Aprendizaje de Reglas de control en Robots Móviles

Rogelio Ferreira Escutia
Instituto Tecnológico de Morelia
Avenida Tecnológico 1500,
Morelia, Michoacán, México
rferreir@antares.tecmor.mx

Eduardo Morales Manzanares
ITESM - Campus Morelos
Paseo de la Reforma 182-A,
62589, Temixco, Morelos, México
emorales@campus.mor.itesm.mx

Resumen

Se describe un sistema capaz de aprender reglas de control para navegación de un robot móvil. El aprendizaje se realiza por medio de imitación, donde un usuario, le dice al sistema qué hacer en algunas situaciones prototípicas, y el sistema es capaz de utilizar lo aprendido en situaciones no vistas. Para esto, se desarrollo también un algoritmo de evitación de trampas. En particular, se prueba el sistema en ambientes de oficina (paredes con ángulos rectos) obteniendo resultados satisfactorios en simulaciones en computadora y en ambientes reales. Una vez aprendido un conjunto de reglas el sistema es capaz de irse de un punto inicial a uno final en una ambiente desconocido. Las reglas fueron reducidas posteriormente siguiendo un esquema de aprendizaje adicional y es útil inclusive en ambientes dinámicos.

1 Introducción

Los robots han empezado a salir de los laboratorios de experimentación, para llegar cada día a un mayor número de lugares y con nuevas e interesantes aplicaciones de la vida real [Augustun and Allen, ; Begley, 1997; Fu and González, 1987]. En la actualidad es común la utilización de robots industriales, los cuales realizan un conjunto fijo de acciones [Barr and Feigenbaum, 1986]. En el caso de los robots móviles que se desplazan en ambientes desconocidos es necesario tener una mayor flexibilidad [Brooks, 1986; 1997]. Una posibilidad es instruir al robot con todas las posibilidades en un ambiente conocido. Las principales desventajas son la gran cantidad de posibilidades y la dependencia del ambiente. La idea central de este trabajo es enseñarle a un robot qué acciones realizar en ciertas situaciones y utilizar lo aprendido en situaciones nuevas. En este sentido, se realiza un aprendizaje por imitación [Morales, 1996; Friedrich *et al.*, 1996]. Lo que se pretende en este trabajo, es que una vez aprendido el sistema un conjunto de reglas, sea capaz de ir desde un punto inicial a uno final en un ambiente desconocido. En este trabajo se asume un ambiente de oficinas (paredes con ángulos rectos). El problema de caer en ciclos es atacado por un algoritmo novedoso de introducción de metas temporales.

La sección 2 describe las condiciones en las cuales se realizaron los experimentos, el robot utilizado y su configuración de sensores. La sección 3 presenta el algoritmo de navegación y el de aprendizaje utilizado, para construir las reglas de control. En la sección 4 se muestran los experimentos realizados y los resultados obtenidos, tanto en ambientes simulados como reales. Finalmente, se presentan las conclusiones y trabajo futuro en la sección 5.

2 Entorno del robot y creación de reglas

La idea principal para aprender reglas de control, es que un usuario le indica al robot qué hacer cuando se encuentra en una situación determinada y en un cierto entorno conocido. El robot utiliza la información del entorno y la acción sugerida por el usario para construir una regla. Con el conjunto de reglas aprendido se coloca al robot en un ambiente similar al de entrenamiento, y es capaz de llegar a la meta propuesta en forma autónoma.

El algoritmo de aprendizaje se montó sobre un robot Nomad 200 (de Nomadic Technologies). El Nomad 200 cuenta con diferentes instrumentos con los cuales puede obtener información del entorno y así poder ejecutar alguna operación en base a la información obtenida.

Los principales sistemas con que cuenta el robot son los siguientes [Nomadic-Technologies, 1996]:

- Sensores táctiles (detectores de choques)
- Sistema de sonares
- Cámara de video
- Sistema de comunicación (transmite datos a una computadora)

Para detectar las condiciones del entorno, en estos experimentos se utilizaron únicamente los datos de los sonares.

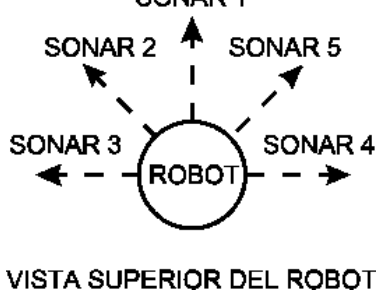

Figura 1: Disposición de los Sonares.

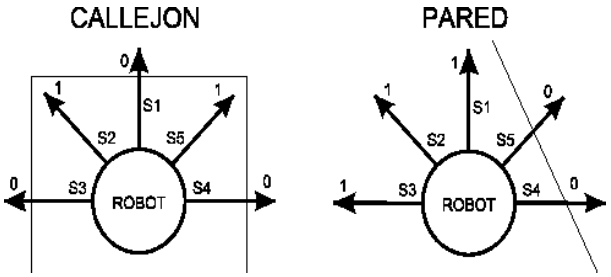

Figura 2: Representación del entorno.

2.1 Sonares

El robot Nomad 200 cuentan con 16 sonares que se encuentran colocados alrededor del robot en forma uniforme en la parte superior. El sistema de sonar tiene una capacidad de detectar objetos que se encuentren a una distancia que va desde 5 hasta 255 pulgadas (12.7 hasta 647.7 cm.).

Para los experimentos sólo se utilizaron los sonares que se encuentran al frente del robot (Figura 1) para tomar 5 lecturas. Los sensores tactiles se usaron unicamente por cuestiones de seguridad.

2.2 Discretización

Los valores de los sonares se discretizaron en dos valores: 0 o *cerca*, que significa que se detecta un obstáculo a menos de 20 pulgadas y 1 o *lejos* que significa que no existe un obstáculo a menos de 20 pulgadas. Como veremos más adelante, este valor puede ser variado, dependiendo de las condiciones del ambiente y no afecta las reglas aprendidas anteriormente.

2.3 Representación del entorno

Se analizaron todos los posibles casos de obstáculos que podría detectar el robot con la disposición de sonares de la Figura 1, para ver si eran cubiertos después de la fase de entrenamiento (descrita posteriormente). La figura 2 presenta dos ejemplos.

2.4 Alineación del robot con el mundo real

Además de las condiciones del entorno (tomadas de los sonares) se tomaron las condiciones de la posición de la meta relativa al robot. Para esto, es necesario primero alinear al robot con respecto a un marco de referencia, que en esta caso fue una marca sobre el piso dentro del laboratorio donde se realizaron las pruebas (ver figura 3).

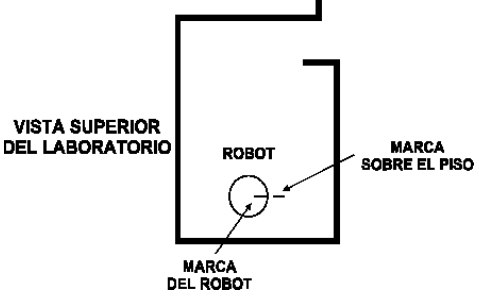

Figura 3: Alineaciín del robot.

2.5 Dirección de la meta

La posición de la meta (con respecto a este marco de referencia del robot) puede tomar los siguientes valores: en dirección vertical: 0 (mismo altura), 1 (arriba), 2 (abajo), y en dirección horizontal: 0 (misma alineación), 1 (a la izquierda), y 2 (a la derecha). Por ejemplo, si la meta esta arriba y a la izquierda del robot (con respecto a su frente dado por su alineación), los valores de la meta serían: 11.

2.6 Acciones de control

Las acciones básicas de control consideradas en estos experimentos son 5 y se codifican de la siguiente manera:

0: detenerse y girar 90^o
1: detenerse y girar 45^o
2: avanzar
3: detenerse y girar -45^o
4: detenerse y girar -90^o

2.7 Creación de una regla

Dada la descripción del entorno y las acciones podemos definir la sintaxis de las reglas como una cadena en donde los primeros 5 lugares representan los valores de los sonares, los siguientes dos la posición de la meta y el siguiente la acción a realizar.

S1, S2, S3, S4, S5, Meta(2), Acción, Peso

Por ejemplo, si el robot se encuentra en un pasillo y la meta se encuentra hacia delante, el usuario le indicará al robot que para este punto en especial la acción a ejecutar será avanzar (Figura 4). La codificación para esta regla sería:

11001022100

donde:

Sensor1 = 1 (pared lejos)
Sensor2 = 1 (pared lejos)
Sensor3 = 0 (pared cerca)
Sensor4 = 0 (pared cerca)

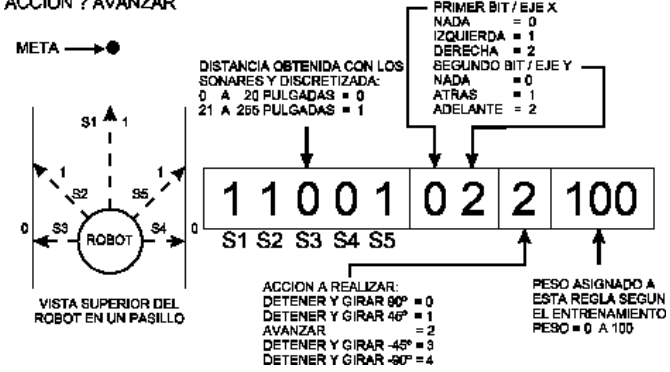

Figura 4: Creación de una regla.

Sensor5 = 1 (pared lejos)
Meta1 = 0 (ni izq, ni der.)
Meta2 = 2 (adelante)
Accion = 2 (avanzar)
Peso = 100

2.8 Peso de una regla

El peso nos identifica la importancia que se le da a la regla. Se planteo originalmente para darle diferente prioridad a reglas que aplican en las mismas condiciones del ambiente y que este se fuera aprendiendo por refuezo [Sutton and Barto, 1998]. Cuando se crea una regla nueva se le asigna un peso de 100. Cuando existen dos reglas se les asigna un peso igual a las dos de 50. Cuando existen 3 reglas, donde una tiene prioridad sobre las otras 2, se les asignó un peso de 60, 20 y 20 respectivamente.

3 Entrenamiento y navegación

Una vez que se crea una regla, el robot ejecuta la acción que le indica la regla. Una vez ejecutada la acción el robot realiza nuevamente mediciones del ambiente, y si uno de los datos cambia (datos de los sonares ó dirección de la meta), entonces revisa si existe una regla que aplique para este caso. Si existe una regla, le pregunta al usuario si la aplica. Si acepta, la aplica, si no, entonces le pregunta que acción debe de ejecutar, y de esta manera crea una nueva regla.

Después del entrenamiento donde se asignan diferentes metas en diferentes entornos, se probó si el robot era capaz de navegar en forma autónoma y llegar a una meta en un entorno parecido. Con el conjunto de reglas aprendidas se posiciona al robot en un ambiente parecido, se le asigna un punto inicial y su ángulo, y una meta, y el robot trata de llegar a la meta, siguiendo el algoritmo de navegación de la figura 5.

4 Pruebas realizadas

El robot Nomad 200 cuenta con un simulador, en el cual se pueden probar los algoritmos, antes de utilizar el robot real.

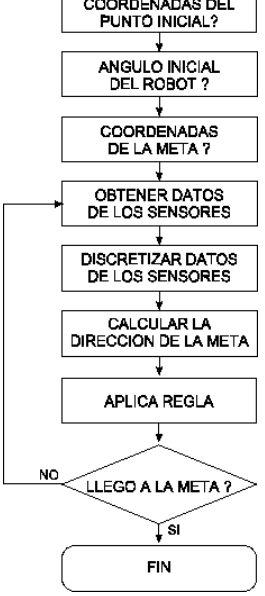

Figura 5: Algoritmo de navegación.

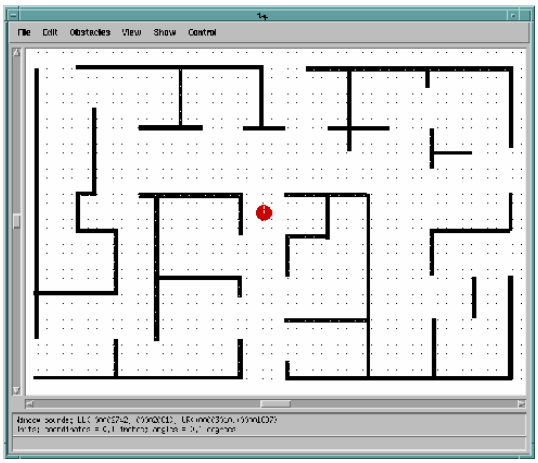

Figura 6: Simulador y entorno propuesto.

4.1 Simulación

Para llevar a cabo la simulación se programa el robot en lenguaje ANSI C, bajo una plataforma IBM UNIX AIX, usando una librería de instrucciones especiales para controlar el robot. Por medio del simulador se pueden crear entornos, en donde se puede probar el robot, y en forma gráfica observar las acciones que el robot realiza. Para las primeras pruebas se propuso un entorno que simula un ambiente de oficina, donde existen paredes y puertas, las cuales se muestran desde la parte superior (ver figura 6).

Una vez que se tiene el entorno en el simulador y el software se procede al entrenamiento y navegación, descritos anteriormente, para finalmente encontrar la meta desde cualquier punto inicial (v.g., ver figura 7).

El proceso de aprendizaje se realizó para diferentes en-

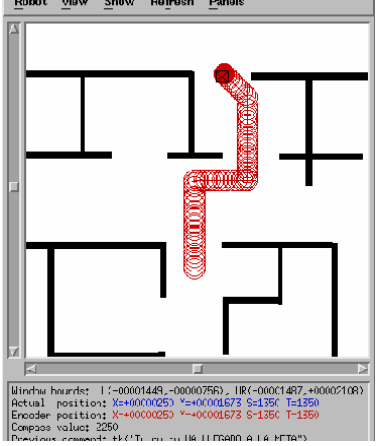

Figura 7: Robot llegando a la meta.

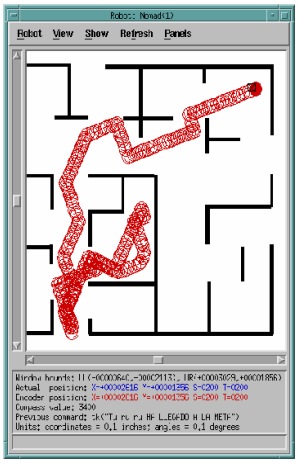

Figura 8: Ejemplo de navegación saliendo de trampas.

tornos y en diferentes posiciones, creando un total de 263 reglas. El proceso de aprendizaje consistió básicamente en la memorización de todos los posibles casos con los que el robot se pudiera encontrar.

4.2 Detectar y salir de trampas

Las reglas pueden caer en trampas al tomar decisiones en base a información local. Para evitar las trampas se diseñó un algoritmo el cual es capaz de salir de trampas sencillas (se pueden diseñar trampas rebuscadas en las cuales el algoritmo se cicla). En todas las pruebas realizadas el algoritmo de detección y salida de trampas funcionó adecuadamente.

El algoritmo detecta una trampa cuando regresa al mismo lugar (con cierto valor de tolerancia) después de una serie de acciones. Una vez detectada una trampa, se crea una meta virtual a un lado de la meta original. El sistema, entonces trata de alcanzar esa meta. Después de N acciones se regresa la meta original. Esta estrategia sencilla resultó adecuada para las pruebas realizadas (Figura 8).

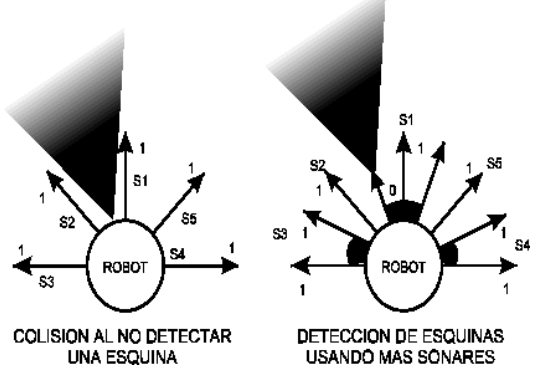

Figura 9: Detección de esquinas.

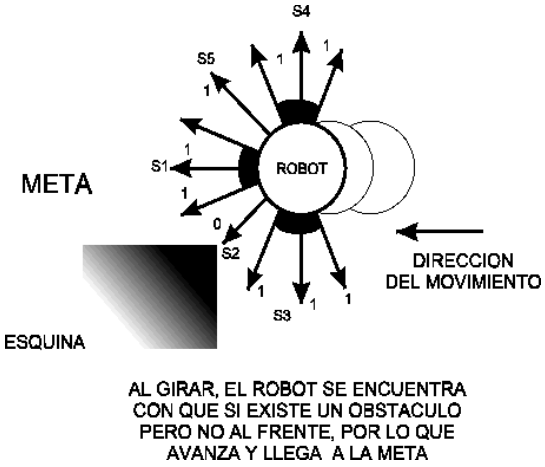

Figura 10: Utilización de más sonares.

4.3 Navegación en ambientes reales

El algoritmo funciona adecuadamente en ambientes simulados, inclusive para salir de trampas. Sin embargo, en ambientes reales, los valores de los sonares son poco confiables por lo cual no se puede depender de la lectura de uno solo.

En algunos casos, cuando el robot se encontraba una esquina, no la detectaba, ocasionando una colisión. Otro problema que se presentó en el robot, fue cuando después de pasar por una pared y llegar a la esquina, trataba de girar hacia la izquierda para llegar a la meta, cuando ya no detectaba obstáculos. Una vez que realizaba el giro detecta que existía un obstáculo y no podía llegar a la meta. Ambos problemas se resolvieron utilizando otros 2 sensores, uno para la parte izquierda y otra para la derecha, del sensor original y quedandose con la menor lectura de los tres. Esto permitio darle más robustez al sistema sin necesidad de cambiar el algoritmo o las reglas propuestas (ver figura 9 y 10).

5 Resultados

El algoritmo generó 263 reglas con las cuales es capaz de navegar para ir de un lugar inicial a uno final en ambi-

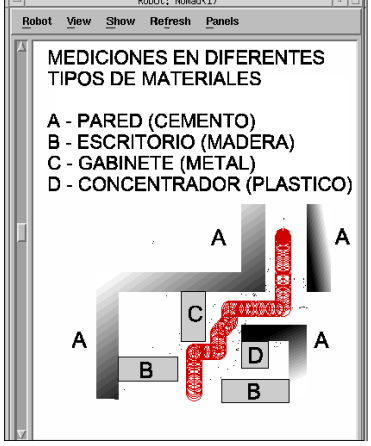

Figura 11: Obstáculos en el laboratorio.

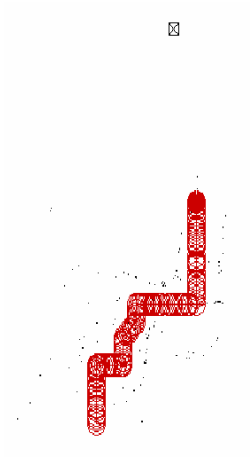

Figura 12: Trayectoria del robot.

entes desconocidos, tanto en pruebas simuladas como en pruebas reales. Dado que el sistema no cuenta con algoritmos de localización, en ambientes reales se hicieron pruebas con longitudes relativamente pequeñas (no mas de 10 m.) en donde el odómetro seguía dando estimaciones adecuadas.

Otra de las pruebas realizadas con el robot real, fue el tratar de observar las mediciones que realizan los sonares sobre diferentes tipos de materiales. Para ello se colocó el robot en el centro del laboratorio, se le asignó una meta en el pasillo que se encuentra a la entrada del mismo. Para lograr la meta propuesta el robot tenía que pasar cerca de escritorios, gabinetes y paredes de diferentes materiales y se observaría el comportamiento del algoritmo ante los diferentes obstáculos que encontraría.

Los resultados fueron buenos, ya que el robot logró detectar estos obstáculos y llegar a la meta propuesta. Sin embargo en algunas ocasiones, llegó a tener colisiones con algunos de estos obstáculos y no logró llegar a la meta (Figura 11 y 12).

Al algoritmo se le agregó un módulo, el cual, con la

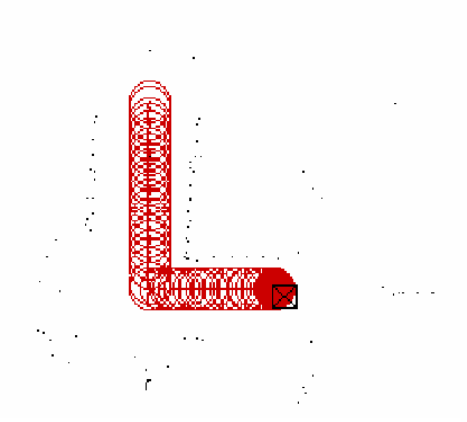

Figura 13: Resultados con valor de umbral bajo.

ayuda del simulador, se encarga de dibujar puntos en la pantalla, los cuales nos indican la posición aproximada de los objetos que se encuentran alrededor del robot. Por medio de estos puntos es posible reconstruir el entorno por donde se mueve el robot, y sirvió para observar las mediciones que realiza el robot real en el entorno donde se mueve (Figura 12).

Al desarrollar el algoritmo, se propuso un valor de umbral de aproximadamente 20 pulgadas (50.8 cm.), por lo que si las mediciones realizadas por los sonares son inferiores al umbral, decimos que el objeto esta cerca, en caso contrario se considera que está lejos.

Cuando se utilizó el simulador se inició con un valor de umbral de 20 pulgadas pero se redujo a 15 pulgadas (38.1 cm.) ya que daba mejores resultados, al pasar por lugares más estrechos sin llegar a tener una colisión. Cuando se utilizó el robot real con este nivel de umbral, resultó que ciertos objetos que se encontraban a una distancia que no ocasionaban una colisión con el robot, sucedía que ocasionaba que el robot las detectara como un objeto, y se detuviera, por ejemplo, como pasar la puerta del laboratorio después de recorrer el pasillo de entrada. Para que el robot pudiera pasar por la puerta del laboratorio fue necesario reducir este nivel de umbral a 10 pulgadas (25.4 cm) y realizar nuevamente la prueba (ver figura 13).

El valor que se le asigna al umbral es un parámetro ajustable, que no afecta la sintáxis de las reglas aprendidas.

Las reglas generadas se alimentaron posteriormente a un sistema de aprendizaje (CN2) buscando lograr una reducción de ellas. CN2 produjo 45 reglas de las cuales algunas de ellas eran no determinísticas. Después de una revisión manual de las reglas producidas por CN2, se construyó el conjunto final de 160 reglas.

El enfoque propuesto y posteriormente su implantación dieron buenos resultados ya que se logró obtener 0% de colisiones en la simulación y el robot lo-

gró llegar a la meta en 90% de las pruebas realizadas. Cuando se agregaron trampas al robot, éste logra salir de ellas y llegar a la meta en 85% de las veces. En la parte de experimentación con el robot real se logró llegar en 75% de las ocasiones a la meta.

6 Conclusiones y Trabajo Futuro

En este trabajo de investigación podemos resaltar los puntos más importantes como son los siguientes:

1. Se propuso un algoritmo de adquisición de reglas para navegación por demostración. Éste resultó ser bastante eficiente, simplificando grandemente la programación del mecanismo de control del robot.
2. Se propuso un esquema de representación del entorno relativamente simple y sin embargo suficientemente poderoso, logrando satisfacer los propósitos de esta investigación.
3. El valor del umbral para la detección de los obstáculos, es un parámetro que puede ajustarse independiente de las reglas aprendidas.
4. Se implantó un esquema de combinación de información de 3 sonares logrando evitar algunos problemas de detección de esquinas y bordes, sin alterar la representación propuesta.
5. Se propuso e implantó un algoritmo de detección de trampas, basado en la creación de metas virtuales, dando buenos resultados en las pruebas realizadas.
6. Se redujeron las reglas originales mediante un proceso de generalización, sin afectar el desempeño del algoritmo.
7. Las reglas aprendidas en un ambiente de entrenamiento, sirvieron para navegar en ambientes simulados parecidos, con distintas ubicaciones de metas y "trampas".
8. El sistema se probó con el robot real, logrando salir y entrar por puertas estrechas, y navegar en pasillos sin colisiones.
9. Se observó que las reglas aprendidas sirven para evitar obstáculos, navegar en ambientes dinámicos no predecibles y metas en movimiento, pero aún se requiere más trabajo de investigación.
10. Se desarrolló un algoritmo para dibujar el entorno del robot, con el fin de detectar lo que el robot real estaba percibiendo y ajustar el umbral. Sin embargo, se vió la viabilidad de este algoritmo para la construcción de mapas.

A pesar de los buenos resultados obtenidos, existen muchas líneas de investigación para desarrollos futuros, en particular los que se mencionan a continuación:

1. Agregar otro tipo de sensores para mejorar la adquisición de información del entorno, por ejemplo, láser, infrarrojo, cámara de video, etc.
2. Mejorar el algoritmo para lograr una mejor ubicación del robot y construir mapas a partir de la navegación.
3. Mejorar el sistema de aprendizaje, agregando más acciones de control para mejorar los movimientos del robot.
4. Caracterizar el algoritmo de trampas y su posible mejora.
5. Ajustar dinámicamente la velocidad de desplazamiento y el valor del umbral.

Referencias

[Augustun and Allen,] K. Augustun and A. Allen. Journeys to the past. *Designnews, VOLUME =*.

[Barr and Feigenbaum, 1986] A. Barr and E.A. Feigenbaum. *Handbook of Artificial Intelligence*. Addison-Wesley, 1986.

[Begley, 1997] A. Begley. Greetings from Mars. *Newsweek*, 1997.

[Brooks, 1986] R.A. Brooks. A Robust Layered Control System for a Mobile Robot. *IEEE transaction on Robotics and Automation*, pages 14–23, 1986.

[Brooks, 1997] R.A. Brooks. Attila-II. Technical Report http://www.fzi.de/divisions/ipt/WMC/preface/node26.html, IS Robotics Corporation, Westlake Village, California, 1997.

[Friedrich *et al.*, 1996] H. Friedrich, S. Munch, and R. Dillmann. Robot Programming by Demonstration (RPD). *Machine Learning*, 1996.

[Fu and González, 1987] K.S. Fu and C.S. González. *Robótica: Control, Visión e Inteligencia*. McGraw Hill, 1987.

[Morales, 1996] E. Morales. On Learning How to Play. In H.J. van den Herik and J.W.H. M. Uiterwijk, editors, *Advanced in Computer Chess*. Universiteit Maastricht, The Netherlands, PAGES =, 1996.

[Nomadic-Technologies, 1996] Nomadic-Technologies. User's Manual. Technical Report Nomad 200, Nomadic Technologies Inc., Mountain View, CA, 1996.

[Sutton and Barto, 1998] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, Massachusetts, 1998.