

**UNIVERSIDAD DE CARABOBO
AREA DE POSTGRADO
CURSO INTELIGENCIA ARTIFICIAL**

**IMPLEMENTACION DE UN PARADIGMA HEBBIANO – APRENDIZAJE POR
REFUERZO EN EL COMPORTAMIENTO DE ESQUIVAR OBSTÁCULOS DE UN
VEHÍCULO ROBOTICO**

Elaborado por : Ing. Ricardo Alonso Carbonell, C.I. 5.379.770

Abril 2006

SUMARIO

Los robots vehiculares autónomos son una clase de robots que requieren operar en un medio no estructurado y no invariable en el tiempo, lo cual implica que paradigmas de programación rígidos y estrictos no son los adecuados para la implementación de este tipo de máquinas. Otros paradigmas parecen más prometedores, pero en este trabajo se evalúa la adecuación de un paradigma combinado de no supervisado del tipo Hebbiano y el de aprendizaje por refuerzo, a fin de programar dicho agente a fin de adaptar el comportamiento del mismo a la tarea de evitar obstáculos en un mundo inestructurado.

Los resultados obtenidos indican que dicha implementación puede hacerse de manera satisfactoria y sencilla, fin de utilizar este paradigma en agentes vehiculares compactos con el concurso de hardware relativamente poco poderoso, lo cual repercute de manera positiva en la concepción económica y simple de dicho tipo de máquinas.

1. Objetivos generales y específicos

El objetivo general del trabajo es el de desarrollar un agente inteligente en la forma de un robot vehicular similar al kepera, el cual sea capaz de adaptar un comportamiento de evitar obstáculos a un medio ambiente no estructurado y no invariable en el tiempo, haciendo uso de un paradigma combinado de aprendizaje Hebbiano y por Reforzamiento.

Como objetivos específicos tendremos :

- a. Desarrollar el modelo de red neural que se utilizara en el robot real
- b. Desarrollar las ecuaciones de Aprendizaje Hebbiano que reflejen la política de ajuste de pesos de la red determinada en el objetivo a.
- c. Desarrollar el o los criterios usados en el aprendizaje por Refuerzo.
- d. Implementar el programa definitivo del agente y probar su adecuación a un entorno de mediano grado de complejidad (una arena de pruebas).
- e. Implementar el programa del agente en una maquina real desarrollada por el autor para la investigación en el campo de robots basados en comportamientos.
- f. Validar los resultados obtenidos en el simulador en el robot real dentro de una arena similar a la usada en el objetivo d.

2. Descripción del problema

Se dispone de un vehículo experimental dotado de dos motores del tipo on - off (prendido – apagado), controlado por medio de un circuito integrado que le da a los motores la capacidad de gira en los sentidos de avance y retroceso. Como sensores, el robot dispone de dos sensores del tipo Infrarrojo, con alcance de 5 – 10 cm, los cuales le dan capacidad de detección sensorial a media distancia al aparato. Además de este dispositivo sensorial, existen dos sensores del tipo bumper que le informan al vehículo cuando este ha topado con un obstáculo sólido.

Hasta ahora, se ha programado en el vehículo un modelo basado en 3 o 4 comportamientos preprogramados, los cuales interactúan con ellos mismos de acuerdo al paradigma de subsunción.

Uno de los comportamientos programados es el llamado AVOID (evitar) el cual se acciona cuando los sensores infrarrojos detectan un obstáculo y hacen que los motores del hardware se detengan y accionen de manera coordinada a fin de sortear el obstáculo; de allí el nombre del comportamiento.

Sin embargo, existen muchos tipos de programar este comportamiento. Uno de ellos se basa en parar el avance del motor de lado opuesto al sensor IR que detecta el obstáculo, alejando su rumbo de avance de dicho obstáculo. Este comportamiento tiene la desventaja de que cuando el sistema pasa una abertura entre dos obstáculos vecinos, el robot suele quedarse estático. Se ve sometido a un dilema similar al que

enfrento Odiseo cuando tuvo que llevar su barco entre Scila y Caribdis durante su regreso a su patria.

A diferencia de Odiseo, el cual buscò un equilibrio entre ambos obstáculos, nuestro programa muestra una tendencia no deseable y nada inteligente de quedarse estático ante la disyuntiva.

Una solución es adoptar ante ambos casos que el robot gire de manera dextrógira o levógira, hasta que uno de los dos sensores deje de detectar el objeto. Sin embargo, esta solución evita que el robot pueda lograr la alternativa de pasar de un ambiente a otro al penetrar por cosas así como una puerta o una tubería (caos de robots patrulleros o robots que inspecciones conducciones o canalizaciones industriales).

Por esta razón se desea lograr en nuestro autómatas un comportamiento mas adecuado al planteado por programación secuencial, el cual haga que el mismo se adapte en lo más posible a las situaciones diversas que se le puedan presentar al mismo en la ejecución de recorridos reales. A la vez, que permita al vehículo compensar cualquier deficiencia sensorial generada en los sensores IR y/o el medio ambiente circundante.

3. **Análisis del problema**

El problema básicamente es el conseguir una función que permita al agente descrito el tomar la una decisión optima. La decisión es la política de uso de los motores izquierdo y derecho (avance o retroceso). Para tomar la decisión, el robot necesita sentir el estado del entorno en cuanto a las variables relevantes para toma de decisión. En nuestro caso, el estado sensorial se refiere al estado de activación o no de los sensores Infrarrojo derecho e izquierdo.

Dicha función es hasta ahora, una serie de instrucciones IF ... THEN en un microprocesador ATOM28. Sin embargo, ya en el pasado se ha configurado en el mismo robot un comportamiento AVOID fundamentado en una estructura la cual ya ha dado buenos resultados en nuestro caso. Dicha estructura es el de las Redes Neuronales.

En esa oportunidad, se hizo un mapa de entrada – salida basado en una tabla de decisión. La red neuronal fue adiestrada para reproducir la tabla, haciendo uso del paradigma de adiestramiento supervisado retropropagación.

El problema de esta solución es que requiere de esfuerzo de diseño humano para prever todas las posibles contingencias y situaciones operacionales a las cuales se debe enfrentar el robot inmerso en un entorno no estructurado y no invariable en el tiempo.

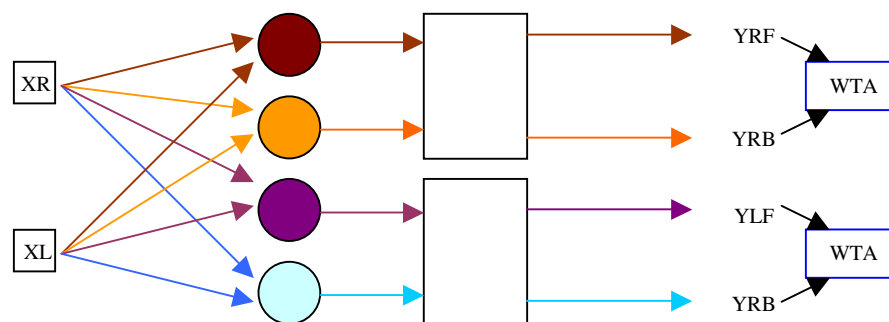
4. Determinación de métodos y herramientas a utilizar.

Por las razones antes expuestas se evalúa la adopción de reemplazar este esfuerzo con otro paradigma neuronal diferente al usado; el paradigma de entrenamiento no supervisado, mas específicamente el adiestramiento Hebbiano no supervisado, el cual ajusta los pesos de la red neural de forma tal que incrementa al máximo la ocurrencia del evento de una salida postsinaptica deseada, ante la ocurrencia de un determinado evento presinaptico en la red. Dicho procedimiento comparte con el algoritmo de retropropagación la existencia de un parámetro de velocidad de aprendizaje, cuya selección es crítica; valores muy alto de este parámetro no garantizan la estabilidad de la convergencia de la solución de adiestramiento; valores muy bajos, hacen lenta la convergencia.

Por esta razón, se decide combinar el algoritmo Hebbiano con otro paradigma: el de adiestramiento por refuerzo o por premio; solo se hará adiestramiento cuando el comportamiento AVOID no realice de manera satisfactoria su función de evitar obstáculos. El criterio de éxito es que el número de veces que el algoritmo Hebbiano se dispara sea lo menor posible, lo cual es consistente con el algoritmo de aprendizaje por refuerzo. Además de esto, es posible seleccionar velocidades de aprendizaje relativamente elevados (5 al 10%).

Así, el modelo básico de adiestramiento Hebbiano será el siguiente :

Para una red neural de dos entradas y 4 salidas; dos por cada motor siendo una salida la que acciona el avance del motor y la otra su retroceso. Solo una de las dos salidas estará activa (1) y la otra inactiva (0), lo cual se garantiza por la adopción de una función de transferencia llamada "Winner Takes All". Vea en el siguiente diagrama la descripción de la topología de la red neural adoptada



El sistema tiene un total de 8 pesos, los cuales deben ser adiestrados usando el paradigma Hebbiano. Para ello usaremos la siguiente ecuación

$$W = W + \alpha * (1 - Y) * X - \gamma*(1-Y)*W \quad (1)$$

La ecuación esta en la forma matricial. Las matrices Y y X son las salidas pre y postsinápticas, respectivamente, de la red.

$$X = \begin{pmatrix} XR \\ XL \end{pmatrix}$$

$$Y = \begin{pmatrix} YRF \\ YRB \\ YLF \\ YLB \end{pmatrix}$$

Debe hacerse resaltar que los pesos serán modificados si y solo si un elemento de Y es uno cuando las entradas también valen 1. Esto tiene sentido cuando consideramos que es indeseable que el motor del robot este en avance cuando un obstáculo esta presente. Para el retroceso, se mantiene la política de adiestramiento por consistencia de calculo, aunque las pruebas realizadas demuestran que no hay un efecto sensible al adoptar la política opuesta.

Además de esto, se observa un ultimo termino, llamado termino de olvido, caracterizado por un parámetro gamma. Este termino evita que el peso crezca de manera indefinida (posiblemente hasta el infinito). Dicho parámetro gamma debe ser la décima parte de alfa, a fin de garantizar que W no crezca mas allá de 10/-10.

Los valores de alfa y gamma se seleccionan por la ejecución de algunos ensayos en el simulador de Matlab, encontrándose que los mejores valores son :

$$\alpha = 0.01$$

$$\gamma = 0.001$$

Por supuesto, la ecuación (1) será ejecutada solamente cuando los bumper del robot se activen, indicando asi que una colisión ha ocurrido (o que el comportamiento AVOID no ha sido satisfactorio).

5. Desarrollo de la solución.

En una primera fase, se prueba el modelo en un simulador denominado SIMROBOT, el cual esta elaborado enteramente en Matlab y el cual es un toolbox desarrollado por El Departamento de Control, Medicion e Instrumentación de la Facultad de Ingenieria Electrica y Computación de la Universidad Brno de la Republica de Checa.

El programa, escrito en lenguaje M es el siguiente :

```
function new = Luigi(simrobot,matrix,step)
```

```
global W B Y X ciclo d jump tiempo
global motor_derecho motor_izquierdo
```

```
a=0.2;
gamma = 0.01;
```

```
% sensor reading
```

```
[DFL,num] = readusonic(simrobot,'sens1',matrix);
[DFR,num] = readusonic(simrobot,'sens2',matrix);
```

```
% Calculo red neuronal
```

```
neto = W*X + B;
```

```
if neto(1,1) >= neto(2,1)
```

```
    neto(2,1) = -1;
```

```
    neto(1,1) = 1;
```

```
else
```

```
    neto(1,1) = -1;
```

```
    neto(2,1) = 1;
```

```
end
```

```
if neto(3,1) > neto(4,1)
```

```
    neto(4,1) = -1;
```

```
    neto(3,1) = 1;
```

```
else
```

```
    neto(3,1) = -1;
```

```
    neto(4,1) = 1;
```

```
end
```

```
Y = hardlim(neto);
```

```
% Calculo acciones de control en el driver
```

```
YRF = Y(1,1);
```

```
YRB = Y(2,1);
```

```
YLF = Y(3,1);
```

```
YLB = Y(4,1);
```

```
motor_derecho = 0.5* sign(YRF - YRB);
```

```
motor_izquierdo = 0.5* sign(YLF - YLB);
```

```
if iscrashed(simrobot)
```

```

for i=1:4
    for j=1:2
        W(i,j) = W(i,j) + a*(1 - Y(i,1))*X(j,1) - gamma*(1-Y(i,1))*W(i,j);
    end
end
load gong
sound(y,Fs)
halt
end

```

```

% Lectura sensores

```

```

if DFR < d
    X(1,1) = 1;
else
    X(1,1) = 0;
end

```

```

if DFL < d
    X(2,1) = 1;
else
    X(2,1) = 0;
end

```

```

% Reporte de los resultados y movimientos del robot
a
[W B]
X'
Y'
simrobot = setvel(simrobot,[motor_izquierdo motor_derecho ]);

```

```

end

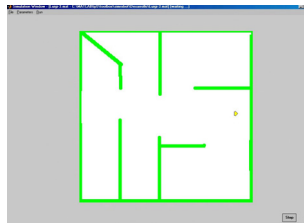
```

```

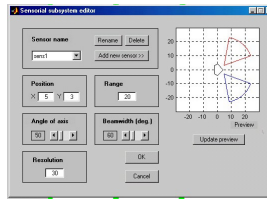
new = simrobot;

```

El robot fue simulado en una arena como la siguiente :



Con un robot con las siguientes especificaciones sensoriales :



Para empezar la simulación, se uso el siguiente conjunto de pesos obtenidos al azar :

1.8005	1.5652
-1.0754	1.0484
0.42737	-0.17413
-0.05607	-1.926

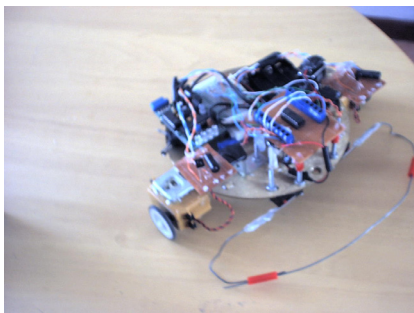
Con bias de 0, -1, 0 y -1 en cada neurona

Después de un adiestramiento por el algoritmo adoptado, se logro un comportamiento satisfactorio del robot con el siguiente conjunto de pesos

1.8005	1.5652
0.6277	2.625
0.42737	-0.17413
1.4675	0.17443

Posteriormente, fue construido un robot real, dotado con los siguientes equipo :

- Dos motores de tipo servo modificados de alto torque y 9 v
- Dos sensores infrarrojos para una frecuencia de 40KHz (5 cm de alcance)
- Dos bumper
- Un microcontrolador ATOM 28 (Microbasic) con 20KB de capacidad RAM, 256 B de capacidad Flash, programable en dialecto Basic.



Vista del robot mostrando sus sensores IR, su bumper y su microcontrolador. Además se ve el driver de los dos motores, los cuales se ven como cajas amarillas (foto del autor).

El programa elaborado para el mismo fue el siguiente

```
' Programa de Robot controlado por redes neurales
' El Robot solo tendra como sensores los dos interruptores de posicion delan
' teros
' ubicados a la derecha e izquierda de la linea central del robot.
' La logica controlara el encendido en directo o reversa de los motores dere
' chos e izquierdo del robot

' Pesos iniciales del Robot escritos en codigo byte

Data 204,194,82,172,146,120,125,46

clear

' PROGRAMA PRINCIPAL

input 2          ' P2 es la entrada del IR derecho
input 3          ' P3 es la entrada del IR izquierdo
input 10
input 11

' Configuramos las salidas de los motores para ir hacia adelante

high 4 'Salida avance motor derecho
Low 5 'Salida retroceso motor derecho
high 6 'Salida avance motor izquierdo
Low 7 'Salida retroceso motor izquierdo

' Declaramos las variables de calculo de la red

XD var long      'Valor estado sensor IR derecho
XI var long      'Valor estado sensor IR izquierdo
BD var long      'Valor del estado del bumper derecho
BI var long      'Valor del estado del bumper izquierdo
YFD var long     'Salida motor derecho hacia adelante
YBD var long     'Salida motor derecho hacia atras
YFI var long     'Salida motor izquierdo hacia adelante
YBI var long     'Salida motor izquierdo hacia atras
IRD var bit      'Estatus del sensor infrarojo derecho
IRL var bit      'Estatus del sensor infrarojo izquierdo
W1 var long
W2 var long
W3 var long
W4 var long
W5 var long
W6 var long

W7 var long
W8 var long
B1 var long
B2 var long
B3 var long
B4 var long

tempo1 var byte
tempo2 var byte
tempo3 var byte
tempo4 var byte
tempo5 var byte
tempo6 var byte
tempo7 var byte
tempo8 var byte
inicio var long
cociente var long
iniciol var long
cocientel var long

alfa var long
```

```

gamma var long
'   Asigna valores iniciales a los parametros de la red neural
B1      = 5.00
B2      = 4.00
B3      = 5.00
B4      = 4.00
alfa    = 0.01
gamma   = 0.001
inicio  = fneg 3.00
inicio1 = 3.00
cociente = 0.023529412
cociente1 = 42.5
'   Se cargan en memoria los valores de inicio de la matriz de pesos de la memoria FLASH

read 0,tempo1
read 1,tempo2
read 2,tempo3
read 3,tempo4
read 4,tempo5
read 5,tempo6
read 6,tempo7
read 7,tempo8

'   Se llevan los pesos en codigo Byte ( 0- 255 ) al rango -3 a 3

W1 = (cociente fmul float tempo1) fadd inicio
W2 = (cociente fmul float tempo2) fadd inicio
W3 = (cociente fmul float tempo3) fadd inicio
W4 = (cociente fmul float tempo4) fadd inicio
W5 = (cociente fmul float tempo5) fadd inicio
W6 = (cociente fmul float tempo6) fadd inicio
W7 = (cociente fmul float tempo7) fadd inicio
W8 = (cociente fmul float tempo8) fadd inicio

ciclo:
'   Se transmiten los valores reales de los pesos en modo serial
SEROUT S_OUT, i9600, [real W1," ", real W2, " ", real W3, " ",real W4, " ", real W5,
" ", real W6, " ", real W7, " ", real W8,13]

gosub vigilancia ' Para detectar el estatus de los sensores IR

'   Se calculan las salidas de la red neural.
'   Se compara que salida de cada motor es la mas alta y se aplica el WTA

YFD = ((W1 fmul XD) fadd (W2 fmul XI)) fadd B1
YBD = ((W3 fmul XD) fadd (W4 fmul XI)) fadd B2
YFI = ((W5 fmul XD) fadd (W6 fmul XI)) fadd B3
YBI = ((W7 fmul XD) fadd (W8 fmul XI)) fadd B4

If YFD >= YBD then
    high 4
    low 5
    YFD = 1.00
    YBD = 0.00
Else
    low 4
    high 5
    YFD = 0.00
    YBD = 1.00
Endif

If YFI >= YBI then
    high 6
    low 7
    YFI = 1.00

```

```

        YBI = 0.00
    Else
        low 6
        high 7
        YFI = 0.00
        YBI = 1.00
    Endif

'Lectura del estado del bumper

    BD = IN10
    BI = IN11

    if BD = 0 or BI = 0 then gosub actualizacion

goto ciclo          ' Retorno al inicio para otra consulta

vigilancia:

    freqout 8,1,25000
    IRD = IN2
    If IRD Then
        XD = 0.0
    Else
        XD = 1.0
    Endif
    freqout 9,1,25000
    IRL = IN3
    If IRL then
        XI = 0.0
    Else
        XI = 1.0
    Endif

return

actualizacion:

' Me detengo unos segundos para actualizar

    low 4
    low 5
    low 6
    low 7

' Actualizacion de los pesos

    W1 = (W1 fadd ((alfa fmul (float 1 fsub YFD)) fmul XD)) fsub ((gamma fmul (float 1
fsub YFD)) fmul W1)
    W2 = (W2 fadd ((alfa fmul (float 1 fsub YFD)) fmul XI)) fsub ((gamma fmul (float 1
fsub YFD)) fmul W2)
    W3 = (W3 fadd ((alfa fmul (float 1 fsub YBD)) fmul XD)) fsub ((gamma fmul (float 1
fsub YBD)) fmul W3)
    W4 = (W4 fadd ((alfa fmul (float 1 fsub YBD)) fmul XI)) fsub ((gamma fmul (float 1
fsub YBD)) fmul W4)
    W5 = (W5 fadd ((alfa fmul (float 1 fsub YFI)) fmul XD)) fsub ((gamma fmul (float 1
fsub YFI)) fmul W5)
    W6 = (W6 fadd ((alfa fmul (float 1 fsub YFI)) fmul XI)) fsub ((gamma fmul (float 1
fsub YFI)) fmul W6)
    W7 = (W7 fadd ((alfa fmul (float 1 fsub YBI)) fmul XD)) fsub ((gamma fmul (float 1
fsub YBI)) fmul W7)
    W8 = (W8 fadd ((alfa fmul (float 1 fsub YBI)) fmul XI)) fsub ((gamma fmul (float 1
fsub YBI)) fmul W8)

'
    Se transforman los pesos del rango -3 a 3 al rango byte de 0 - 255

    tempo1 = int ((W1 fadd inicio1) fmul cocientel)
    tempo2 = int ((W2 fadd inicio1) fmul cocientel)
    tempo3 = int ((W3 fadd inicio1) fmul cocientel)
    tempo4 = int ((W4 fadd inicio1) fmul cocientel)
    tempo5 = int ((W5 fadd inicio1) fmul cocientel)

```

```

tempo6 = int ((W6 fadd inicio1) fmul cociente1)
tempo7 = int ((W7 fadd inicio1) fmul cociente1)
tempo8 = int ((W8 fadd inicio1) fmul cociente1)

'
Se describen los pesos en la memoria FLASH

write 0,tempo1
write 1,tempo2
write 2,tempo3
write 3,tempo4
write 4,tempo5
write 5,tempo6
write 6,tempo7
write 7,tempo8

' Y ahora retocedo 6 segundos

low 4
high 5
low 6
high 7

pause 6000

return

```

6. Pruebas

Las pruebas efectuadas consistieron en :

- Ensayo a nivel del simulador del mejor modelo matemático a adoptar en el robot. En concreto, se comprobó el esquema de comportamiento Hebbiano puro versus el comportamiento Hebbiano puro controlado por el aprendizaje por refuerzo. Se pudo comprobar que para evitar problemas de estabilidad de aprendizaje, el modelo Hebbiano puro requería de velocidades de adiestramiento extremadamente bajas, lo cual causa que el robot tenga problemas de lentitud de adaptación ante situaciones nuevas. Inclusive, se pudo observar que al llegar a un mínimo local, la inestabilidad era tal (aun con velocidades de aprendizajes tan bajas como 0.1%) que el robot lo que hacía era dar vueltas en círculo o chocar con las paredes del obstáculo, alternativamente.

En cambio, la adopción del aprendizaje Hebbiano – por Refuerzo hace que la inestabilidad no sea problema aun cuando se adoptan valores de aprendizaje mas altos (1 – 5%), ya que el ajuste de los pesos se hace solo cuando sea necesario por la ocurrencia de un evento de choque.

- La segundo fase fue la de pruebas del robot a nivel real. Se sometio a un entorno normal como el que se conseguiria en una casa o edificio de oficinas. La programación adopto el algoritmo identificado en el simulador.

En este caso, se pudo observar un comportamiento mejor que en el simulador, ya que obstáculos que no pudieron ser superados por el agente simulado, si lo fueron por el agente real. Un obstáculo resaltante fue el formado por una

esquina en angulo agudo, la cual fue superada por el agente mediante el expediente de cambiar momentáneamente su set de pesos para sortear con un giro levógiro de casi 180°. Esto significo que todos los obstáculos eran sorteados con giros levógiros de ahí en adelante. Sin embargo, uego, al chocar con otros obstáculos que no podian ser sorteados de este modo, el robot cambiaba su set de pesos por un esquema de giros levógiros y dextrogiros adecuados a la detección de obstáculos con el sensor IR derecho, o izquierdo, respectivamente. Se asume como explicación que existe algun tipo de degradación sensorial tipica del IR que permite ese comportamiento característico, por cuanto el simulador adopto el esquema de un sensor ultrasónico, el cual tiene un comportamiento menos difuso que el IR. Sin embargo, se pudo determinar que hubieron otros obstáculos, no detectables con el IR que afectaban el adiestramiento del robot y su comportamiento

7. Conclusiones

Como resultado de las pruebas realizadas hasta ahora en el adiestramiento del robot y comportamiento subsecuente, se puede concluir lo siguiente:

- La combinación de ambos esquemas permite una codificación compacta y sencilla de implementar en un hardware compacto y economico.
- Favorece la tendencia a que los vehículos robóticos vayan a esquemas de tamaño cada vez mas pequeños.
- La degradación sensorial contribuye positivamente en el comportamiento de un robot con este esquema de control, ya que causa un comportamiento mucho mas adecuado dada lo difuso que era la información suministrada por ellos.
- Sin embargo, dicha limitación causa problemas con obstáculos no detectables, debido a la baja reflectividad IR de los mismos. En estos casos, el robot recibe el refuerzo de que no es correcto avanzar cuando no se detectan obstáculos lo cual produce comportamientos completamente no adecuados.
- Se debe combinar varios tipos de sensores (fusion temporal) para cubrir una amplia gama de espectros de emisión radiante, sin que se pierda la ventaja que la información sensorial difusa. Esto puede ser la combinación de ultrasonicos con IR.

8. Bibliografía

- Basic Micro , User`s Guide for Atom28.
- AMRT, AUTONOMOUS MOBILE ROBOTICS TOOLBOX User's manual.
- **Eden, Tim / Knittel, Anthony / Uffe, Raphael Van**, REINFORCEMENT LEARNING (2001), www.cse.unsw.edu.au/~cs9417ml/RL1.
- **Fleifel Tapia, farid**, ROBOTS AUTÓNOMOS Y APRENDIZAJE POR REFUERZO, www.fleifel.net/ia/robotsyaprendizaje.php

- **Kaelbling, Leslie Pack**, ADAPTIVE INTELLIGENT MOBILE ROBOTS (1999), people.csail.mit.edu/lpk/mars/kickoff99.ppt
- **Harmon, Mance & Harmon, Stephanie**, REINFORCEMENT LEARNING : A TUTORIAL (2000),
<http://www.nbu.bg/cogs/events/2000/Readings/Petrov/rltutorial.pdf>