ruta se desvíe ligeramente; especialmente en los giros de 90°. Por esta razón es necesario incluir la etapa de corrección de ruta en la forma de un control de lazo cerrado con retroalimentación en base a información 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 detección y seguimiento de algún objeto [14]. 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 IV y su representación gráfica se muestra en la Figura 8.

(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 IV
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.

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 [14]. 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 V.

TABLA V
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

Con la finalidad de comprobar el buen funcionamiento del control difuso, se realiza una simulación haciendo uso de la herramienta gráfica del Sistema de Inferencia Difuso de Matlab. Se obtienen los gráficos de la Figura 9, en la

cual se observa el comportamiento de las funciones de membresía.

E. Aplicación del sensor de Ultrasonido

Con la finalidad de construir un algoritmo en el cual el robot sea capaz de explorar el entorno, buscar las colisiones presentes y reportar la localización de los mismos. Se implementó una rutina con el uso del sensor de ultrasonido montado en el robot.

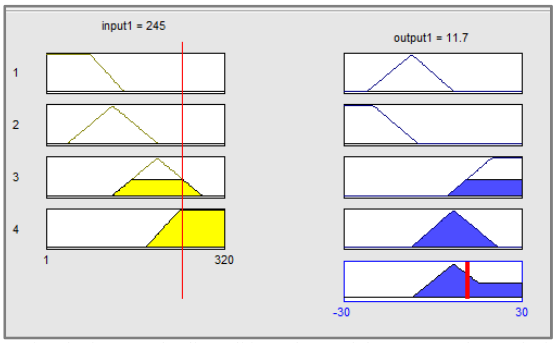


Fig. 9. Simulación en el editor difuso de Matlab. La entrada es el pixel 245 (Right) y la salida es un ángulo de 11.7° (TurnLittleLeft)

En esta última, el usuario establece únicamente la posición de los nodos INICIO y META; posteriormente se ejecuta el algoritmo de Dijkstra como si no existieran obstáculos presentes en el ambiente. Si el sensor de ultrasonido detecta un obstáculo a una distancia menor de 20 cm, se ordena detener el robot. El algoritmo para hallar la ruta más corta vuelve a ser ejecutado, siendo la posición actual de robot el nuevo nodo INICIO. Este proceso se sigue repitiendo hasta que el robot alcance llegar a la meta establecida inicialmente.

F. Interfaz Gráfica de Usuario

El sistema se incorpora en una sola interfaz de usuario realizada bajo el ambiente GUI en Matlab. Con la finalidad de que el sistema sea fácil de manejar es necesario incluir las funciones principales dentro del diseño de la interfaz. La distribución de dichas funciones se muestra en la Figura 10.

Fig. 10. Interfaz gráfica de usuario.

A continuación se enlista una breve descripción de las funciones dentro de la interfaz.

1) Bluetooth Conexion Status: Habilita y deshabilita la comunicación entre el Lego y la computadora (ambas direcciones) usando la conexión inalámbrica Bluetooth.

2) Create Enviroment: Permite establecer la posición del nodo INICIO *(Start)* y el nodo META *(Goal)*; así como también el número y posición de los obstáculos *(Obstacles)*. Esta sección incluye también la función para poder definir el número de filas y columnas existentes en el ambiente virtual. Este se modifica al pulsar la tecla *Modify*. En caso de no establecer el número y posición de los obstáculos, en la sección *Camera on top* se incluye la función de Captura del Ambiente real descrita en la sección B.

3) Dijkstra's Algorithm: En esta sección se ejecuta el algoritmo de Dijkstra. *Execute-A* lo realiza con movimientos ortogonales y *Execute-B* con movimientos omnidireccionales. Cabe mencionar que estas dos funciones pueden ser ejecutadas con la conexión Bluetooth deshabilitada. En esta sección se incluyen las siguientes operaciones: *Graph* permite visualizar el grafo creado por el algoritmo, *Ultrasensor* habilita la función descrita en la sección E y *Reset* borra la ruta trazada por el algoritmo y la ubicación de los nodos INICIO y META, permitiendo al usuario definir una nueva.

4) Fuzzy correction: Activa la cámara montada en el robot Lego NXT, ya que ésta es indispensable para realizar la corrección de la ruta.

5) Send Route: Envía la ruta establecida por el algoritmo hacia el robot Lego. Para llevar a cabo esta función es forzosamente necesario habilitar la conexión Bluetooth y la corrección por control difuso.

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 1m^2 mostrada en la Figura 11a. Se comprobó la funcionalidad del sistema desarrollado a través de varias pruebas con diversas disposiciones de obstáculos. La Tabla VI muestra un resumen de los resultados obtenidos en las diferentes etapas del sistema.

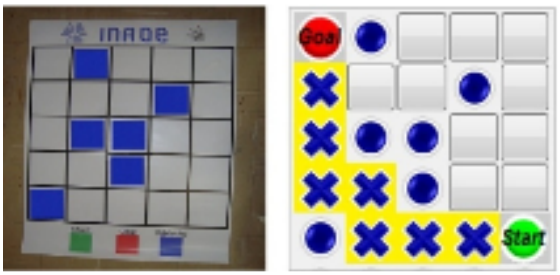
TABLA VI
RESUMEN DE RESULTADOS DEL SISTEMA DE NAVEGACIÓN

Etapa	TP	P E	PF	%E
Implementación del Algoritmo de Dijkstra.	50	50	0	100%
Captura del Ambiente real	30	27	4	90%
Recorrido de la ruta más corta con la alineación correspondiente	30	25	5	83%
Aplicación del Sensor de Ultrasonido	20	16	4	80%

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

En todos los experimentos llevados a cabo, el algoritmo de Dijkstra estableció adecuadamente la ruta más corta, misma que fue completada con éxito en un 83% de las pruebas realizadas por el vehículo. 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. Cabe mencionar, que dadas las dimensiones del entorno real, aquellas rutas que el robot recorrió en más de 3 minutos se consideraron fracasadas. Se realizó la traducción del ambiente real captado por la cámara hacia un ambiente virtual representado en forma de grafo. Las condiciones de iluminación y el color de fondo son primordiales en el desempeño eficiente de esta etapa, ya que las pruebas reportadas como fracasadas se realizaron durante la noche con luz blanca fluorescente y con colores claros (blanco y amarillo) en el fondo. Un ejemplo del desempeño del sistema se ilustra en las Figuras 11 y 12. Por último, el robot logró explorar el entorno y reportó la localización de los obstáculos detectados por el sensor de ultrasonido. De igual manera, el tiempo de duración del recorrido es el criterio para determinar pruebas falladas.

El desempeño del sistema es satisfactorio al reportar al menos un 80% de éxito en los experimentos realizados.



(a) (b)

Fig. 11. (a) Captura del ambiente real por la cámara aérea. (b) Ruta establecida por el algoritmo de Dijkstra en el ambiente virtual.

Fig. 12. Desplazamiento del robot en el ambiente real con obstáculos.

IV. Conclusiones

Se presentó un sistema de navegación autónoma basado en visión por computadora y control difuso, 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 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] 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.

[3] 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.

[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] N.N Flan., K.L Moore., "A small mobile robot for security and inspection", *Control Engineering Practice*, vol. 10, issue 11, pp. 1265- 1270, 2004.

[6] J.N. Wilson, "Guidance of agriculture vehicle: a historical perspective", Computers and Electronics in Agriculture, vol. 25, No. 2, pp. 3-9, 2003.

[7] D. Stipanicev , M. Cecic. "Eye-Hand coordination based on fuzzy visión" in *Fuzzy System Conference IEEE International,* 1992, pp. 29-35.

[8] M. Usov "Vision Based Mobile Robot Navigation", M.Sc. Thesis, *University of Twente EEMCS / Electrical Engineering Control Engineering,* June, 2006.

[9] R. López Padilla, V. Ayala Ramirez , R. Sánchez- Yañez , "Some Experiments on Reactive Obstacle Avoidance for a Mobile Robot ", Universidad de Guanajuato, *18th International Conference on Electronics Communications and Computers, IEEE Computer Society,* pp. 193-196, Puebla, Mexico, 2008.

[10] 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.

[11] A. Benitez, C.J. Moreno, D. Vallejo, "Localization Control LEGO Robot's Navigation", *18th International Conference on Electronics Communications and Computers, IEEE Computer Society,* pp. 187-192, Puebla, Mexico, 2008.

[12] J.E. Naranjo, et al., "Using Fuzzy Logic in automated Vehicle Control", *IEEE Intelligent Systems,* 2007, pp. 36-45.

[13] 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.

[14] 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.

[15] A. Koning, "Shortest path algorithms base on component hierarchies", *Ph.D. Thesis, University of Utrecht, department of Mathematics in cooperation with ORTEC*, 2007.

[16] http://www.mathworks.com/programs/mindstorms/