Diseño de un Controlador Fuzzy para Guiado de un Robot Móvil

Cela A.*; Sotomayor J.*; Sánchez F.**

**Escuela Politécnica Nacional, Facultad de Ingeniería Eléctrica y Electrónica, Quito, Ecuador*
e-mail: {andres.cela, juan.sotomayor}@epn.edu.ec
*** Universidad Carlos III de Madrid, Facultad de Ingeniería Telemática, Madrid, España*
e-mail: franklin.sánchez@alumnos.uc3m.es

Resumen: *En este trabajó se diseñó un controlador Fuzzy de tipo Mamdani para guiar un robot móvil por un entorno desconocido. El controlador tiene cuatro entradas y dos salidas. Las entradas contienen información de los sensores del robot mientras que las salidas corresponden a la velocidad lineal y angular del robot. Como parte del controlador se diseñó un conjunto de reglas en base a un conocimiento heurístico del comportamiento del robot y de los sensores. El control permite que el robot pueda evitar obstáculos los cuales son detectados por sensores ultrasónicos. El sistema ha sido simulado en el entorno virtual de Player Stage y probado en el robot móvil Amigobot el cual se mueve con tracción diferencial. Los resultados son satisfactorios en vista de que el diseño ha sido implementado en un robot real y se conoce con certeza que el objetivo se ha cumplido plenamente.*

Palabras clave: *control Fuzzy, robótica móvil, control, electrónica*.

Abstract: *In this work a Mamdani Fuzzy controller has been designed and implemented in a mobile robot Amigobot. The goal is guiding the robot in an unknown environment avoiding obstacles. Controller has four inputs and 2 outputs. The inputs are based on sensors information and outputs are angular and lineal velocities of robot. A set of rules has been designed as part of controller. The rules were designed using heuristic information about robots and sensors behavior in different maps. The control allows the robot avoids obstacles, which are detected by ultrasonic sensors. The whole system has been simulated in Player Stage environment and tested in the real mobile robot Amigobot which has differential traction. The aim has been accomplished satisfactory and real robot avoids obstacles.*

Keywords: *Fuzzy logic, robotic movil, control, electronic*.

1. INTRODUCCION

Hoy en día se puede observar el auge de la robótica móvil tanto en el ámbito académico como en el industrial. Aunque este campo es relativamente nuevo, 20 años [1], se observa un desarrollo acelerado debido al auge también de otras disciplinas como la electrónica y la inteligencia artificial. El robot móvil ha pasado de ser un mero elemento de estudio, investigación, competición [2] y hasta juego, a un dispositivo útil en el hogar, como se evidencia en las múltiples empresas que han desarrollado robots móviles de limpieza, [3] y [4]. Tomando este ejemplo, se puede notar que este tipo de robots de servicio necesita un sistema computacional que a más de controlar la tarea para lo cual está diseñado, también realice tareas básicas de guiado y navegación, lo cual se considera como elementos básicos que un robot móvil debe tener.

La terea de guiado se considera esencial ya que describe las trayectorias que el robot puede seguir o las dirección que puede tomar. La navegación permite que robot siga trayectorias reconociendo o no el entorno pero es fundamental que lo haga sin colisionar con obstáculos o paredes.

En este sentido existen diferentes formas de "*sensar*" el entorno. En [5] se propone usar una cámara de tiempo de vuelo y en combinación de odometría se consigue planificar trayectorias para un robot móvil. Otra forma de detectar el entorno es usando sensores infrarrojos. Se han usado los sensores ultrasónicos que son parte del robot móvil Amigobot ya que el software de programación ofrece suficiente soporte para ello.

Para generar las trayectorias existen diferentes métodos, como se muestra en [6] y [7], para los cuales el objetivo es encontrar el camino más eficiente en términos de distancia y seguridad en base a un mapa conocido y representado en grillas, del cual se obtiene un mapa topológicamente organizado.

Para entornos desconocidos esta tarea es irrelevante, y el sistema se limita, como en este trabajo, a la navegación, la cual puede tener objetivos como reconstrucción de entornos [8], [9] conocido como mapeado, con el cual se obtiene un mapa a partir de la información de los sensores y de la odometría del robot, o reconocimiento de objetos [10].

En cuanto a los métodos usados para controlar un robot móvil se encuentran desde los más comunes, como controladores PID [11] o Híbridos [12], hasta los más actuales como algoritmos genéticos y redes neuronales [13].

En este trabajo se propone usar un controlador borroso para controlar las velocidades asociadas al robot móvil, siendo

estas la velocidad lineal y la angular. Además, como se concluye en [14], este controlador es fácil de implementar en robots para aplicaciones de tiempo real. Por otro lado, el controlador borroso puede ser diseñado en programas como Matlab, o X-Fuzzy y luego ser exportado como un archivo de extensión .fis, al cual se puede acceder mediante una llamada desde cualquier compilador en C++. De esta forma se libera carga del programa principal que controla todos los dispositivos del robot y se accede al controlador cuando se tiene datos disponibles de los sensores.
El entorno de simulación Player Stage (PS) permite probar los algoritmos diseñados antes de llevarlos al robot real. Player Stage funciona bajo el sistema operativo Linux y permite conexión con algunos robots como Amigobot o Pionner. La ventaja de este simulador es que permite cargar diferentes mapas y estudiar el comportamiento del robot en cada uno de ellos. Si bien es cierto que los datos de sensores, velocidades, posiciones, etc. tienen ciertos errores por el hecho de ser señales simuladas, el poder de abstracción es tal que permite simplificar los algoritmos con resultados aceptables en el entorno real como se observará en los siguientes apartados de este trabajo.
En el siguiente apartado se muestra el diseño del controlador. En el apartado 3 se estudia las pruebas y en el 4 los resultados del sistema, tanto del real como del simulado. Finalmente en el apartado 5 se dan conclusiones sobre el trabajo.

2. ROBOT AMIGOBOT

2.1 Robot móvil Amigobot

El robot móvil que se usa para las pruebas es el Amigobot, aunque también se realizaron pruebas con el robot Pionner 2. Amigobot tiene un arreglo de 6 sensores ultrasónicos en su parte frontal con alcance de aproximadamente 4 [m]. Se mueve mediante un sistema de tracción diferencial a una velocidad máxima lineal de 0.9 [m/s]. Se usa odometría para determinar la posición relativa del robot mientras está en movimiento. El control total del robot se realiza usando una computadora enlazada a la red LAN mediante la cual se transmiten los datos [15]. La Fig. 1 muestra el robot Amigobot en el cual se aprecian los sensores ultrasónicos.

Figura 1. Robot Móvil Amigobot [15].

2.2 Robot simulado en PS

Una ventaja de PS sobre otro tipo de simuladores como Matlab, es que el robot móvil, cualquiera sea el fabricante, se simula como un objeto rectangular de 3 dimensiones, de esta manera se toma en cuenta aspectos como la inercia al acelerar o frenar, y las esquinas sólidas del robot. Una variable que no se puede simular es la reflexión del ultrasonido en los objetos, sean éstos paredes u obstáculos.
En la Fig. 2 se observa la el robot Amigobot simulado en PS. Se usan todos los sensores del robot, siendo estos S1 y S6 los laterales, S2 y S5 los esquineros y S3 y S4 los frontales.

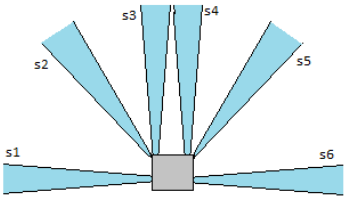


Figura 2. Robot Móvil Amigobot.

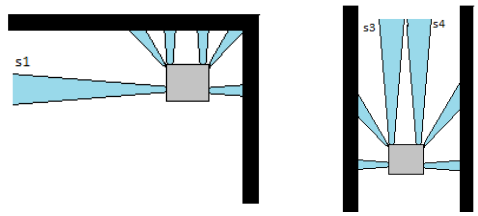
Cuando el robot choca con algún objeto la simulación genera un aviso de colisión. El usuario o programador debe detener la simulación para evaluar el problema.

3. DISEÑO DEL CONTROLADOR FUZZY

Se ha seleccionado el controlador Fuzzy debido a su operatividad en sistemas MIMO (Multiple Input - Multiple Output) por sus siglas en inglés, lo cual es una ventaja frente al control clásico.
Se ha seleccionado el método de Inferencia Borrosa Mamdani, el cual es usado en casos prácticos y cuando el número de variables lingüísticas es reducido. La estructura de este método se basa en condiciones de mínimos y máximos, lo cual hace que el controlador sea rápido al computar operaciones simples [16].

3.1 Selección de Entradas y Salidas

El controlador Fuzzy permite controlar una planta sin necesidad de tener el modelo de la misma, por esta ventaja se ha decidido usar este método. En un controlador Fuzzy se usa como entradas conjuntos borrosos, los cuales dependen del estado del robot, que en este caso está dado por la distancia medida por los sensores. Las salidas del controlador también son borrosas las cuales se desborrosifican para obtener un valor lógico para el sistema de tracción.
Considerando algunos posibles escenarios, como los de la Fig. 3, se decidió usar 4 entradas y dos salidas borrosas La entrada 1 corresponde al sensor S3, la entrada 2 al sensor S4, la entrada 3 corresponde a la diferencia sensor S2 - sensor S5 y la entrada 4 a la diferencia sensor S1- sensor S6. Las salidas son dos y corresponden a las velocidades lineal y angular del robot.



(a) (b)

Figura 3. Dos Posibles escenarios.

En la Fig. 3(a) se observa el robot cuando está frente a la esquina de una pared. La información de los dos sensores frontales S3 y S4 es prácticamente la misma porque están a la misma distancia de la pared, lo mismo sucede con los sensores esquineros, de tal manera que la información útil para salir la ofrece el sensor S1. La ubicación del los sensores laterales permite saber si existe espacio disponible para que el robot vaya en esa dirección, en cuyo caso el robot gira para tomar ese camino. Para reducir el número de entradas, y por consiguiente de reglas del sistema, no se usan los sensores S1 y S6 como entradas independientes, sino que su diferencia genera una sola entrada. De la misma forma se usa la diferencia de los sensores esquineros S2 y S5 como otra entrada, En la Fig. 3 (b) se observa que los sensores frontales aportan mucha información acerca del entorno ya que indican si existe espacio disponible para navegar, por tanto, los sensores S3 y S4 se usan como entradas independientes. De esta manera quedan definidas las 4 entradas del controlador Fuzzy que se muestran en la Fig. 4 y que se describen a continuación.

3.2 Diseño de los Conjuntos de Entrada y Salida Borrosos

La entrada 1, con información del sensor S3, tiene dos subconjuntos trapezoidales, Pequeño (P) y Grande (G), y un triangular central, Mediando (M), esto permite que en el rango central de la variable exista más dinámica que en los extremos, en donde se hacen correcciones extremas como: girar lo máximo a derecha o izquierda, etc.
La entrada 2, con información del sensor S4 tiene la misma configuración y distribución que la entrada 1.
La entrada 3, con información de la diferencia entre S2 y S5, se distribuye en tres subconjuntos similares a la entrada 1 y con nombres Negativo (N), Cero (Z) y Positivo (P).
La entrada 4, con información de la diferencia entre los sensores S1 y S6 tiene dos subconjuntos trapezoidales ubicados de tal forma que el valor borroso es prácticamente un valor digital, así cuando el subconjunto Negativo (N) está en 1 el subconjunto Positivo (P) está en 0, y viceversa.
La escala de las variables está dada en cm ya que es la escala usada en el simulador.

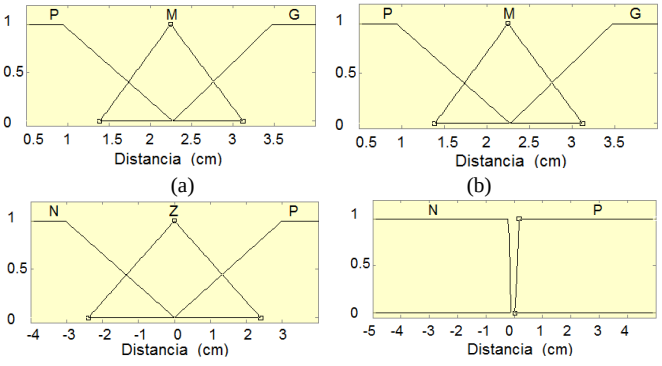


Figura 4. Conjuntos borrosos de las entradas. (a) Entrada 1,sensor frontal S3. (b) Entrada 2, sensor frontal S4. (c) Entrada 3, S2 - S5. (d) Entrada 4, S1- S6.

En la Fig. 5(a) se observa el conjunto borroso de la salida1 correspondiente a la velocidad lineal, para este conjunto la escala varía de 0 a 1 donde 0 es el mínimo y 1 el máximo.
Se han considerado 7 subconjuntos triangulares distribuidos uniformemente a lo largo del rango de la velocidad lineal, lo que permite tener una salida dinámica para todo su rango. Los subconjuntos borrosos son: Pequeño Pequeño (PP), Pequeño Medio (PM), Medio (M), Grande Medio (GM) y Grande Grande (GG).
Para la salida de velocidad angular se consideró los siguientes 7 subconjuntos borrosos: Negativo (N), Negativo Medio (NM), Cero Negativo, (ZN), Cero (Z), Cero Positivo (ZP), Positivo Medio (PM) y Positivo (P), como se observa en la Fig. 5 (b). Los subconjuntos N y P se ubican en los extremos del rango de la velocidad angular mientras los demás subconjuntos están alrededor de la velocidad nula. Con esta distribución la dinámica de la respuesta es tal, que es más sensible a cambios alrededor de 0 (rad/s).

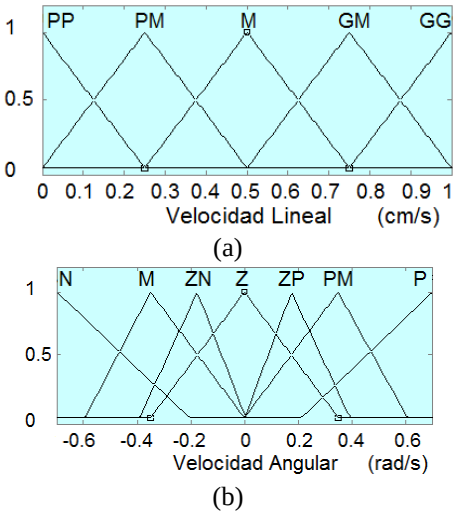


Figura 5. Conjuntos borrosos de las salidas. (a) Velocidad Lineal. (b) Velocidad angular.

3.3 Diseño del Conjunto de Reglas de Control.

Un controlador Fuzzy se considera completo si existe un número de reglas igual a todas las posibles combinaciones de los subconjuntos de entrada. En este trabajo el controlador completo debería tener 2^{11} reglas, pero considerar este número de reglas resulta intratable, razón por la cual, usando conocimiento heurístico del sistema se diseñó el conjunto de reglas basado en la forma general expresada en (1), en donde ***x*** representa los conjuntos y **A** los subconjuntos de entrada y, ***y*** los conjuntos y **B** los subconjuntos de salida.

$$\mathrm{R}^{\mathrm{i}}:\mathrm{IF\,f}(x_1\ \text{es}\ \mathrm{A}_1,\ldots,x_k\ \text{es}\ \mathrm{A}_{\mathrm{k}})\ \ \mathrm{THEN}\ y=\mathrm{B}_{\mathrm{m}} \qquad (1)$$

Un subconjunto está activo cuando el valor del rango de la variable da un valor borroso mayor de cero. Así por ejemplo, si la entrada 1 tiene un valor de 2 (cm), los subconjuntos activos son P y M con valores borrosos de 0.2 y 0.6 aproximadamente. El subconjunto G tiene un valor borroso de cero, por tanto está desactivado.
La regla debe considerar la combinación de las diferentes entradas. Es decir, pueden estar activados subconjuntos de diferentes entradas al mismo tiempo, como se indicó arriba. Esta combinación se define mediante el operador **AND** que reemplazando en (1) da como resultado

$$\begin{aligned}\mathrm{R}^{\mathrm{i}}:\mathrm{IF\,f}(&x_1\ \text{es}\ \mathrm{A}_{\mathrm{k}}\ \mathrm{AND}\ x_2\ \text{es}\ \mathrm{A}_{\mathrm{k}}\ \mathrm{AND}\ x_3\ \text{es}\ \mathrm{A}_{\mathrm{k}}\ldots\\ &\mathrm{AND}\ x_4\ \text{es}\ \mathrm{A}_{\mathrm{k}})\ \ \mathrm{THEN}\ y_1\ \text{es}\ \mathrm{B}_{\mathrm{m}},\ y_2\ \text{es}\ \mathrm{B}_{\mathrm{m}}\end{aligned} \qquad (2).$$

El set de reglas se compone de 26 combinaciones en donde ***I1*** es la variable lingüística ***Distancia 1*** de la entrada 1, ***I2*** es la variable ***Distancia 2*** de la entrada 2, ***I3*** es la diferencia de distancias correspondiente a la entrada 3, ***I4*** es la diferencia de los sensores laterales, entrada 4. Las premisas básicas para la construcción del set de reglas son las siguientes:

- Si los sensores frontales no detectan obstáculos se debe incrementar la velocidad lineal para avanzar rápido.
- Si se detecta un obstáculo con el sensor frontal izquierdo a media distancia se debe girar un poco, "medio", hacia la derecha y reducir la velocidad lineal. Si el obstáculo se detecta más cerca, se reduce aun más la velocidad lineal.
- Si se detecta obstáculo muy cerca, "pequeña distancia", por los dos sensores frontales, entonces se debe reducir la velocidad lineal al mínimo y girar hasta encontrar vía libre de obstáculos con alguna otra entrada, como I3 o I4, así el robot gira hacia la derecha o la izquierda buscando una vía alternativa.
- Si ningún sensor detecta obstáculos, se debe dar prioridad a las reglas de los sensores delanteros, es decir, se incrementa la velocidad lineal para ir más rápido hacia adelante.

De ésta forma se define el set de 26 reglas que se muestra a continuación.

R[1]: If (***I1*** is P) and (***I2*** is P) and (***I3*** is N) and (***I4*** is N)then(***V*** is PP)(***W*** is N)
R[2]: If (***I1*** is P) and (***I2*** is P) and (***I3*** is N) and (***I4*** is P)then (***V*** is PP)(***W*** is P)
R[3]: If (***I1*** is P) and (***I2*** is P) and (***I3*** is Z)and (***I4*** is N) then (***V*** is PP)(***W*** is N)
R[4]: If (***I1*** is P) and (***I2*** is P) and (***I3*** is Z) and (***I4*** is P) then (***V*** is PP)(***W*** is P)
R[5]: If (***I1*** is P) and (***I2*** is P) and (***I3*** is P) and(***I4*** is N) then (***V*** is PP)(***W*** is N)
R[6]: If (***I1*** is P) and (***I2*** is P) and (***I3*** is P) and (***I4*** is P) then (***V*** is PP)(***W*** is P)
R[7]: If (***I1*** is M) and (***I2*** is P) and (***I3*** is N) then (***V*** is PM)(***W*** is N)
R[8]: If (***I1*** is M) and (***I2*** is P) and (***I3*** is Z) then (***V*** is M)(***W*** is PM)
R[9]: If (***I1*** is M) and (***I2*** is P) and (***I3*** is P) then (***V*** is GM)(***W*** is P)
R[10]: If (***I1*** is M) and (***I2*** is M) and (***I3*** is N) then (***V*** is M)(***W*** is N)
R[11]: If (***I1*** is M) and (***I2*** is M) and (***I3*** is Z) then (***V*** is GM) (***W*** is Z)
R[12]: If (***I1*** is M) and (***I2*** is M) and (***I3*** is P) then (***V*** is M)(***W*** is P)
R[13]: If (***I1*** is M) and (***I2*** is G) and (***I3*** is N) then (***V*** is GG)(***W*** is NM)
R[14]: If (***I1*** is M) and (***I2*** is G) and (***I3*** is Z) then (***V*** is GG)(***W*** is Z)
R[15]: If (***I1*** is M) and (***I2*** is G) and (***I3*** is P) then (***V*** is M)(***W*** is NM)
R[16]: If (***I1*** is P) and (***I2*** is M) and (***I3*** is N) then (***V*** is GM)(***W*** is N)
R[17]: If (***I1*** is P) and (***I2*** is M) and (***I3*** is Z) then (***V*** is M)(***W*** is NM)
R[18]: If (***I1*** is P) and (***I2*** is M) and (***I3*** is P) then (***V*** is PM)(***W*** is P)
R[19]: If (***I1*** is G) and (***I2*** is G) and (***I3*** is N) then (***V*** is GG)(***W*** is N)
R[20]: If (***I1*** is G) and (***I2*** is G) and (***I3*** is Z) then (***V*** is GG)(***W*** is Z)
R[21]: If (***I1*** is G) and (***I2*** is G) and (***I3*** is P) then (***V*** is GG)(***W*** is P)
R[22]: If (***I1*** is G) and (***I2*** is M) and (***I3*** is N) then (***V*** is M)(***W*** is PM)
R[23]: If (***I1*** is G) and (***I2*** is M) and (***I3*** is Z) then (***V*** is GG)(***W*** is Z)
R[24]: If (***I1*** is G) and (***I2*** is M) and (***I3*** is P) then (***V*** is GG)(***W*** is PM)
R[25]: If (***I1*** is G) and (***I2*** is P) then (***V*** is M)(***W*** is ZP)
R[26]: If (***I1*** is P) and (***I2*** is G) then (***V*** is M)(***W*** is ZN)

Las Variables lingüísticas de velocidades lineal y angular están representadas por ***V*** y ***W*** respectivamente. Sus subconjuntos borrosos toman el nombre de consecuentes.
Las 6 primeras reglas ayudan a tomar decisiones extremas cuando el robot está en circunstancias similares a las de la Fig. 3(a). En esos casos prima el giro del robot a altas velocidades angulares y bajas lineales.
Las siguientes 18 reglas trabajan en el rango dinámico de las combinaciones de los subconjuntos de entrada premiando o castigando las velocidades dependiendo de las distancias a los obstáculos.
Finalmente las dos últimas reglas, premian el sistema con altas velocidades lineales cuando los sensores frontales detectan que existe "*Grande*" vía libre para la navegación, como se indicó en las premisas básicas.

3.4 Características del Controlador

El Operador **MIN** se usa en casos prácticos para ajustar el Operador **AND** que une las variables lingüísticas. El operador **MIN** se define por

$$\mathrm{MIN}(\mathrm{r_x}=\min(\mathrm{A}_1,\mathrm{A}_2,\ldots\mathrm{A}_{\mathrm{k}})) \qquad (3).$$

El ajuste del operador **AND** se hace con el operador producto (**PROD**) en casos de estudio de estabilidad de sistemas [17]. La agregación, es decir cuando dos o más reglas actúan sobre el mismo consecuente, se resuelve por máximos y se define por

$$\mathrm{r_y}=\max\left[\mathrm{V}^{\mathrm{N}}{}_{\mathrm{i}=1}\ \mathrm{R}^{\mathrm{i}}\rightarrow B_m\right] \qquad (4).$$

En donde **V** es el conjunto de valores de los consecuentes repetidos de la Regla $\mathbf{R}^{i}$ correspondiente [17].
La *desborrosificación* está dada por el método de los centroides de acuerdo al método Mamdani [16].
El controlador borroso tiene las siguientes características:

- Entradas: 4
- Salidas: 2
- Método Borroso: Mamdani
- Método AND: Mínimos
- Método OR: Máximos
- Agregación: Máximos
- Desborrosificación: Método de los Centroides.

Una vez diseñado el controlador borroso se simuló los resultados de las variables lingüísticas en el programa X-Fuzzy, el cual permite diseñar, simular y evaluar controladores borrosos.
Las superficies de control son superficies que indican las tendencias del sistema de control para corregir el error. Estas están formadas por una combinación de variables de entrada y salida.
En la Fig. 6 se observa dos superficies de control de tres dimensiones. El plano xy lo forman los rangos de las entradas I1 e I2, mientras el eje Z lo forma la Velocidad Lineal del robot. Se puede observar que, si las distancias de los sensores frontales son pequeñas, alrededor de 1, la velocidad es 0.5 (cm/s), pero conforme las distancias crecen la velocidad aumenta de acuerdo a lo establecido por las reglas. La máxima velocidad lineal se da cuando los dos sensores miden una distancia de 4 (cm), lo que indica que el robot puede navegar de frente a máxima velocidad ya que no existe obstáculos.
Existen tres superficies adicionales que se forman por la combinación de las entradas I3 e I4 y la velocidad lineal.

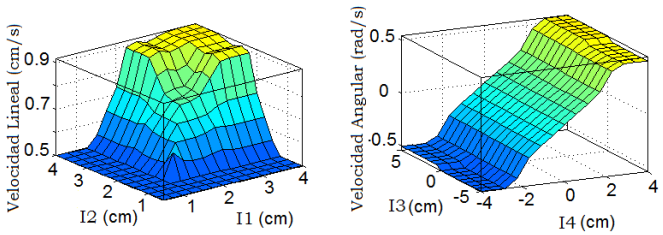


Figura 6. Superficies de control. (a) Respecto de la Velocidad Lineal. (b) Respecto de la Velocidad Angular

En la Fig. 6(b) se observa la superficie de control formada por las entradas I3, I4 y la velocidad angular del robot. Cuando la entrada I4 indica 4 (cm) de distancia hacia la izquierda, lo que sugiere que no hay obstáculos, el robot gira en esa dirección para tomar ese camino. Lo mismo ocurre si no hay obstáculos al otro lado del robot.
La velocidad angular sigue una distribución aproximadamente lineal a lo largo de las distancias de I3 e I4, lo cual indica que el robot no gira cuando no hay distancia disponible, como ocurre en I3=0 (cm) e I4=0 (cm).

Existen otras tres superficies de control formadas por las entradas I1, I2 y la velocidad angular que tienen similares características.
El lazo de control se representa en la Fig. 7, en donde se observa que el Punto de consigna (Set Point) forma parte del controlador Fuzzy el cual entrega las variables de velocidad. Las 4 entradas del controlador vienen de la realimentación de los 6 sensores que miden la distancia del robot al obstáculo. Los bloques Vel. L y Vel A. acondicionan y escalan las variables de velocidad lineal y angular respectivamente, al rango de velocidades del robot real. El sistema es MIMO.

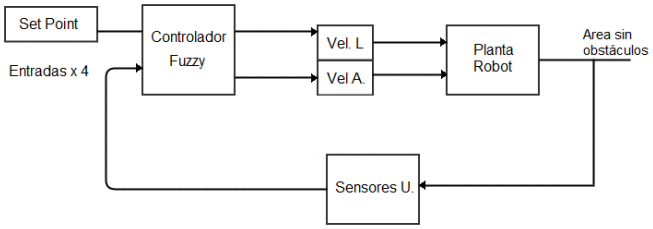


Figura 7. Lazo de control del Robot (Sistema MIMO).

4. PRUEBAS Y RESULTADOS

Las pruebas de simulación se realizaron en PS en el cual se pueden enlazar el programa de simulación, el controlador borroso, el robot y el mapa, todo bajo SO Linux.
El programa de simulación tiene como tarea principal leer los sensores y enviar los datos del controlador al robot. Además, como tareas adicionales se pinta puntos en el mapa en el lugar de detección de los sensores y en el camino recorrido por el robot.
La Fig. 8 muestra la simulación realizada, en donde se observa el robot como un cuadro de color rojo y el alcance de sus sensores en color celeste. El recorrido realizado por el robot se ha pintado como finos puntos de color rojo, mientras los obstáculos detectados se han pintado en puntos de color azul. Este mapa es básico ya que tiene un camino en forma de rectángulo en donde la detección de los sensores frontales es la máxima en todo el recorrido excepto en las esquinas. Se puede observar que en el camino el robot no ha colisionado con ningún obstáculo o pared. El tiempo de recorrido del robot en una vuelta fue de 50 (s).

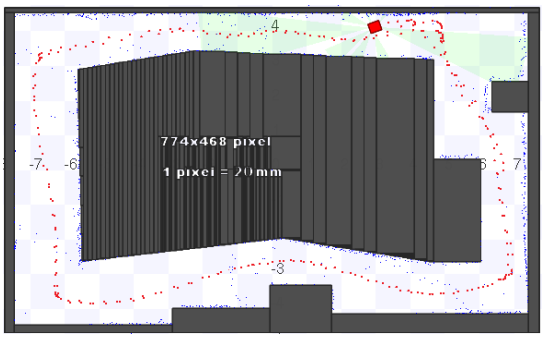


Figura 8. Pruebas en Mapa 1.

La Fig. 9 muestra el recorrido en el Mapa 2, el cual tiene una dificultad mayor que el anterior por las curvas en "U" y las paredes multiformes. La acumulación de puntos rojos, especialmente en las curvas, indica que el robot ha reducido su velocidad para no colisionar. El tiempo para recorrer este mapa fue de 85 (s). No se detectaron colisiones.

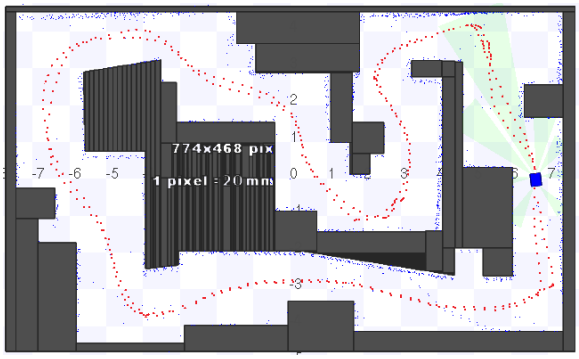


Figura 9. Pruebas en Mapa 2.

Una vez pasadas las pruebas de simulación se enlazó el sistema al robot Amigobot. Para ello se hizo un programa de conexión, similar al de simulación pero que no pinte puntos en el recorrido ni en la detección de los sensores.
El entorno de recorrido fue el Laboratorio de Inteligencia Artificial de la Politécnica de la Universidad de Alcalá.
Se observó que los sensores no detectan objetos muy delgados como las patas de una mesa. También tienen problemas de detección de objetos con una superficie de textura lisa.
El recorrido se hizo poniendo como obstáculos objetos grandes como: cajas, personas, otro robot, etc. y los resultados fueron satisfactorios ya que el robot no colisionó con estos objetos sino que tomó rutas alternativas.

5. CONCLUSIONES

Las conclusiones de este trabajo son las siguientes:
Un controlador Fuzzy permite usar varias entradas para supervisar varias salidas. La ventaja de este tipo de control en sistemas MIMO es evidente ya que un controlador clásico PID necesitaría un lazo para cada variable de control, mientras que el Fuzzy trabaja con todas las variables como un solo bloque. Por otro lado, el algoritmo del control Fuzzy ocupa más memoria de programa, y por tanto necesita más tiempo para ejecutarse que un PID, aunque este punto es irrelevante cuando el controlador está funcionando sobre un procesador de altas prestaciones como los de un computador personal.
Aunque este tipo de control trabaja fácilmente con sistemas MIMO hay que considerar seriamente el número de variables de entrada y salida, pues esto repercute en el número de reglas que definen el sistema. Además, mientras más variables tiene el controlador mayor debe ser el conocimiento del comportamiento del sistema ya que las reglas se construyen en base a este conocimiento.
Un set de reglas completo ofrece mayor dinámica de control por cuanto para cada combinación entrada-salida ofrece una regla consecuente, pero un set de reglas reducido, como en este caso, permite eliminar algunas combinaciones entrada-salida que no se cumplen, y por tanto simplifica la solución del controlador.
La facilidad que presentan programas como XFuzzy o Matlab para diseñar controladores Fuzzy permite que el desarrollo de este tipo de sistemas sea más rápido. Además, con la ayuda del software de simulación Player Stage se observa de primera mano los resultados del controlador sobre el robot. Aunque el medio de simulación tenga sus limitaciones, es una herramienta poderosa para depurar el algoritmo, empezando desde las entradas, salidas, reglas y método de control Fuzzy.
Los sensores ultrasónicos ofrecen una buena solución para medir distancia entre los rangos de 2 (cm) a 4 (m). En el robot Amigobot están distribuidos de tal forma que se puede "observar" obstáculos alrededor de los 180º frontales del robot a una distancia mínima de 5 (cm) y máxima de 4 (m).
Antes de seleccionar el tipo de sensores para medir distancia se debe conocer el tipo de obstáculos a detectar. Por ejemplo, los sensores del robot Amigobot no detectan obstáculos con superficies de textura lisa como madera lacada, baldosa lisa, etc., u obstáculos de grosor inferior a 3 o 4 (cm), pero ofrecen buenos resultados al detectar personas, cajones u otros robots similares.
Este trabajo demuestra que un controlador Fuzzy puede ser implementado exitosamente en un robot móvil para controlar su dinámica. Aunque podría tener limitantes en el tipo de procesador que use dicho robot, pues, como se ha comentado anteriormente, este tipo de control necesita muchos recursos computacionales que en robots básicos no se tiene.
Una forma de implementar este tipo de control en procesadores básicos de 8 bits y 20 [MHz], es diseñar los conjuntos borrosos como líneas que se intersecan, o hacer tablas de valores. El problema con la primera opción es la dificultad de la programación, el de la segunda es la cantidad de memoria, aun más, en cualquiera de los dos casos, si se desea mover los subconjuntos se los debe reprogramar. Por eso se recomienda usar procesadores de mayores prestaciones como los ARM o Core 2 DUO, o como en este trabajo, usar un computador que realice el trabajo de control y que envíe datos al robot mediante una red inalámbrica.
Aunque el objetivo de este trabajo fue diseñar el controlador para que el robot evite obstáculos, un aporte adicional se puede observar en las Figs. 8 y 9, en dónde, con los puntos azules que representan los obstáculos detectados por los sensores, se puede reconstruir el mapa del entorno recorrido. Este avance proyecta un trabajo futuro de reconocimiento y reconstrucción de entornos desconocidos, también llamado mapeado o "mapping" por su traducción al inglés, en el cual se usará el controlador Fuzzy diseñado en este trabajo.

REFERENCIAS

[1] A. Aksamovic, M. Hebibovic, y S. Konjicija, «Similarities in development of digital computers and mobile robots», en *IEEE EUROCON 2009, EUROCON '09*, 2009, pp. 740-745.

[2] J. Green y S. Plumb, «Mobile robot competition», en *AFRICON, 2011*, 2011, pp. 1-6.

[3] «Robot aspirador LG: limpieza automática de tu casa | LG España». [En línea]. Disponible en: http://www.lg.com/es/robot-aspirador. [Accedido: 13-sep-2013].

[4] «Aspiradoras | SAMSUNG». [En línea]. Disponible en: http://www.samsung.com/latin/consumer/home-appliances/vacuum-cleaners/?cid=ec_search_google_alwayson_ha_20130304. [Accedido: 13-sep-2013].

[5] H. Fukai, Y. Mitsukura, y G. Xu, «The calibration between range sensor and mobile robot, and construction of a obstacle avoidance robot», en *2012 IEEE RO-MAN*, 2012, pp. 737-742.

[6] A. R. Willms y S. X. Yang, «An efficient dynamic system for real-time robot-path planning», *IEEE Trans. Syst. Man Cybern. Part B Cybern.*, vol. 36, n.º 4, pp. 755-766, 2006.

[7] N. Sedaghat, «Mobile robot path planning by new structured multi-objective genetic algorithm», en *2011 International Conference of Soft Computing and Pattern Recognition (SoCPaR)*, 2011, pp. 79-83.

[8] R. Ouellette y K. Hirasawa, «Mayfly: A small mapping robot for Japanese office environments», en *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, 2008. AIM 2008*, 2008, pp. 880-885.

[9] P. Mirowski, R. Palaniappan, y T. K. Ho, «Depth camera SLAM on a low-cost WiFi mapping robot», en *2012 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*, 2012, pp. 1-6.

[10] K. Balasubramanian, R. Arunkumar, J. Jayachandran, V. Jayapal, B. A. Chundatt, y J. D. Freeman, «Object recognition and obstacle avoidance robot», en *Control and Decision Conference, 2009. CCDC '09. Chinese*, 2009, pp. 3002-3006.

[11] T.-Y. Kuc y W.-G. Han, «Adaptive PID learning of periodic robot motion», en *Proceedings of the 37th IEEE Conference on Decision and Control, 1998*, 1998, vol. 1, pp. 186-191 vol.1.

[12] X. Lai, S. Zhu, y W. Wu, «Research on driving wheel control of cleaning robot based on fuzzy adaptive tuning PID», en *International Conference on Mechatronics and Automation, 2009. ICMA 2009*, 2009, pp. 535-540.

[13] X. Yang y M. Meng, «A neural network approach to real-time motion planning and control of robot manipulators», en *1999 IEEE International Conference on Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings*, 1999, vol. 4, pp. 674-679 vol.4.

[14] S. H. Lian, «Fuzzy logic control of an obstacle avoidance robot», en , *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems, 1996*, 1996, vol. 1, pp. 26-30 vol.1.

[15] «Map Making». [En línea]. Disponible en: http://www8.cs.umu.se/~c06aag/mapmaking/. [Accedido: 13-sep-2013].

[16] K. Michels, *Fuzzy control: fundamentals, stability and design of fuzzy controllers*. Springer, 2006.

[17] Cela, A, Yebes, J.J, Arroyo, R, Bergasa, L.M, Barea, R, y López, E, «Complete Low-Cost Implementation of a Teleoperated Control System for a Humanoid Robot», *Sensors 2013*, vol. 13, pp. 1385-1401, ene. 2013.