Desplazamiento de un hexápodo utilizando técnicas de aprendizaje por refuerzo

Matías Ariel Scremin[1,3], Juan Carlos Gómez[1,2]; Claudio Abel Verrastro[1,3]; Pablo Folino[3]; Sergio Alberino[3]; Juan Alarcon[1,3]; Elias Daponte[1,3]
[1]UTN FRBA GIAR; [2]INTI; [3]CNEA
scremin@cae.cnea.gov.ar ; juanca@inti.gob.ar ; cverra@cae.cnea.gov.ar

***Resumen*:**
El presente trabajo aborda la programación del modo de desplazamiento de un robot con una sola extremidad con tres articulaciones, por medio de técnicas de aprendizaje por refuerzo. En esta primera etapa se parte de un modelo simplificado implementado en un entorno simulación con el objeto generar la base del aprendizaje que, con posterioridad, se utilizará en primera instancia para modelizar un hexápodo de 6 extremidades y 18 articulación, y posteriormente sobre un modelo físico.
Se modela la mecánica utilizando un programa que recrea las condiciones físicas del entorno y del robot, el cual permite desde otro entorno de programación comandar el autómata y leer de ciertos parámetros físicos por medio de sensores.
Como una primera etapa se plantea un modelo simplificado del robot con el fin de hallar desplazamientos con distintas técnicas de aprendizaje, adquiriendo una experiencia previa para abordar el trabajo sobre el hexápodo.

***Palabras clave:* RL, Hexapodo,autómata,SARSA(λ), learning**

I. INTRODUCCIÓN

El continuo avance tecnológico de las últimas décadas, se ve reflejado en el área de la robótica, buscando principalmente agentes que sean capaces de realizar tareas de mayor complejidad, para su programación se requiere de herramientas cada vez más sofisticadas.

Este desarrollo tecnológico mejoró aspectos prácticos y cotidianos en la vida de las personas, impulsó la necesidad de un manejo de mayor cantidad de información, creando nuevas aéreas de investigación, para la exploración y procesamiento de grandes volúmenes de información en menor tiempo.

La programación de sistemas multivariable como robots de múltiples grados de libertad presentan una complejidad tal, que para su abordaje con las metodologías convencionales presenta inconvenientes difíciles de resolver, derivados de la variación de condiciones de trabajo, del ambiente en que se mueve, precisión de los sensores, un conocimiento inexacto del modelo dinámico del mismo etc. El aprendizaje por refuerzo (RL) [1], es una técnica inspirada en la psicología conductista, donde cada acción tiene asociado un premio o castigo, y un peso o calificación sobre la información que recolecta el autómata interactuando en su ambiente [1]. Se logra así generar un comportamiento del agente que mejora las recompensas recibidas, considerando además, estrategias de exploración en la toma de decisiones. La técnica de RL ofrece una alternativa alentadora para la solución a este tipo de problemas. En este trabajo se la aplica para programar el modo de caminar de un robot [2].

La expansión combinatoria del espacio de búsqueda de soluciones, es una doble amenaza para los sistemas de aprendizaje, por un lado, los recursos computacionales crecen exponencialmente con la cantidad de estados y acciones. Por otro, el método de aprendizaje requiere la exploración reiterada de esos estados y acciones. El estado actual de la técnica de RL ofrece alternativas para atacar este problema, incluyendo los casos de espacios de estados continuos [3]. Mediante RL se puede caracterizar un problema en un mundo totalmente desconocido y resolverlo por medio de reglas y formas de adquirir información acerca de las consecuencias de una acción elegida, indicando cuan apropiada resulto.

Este trabajo unifica dos de los conceptos mencionados, por un lado algoritmos de aprendizaje para realizar una tarea, por medio del concepto de RL, y por otro un mecanismo de desplazamiento, con el objetivo de construir un autómata que aprenda los movimientos para desplazarse de una manera eficaz y explorar nuevos modos sin necesidad de un algoritmo que describa exactamente sus acciones.En cuanto a la geometría del autómata, se adoptó un robot de 6 patas que forman un hexágono, con la posibilidad de dirigirse en cualquier dirección sin la necesidad de girar, con el movimiento de un par de piernas opuestas. Además esta configuración permite modos de superación de obstáculos que están vedadas a robots con ruedas.

En la sección II se presenta el entorno de desarrollo, sus características y su empleo en el modelo de robot elegido. En la III se describe el robot con las

simplificaciones realizadas para comenzar el desarrollo. En la IV se describe el algoritmo utilizado. En la V los resultados obtenidos. En la VI se exponen las conclusiones y finalmente en la VII, se discuten alternativas para la continuación de este desarrollo.

II. ENTORNO DE DESARROLLO

Para la simulación de un ambiente y el estudio del comportamiento del robot de acuerdo a sus criterios físicos de construcción: forma, servos utilizados, material, peso, fuerzas, se requiere de un entorno de simulación confiable y versátil, por experiencias previas con resultados positivos se opto por el uso de un software de simulación de robots llamado "Anykode Marilou" [http://www.anykode.com], el cual permite la construcción de modelos físicos, generar perturbaciones y medir respuestas.

Este software posee elementos básicos para la creación de un prototipo, motivo por el cual, la geometría del autómata se proyectó con un programa de diseño, logrando figuras complejas, muy similares al robot (ver Fig. 1). Cuenta con la posibilidad de conectarle actuadores (motores y servo motores), definir distintos tipos de comportamiento en sus articulaciones y asociarles dispositivos de sensado de distintas magnitudes, recreando un contexto similar al pretendido en la realidad. Dicha simulación sin embargo, no es controlada desde el programa sino desde otro a través del protocolo de comunicación TCP/IP, dando una ventaja de poder crear algoritmos en el lenguaje se adapte mejor al problema. En este caso se utilizo algoritmos programados en C#, desde donde se comandan los actuadores y se leen los sensores.

III. EL ROBOT

Este robot posee 3 articulaciones en cada una de las 6 extremidades, con un servo motor en cada una, totalizando 18 grados de libertad, permitiendo la estabilidad del robot, en superficies irregulares sin la necesidad de tener todas sus extremidades apoyadas, facilitando su desplazamiento. El modelo puede observarse en la Figura 1.

Para esta primera etapa de desarrollo se realizó una simplificación con el objeto de ganar experiencia en los aspectos concernientes al manejo del entorno de simulación y su control empleando técnicas de RL. Se comenzó con un modelo de una sola extremidad que arrastra un carro con ruedas. Se busca que el autómata aprenda a realizar movimientos para desplazarse en una determinada dirección. Dicho modelo se puede observar en la Figura 1, este modelo tiene parámetros físicos teóricos de los materiales que componen el robot, tales como, peso, tamaño, fricción, densidad, etc.

Por medio de un algoritmo implementado en C# se establece comunicación con el software que simula el robot y su entorno, enviando comandos de movimientos sobre las articulaciones de las extremidades, y recibiendo los datos de distintos sensores para determinar su desplazamiento.

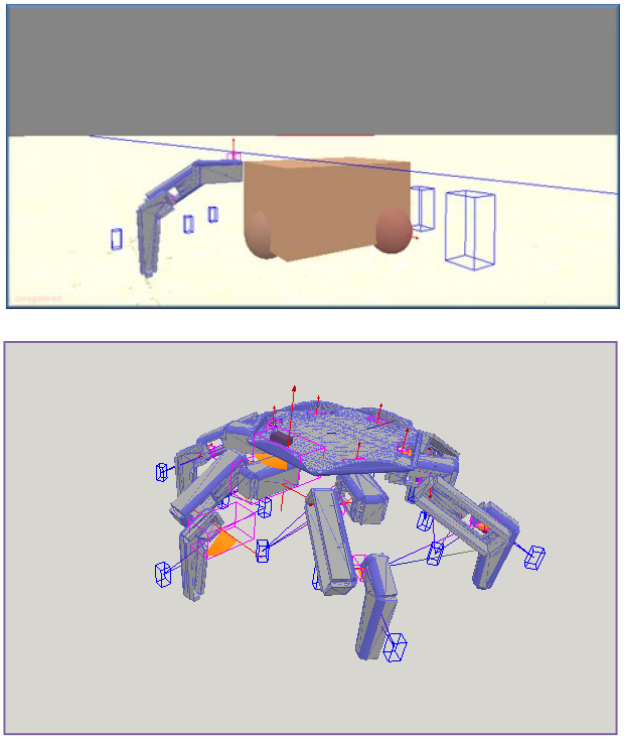

Figura 1. **Modelización del robot con el software Marilou, por un lado el modelo final con sus 18 articulaciones por el otro el modelo simplificado con una sola extremidad**

IV. EL ALGORITMO

El algoritmo en grandes rasgos consiste en calificar las acciones de un agente, mediante la observación y estipulando criterios de éxito, buscando que el agente logre cumplir un objetivo como desplazarse y al mismo tiempo maximizar una variable (consumo, velocidad, área de exploración, etc.).A partir de esto, se asocia para cada acción una recompensa, que consiste en un valor numérico que puede ser interpretado como un castigo o un premio para la acción tomada (A) a partir de un estado previo (S) y alcanzando uno nuevo (S´).

El par estado-acción conforma una matriz, la cual es actualizada bajo cierto criterio, cada vez que sus índices cambian. El modelo acción-estado puede ir variando en su complejidad, comenzando por una discretización o utilizando funciones [5], que determinen un espacio continuo a través de distintas técnicas.

Dentro del RL existen diferentes métodos, uno de ellos es el denominado SARSA (λ) State-Action-Reward-State-Action, siendo uno de los métodos de diferencia temporal (TD), el cual fue elegido en este trabajo, siendo métodos básicos para la estimación de funciones de valor .Estas funciones determinan valores de los pares (S, A), donde se evalúa una instancia de uno o más pasos adelante, siendo en particular este el caso donde λ es 0 (TD[0]).

En un algoritmo de este tipo, existen ciertas características, que diferencian el aprendizaje. Una de ellas es la cantidad de pasos se toman en cuenta para actualizar la función de valor, al cual se lo llama backup y otra referida a la cantidad de estados que se actualizan, se denominada simple Backup cuando se actualizan una cantidad limitada de estados previos y full Backup cuando se actualizan todos los estados previos.

Mediante estos dos conceptos, la recompensa obtenida por una acción tomada se refleja sobre el valor de la función recompensa del estado-acción *Q(s,a)*, de formas diferentes por medio de coeficientes, los cuales influyen en la actualización de *Q(s,a)*, de acuerdo a cuantos pasos previos a este estado se actualizan, y con qué frecuencia es visitado, con el fin de lograr un aprendizaje exploratorio y que no produzca oscilación , iterando siempre entre las mismas decisiones por parte del autómata.

Otro papel importante es la política de exploración y de toma de decisiones la cual implementa distintos criterios para elegir la siguiente acción, se puede utilizar una función totalmente aleatoria, u otra conocida como "greedy" la cual siempre toma el camino con mayor recompensa, una posibilidad más completa es una combinación de estas dos denominada "ε-greedy", con una parte aleatoria y otra "greedy".

En este trabajo se utilizo la "ε -greedy", probando distintos rangos aleatorios, lo cual permite controlar la expansión de la búsqueda.

Las ecuaciones (1) y (2) describen este modelo simple, existen otras ecuaciones donde se actualiza de acuerdo a la esperanza del suceso, es decir, la estimación del camino tomado a partir de esa acción, en este caso solo se tiene en cuenta el siguiente par *Q(s´,a´)* en el paso t + 1, mientras que denominamos al par Q[s:a] al estado y acción actual.

Tenemos un factor de aprendizaje (α), que es un numero entre 0 y 1, que pondera la actualización de la matriz, cuanto más pequeño dicho valor menos peso tiene el nuevo valor que se suma. Para nuestro caso utilizamos valores pequeños del orden de los 0.3 a 0.1, para evitar una actualización brusca del valor de *Q(s:a)*, permitiendo que el aprendizaje sea relativamente lento.

Por otro lado está el factor γ (entre 0 y 1) que indica el peso que el valor del siguiente paso a seguir tiene sobre el actual, dando un valor de γ pequeño la diferencia en la ecuación disminuye dando un valor menor al término.

$$Q'(s,a) = Q(s,a) + \propto \delta(s,a) \quad (1)$$

$$\delta(s,a) = r + \gamma\, Q(s',a') - Q(s,a) \quad (2)$$

r : recompensa.

Q´(s,a): Valor a ser actualizado.

Q(s,a): Valor actual.

Q(s´,a´): Valor del siguiente paso a visitar.

$\propto$: Factor de aprendizaje.

Una vez detallado esto, la secuencia de código es la siguiente.

```
Estado inicial;
Elijo primera acción (A0);

Repetir (CANTIDAD DE PASOS)
          {
          Ejecuto A0
          Averiguo nuevo estado S1
          Elijo nueva acción A1
          Calculo recompensa de la acción A0
          Actualizo Q(S0:A0) en función de y S1, A1

          }
```

Cada conjunto de código es un "Episodio", donde se va ejecutando y actualizando la matriz con parámetros fijos, una vez terminado esto, el algoritmo modifica los valores de aprendizaje y corre nuevamente el Episodio, pero continuando con los valores obtenidos en la matriz Q(s,a), que dejo el episodio anterior.

Cuando se ejecutó una cantidad de episodios, se calculan indicadores que evalúan la eficacia del algoritmo, en este caso el indicador comparativo es el desplazamiento en una dirección dado un número fijo de pasos, comparado con una secuencia óptima generada por un programador. También podemos analizar una curva de aprendizaje en función a los resultados de cada episodio, donde variamos los parámetros de aprendizaje.

V. RESULTADOS

Existen dos parámetros de repetición de algoritmo, uno es la cantidad de pasos que conforman un episodio y por el otro la cantidad que de veces que se repite, por cada episodio se pueden definir condiciones de aprendizaje diferentes, adoptando como criterio episodios de 500 pasos, y ejecutar un total de 40 episodios. Esto posee 2 etapas diferenciadas, la primera de exploración con un ε del orden del 90%, luego disminuido exponencialmente

en episodios sucesivos para dar lugar a la etapa de aprendizaje.

Para el modelo simplificado la variable sensible a mejorar es la velocidad, la cual se mide tomando en cuenta cuanto pudo desplazarse el autómata en cada episodio.

VI. CONCLUSIONES

Como podemos observar en la Figura 2, el autómata comienza con una etapa aleatoria donde incrementa su desplazamiento, con un poco influencia de aprendizaje, luego de 7 episodios ε, se reduce al orden del 10%, tomando importancia las acciones cuya recompensa fueron mayores, llegando a un valor estacionario muy próximo al desplazamiento logrado por el algoritmo de referencia. Debemos tener en cuenta que si bien esta diferencia es mínima para una mayor cantidad de episodios podría disminuir un poco más. En la Figura 3 se representa la frecuencia de visita para cada par estado-acción, este histograma resalta aquellos que pertenecen a la secuencia que logra el mejor desplazamiento.

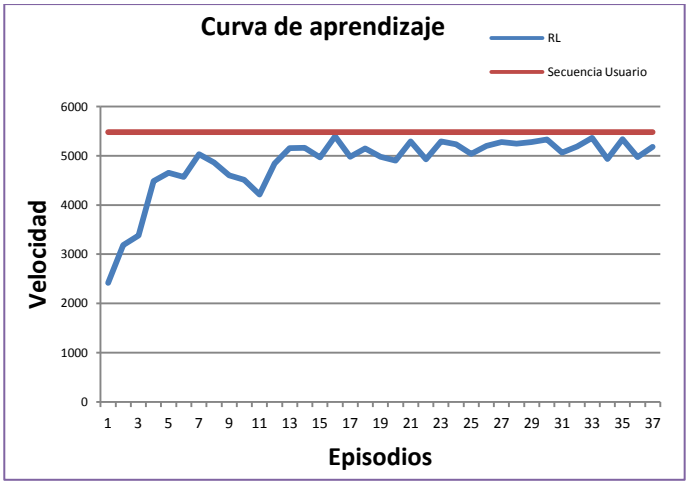

Figura 2-. **Grafico comparativo sobre la influencia del factor ε en una política ε – greedy.**

VII. TRABAJOS FUTUROS

Como se detalló en el comienzo del trabajo, esta modelización simple es el paso necesario para impulsar los algoritmos de aprendizaje para el hexápodo, si bien este trabajo obtiene resultados similares a una programación estándar, donde para este modelo simple el aprendizaje no sería necesario, lo importante es que esta metodología puede aplicarse en un modelo futuro con muchas variables.

Sin embargo para evitar la expansión exponencial del numero de estado-acción, para el caso de los 18 servomotores, con posiciones de forma continua entre 0° y 90°, se cuantificaran los estados utilizando el modelo "*Tile Coding*" [3], combinando con un control difuso [4].

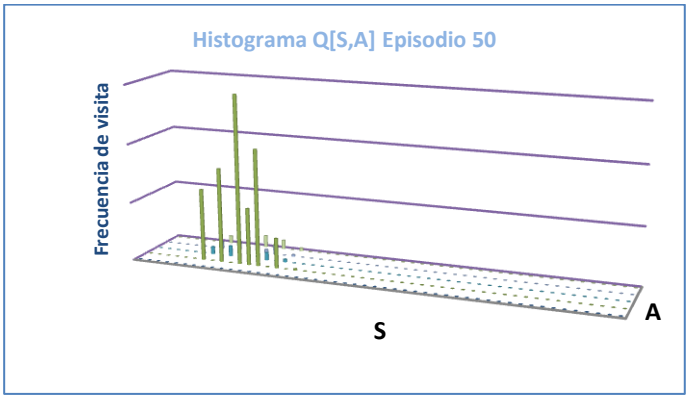

Figura 3-. **Frecuencia con la que se visitan los pares estado-acción en el episodio N°50 tomando una política e-greedy.**

REFERENCIAS

[1] L. Kaelbling, M. Littman y A. Moore, «Reinforcement Learning: A Survey,» de *Journal of Artificial Intelligence Research 4*, 1996.

[2] J. Kober, J. A. Bagnell y J. Peters, «Reinforcement Learning in Robotics: A Survey,» *International Journal of Robotics Research,* vol. 32, nº 11, pp. 1238-1274, 2013 July.

[3] R. Sutton y A. Barto, Reinforcement Learning: An Introduction, Cambridge, MA: MIT Press, 1998.

[4] J. C. Gómez, *Fuzzy Control,* Buenos Aires: edUTecNe - Editorial Universitaria de la Universidad Tecnológica Nacional, 2008.

[5] R. y. D. Busoniu, «Reinforcement learning and dynamic programming using function approximators,» *FL: CRC Press, Boca de ratón,* 2010.