Memorias del Congreso Nacional de Control Automático
ISSN: 2594-2492

Planeación de trayectorias para evasión de obstáculos en robots móviles, mediante sensores ultrasónicos

Iván Hernández González*. Jaime Álvarez Gallegos*.
Rafael Castro Linares*

**Centro de Investigación y de Estudios Avanzados del IPN, Cd. Mx., México*
(e-mails: {ivan.hernandezg, jalvarez, rcastro}@cinvestav.mx)

Resumen: Se diseña un algoritmo para la evasión de obstáculos, basado en un control conmutado de planeación de senderos, bajo una estructura de trayectorias definidas como una cadena de funciones paramétricas. El control conmutado de planeación de senderos consiste en dos funciones básicas, evasión y seguimiento. La estrategia de control usada para llevar al robot móvil a un punto determinado es denominada control práctico PD. Las trayectorias lineales son construidas definiendo una función lineal temporal auxiliar y una función espacial de splines cúbicos para cada coordenada de posición, (u,v), del robot.

Palabras clave: Robots móviles, sistemas robóticos autónomos, controladores inteligentes.

1. INTRODUCION

Actualmente se investigan algoritmos en robótica móvil que contribuyan al mejoramiento de la calidad de vida humana, y en este sentido, se pretende alcanzar un grado de autonomía para el robot que permita evitar colisiones y la reducción de bloqueos de tránsito. Rong-Jong et al. (2011) dividen en términos generales la planeación de trayectorias en dos categorías: planeación de trayectoria global, en la cual se tiene conocimiento previo del espacio de trabajo; y planeación de trayectoria local, en la cual el ambiente se percibe mediante sensores, por ejemplo, láseres y sistemas de visión artificial a bordo. Existe un problema en la evasión de obstáculos cuando se está en un ambiente parecido al de un laberinto, en el cual se puede entrar en una parte del mismo que no tiene salida. Para resolver esto, se puede memorizar el recorrido, pero se requiere de una gran cantidad de memoria y, por lo tanto, de una gran carga computacional. Lo anterior puede generar grandes retardos en las transmisiones durante los procesos de control. Una alternativa es la implementación de un esquema de control de planeación de trayectoria conmutado. Otras opciones son los métodos propuestos por C. Ye et al. (2003) y J. H. Lilly (2007) quienes proponen algoritmos basados en lógica difusa para resolver el problema de la evasión de obstáculos. Por otro lado, Chengchen et al. (2017) proponen un algoritmo tomando en cuenta un ambiente dinámico. Mientras que Jang-Ho Cho et al. (2018) presentan una versión basada en los algoritmos de campos potenciales, que denominan Campos potenciales Gaussianos, como un método de evasión de obstáculos en tiempo real.

En este trabajo se propone un esquema de planeación de trayectorias lineales con continuidad en la velocidad de un robot móvil, para lograr la evasión de obstáculos, sin conocimiento previo del entorno. Esto se logra a través de un algoritmo de control conmutado simple, que no requiere de una función de seguimiento de pared como la propuesta por W. Tsui, et al. (2008). Para ello se diseña una función de evasión modulada en amplitud por la distancia al obstáculo, además de una función intermedia que evita los cambios bruscos entre el cambio de evasión y seguimiento.

2. MODELO DEL ROBOT

2.1 Robot diferencial como uniciclo

La cinemática del robot móvil tipo uniciclo descrita por J.Velagic et al. (2006) está definida como

$$\dot{u} = \vartheta \cos\theta, \quad \dot{v} = \vartheta sen\theta, \quad \dot{\theta} = \omega, \tag{1}$$

donde ϑ es la velocidad traslacional del robot, perpendicular al eje formado por la unión entre las dos ruedas, y ω es la velocidad rotacional del robot. P(u,v) es la posición del robot, y θ la orientación respecto al marco de referencia UV, (Fig. 1). Las señales ϑ y ω están relacionadas con las velocidades tangenciales de cada rueda v_r (rueda derecha) y v_l (rueda izquierda) por

$$\vartheta = \frac{v_r + v_l}{2}, \quad \omega = \frac{v_r + v_l}{2b}, \tag{2}$$

donde 2b es la distancia entre las ruedas.

En el diseño de un esquema de control para este modelo del robot hay que tener en cuenta la restricción no holonómica de no deslizamiento, $\dot{v}cos\theta\text{-}\dot{u}sen\theta = 0$.

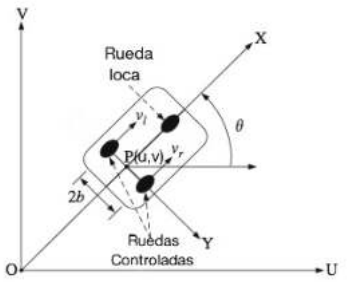

Fig. 1. Sistemas de referencia del robot (marco de referencia UV)

2.2 Modelo discreto

De acuerdo a Velasco-Villa et al. (2007), bajo las siguientes suposiciones se puede obtener un modelo discreto en el tiempo del robot diferencial:

i) Se considera un tiempo de muestreo T>0.

ii) Se define un intervalo de tiempo t_k como

$$t_k = t\epsilon[kT, kT+T], \text{ donde } k\epsilon Z^+ \cup \{0\}.$$

iii) Las señales de entrada ϑ y ω son constantes en los intervalos t_k. Esto es, $\vartheta(t) = \vartheta(kT)$ y $\omega(t) = \omega(kT) \ \ \forall t \in [kT, kT+T]$.

El modelo del robot diferencial en tiempo discreto está entonces dado por

$$u(kT+T) = u(kT) + T\vartheta(kT)sinc\left(\frac{T}{2}\omega(kT)\right)\cos\left(\theta(kT)+\frac{T\omega}{2}\right),$$

$$v(kT+T) = v(kT) + T\vartheta(kT)sinc\left(\frac{T}{2}\omega(kT)\right)\sin\left(\theta(kT)+\frac{T\omega}{2}\right),$$

$$\theta(kT+T) = \theta(kT) + T\omega(kT), \tag{3}$$

donde

$$sinc\left(\frac{T}{2}\omega(kT)\right) = \frac{\sin\left(\frac{T}{2}\omega(kT)\right)}{\frac{T}{2}\omega(kT)} \tag{4}$$

Nótese que cuando $T\omega = 0$, $sinc(T\omega) = 1$, pero cuando $T\omega = n\pi$, para $n \in N$, $sinc(T\omega) = 0$. Entonces la relación entre el periodo de muestreo y la velocidad angular del robot quedan limitadas por esta singularidad. En general se observa que esto ocurrirá para velocidades angulares muy grandes, ya que los tiempos de muestreo son relativamente pequeños.

3. SISTEMA DE CONTROL

3.1 Control Conmutado

El algoritmo se diseña para el Robot Pioneer P3DX, que es un robot diferencial provisto de 8 SONAR frontales, distribuidos sobre 180 grados. Las acciones básicas del algoritmo son evasión y seguimiento (Fig. 2). Estas acciones consisten en definir un punto a alcanzar; este objetivo se logra al implementar el control descrito en la siguiente sección (control práctico). La acción evasiva consiste en identificar el sensor que mide una menor distancia al obstáculo; esto significa identificar el punto de mayor cercanía con el obstáculo detectado por los sensores. El punto a alcanzar para evadir el obstáculo se calcula de manera proporcional a la cercanía con él, de esta forma, al acercarse más, la evasión será con mayor repulsión y, cuando sólo se esté rozando el obstáculo, el robot prácticamente seguirá en forma casi paralela al obstáculo (semejante al resultado obtenido por los algoritmos seguidores de pared). El robot se encontrará entonces en modo evasión, pero los efectos repulsivos serán mínimos, es decir, la fuerza con la cual se alejará del obstáculo será pequeña. En el modo evasión se calcula un punto a alcanzar con el cual se logra evadir al obstáculo.

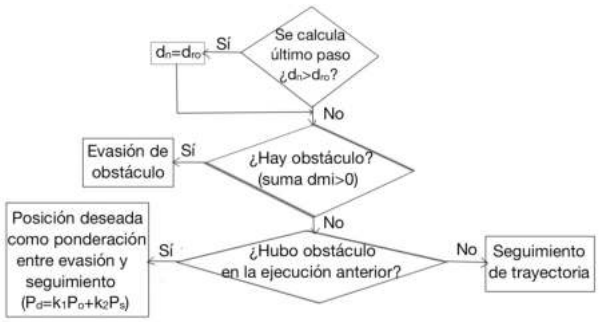

Fig. 2. Algoritmo de control conmutado

En el algoritmo de la Fig. 2 d_n es la longitud del paso de avance, definido por el usuario, d_{ro} es la distancia del robot al objetivo, $d_{mi} = a\text{-}d_{si}$, siendo d_{si} la distancia medida por el sensor, y a el alcance del sensor.

La posición a alcanzar para evadir el obstáculo se define como

$$u_d = u + d_n\cos\theta_d, \quad v_d = v + d_n\sin\theta_d, \tag{5}$$

y el ángulo para el punto deseado como

$$\theta_d = \theta + \theta_k, \tag{6}$$

siendo θ el ángulo de orientación del robot y θ_k el ángulo de evasión dado por

$$\theta_k = \frac{dmi * \theta_{si}}{a}, \tag{7}$$

donde θ_{si} es el ángulo máximo deseado en evasión. Esta modulación del ángulo de evasión, a través de la distancia al obstáculo, permite una repulsión controlada. Mientras que en modo seguimiento se calcula la siguiente posición sin obstáculos como

$$u_d = u + d_n \cos\theta_0, \quad v_d = v + d_n sen\theta_0, \tag{8}$$

donde θ_0 es el ángulo del objetivo medido respecto al eje inercial horizontal. Cuando no se encuentra ningún obstáculo, pero se sabe que hubo obstáculo previamente, se calcula el punto a alcanzar, Pd, como la suma vectorial ponderada del vector de evasión, Po, (en la dirección de la evasión previa) y el vector de seguimiento de trayectoria, Ps, a partir de la posición del robot, Pr, (Fig. 3). De esta forma, el avance del robot es más suave, puesto que no se da un cambio brusco entre evasión y seguimiento, lo que, junto con la evasión modulada por la distancia al obstáculo, permiten evadir obstáculos no convexos pequeños. Para todos los ángulos deseados, dado que se implementa su cálculo con atan2, en los primeros dos cuadrantes se tiene un ángulo positivo, mientras que los otros dos son negativos. Esto genera conflictos en las rotaciones ya que al pasar de un ángulo positivo a negativo o viceversa el robot tiene giros de más de π /2 sin ser necesarios, produciendo un comportamiento no deseado por la excesiva rotación. El problema se resuelve implementando un algoritmo que decide en qué sentido rotar de tal forma que el giro sea mínimo en los casos mencionados, esto se logra comparando las distancias entre el sentido horario y antihorario, eligiendo la menor. Esto se muestra en el siguiente algoritmo:

1. Se calcula θ_d
2. Se calculan dos variables auxiliares $\theta_{aux} = \theta_d + 2\pi$ y $\theta_{aux1} = \theta_d$-2π
3. Si θ-$\theta_d > |\theta$-$\theta_{aux}|$
4. Entonces $\theta_d = \theta_{aux}$
5. Si θ-$\theta_d > |\theta$-$\theta_{aux1}|$
6. Entonces $\theta_d = \theta_{aux1}$

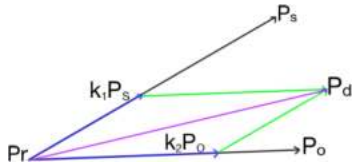

Fig. 3. Suma vectorial ponderada para cálculo de posición deseada tras haber encontrado un obstáculo previamente. $P_d = k_1 P_S + k_2 P_o$. k_1 y k_2 son constantes de proporcionalidad.

3.2 Control Práctico

De acuerdo a Arteaga E. (2016), se definen controles virtuales

$$u(k+kT) = v_1, \quad v(k+kT) = v_2, \quad \theta(k+kT) = v_3. \tag{9}$$

El sistema discreto obtenido en la sección 2.2 se reescribe entonces como

$$\begin{aligned} v_1 &= u(kT) + \frac{\vartheta(kT)}{\omega(\mathrm{kT})}\Big[sen(\theta(kT) + T\omega(kT)) - sen(\theta(kT))\Big], \\ v_2 &= v(kT) - \frac{\vartheta(kT)}{\omega(kT)}\Big[cos(\theta(kT) + T\omega(kT)) - cos(\theta(kT))\Big], \\ v_3 &= \theta(kT) + T\omega(kT). \end{aligned} \tag{10}$$

Simplificando la notación, se obtiene

$$\begin{aligned} \vartheta &= \frac{(v_1 - u)cos(\theta + T\omega) + (v_2 - v)sen(\theta + T\omega)}{Tsinc(T\omega)}, \\ \omega &= \frac{v_3 - \theta}{T}. \end{aligned} \tag{11}$$

Haciendo

$$\begin{aligned} v_1 &= u_d(k+kT) + \kappa_1(u - u_d), \\ v_2 &= v_d(k+kT) + \kappa_2(v - v_d), \\ v_3 &= \theta_d(k+kT) + \kappa_3(\theta - \theta_d), \end{aligned} \tag{12}$$

donde u_d, v_d, θ_d son señales de referencia deseadas que, en este trabajo, son constantes, se tiene que

$$\begin{aligned} v_1 &= u_d + \kappa_1(u - u_d), \\ v_2 &= v_d + \kappa_2(v - v_d), \\ v_3 &= \theta_d + \kappa_3(\theta - \theta_d), \end{aligned} \tag{13}$$

donde κ_1, κ_2 y κ_3 son las ganancias del controlador.

4. PLANEACIÓN DE TRAYECTORIA

4.1 Trayectoria en línea recta por parametrización respecto al tiempo

Una implementación directa del control práctico al modelo del robot genera desplazamientos sobre trayectorias curvas, con picos entre cada paso, lo que produce cambios bruscos en la velocidad traslacional (Fig. 4). Con el fin de evitar este comportamiento, se busca que la trayectoria a seguir sea una línea recta. Entonces se realiza la interpolación entre dos puntos mediante funciones paramétricas, quedando descritas como

$$u = m_u(t - t_0) + u_0, \quad v = m_v(t - t_0) + v_0, \tag{14}$$

donde

$$m_u = \frac{u_1 - u_0}{t_1 - t_0}, \quad m_v = \frac{v_1 - v_0}{t_1 - t_0}, \tag{15}$$

t_0 y t_1 son el tiempo inicial y el tiempo final que definen el tiempo en que se desea alcanzar el objetivo de punto a punto, es decir en un paso.

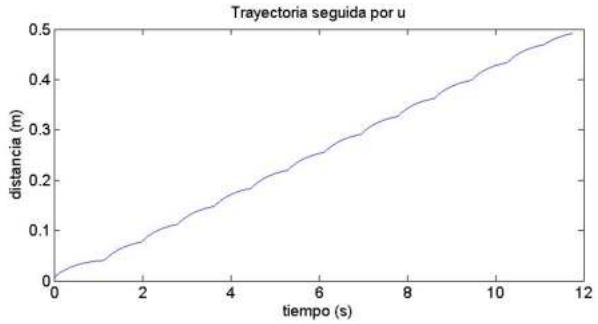

Fig. 4. Trayectoria seguida por u (u,v son las coordenadas del robot)

Con esto, la trayectoria seguida por el robot es una línea recta, y no hay cambios bruscos entre cada paso, prácticamente es el comportamiento de una función suave; dado que la función corresponde a una línea recta, su derivada, que corresponde con su velocidad, es una función constante. Al terminar de ejecutarse el algoritmo, la velocidad no desciende lentamente, sino que sólo se hace cero instantáneamente y se mantiene ahí, esto significa que el robot llegará con esa velocidad y al llegar al objetivo se apagará para que el robot se quede ahí, pero esto sólo es cierto en un ambiente simulado, ya que en un ambiente real, llegar con una velocidad distinta de cero y apagar el robot hará que la inercia del mismo ocasione que el robot no permanezca en el punto objetivo, sino que avance hasta finalmente vencer la inercia y detenerse.

4.2 Trayectoria en línea recta por interpolación mediante splines cúbicos

Un comportamiento en el que el robot no se detenga en el objetivo no es deseado. Con el fin de obtener una trayectoria donde se tenga un comportamiento seudosuave en la velocidad lineal, esto es, donde no existan cambios bruscos distintos de los propios de la discretización, es necesario que exista una continuidad en cada paso, lo que implica que la velocidad debe ser igual entre el término de un paso y el inicio del siguiente.

Por otro lado, se busca que la trayectoria seguida en cada paso sea una línea recta. Al implementar una trayectoria de esta naturaleza, se tiene que la velocidad será constante e igual a la pendiente de la recta, pero esto produciría cambios bruscos no deseados entre pasos puesto que se terminará con la velocidad de la pendiente de la trayectoria del paso anterior y se iniciará con una velocidad distinta para seguir el siguiente tramo de trayectoria recta, más aún, al llegar al fin de la trayectoria en cada paso, la velocidad no será cero, lo que significa que se llegará al destino deseado pero no se permanecerá ahí por cuestiones inerciales. Para resolver este problema se diseña una trayectoria que satisfaga todos los requerimientos mencionados.

La trayectoria debe ser una línea recta, se debe iniciar con velocidad cero y terminar con velocidad cero. Esto no se puede lograr con una parametrización sencilla de una línea recta como función de dos puntos, a saber, el de la posición en el tiempo t_n y el de la posición en el instante t_{n+1}. Para resolver el problema se propone una parametrización en cadena, de tal manera que exista una parametrización espacio temporal lineal y una parametrización intermedia espacial cúbica como se describe a continuación:

Primero se parametrizan las posiciones auxiliares x y y en línea recta en función del tiempo, es decir

$$x(t) = m_u(t - t_0) + u_0, \quad y(t) = m_v(t - t_0) + v_0, \qquad (16)$$

donde

$$m_u = \frac{u_1 - u_0}{t_1 - t_0}, \quad m_v = \frac{v_1 - v_0}{t_1 - t_0}, \qquad (17)$$

y la parametrización intermedia se define como una función que mapea los puntos parametrizados x y y a los puntos espaciales u y v mediante splines cúbicos que satisfacen lo siguiente:

1. Las funciones $y_1(x(t))$ y $y_2(y(t))$, que describen las trayectorias de u y v, respectivamente, son suaves.
2. Tienen derivadas $\dot{y}_1(x_n) = 0$ y $\dot{y}_2(x_n) = 0$, es decir las velocidades son 0 al iniciar la trayectoria en cada paso.
3. Tienen derivadas $\dot{y}_1(x_{n+1}) = 0$ y $\dot{y}_2(x_{n+1}) = 0$, es decir las velocidades son 0 al terminar la trayectoria en cada paso
4. Las funciones en t_n son iguales al punto inicial en cada paso
5. Las funciones en t_{n+1} son iguales al punto final en cada paso

Lo anterior produce una función continua a lo largo de toda la trayectoria, además de una velocidad continua, ya que la velocidad es cero entre cada paso. Para encontrar la función $y_1(x) = ax^3 + bx^2 + cx + d$ que satisface las condiciones anteriores se resuelve el siguiente sistema:

$$\begin{aligned} y_1(x_n) &= ax_n^3 + bx_n^2 + cx_n + d = y_n, \\ y_1(x_{n+1}) &= ax_{n+1}^3 + bx_{n+1}^2 + cx_{n+1} + d = y_{n+1}, \\ \dot{y}_1(x_n) &= 3ax_n^2 + 2bx_n + c = 0, \\ \dot{y}_1(x_{n+1}) &= 3ax_{n+1}^2 + 2bx_{n+1} + c = 0, \end{aligned} \qquad (18)$$

quedando como solución

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} x_n^3 & x_n^2 & x_n & 1 \\ x_{n+1}^3 & x_{n+1}^2 & x_{n+1} & 1 \\ 3x_n^2 & 2x_n & 1 & 0 \\ 3x_{n+1}^2 & 2x_{n+1} & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} y_n \\ y_{n+1} \\ 0 \\ 0 \end{bmatrix} \quad (19)$$

Con estas trayectorias se asegura que la velocidad en los extremos de cada paso será cero, ya que las trayectorias espaciales tienen pendiente cero, como se definen en los requisitos 1 y 2 para la construcción de dichas trayectorias, y

$$\frac{\partial y_1(x(t))}{\partial t} = \frac{\partial y_1}{\partial x}\frac{dx}{dt}, \quad (20)$$

$$\frac{\partial y_1(x(t))}{\partial t}\Big\|_{x=x_n} = 0 \frac{dx}{dt} = 0 \quad (21)$$

Es claro que ocurre lo mismo en $t = T + kT$, donde $x = x_{n+1}$. De forma análoga se resuelve para $y_2(y)$.

Implementando estas trayectorias se tiene el resultado para la coordenada u como una sucesión de funciones cúbicas enlazadas, manteniendo continuidad en cada enlace (Fig. 5). Se observa además que la trayectoria resultante no tiene picos, por lo que la velocidad, dada por la pendiente, será continua. Más aún la trayectoria seguida por el robot es prácticamente una línea recta, como se deseaba (Fig. 6). Existe evidentemente un transitorio y una curvatura en la parte final debido a que el último avance se calcula como la distancia restante al objetivo, mientras que los pasos previos son fijos.

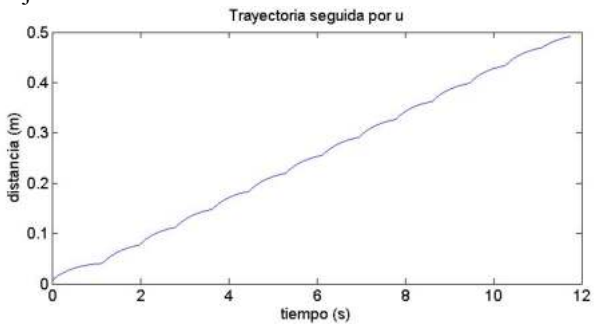

Fig. 5. Trayectoria seguida por u, utilizando splines cúbicos

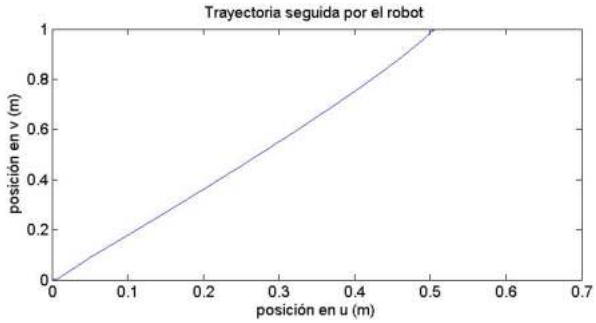

Fig. 6. Trayectoria seguida por el robot

6. SIMULACIONES Y RESULTADOS EXPERIMENTALES

Las pruebas se realizaron en tres escenarios:

El primero con la implementación del modelo cinemático del Robot en Matlab. El programa es completamente implementado en Matlab, no se utilizan funciones de ARIA, ya que el robot aquí está solo definido por su modelo cinemático, la estructura (y dimensiones) del robot, así como los sensores se implementan de igual forma como una idealización en Matlab. ARIA (Por sus siglas en inglés) es una Interfaz para Aplicaciones de Robots Avanzados de MobileRobots constituida como funciones en C++.

El segundo escenario consiste en el uso de MobileSim, que como se describe en la página web de MobileRobots, es un software de simulación para las plataformas MobileRobots\ActivMedia cuya finalidad es la depuración y experimentación con ARIA. MobileSim utiliza la información de obstáculos de un mapa de MobileRobots para simular paredes y otros obstáculos en el ambiente. Sugiere utilizar Mapper3 o Mapper3Basic para crear mapas, no obstante, se realizaron escribiendo los programas directamente desde un block de notas tomando como base uno de los mapas de ejemplo que se incluyen en el paquete de instalación de ARIA.

El último escenario, es el escenario experimental, se basa en la interfaz entre Matlab y el Robot Pioneer P3DX mediante ARIA, el material con el que se construyeron los obstáculos fue cartón corrugado. El programa ejecutado es idéntico a la simulación salvo en los parámetros de repulsión, estos también son distintos para el caso de la implementación por el modelo cinemático. De igual manera, la ganancia en la simulación con el modelo cinemático es distinta para el control del ángulo de orientación θ del robot. La adquisición de los sonares se hace mediante una multiplexación por división de tiempo a 25 Hz como se describe en el manual del Pioneer P3DX.

Al implementar cada uno de los escenarios se observa que en general el comportamiento es el mismo, a continuación, se detallan los resultados en cada caso. El entorno de experimentación tiene las mismas dimensiones y se trata de un conjunto de 3 obstáculos convexos. Para los tres escenarios el objetivo es ir del punto (0,0) al punto (3.5,0). Como puede observarse (Fig. 7) el robot llega al objetivo logrando evadir los obstáculos satisfactoriamente en el entorno simulado completamente en Matlab, el tiempo en que logra llegar al objetivo es de 32.4 s. En la ejecución del algoritmo sobre MobileSim se observa nuevamente que el robot alcanza el objetivo satisfactoriamente (Fig. 8), pero ahora en un tiempo mayor, a saber, 45.5 s.

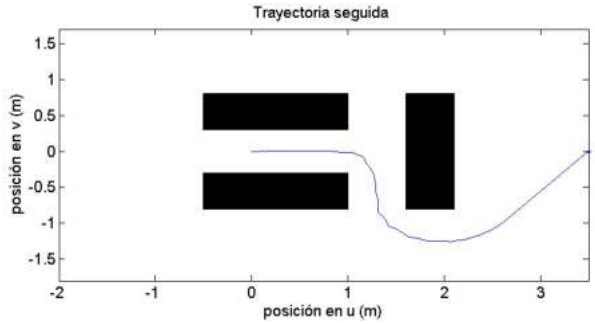

Fig.7. Trayectoria seguida en el entorno simulado en Matlab

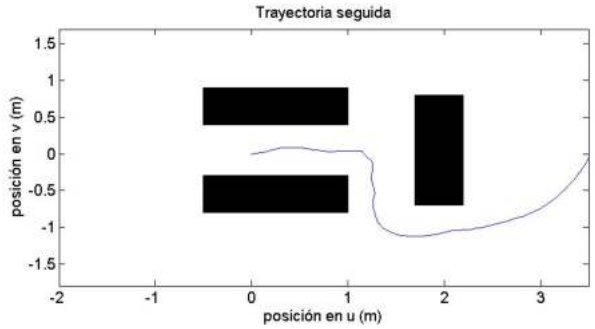

Fig. 8. Trayectoria seguida con MobileSim

Finalmente se muestran los resultados de la ejecución del algoritmo en la plataforma experimental, el robot Pioneer P3DX, bajo el mismo entorno de obstáculos, con las mismas dimensiones y ubicados en el mismo arreglo (Fig. 9).

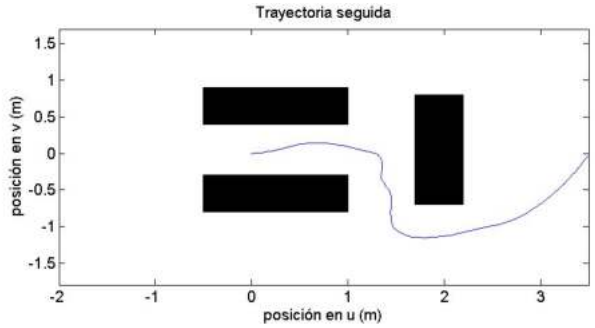

Fig. 9. Trayectoria obtenida al ejecutar el algoritmo sobre el robot Pioneer 3DX

Nuevamente se observa que se logra llegar al objetivo, evadiendo satisfactoriamente los obstáculos. El tiempo que le toma al robot llegar al objetivo es de 38.4 s, que es mayor al obtenido dentro de la simulación con sólo el modelo ideal, y menor que el obtenido dentro de la simulación con MobileSim. Con estos resultados se verifica la funcionalidad del algoritmo y esquemas de control diseñados para evadir obstáculos convexos.

7. CONCLUSIONES

Con estos resultados se verifica la funcionalidad del algoritmo y esquemas de control diseñados para evadir obstáculos convexos. Actualmente se trabaja en una ampliación del algoritmo, en particular agregando un modo en el que el robot sigue el espacio libre, con lo que se logra resolver el problema de evasión para obstáculos no convexos como los obstáculos en forma de G con un algoritmo simple, comparado con el propuesto por O.R.E Motlagh, et al. (2009), pudiendo con esto resolver el problema de navegación autónoma dentro de un laberinto.

AGRADECIMIENTOS

Este trabajo se realiza con el apoyo del CONACYT a través del proyecto: CB-2015-01, 254329, además de ser becario CONACYT.

REFERENCIAS

Arteaga, E., (2016). Teleoperación de un robot móvil, *Tesis para obtener el grado de maestría por el CINVESTAV-IPN,* México.

Chengchen Zhuge, Yunfei Cai, Zhenmin Tang (2017). A novel dynamic obstacle avoidance algorithm based on collision time histogram. *Chinese Journal of Electronics.*

Craig, John J. (2006). *Robótica*, Pearson Education, México.

C. Ye, H. C. Yung, D. Wang. (2003) A fuzzy controller with supervised learning assisted reinforcement learning algorithm for obstacle avoidance, *IEEE Trans. Syst. Man Cybern.* B 33, pp. 17-27.

Jang-Ho Cho et al (2018). A Real-Time Obstacle Avoidance Method for Autonomous Vehicles Using an Obstacle-Dependent Gaussian Potential Field. Journal of Advanced Transportation. (A ser publicado).

J. H. Lilly. (2007) Evolution of a negative-rule fuzzy obstacle avoidance controller for an autonomous vehicle, *IEEE Trans. Fuzzy Syst.* 15, pp. 718-728.

J. Velagic, B. Lacevic, B. Perunicic. (2006). A 3-level autonomous mobile robot navigation system designed by using reasoning/search approaches, *Robot. Autonomous Syst. 54,* pp. 989-1004.

O.R.E. Motlagh, T.S. Hong, N. Ismail. (2009). Development of a new minimum avoidance system for a behavior-based mobile robot, Fuzzy Sets Syst. 160, pp. 1929-1946.

Rong-Jong Wai, Chia-Ming Liu, You-Wei Lin. (2011). Design of switching path-planning control for obstacle avoidance of mobile robot, *Journal of the Franklin Institute.348,* pp. 718-737.

Velasco-Villa, A. Alvarez-Aguirre y G. Rivera-Zago. (2007). Discrete-Time Control of an Omnidireccional Mobile Robot Subject to Transport Delay, *Proceedings of the 2007 American Control Conference*, New York City, USA, pp. 2171-2176.

W. Tsui, M.S. Masmoudi, F. Karray, I. Song, M. Masmoudi. (2008). Soft-computing-based embedded design of an intelligent wall/lane-following vehicle, IEEE/ASME Trans. Mechatron. 13, pp 273-283.

