APRENDIZAJE POR REFUERZO EN ROBOTS AUTONOMOS PARA EL PROBLEMA DE SEGUIMIENTO DE OBJETIVOS MOVILES Y SU APLICACION EN FORMACIONES

Tesista

Diego Ariel Bendersky

Director

Juan Miguel Santos

DEPARTAMENTO DE COMPUTACION
FACULTAD DE CIENCIAS EXACTAS Y NATURALES
UNIVERSIDAD DE BUENOS AIRES

Indice general

Indice de tablas

Indice de figuras

Resumen

La tarea de seguimiento de objetivos móviles (*target following*) consiste en detectar un objeto que se mueve con una trayectoria desconocida y mantener acotada la distancia al mismo a lo largo del tiempo. En este trabajo estudiamos este problema en el contexto de robots autónomos con sensado local. La complejidad de la tarea depende en gran medida de la capacidad de sensado del robot: en nuestro caso, trabajamos con robots dotados únicamente de sensores infrarrojos de muy corto alcance, lo que hace que el problema sea difícil de resolver. Analizamos tres técnicas para la síntesis de comportamientos y vimos que todas son difíciles de aplicar a la tarea de seguimiento con estas restricciones en el sensado: el grado de error de los sensores es muy grande y depende de variables externas, no es sencillo generar un modelo del sistema a partir de los valores de los sensores, el espacio de situaciones y acciones es demasiado grande y su exploración se ve dificultada por el reducido alcance de los sensores y, por último, la cantidad de acciones buenas asociadas a cada situación es muy baja.

Para solucionar este problema, presentamos un método de síntesis de comportamientos que nos permite aplicar técnicas de aprendizaje por refuerzo y obtener comportamientos en tiempos de aprendizaje muy cortos. Este método está basado en: 1) la división de la tarea de seguimiento a partir de la transformación del espacio de acciones, 2) el aprendizaje de las subtareas con el objetivo detenido, 3) la ejecución simultánea de las acciones de ambos subcomportamientos a partir de su combinación en una única acción y 4) la superposición de los ciclos de control. Utilizamos también un método de clustering basado en redes neuronales artificiales para la representación

del espacio de situaciones, acciones y utilidad (valores q), preservando el amplio rango de valores posibles de los distintos espacios.

Los comportamientos de seguimiento generados fueron aplicados con éxito en un grupo de robots para el desarrollo de formaciones de distintas configuraciones geométricas (en columna y en diagonal). Las mismas fueron obtenidas en forma emergente, sin comunicación entre los robots, sin control centralizado y sin sensores externos. Para lograr formaciones con distintas configuraciones geométricas, se modificó únicamente la función de refuerzo, sin alterar el método de aprendizaje. Esto último nos permite verificar la generalidad del método presentado y la posibilidad de utilizarlo en otras tareas relacionadas con la navegación.

The target-following behavior implies the detection of a mobile target and the maintenance of a bounded distance from it over time. In this work, we study this problem in the context of autonomous robots with local sensing. The difficulty to synthesize such a behavior depends on the sensing capabilities of the robot. In our case, we have used robots that have proximity sensors with a limited detection range. This fact turns the synthesis problem into a hard one compared to those cases where the robot has information about the absolute location of the target or about the relative location of the target but without range limit. We analyze three behavior synthesis techniques and we found that all of them are difficult to apply in this problem: the sensors are imprecise and its data varies with external variables, it is difficult to build a model of the system from the sensors data, the situation/action space is big and hard to explore, and the number of good actions associated with each state is low.

To solve this problem, we propose a behavior synthesis method based on Reinforcement Learning that allows us to obtain the required behavior in short time lapses. The method consists in 1) a task decomposition technique based on an action space transformation, 2) the learning of the subtasks in limited environments, with the target stopped, 3) the concurrent execution of the subtasks and the combination of their actions, and 4) the overlapping of the control loops. We also use a clustering technique, based on Artificial Neural Networks for the representation of the situations, actions and q-values.

The resulting behaviors were successfully applied to a group of robots to build formations with different shapes. The formations were obtained without inter-robot communication, centralized control or global sensors. In order to achieve formations with shapes other than column, we changed the reinforcement function only, leaving the learning method unmodified. This last experiment gives us a feedback over the generality of this method and the possibility to use it in the synthesis of other navigational behaviors.

Capítulo 1

Introducción

La tarea de seguimiento de objetivos móviles (*target following)* consiste en detectar un objeto que se mueve con una trayectoria desconocida y mantener acotada la distancia al mismo a lo largo del tiempo. La importancia de esta tarea en el contexto de robots autónomos radica en que los comportamientos que resuelven ésta tarea u otras similares son utilizados para la construcción de soluciones a problemas colectivos y cooperativos, como por ejemplo formaciones, observación de múltiples objetivos, recolección y manipulación de objetos.

La dificultad para obtener comportamientos de seguimiento depende en gran medida de la capacidad de sensado del robot. Si se cuenta con sensores globales (GPS, radares, cámaras fijas) o con equipamiento de comunicación entre el robot y el objetivo la resolución de este problema se ve facilitada, ya que si la posición del robot y del objetivo pueden estimarse con precisión, entonces el problema puede resolverse a partir de un modelo físico del sistema y la formulación de la trayectoria óptima para el robot. Los robots que utilizan este tipo de equipamiento, si bien pueden ser autónomos desde el punto de vista del control, no lo son respecto de los sensores, porque para su funcionamiento dependen de equipos externos, que no controlan. Muchas veces no es posible utilizar este tipo de sensores, ya sea porque requieren de una infraestructura previa (el GPS, que no puede usarse por ahora en otros planetas), una adaptación del entorno (las cámaras de video, que deben ser colocadas antes de

ejecutar las tareas) o un acuerdo previo entre los robots (la comunicación por radio, para la cuál los robots deben acordar los parámetros). Otras veces, el uso de estos sensores no es conveniente: para un grupo de robots, el uso de estos equipos hace que el sistema global sea menos robusto, ya que una falla en el sensor o en el medio de comunicación afecta a todos los robots. Los comportamientos basados exclusivamente en información de sensores propios se denominan de *sensado local*. Estos son en general más robustos (los problemas en un sensor afectan solamente al robot propietario del sensor), más escalables (no requieren cambios en el entorno si se agregan nuevos robots), y permiten el uso de robots heterogéneos, que interactúen entre sí únicamente a través del entorno, o incluso robots enemigos o con objetivos contradictorios.

En este trabajo estudiaremos el problema de seguimiento en el contexto de *robots autónomos con sensado local*, para robots miniatura dotados de sensores infrarrojos. Estos sensores son difíciles de utilizar en un número importante de tareas, en particular en la tarea de seguimiento. La dificultad más importante radica en su reducido alcance: el seguimiento debe ser muy preciso y ágil, porque de lo contrario una mala acción del robot o un movimiento brusco del objetivo harían que éste sea perdido de vista por el robot. La precisión requerida se ve dificultada por el alto error y la variabilidad que en general tienen estos sensores. Si bien el valor que devuelven depende de la distancia a la cuál se detectó el objeto, la función $valor \rightarrow distancia$ varía en forma considerable con una gran cantidad de variables externas: color y material del objeto, orientación del sensor respecto del piso, color y material del piso y características de la luz ambiente: nivel de luminosidad y tipo de fuente de luz (natural, tubos fluorescentes o lámpara). Esta función es distinta también para cada sensor, aún de la misma marca y modelo, y varía a lo largo de su vida útil. A pesar de estos inconvenientes, los sensores infrarrojos tienen también ventajas: no requieren mucha energía para su funcionamiento ni gran capacidad de cómputo, y funcionan en condiciones externas muy variadas, aún en ambientes en los cuales otros sensores no pueden ser utilizados (por ejemplo, en ambientes pobremente iluminadas).

Para este problema, y con las limitaciones de sensado mencionadas, el método

de aprendizaje por refuerzo (RL) tiene ventajas sobre otros métodos de síntesis de comportamientos, como la descripción algorítmica del comportamiento y los métodos de aprendizaje supervisado. Al no conocer la relación entre los valores de los sensores y las variables físicas (como distancia y ángulo), resulta difícil definir el comportamiento por medio de un algoritmo escrito a mano. Por otro lado, el gran tamaño del espacio de situaciones y acciones, y el alto nivel de precisión y de reacción que requiere el comportamiento, dificultan el manejo manual del robot mientras se ejecuta un algoritmo de aprendizaje supervisado. Los algoritmos RL permiten, en cambio, obtener comportamientos sin necesidad de conocer la política óptima ni la dinámica del sistema, y sin tener que generar un modelo del mismo.

Sin embargo, cuando se intenta utilizar algoritmos RL en robots autónomos, se presentan varias dificultades. Entre las más importantes figuran el gran tamaño del espacio de situaciones y de acciones, el tiempo que demora cada experimento que limita la cantidad de iteraciones del aprendizaje, y algunas características propias de los robots reales: las situaciones son imprecisas e incompletas, los refuerzos intermitentes y las acciones falibles. Además, los métodos RL fueron pensados para procesos discretos, y en el caso de los robots reales las acciones no son instantáneas ni las mediciones de los sensores simultáneas. Además de estos problemas, en el caso de seguimiento se agregan otros: la dificultad para explorar el espacio de situaciones/acciones manteniendo al objetivo dentro del área de sensado y la reducida cantidad de acciones buenas en cada situación, que se traduce en una baja proporción de refuerzos positivos durante el aprendizaje.

Para superar las limitaciones de las técnicas de RL en el problema de seguimiento, presentamos en este trabajo un método basado en 1) la *división de la tarea* de seguimiento en otras más sencillas, a partir de la transformación del espacio de acciones, 2) el aprendizaje de versiones simplificadas de las mismas en *entornos reducidos*, 3) la utilización de un algoritmo de *clustering* para representar el espacio de situaciones/acciones/utilidades y 4) la superposición de los ciclos de control. Este método permite obtener los comportamientos en tiempos de aprendizaje muy acotados.

Las acciones del robot, originalmente expresadas con una velocidad independiente para cada rueda, son representadas con una componente de velocidad lineal y una de velocidad angular. Basándonos en esta nueva representación de las acciones, una forma natural de dividir la tarea de seguimiento consiste en definir una tarea independiente para cada una de las componentes. Así, a partir de la componente velocidad angular, desarrollamos una tarea que consiste en mantener el frente del robot alineado con el objetivo (tarea *mantener-ángulo*) mediante rotaciones en el lugar. A partir de la velocidad lineal, la tarea que queda definida consiste en mantener la distancia entre el robot y el objetivo dentro de un rango predefinido (tarea *mantener-distancia*) con movimientos en dirección adelante/atrás. Para obtener un comportamiento que solucione la tarea de seguimiento, ambos subcomportamientos son ejecutados en forma simultánea y sus acciones (las velocidades angular y lineal) son combinadas y transformadas nuevamente en velocidades independientes para las ruedas izquierda y derecha.

A partir de las nuevas definiciones de tareas, los comportamientos son más simples y tienen un espacio de acciones mucho menor, continuo y más fácil de generalizar y de representar. Además, las nuevas tareas son más fáciles de cumplir, lo que permite aprenderlas en un tiempo mucho menor.

Para solucionar las dificultades en el aprendizaje producto del reducido alcance de los sensores, las subtareas son aprendidas en entornos estáticos, es decir, con el objetivo detenido. Con esta configuración del entorno, es posible mantener al objetivo dentro del área de sensado del robot durante todo el proceso de aprendizaje, que era uno de los problemas principales de aplicar RL en forma directa. Además, es posible reducir el tamaño del espacio de situaciones, ya que se requiere menor cantidad de información para poder deducir el estado actual (por ejemplo, no es necesario incluir información que nos permita estimar la velocidad del objetivo).

Sin embargo, un comportamiento aprendido con el objetivo detenido no es necesariamente el mejor si se lo utiliza con un objetivo móvil. Durante la ejecución, el resultado de las acciones será distinto que el obtenido durante el aprendizaje, y

muchas de las acciones no serán las adecuadas. Para aliviar este problema, nuestra propuesta consiste en *superponer las acciones*, iniciando el nuevo ciclo de control (sensado, seleccion de la acción, ejecución) antes de que terminen de ejecutarse la acción anterior. Así, una acción inadecuada es inmediatamente corregida por el siguiente ciclo, logrando que sólo sea ejecutada en una pequeña parte. Las acciones correctas, en cambio, son ratificadas por el siguiente ciclo, ejecutándose en forma continua hasta que cambien las condiciones del entorno.

Una vez obtenidos los comportamientos de seguimiento, nuestro siguiente propósito es utilizar los mismos para resolver, en forma emergente, tareas grupales. En este marco, nos proponemos estudiar las formaciones de robots. Esta tarea consiste en lograr que un grupo de robots alcancen una determinada configuración espacial y la mantengan mientras se traslada de un lugar a otro. Para lograr este objetivo, presentamos modificaciones a la definición de la tarea de seguimiento que nos permite generar, sin alterar el método de aprendizaje presentado en este trabajo, formaciones de diversas configuraciones geométricas, en particular fila y diagonal. La utilización del método de aprendizaje para resolver variantes de la tarea de seguimiento nos permiten comprobar que el mismo puede ser aplicado en otros problemas similares.

El resto del trabajo está organizado de la siguiente manera: el capítulo 2 presenta una introducción al tema de robots autónomos, al aprendizaje por refuerzo aplicado a problemas de robots reales y contiene además un repaso por distintas implementaciones de problemas similares al de seguimiento. El capítulo 3 muestra en detalle los problemas que presenta el uso de RL en esta aplicación en particular y brinda una descripción de la solución presentada. También se analiza aquí de qué manera se solucionan cada uno de los problemas de RL y se señalan algunas características y propiedades de los comportamientos obtenidos con el método propuesto. En el capítulo 4 se presentan los resultados experimentales para la tarea de seguimiento, implementada con robots reales siguiendo diferentes trayectorias. El capítulo 5 brinda una introducción a los sistemas multi-robot y a la tarea de formaciones y describe

la aplicación de los comportamientos de seguimiento en esta tarea, mostrando los resultados experimentales obtenidos en formaciones en fila y en diagonal. Finalmente, presentamos en el capítulo 6 las conclusiones de este trabajo y las posibles líneas de investigación que permitirían continuar este trabajo en el futuro.

Capítulo 2

La tarea de seguimiento en robots autónomos móviles

Los robots autónomos son máquinas inteligentes que interactúan con su entorno para alcanzar objetivos específicos, sin ningún tipo de intervención humana [4]. Los campos de aplicación más importantes incluyen tareas de reconocimiento, rescate y vigilancia, limpieza, traslado y manipulación de objetos [31]. El principal potencial para la utilización de robots autónomos en estas áreas radica en que los robots pueden realizar su trabajo en entornos hostiles al hombre: rescate de personas en catástrofes (ambientes radiactivos, incendios, etc.), búsqueda de minerales bajo tierra y exploración de otros planetas. Desde el punto de vista de la psicología y la biología, el campo de los robots autónomos brinda un marco ideal para estudiar aspectos relacionados con la inteligencia, el comportamiento humano y animal, y el proceso de aprendizaje. Desde el punto de vista computacional y de teoría de control, los robots autónomos plantean problemas interesantes, muchos de los cuáles surgen del hecho de que la interacción con el mundo real es más compleja que cualquier simulación: la cantidad de variables en juego es muy grande, todos los sensores existentes e imaginables (aún nuestros propios sentidos) son imprecisos, parciales y muchas veces contradictorios, los actuadores (el equipamiento que nos permite modificar nuestro entorno, como

nuestros brazos y piernas) fallan o no responden como quisiéramos. Para que un robot alcance su objetivo todas estas características deben estar contempladas en los algoritmos de control que lo comandan, exigiendo muchas veces soluciones distintas a las utilizadas para problemas de computación tradicionales.

Muchos autores comenzaron hace unos años a estudiar comportamientos en grupos de robots, y a pensar de qué forma los robots pueden colaborar entre sí para resolver una única tarea. Los grupos de robots permiten no sólo escalar en tamaño las aplicaciones, sino que brindan soluciones más eficientes, económicas, robustas y versátiles a los mismos problemas [43].

Actualmente, el campo de los sistemas multi-robot presenta una intensa actividad en el estudio y desarrollo de comportamientos sociales [25], en las arquitecturas de control distribuido [33] y en el desarrollo de técnicas de aprendizaje cooperativas [35]. Parker [34] presenta un estado del arte en el área de robots autónomos móviles distribuidos, identificando ocho áreas de estudio: inspiración en la biología, comunicación, arquitecturas, localización y exploración, transporte y manipulación de objetos, coordinación percepto-motriz, robots reconfigurables y aprendizaje.

En este trabajo estudiaremos la tarea de seguimiento de un objetivo móvil (*target-following*) [8] en el contexto de robots autónomos dotados de sensores infrarrojos. Mataric [27] incluye esta tarea (junto con otras como *dispersarse*, *evitar obstáculos* y *alcanzar una posición*) en su identificación de comportamientos elementales (*basis behaviors*) utilizables para la construcción de comportamientos cooperativos. Algunos problemas colectivos relacionados con el problema de seguimiento son: formaciones (*robot formations*) [5] [13], desplazamiento de cajas (*box pushing*) [33] y observación de múltiples objetivos (*CMOMMT, Cooperative Multi-robot Observation of Multiple Mobile Targets*) [35] [32].

2.1. Tareas relacionadas con seguimiento

En la mayor parte de los trabajos que presentan tareas colectivas relacionadas con el problema de seguimiento, se utiliza sensado global (por ejemplo GPS o sensores externos) o bien sensado distribuido y comunicación entre los robots (por ejemplo odometría y red inalámbrica).

En el trabajo de Balch y Arkin sobre formaciones de robots [5], la tarea de seguimiento es aplicable a una de las estrategias utilizadas, denominada *referencia por el vecino* (*neighbor-referenced*), en la cuál cada robot debe seguir a su vecino más cercano manteniendo ciertas propiedades espaciales en relación a éste. Se presentan dos implementaciones: en la primera cada robot mide por odometría su propia posición absoluta y la transmite a los robots que lo están siguiendo por una red de comunicación inalámbrica. En esta implementación cada robot utiliza una arquitectura *motor-schema* [3] para calcular su trayectoria en función de su propia posición y la del vecino. En la segunda implementación cada robot obtiene su propia posición y la de todos los demás con un sistema de posicionamiento global (GPS diferencial) y utiliza la arquitectura de arbitraje DAMN [36] para controlar su velocidad y orientación.

Otra aplicación de robots cooperativos vinculada a la tarea de seguimiento es la denominada *multi-target observation*. Aquí, dado un grupo de robots y un grupo de objetos móviles, el objetivo de los robots es mantener la mayor cantidad de objetos dentro del área de sensado de alguno de ellos la mayor cantidad de tiempo posible. Parker [32] presenta una solución a este problema basada en la arquitectura de comportamientos distribuidos ALLIANCE [33]. Aquí, cada robot obtiene su propia posición absoluta por medio de un sensor de tipo GPS y la posición relativa de los objetos cercanos a partir de sensores locales (cámara o sonar). Luego, cada robot realiza predicciones sobre la velocidad y trayectoria de los objetos cercanos y transmite esta información a los robots vecinos. De esta manera, cada robot cuenta con información detallada sobre una gran cantidad de objetos.

Los trabajos mencionados hasta aquí utilizan en algún momento sensado no local. Fredslund y Mataric [13] proponen en cambio el uso de sensado local para formaciones

de robots de diferentes formas y tamaños. En su trabajo, cada robot debe realizar la tarea de seguimiento, intentando mantener fijos la distancia y el ángulo que lo separa de otro robot, denominado su *amigo*. Cada robot tiene un sensor de ángulo/distancia formado por un transmisor/receptor láser y una cámara color, ambos ubicados en una pieza mecánica capaz de girar sobre un eje vertical (*panning*). Por medio de la cámara, el robot calcula el ángulo respecto del robot amigo, luego el sensor gira, centrando la imagen de la cámara y apuntando el láser, y finalmente éste mide la distancia. La entrada para el algoritmo de control está formada por los valores de distancia y ángulo. Las acciones son generadas en base a un cálculo analítico de la trayectoria óptima a partir de estos valores.

En comparación con los sensores IR, las cámaras de video requieren mucha mayor capacidad de procesamiento para analizar las imágenes, y son además muy sensibles a la luz ambiente. Los sensores láser, al ser direccionales, deben estar apuntando al objeto en forma precisa para poder funcionar, y además son mucho más grandes y costosos. En cuanto a los sensores de ultrasonido, sus características son similares a los infrarrojos, pero no pueden utilizarse para distancias cortas por el efecto de eco que producen.

2.2. Mejoramiento de la performance de Aprendizaje por Refuerzo para problemas de robots autónomos

Muchas técnicas diferentes han sido propuestas para eliminar los inconvenientes asociados con el uso de RL en tareas de robots reales. Touzet [40] divide estas técnicas en tres grandes grupos: 1) mejoramiento de la exploración, 2) reducción del espacio de situaciones/acciones y 3) simplificación de la tarea. Además, propone una cuarta categoría, en la que incluye aquellas técnicas que no influyen en el aprendizaje sino en la ejecución del comportamiento (*a-posteriori bias*).

Exploración del espacio de situaciones/acciones

La técnica denominada *counter-based exploration*, propuesta por Thrun [38], consiste en asociar un contador con cada posible situación y dirigir la exploración hacia las situaciones que tengan sus contador en valores más bajos, es decir, hacia las situaciones menos visitadas. Esto produce una exploración más uniforme de todo el espacio. Zhang y Canu [44] proponen dirigir la exploración con una medida de incertidumbre basada en la entropía de cada situación: las situaciones con entropía alta (y que serán más exploradas) son aquellas en las que todas las acciones tienen valores q similares. Millán [11] agrega durante el aprendizaje un conjunto de *reflejos*, que evitan que el robot ingrese a regiones del espacio de búsqueda que se consideran poco importantes. El método D-UPA (*dynamic update parameters algorithm*), desarrollado por Santos [37], propone definir las funciones de refuerzo a partir de parámetros que se ajustan dinámicamente durante el aprendizaje para mantener estable la cantidad de refuerzos positivos obtenidos por el robot, logrando un refinamiento sucesivo del comportamiento. Este método presenta además una solución elegante al denominado *dilema exploración/explotación*, que plantea un interrogante acerca de cómo relacionar la exploración con la ejecución efectiva de la política durante el aprendizaje de las políticas.

Respecto de la reducción del espacio de estados, dos de las estrategias más importantes son la descomposición de tareas y el uso de técnicas de clustering.

División de tareas

La estrategia de dividir un problema complejo en otros más simples es muy utilizada tanto en el contexto de RL como en la síntesis de comportamientos para robots autónomos. Para una enumeración y comparación actualizada de los métodos más comunes, remitimos al lector al trabajo realizado por Karlsson [20]. Mencionaremos aquí los métodos más populares y los más relacionados con nuestra propuesta.

En la arquitectura de subsumición [10] (*subsumption architecture*), desarrollada

por Brooks, los comportamientos son estructurados en capas, compuestas por módulos que funcionan en forma asincrónica y que se implementan mediante máquinas de estado finito. Estos suelen emplear algún tipo de información simbólica como entrada y salida, o para representar sus estados internos. Los módulos de una determinada capa pueden influir en los de las capas inferiores a través de enlaces denominados *subsumption links*, que permiten suprimir sus salidas. De esta manera se logra, en forma distribuida, un mecanismo de coordinación entre las distintas partes del sistema.

La arquitectura de *motor schemas*, desarrollada por Ronald Arkin [3], es altamente reactiva y está ligada con el comportamiento animal. Aquí también el comportamiento original es descompuesto en otros más simples, pero en este caso todos los subcomportamientos cooperan entre sí constantemente para lograr un objetivo común. Por este motivo no hay ningún algoritmo de coordinación ni selección, sino que la acción que el robot ejecuta es el resultado de la combinación de las acciones individuales de todos los subcomportamientos. La salida de cada componente es un campo de potencias (*potential fields*) [21], en la que cada situación tiene asociado un vector de fuerza que indica el movimiento que debe realizar el robot. Esos vectores usualmente se generan considerando a los estados objetivo como atractores y a los obstáculos como repulsores.

Mataric [27] propone un método que recorre el camino inverso: mediante la combinación de una serie de comportamientos básicos (*basis behaviors*), se generan, en forma emergente, comportamientos de más alto nivel. El desafío consiste aquí en descubrir los comportamientos básicos y analizar la forma de combinarlos. En el trabajo citado se propone utilizar sumas pesadas para combinar tareas complementarias que involucran desplazamientos, y el ordenamiento y la ejecución secuencial en el caso de comportamientos contradictorios. Mataric lleva esta misma estrategia al área de comportamientos cooperativos. En [25] desarrolla un árbitro que, mediante la activación de la tarea adecuada en cada momento, produce comportamientos sociales en forma emergente. Este árbitro funciona de manera autónoma en cada robot, y es desarrollado por medio de técnicas de aprendizaje por refuerzo. En cambio, Jung y Zelinsky

[17] proponen una arquitectura en la cuál para cada comportamiento se define un conjunto de precondiciones que indican en qué circunstancias el comportamiento se encuentra listo para ser ejecutado. Un algoritmo de selección elige cuál ejecutar de entre todos los comportamientos listos, en función de la importancia de cada uno y de la interacción y dependencias entre ellos. El sistema puede aprender con la experiencia cuáles son las relaciones entre las tareas y ajustar la importancia de cada una.

Para nuestro problema de seguimiento con sensado local basado en sensores infrarrojos, ninguna de estas arquitecturas puede utilizarse en forma sencilla para simplificar la tarea. No parece fácil dividir la tarea en capas, o definir subtareas desde un punto de vista funcional, como en el modelo de subsumición. Respecto del modelo de comportamientos elementales, la tarea de seguimiento es considerada por sí misma una tarea elemental. Aún de ser posible la división no queda claro cómo implementar un algoritmo de selección. Lo mismo ocurre con la arquitectura de *motor schema*: el seguimiento define de por sí un único campo de potencias que difícilmente pueda ser dividido.

Técnicas de *clustering*

Las técnicas de *clustering* son utilizadas para obtener una representación compacta del espacio de búsqueda que permita además generalizar el conocimiento asociado con situaciones o acciones particulares. Las redes neuronales artificiales son herramientas muy poderosas para esto, debido a sus propiedades emergentes de memorización, agrupamiento y generalización [16].

Lin introduce una serie de algoritmos RL que utilizan redes basadas en perceptrones multicapa con aprendizaje supervisado para la representación de las situaciones [24]. Cada acción posible tiene asociada una red, donde la entrada es la situación y la salida el valor q asociado con la acción. La salida deseada es 1 para la red de la acción que produjo un refuerzo positivo y 0 para todas las demás. Touzet [39] expande este

esquema con un algoritmo que permite representar el espacio completo de situaciones/acciones con una única red, permitiendo la generalización sobre las acciones. En ese mismo trabajo presenta también una solución basada en Redes de Kohonen [22]. Aquí, la misma red representa, en forma uniforme, el espacio de situaciones, acciones y valores q. Las redes de Kohonen tienen la ventaja de que los clusters preservan las relaciones espaciales de las situaciones que representan, una característica deseable para la representación de espacios continuos. La red es utilizada como memoria asociativa, ya que se selecciona el cluster ganador a partir de tuplas parciales (por ejemplo situación/valor de utilidad para obtener la mejor acción, y situación/acción para obtener la utilidad). En la implementación de Touzet, la forma de la grilla de neuronas es rectangular, y de tamaño fijo.

Otras técnicas de clustering utilizadas para representar los espacios de búsqueda en problemas de RL son las redes CMAC (*cerebellar model articulation*) [1] y RBF (*radial basis functions*) [29]. Estas dos técnicas permiten, en forma no supervisada, obtener representaciones de espacios continuos. Kretchmar y Anderson presentan una comparación de ambas técnicas para el problema clásico de RL *car on the hill* [23].

Dada la dificultad de calcular de antemano la arquitectura de la red o los parámetros del algoritmo de clustering, algunos autores han implementado técnicas basadas en redes neuronales que modifican su arquitectura (cantidad de neuronas, conexiones, etc.) durante el aprendizaje. Anderson presenta una técnica basada en RBF en la cuál las neuronas menos útiles son reinicializadas si se cumplen ciertas condiciones [2]. Matuk y Santos [28] presentan un método que además de adaptar el tamaño de la red en función del espacio a cubrir, soluciona el problema denominado *clustering aliasing*. Este problema se produce cuando el algoritmo asocia en un mismo cluster situaciones que requerirían acciones distintas, o pares situación/acción con diferentes valores de refuerzo. El método está basado en un modelo de mapas topológicos desarrollado por Fritzke [14] en el cuál la cantidad de neuronas no es fija, sino que crece según las entradas recibidas (los pares situación/acción visitados). Las neuronas que presentan un nivel alto de *clustering aliasing* son divididas, mejorando la representación del

espacio de búsqueda y la performance de los comportamientos.

Capítulo 3

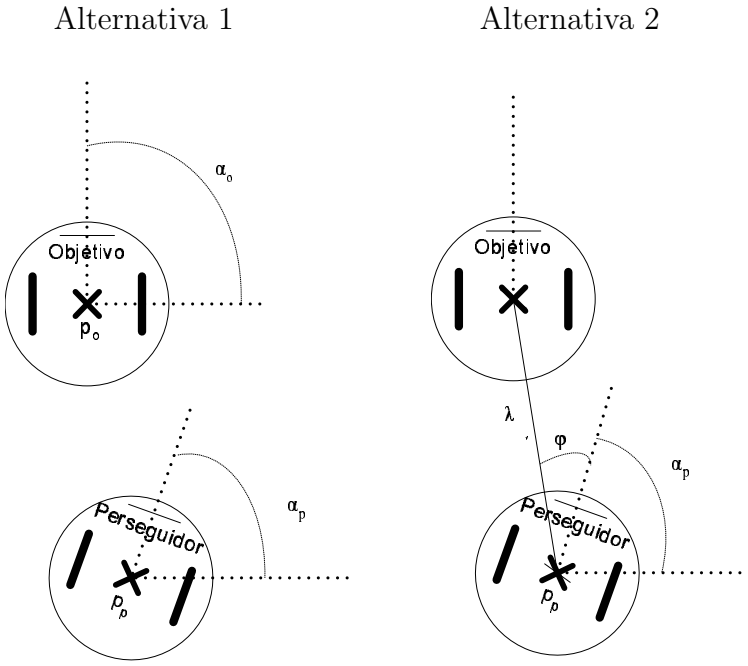
Problemas en la síntesis de seguimiento y solución propuesta

3.1. Definición de la tarea

En esta sección, daremos una descripción del sistema, y describiremos las características de los robots y de su interacción con el entorno. También daremos una definición formal de la tarea de seguimiento.

Descripción del sistema

El sistema que estudiaremos en este trabajo está formado por un área de dos dimensiones y dos robots: el robot objetivo (o simplemente objetivo) y el robot perseguidor. Ambos robots son de cuerpo circular y pueden ser descriptos, en cada momento, por su posición en el plano, su orientación y su velocidad vectorial instantánea. El estado de todo nuestro sistema puede entonces determinarse unívocamente a partir de estas variables (ver tabla 3.1 y figura 3.1). Alternativamente, la posición del robot objetivo puede describirse como diferencia respecto del robot perseguidor. En el entorno no hay ningún otro objeto (el entorno es libre de obstáculos).



p_p, p_o: posición
α_p, α_o: orientación
θ: diferencia de orientación
λ: distancia

Figura 3.1: Variables de estado del sistema para el problema de seguimiento.

Alternativa 1

p_o, p_p	posición en el plano de los robot objetivo y perseguidor $(p = (x, y) \in \Re^2)$
α_o, α_p	orientación de los robots respecto del plano horizontal $(-\pi \leq \alpha \leq \pi)$
v_o, v_p	velocidad instantánea de los robots $(v = (\rho, \phi), \rho \geq 0, -\pi \leq \phi \leq \pi)$

Alternativa 2

p_p	posición en el plano del robot perseguidor
α_p	orientación del robot perseguidor
λ	distancia entre los dos robots $(\lambda \geq 0)$
ϕ	ángulo entre el frente del robot perseguidor y el objetivo $(-\pi \leq \phi \leq \pi)$
v_o, v_p	velocidad instantánea de los dos robots

Tabla 3.1: Variables de estado del sistema para el problema de seguimiento.

Definición de seguimiento

A partir de esta descripción del estado del sistema, definiremos formalmente la tarea de seguimiento mediante el siguiente predicado:

$$seguir(t) \Leftrightarrow d - \epsilon_d \leq \lambda(t) \leq d + \epsilon_d \wedge -\epsilon_\phi \leq \phi(t) \leq \epsilon_\phi. \tag{3.1}$$

Dicho en palabras, el robot perseguidor está siguiendo al objetivo en el instante t si la distancia que los separa es similar a una cierta distancia d, y el robot perseguidor se encuentra apuntando al objetivo. Asimismo, el robot perseguidor cumple con la tarea de seguimiento en un intervalo de tiempo $t_0..t_1$ si:

$$seguir(t_0..t_1) \Leftrightarrow seguir(t_i)\ \forall t_i, t_0 \leq t_i \leq t_1. \tag{3.2}$$

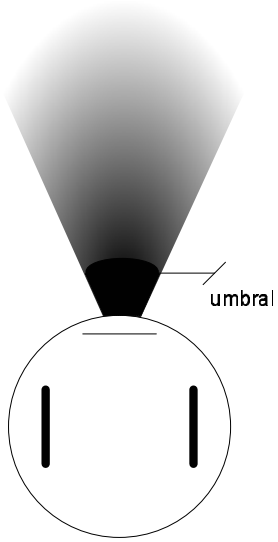


Figura 3.2: Area de sensado de un sensor infrarrojo. El color indica el valor que devuelve el sensor, de negro (valor máximo) a blanco (valor nulo).

Interacción del robot con el entorno

Los robots obtienen la información del entorno a partir de sensores infrarrojos, que permiten detectar la presencia de un objeto que se encuentra dentro de su área de sensado. Dicha área tiene forma cónica y la distancia máxima de sensado es obtenida con el objeto y el sensor alineados. Cada sensor devuelve un número entero, en función de la posición del objeto dentro del área de sensado, que es cero si no se detectó la presencia de ningún objeto y máximo si la distancia entre el objeto y el robot es menor que un determinado umbral. La figura 3.2 muestra un esquema del área de sensado de un sensor.

El robot puede medir en un instante dado el valor de todos sus sensores. La información obtenida puede representarse por un vector

$$s = (s_0, s_1, ..., s_n), 0 \leq s_i \leq s_{max}, \tag{3.3}$$

donde n es la cantidad de sensores, y s_{max} el máximo valor devuelto por los mismos.

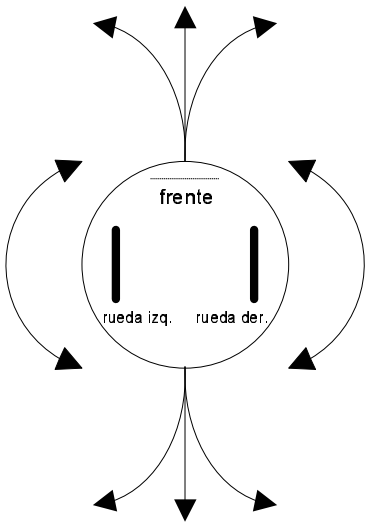


Figura 3.3: Posibles acciones del robot.

En cuanto a los actuadores (el equipamiento que les permite modificar el estado del sistema) los robots cuentan con dos ruedas. Las acciones que éstas ejecutan se pueden definir de dos maneras: mediante una velocidad y un sentido de movimiento para cada rueda, o mediante la distancia que debe recorrer cada una. De esta manera, los movimientos que pueden ejecutarse son: avanzar y retroceder, girar en el lugar y describir arcos (ver figura 3.3).

Las acciones expresadas con velocidades son asignadas a las ruedas en forma directa y ejecutadas de manera continua hasta que se le indique una nueva acción. En cambio, para las acciones expresadas como distancias, el robot define un movimiento suave, a partir de la modificación incremental de las velocidades, y detiene el robot en forma completa al concluir la acción. En principio, no es posible asignar una nueva acción hasta tanto no concluya la acción anterior. Tenemos entonces que:

$$\begin{aligned} a &= (v_{izq}, v_{der}), -v_{max} \leq v_{izq}, v_{der} \leq v_{max}, \\ a' &= (d_{izq}, d_{der}), -d_{max} \leq d_{izq}, d_{der} \leq d_{max}, \end{aligned} \tag{3.4}$$

donde v_{max} es la velocidad máxima, y d_{max} la distancia máxima de movimiento

de cada rueda.

3.2. Síntesis del comportamiento

3.2.1. Análisis de los métodos más utilizados

El aprendizaje por refuerzo parte de una idea atractiva: lograr que el robot aprenda un comportamiento interactuando con su entorno, ayudado por una función de refuerzo, que le otorga *premios y castigos* en función de las acciones que éste ejecuta. De esta manera la información que se le brinda al robot es mínima, permitiendo que el robot descubra por sí mismo el mejor comportamiento, sin verse influenciado por *prejuicios* del diseñador respecto de cómo resolver el problema. Además, suele ser mucho más fácil definir una tarea con una función de refuerzo que mediante la especificación de la política óptima. Por ejemplo, evitar obstáculos puede describirse en forma intuitiva con una función que penalice las colisiones, pero es muy difícil saber de antemano cuál es la mejor acción en cada situación (por ejemplo: ¿Es conveniente girar de inmediato o es preferible avanzar un poco más y girar más tarde? Si hay un obstáculo adelante, ¿es mejor girar a la derecha o a la izquierda?).

En el problema de seguimiento es posible calcular analíticamente una política que produzca una trayectoria óptima a partir de las variables de estado del sistema y de acciones expresadas en términos de esas mismas variables. Sin embargo, el problema está dado por la dificultad de traducir las situaciones y acciones del robot en variables de estado.

Veremos ahora un detalle de los inconvenientes que surgen al intentar utilizar otros métodos de síntesis de comportamientos. Estos problemas justifican la utilización de RL.

Comportamientos codificados a mano

Para que el robot pueda ejecutar una política, esta debe estar expresada en función de la situación observable por el robot y de las acciones que éste puede ejecutar. Por

lo tanto, si contamos con una política calculada a partir de las variables de estado, es necesario estimar el estado del sistema a partir de las situaciones detectadas por el robot, y convertir luego las acciones del modelo en acciones ejecutables por éste. Esto es:

$$\begin{aligned} s &\rightarrow e, \\ e &\rightarrow a_e, \\ a_e &\rightarrow a. \end{aligned} \tag{3.5}$$

donde s es la situación (en el espacio de situaciones del robot), e el estado del sistema, a_e una acción expresada a partir de las variables de estado, y a una acción expresada en el espacio de acciones del robot.

Para un conjunto de sensores infrarrojos, las funciones que transforman la situación observada en un estado del sistema no pueden deducirse analíticamente, son distintas según el robot que se utilice, dependen de la luminosidad (y, por lo tanto, de la hora del día en la que se realiza el experimento) y del color y del material que recubre los robots, entre otros factores. Estas funciones de transformación deberían ser calculadas por medio de alguna técnica de aproximación, como por ejemplo mínimos cuadrados o redes neuronales artificiales con aprendizaje supervisado [9]. Para un muestreo del espacio de situaciones, cada situación debería ser asociada con la distancia y ángulo que representa, calculando estos valores manualmente o por medio de un sensor externo. Este proceso debería repetirse para cada robot y cada sensor. Pero debido al alto grado de error de los sensores infrarrojos (y a su alta variabilidad), las variables de estado obtenidas de esta manera (especialmente aquellas que se deducen de varias mediciones de los sensores, como por ejemplo el vector de velocidad del objetivo) tendrían un error demasiado alto, incompatible con el cálculo de una trayectoria precisa.

La función que transforma acciones del modelo en acciones del robot, también tienen error, distinto en cada robot y para cada velocidad y tipo de movimiento. Este error se debe fundamentalmente a la discretización de las velocidades y al rozamiento de la superficie y depende en gran medida del código de control de velocidades

del robot, y de la función y parámetros empleados para el cambio de velocidades (aceleración y frenado).

Aprendizaje supervisado

Para utilizar técnicas de aprendizaje supervisado es necesario conocer la política óptima, al menos para un conjunto de estados. Sería el caso de una persona capaz de manejar al robot perseguidor mediante un joystick u otro dispositivo de control a distancia, de manera de lograr que persiga al objetivo para distintas trayectorias y velocidades. Aprender a controlar un robot de estas características y lograr que ejecute una política del tipo de seguimiento es una tarea manual difícil, que requiere mucho entrenamiento.

Otro problema que acompaña el aprendizaje supervisado es el de la selección de los conjuntos de entrenamiento y prueba. Para que el aprendizaje tenga éxito, estos deben representar adecuadamente el espacio de estados y no queda claro, en principio, cómo hacer esto en el problema de seguimiento (qué trayectorias usar para el entrenamiento y con qué velocidades, cómo hacer que el objetivo sea detectado en todas las direcciones posibles, etc.).

3.2.2. Dificultades para la aplicación de Aprendizaje por Refuerzo

El algoritmo Q-Learning

Antes de analizar nuestro problema, vamos a repasar brevemente los aspectos básicos del algoritmo Q-Learning [42] que es el método de aprendizaje por refuerzo que usaremos y el más empleado actualmente en problemas de robots reales.

Para la utilización de los algoritmos RL en general, un problema debe ser definido en función de un espacio de situaciones, un espacio de acciones y una función de refuerzo (ver tabla 3.2). La dinámica del sistema está dada por una función de transición T, que indica la situación que se obtiene al ejecutar una determinada acción

Configuración del problema: (S,A,r,T)
Política: $\pi : S \rightarrow A$ donde,

S	Conjunto de situaciones. $S = \{s\}$, $s = (s_0, s_1, ..., s_n)$
A	Conjunto de acciones. $A = \{a\}$, $a = (a_0, a_1, ..., a_n)$
r	Función de refuerzo. $r : SxAxS \rightarrow R$
T	Función de transición (puede ser desconocida) Estocástica. $P : S \times A \times S \rightarrow [0.,1]$ Determinística. $T : S \times A \rightarrow S$

Tabla 3.2: Configuración de un problema de Aprendizaje por Refuerzo.

en una situación dada. Esta función T puede ser probabilística si el problema es no determinístico o si la definición de la situación es incompleta (esto es, si no contiene toda la información necesaria para determinar unívocamente el resultado de una acción). A diferencia de otros algoritmos RL, para la utilización de Q-Learning no se requiere conocer la función T, ya que la dinámica del sistema se descubre durante el aprendizaje junto con la política óptima. Este es uno de los principales motivos de su uso tan difundido para problemas de robots reales. Un comportamiento o política es una función o *mapping* de situaciones en acciones, que indica la acción que se debe ejecutar en cada situación.

Sea $(s_0, s_1, s_2, ...)$ la secuencia de situaciones obtenida por la ejecución de una política a partir de la situación inicial s_0 ($s_i = T(s_{i-1}, \pi(s_{i-1}))$). La siguiente función expresa la utilidad de la política π para el estado s_0:

$$V(\pi, s_0) = \sum_{i=0}^{\infty} \gamma^i r(s_i, \pi(s_i), s_{i+1}), \tag{3.6}$$

donde $0 < \gamma < 1$ se denomina factor de descuento e indica cuánto afectan los refuerzos futuros en la utilidad del estado actual. La política óptima π^* es aquella que

maximiza la suma de la función V sobre el espacio de situaciones:

$$\pi^* = \max_{\pi} \sum_{s \in S} V(\pi, s). \tag{3.7}$$

El objetivo del aprendizaje es encontrar la política óptima. Para ello, el algoritmo Q-Learning mantiene una estimación de los valores de utilidad de la mejor política obtenida hasta el momento en todos los estados. A medida que experimenta nuevas situaciones y acciones, estos valores son ajustados y, de ser necesario, la política óptima es modificada.

Cabe destacar que una política es una función de la situación actual y, por lo tanto, no se consideran situaciones pasadas para la selección de la acción. Sin embargo, las situaciones pueden contener, de ser necesario, información del pasado. Los procesos en los cuales la probabilidad de transición es función exclusiva de la situación actual se denominan Procesos de Decisión de Markov (MDP).

Si la definición de la situación no distingue entre estados que requieren acciones diferentes se produce un efecto denominado *perceptual aliasing* [28], que degrada la performance de los comportamientos. El motivo de este efecto está relacionado muchas veces con limitaciones en los equipos de sensado, pero también puede resultar de una definición incompleta de la situación.

Como ya mencionamos en la introducción, el uso de RL para problemas de robots autónomos presenta varios desafíos. Analizaremos en detalle los más relevantes para el problema de seguimiento y cómo influye cada uno en la síntesis del comportamiento.

Gran tamaño del espacio de situaciones y de acciones

El tiempo necesario para el aprendizaje en los algoritmos RL está relacionado con el tamaño de los espacios de situaciones y acciones [19].

En el problema de seguimiento de objetivos móviles, para evitar el efecto de *perceptual aliasing*, la situación debe contener la medición actual de los sensores, mediciones anteriores (para calcular la velocidad del objetivo a partir de las diferencias en los

sensores del robot en dos instantes de tiempo) y las velocidades de las ruedas del robot perseguidor. Si eliminamos cualquiera de estos elementos, una situación no permitiría definir unívocamente la mejor acción. Supongamos, por ejemplo, que tomáramos sólo la medición actual de los sensores. Entonces, todos aquellos estados en los cuales los robots se encuentran a la misma distancia serían indistinguibles, y se les asignaría una misma acción. En la figura 3.4 vemos tres estados distintos en los cuales los robots están a la misma distancia del objetivo, pero se ve claramente que la acción que debe tomar el perseguidor es distinta en cada caso: debería avanzar, quedarse quieto o retroceder respectivamente. Si, en cambio, considerásemos como situación las mediciones actuales y anteriores de los sensores, pero sin considerar las velocidades de las ruedas del robot perseguidor, igualmente nos veríamos imposibilitados de estimar la velocidad del objetivo a partir de la diferencia de los sensores: en la figura 3.5 se ven dos estados distintos (que requerirían acciones distintas) en los cuales la diferencia entre dos mediciones de los sensores es la misma.

El espacio de situaciones que deberíamos utilizar es entonces el siguiente:

$$S = (s(t), s(t-1), v_{izq}, v_{der}), \tag{3.8}$$

donde $s = (s_0, s_1, ..., s_n)$, $0 \leq s_i \leq s_{max}$ y $-v_{max} \leq v_{izq}, v_{der} \leq v_{max}$. El tamaño total del espacio de situaciones es entonces:

$$|S| = |s| \times |s| \times 2\ v_{max} \times 2\ v_{max} = 4\ {s_{max}}^{2n} \times {v_{max}}^2. \tag{3.9}$$

Suponiendo valores bajos para n, v_{max} y s_{max} de 5, 5 y 10 respectivamente (nuestros robots manejas valores de 8, 127 y 1024 respectivamente), esto representa un tamaño de $4 \times 10^{10} \times 5^2 = 10^{12}$. A pesar de que no todas son situaciones posibles en nuestro entorno (por ejemplo las situaciones que tienen valores distintos de cero en los sensores de la izquierda y de la derecha simultáneamente nunca pueden ocurrir) y muchas situaciones son prácticamente iguales entre sí, este espacio es demasiado grande, y hace que el problema sea imposible de resolver por RL sin aplicar algún tipo de reducción del espacio.

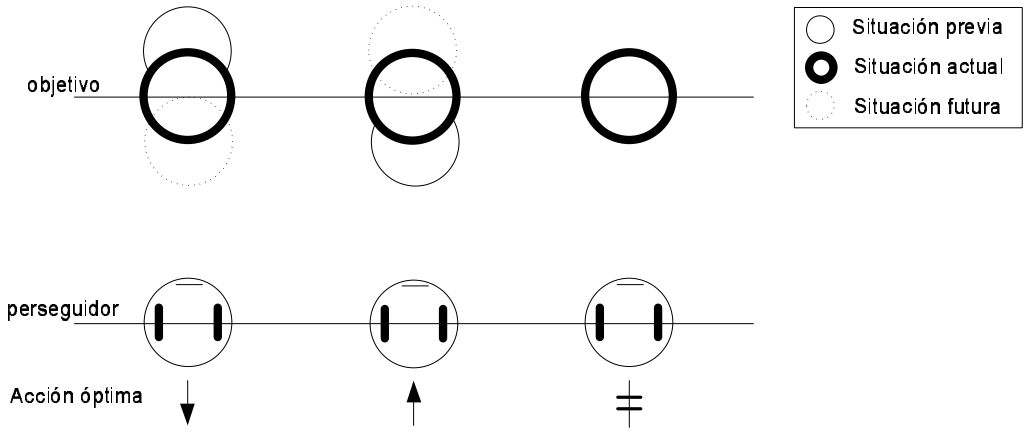


Figura 3.4: Tres estados para los cuales el valor instantáneo de los sensores es el mismo y la acción que el robot debería ejecutar es diferente.

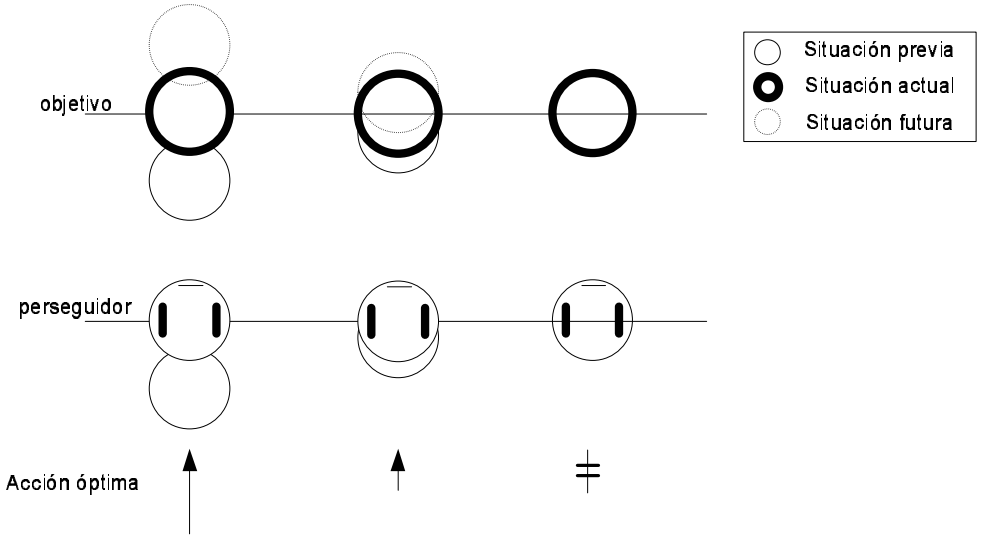


Figura 3.5: Tres estados para los cuales el valor instantáneo de los sensores y la diferencia entre dos mediciones son iguales, pero la acción que el robot debería ejecutar es diferente.

Con respecto a las acciones, para obtener libertad de movimiento y lograr trayectorias suaves vamos a considerar en principio acciones formadas por una velocidad independiente en cada rueda:

$$A = (v_{izq}, r_{der}), \tag{3.10}$$

donde v_{izq} y v_{der} indican las velocidades para la rueda izquierda y derecha respectivamente ($-v_{max} \leq v_{izq}$, $v_{der} \leq v_{max}$). Si consideramos sólo 10 velocidades distintas para las ruedas (5 hacia delante y 5 hacia atrás) la cantidad de acciones posibles asciende a 100. Este número es demasiado grande, si tenemos en cuenta que el tamaño del espacio de búsqueda crece exponencialmente con de la cantidad de acciones [40]. Además, a diferencia del espacio de situaciones, las acciones posibles no dependen de la configuración del entorno ni de la exploración, sino que, en principio, todas las acciones deberían ser probadas en cada situación.

Dificultad para mantener al objetivo dentro del área de sensado

Para que un comportamiento sea aprendido correctamente mediante algoritmos RL, es necesario estimar la utilidad de un conjunto importante de pares situación/acción. En el algoritmo Q-Learning, esto se logra mediante el proceso de exploración, que consiste en ejecutar acciones aleatorias o sub-óptimas. Pero debido al reducido alcance de los sensores infrarrojos y al movimiento del objetivo, la mayoría de estas acciones hacen que el objetivo se aleje más allá del área de sensado del robot y sea perdido de vista por éste, lo que provoca la interrupción del proceso de aprendizaje hasta tanto el objetivo vuelva a ser detectado por el robot perseguidor. El aprendizaje resulta entonces muy difícil, los tiempos de los experimentos aumentan y el proceso de exploración se ve drásticamente limitado.

Pero además, hay estados (en particular aquellos en los que el objetivo se encuentra demasiado cerca del robot) que tienen muy baja probabilidad de ser alcanzados con acciones aleatorias. Estos estados serán poco explorados durante el aprendizaje y, por lo tanto, el comportamiento no podrá ser aprendido correctamente para esas

situaciones.

El problema de los refuerzos

En las técnicas RL, un comportamiento se define implícitamente a partir de la función de refuerzo. Con una función de refuerzo inadecuada, el comportamiento que aprenderá el robot no resolverá la tarea deseada. Muchos autores estudiaron el problema de la función de refuerzo y elaboraron técnicas para su construcción, formalización y verificación [37], [9], [26].

Idealmente, la función de refuerzo debe ser una definición intuitiva y de alto nivel de la tarea, asociada con los resultados y no con una solución específica al problema. El caso extremo de este razonamiento es el de los juegos: en éstos suele asignarse un refuerzo positivo cuando el agente gana y uno negativo cuando pierde, asociando a todas las demás jugadas un refuerzo neutro. En la tarea de seguimiento, la definición de la tarea y del entorno nos lleva a asignar refuerzo positivo en todas aquellas situaciones asociadas con estados en los cuales el robot cumple con el predicado de seguimiento. Los refuerzos neutros y negativos podrían asignarse dependiendo de la distancia de la situación actual a dicha región del espacio de estados. Una posible forma de expresar esta función es:

$$rf(s_0..s_n) = \begin{cases} 1 & \text{si } (i^* = i_f) \wedge (|s_f - 0{,}5| < \sigma_1) \\ 0 & \text{si } (i^* = i_f - 1) \vee (i^* = i_f + 1) \\ & \quad \vee[((i^* = i_f) \wedge (|s_f - 0{,}5| \geq \sigma_1)] \\ -1 & \text{en caso contrario} \end{cases} \tag{3.11}$$

donde $s_0..s_n$ son los valores de los sensores en un instante dado ($0 \leq s_i \leq 1$), σ_1 es un parámetro de la función de refuerzo ($0 \leq \sigma_1 \leq 1$), i_f es el índice del sensor ubicado en el frente del robot y i^* es el índice del sensor con el valor máximo ($i^* = \arg\ \text{máx}_{0 \leq i \leq n}\{s_i\}$). Esto indica que el refuerzo positivo es obtenido cuando el objetivo está alineado y a una distancia cercana al valor de medición medio, nulo cuando el ángulo entre el objetivo y el robot es pequeño o bien cuando están alineados

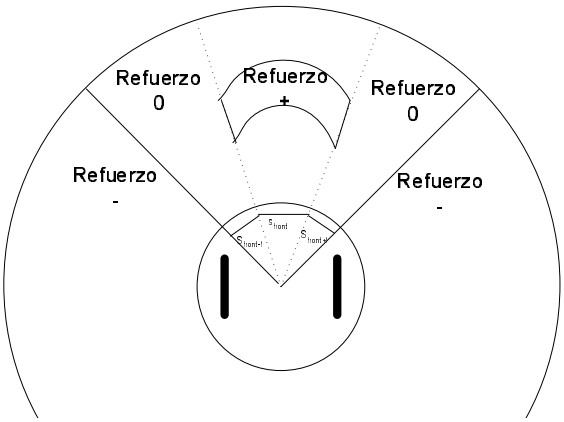


Figura 3.6: Estados objetivo para el problema de seguimiento.

pero no a la distancia correcta, y negativo en cualquier otro caso. La figura 3.6 muestra una representación gráfica de esta función.

Para que el aprendizaje sea exitoso, el robot debe recibir una proporción balanceada de refuerzos positivos, negativos y neutros durante el aprendizaje, y en particular durante la exploración del espacio de estados [37]. El problema con la función de refuerzo que dimos es que, como se puede ver en la figura, la cantidad de situaciones que se asocian con un refuerzo positivo es demasiado baja respecto del total de situaciones. Esto hace que el robot obtenga muy pocos refuerzos positivos, lo cuál dificulta el aprendizaje y aumenta los tiempos de los experimentos. Pero además de ser muy escasos, muchos de estos refuerzos serán demorados, es decir, serán obtenidos luego de ejecutar una secuencia de acciones. Esto no es un problema en sí mismo, ya que las técnicas de aprendizaje por refuerzo fueron pensadas para tratar problemas con estas características. Sin embargo, esta característica dificulta el aprendizaje y aumenta los tiempos de los experimentos [26].

3.3. Descripción de la Solución Propuesta

A partir del análisis de los problemas descriptos, presentamos en este trabajo una reformulación del problema y un método de aprendizaje que nos permite sintetizar, con tiempos de aprendizaje cortos, comportamientos de seguimiento eficaces. Nuestra propuesta está basada en la división del comportamiento, el aprendizaje en entornos reducidos y en una técnica de clustering basada en redes neuronales artificiales. Cada uno de estos elementos soluciona alguno de los problemas planteados, pero a la vez todos ellos están relacionados y se complementan entre sí, formando una solución integral del problema.

3.3.1. División de la tarea

Nuestro enfoque para la división de la tarea involucra una transformación del espacio de acciones: éstas, expresadas originalmente por una velocidad independiente en cada rueda (v_{izq}, v_{der}), son transformadas en tuplas compuestas por una componente de velocidad lineal y una de velocidad angular (v_{lin}, v_{ang}). Esta transformación se logra aplicando,

$$\begin{aligned} v_{lin} &= (v_{der} + v_{izq})/2 \\ v_{ang} &= (v_{der} - v_{izq})/L, \end{aligned} \tag{3.12}$$

donde L es el diámetro del robot (más precisamente, la distancia entre las dos ruedas).

A partir de esta descripción de las acciones, surge una división natural de la tarea de seguimiento: la definición de una tarea independiente para cada componente de las acciones. La primera tarea, que denominamos *mantener-ángulo*, tiene por objeto mantener al robot alineado con el objetivo mediante giros en el lugar ($v_{lin} = 0$). La segunda tarea, *mantener-distancia*, se ocupa de mantener la distancia entre ambos robots dentro de un rango predefinido mediante movimientos con dirección adelante/atrás ($v_{ang} = 0$). De esta manera, a partir del predicado que define la tarea de seguimiento obtenemos los predicados para las subtareas:

$$\begin{aligned} md(t) &\Leftrightarrow d - \epsilon_d \leq \lambda(t) \leq d + \epsilon_d \\ ma(t) &\Leftrightarrow -\epsilon_\phi \leq \phi \leq \epsilon_\phi. \end{aligned} \tag{3.13}$$

Así, el predicado original puede ser formado a partir de la conjunción de los predicados de las subtareas, mediante:

$$\begin{aligned} seguir(t) &\Leftrightarrow d - \epsilon_d \leq \lambda(t) \leq d + \epsilon_d \wedge -\epsilon_\phi \leq \phi \leq \epsilon_\phi \\ &\Leftrightarrow md(t) \wedge ma(t). \end{aligned} \tag{3.14}$$

Ambas subtareas son desarrolladas como comportamientos. El espacio de situaciones es en ambos casos el mismo que el del comportamiento original. Las acciones son transformadas nuevamente de velocidades lineal y angular a una distancia (interpretada como un movimiento en línea recta) en el caso de *mantener-distancia* y un ángulo (interpretado como una rotación en el lugar) en el caso de *mantener-ángulo*. Como veremos más adelante, las acciones expresadas de esta manera facilitan el aprendizaje.

Entonces, a partir de la política de seguimiento:

$$\pi_{seguir}(s) = (v_{izq}, v_{der}), \tag{3.15}$$

se definen las políticas:

$$\begin{aligned} \pi_{md}(s) &= d \quad d \in \Re \\ \pi_{ma}(s) &= \phi \quad \pi \leq \phi \leq \pi, \end{aligned} \tag{3.16}$$

donde s es la situación (la misma que para seguimiento), d, la acción de *mantener-distancia* es una distancia y ϕ, la acción de *mantener-ángulo* es un ángulo.

3.3.2. Configuración del entorno de aprendizaje

En cuanto al aprendizaje de los subcomportamientos, éste es realizado con el objetivo detenido y ubicado, al comenzar el aprendizaje, en la posición óptima, es decir, alineado y a la distancia deseada. Debido a las restricciones de movilidad de los subcomportamientos, durante el aprendizaje de *mantener-ángulo* la distancia que

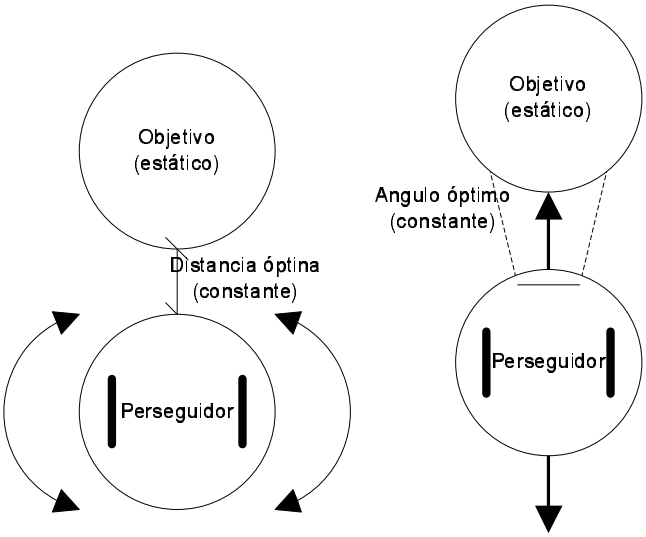


Figura 3.7: Entornos de aprendizaje para a)*mantener-ángulo* y b) *mantener-distancia*.

separa ambos robots se mantendrá constante, y durante el aprendizaje de *mantener-distancia*, será el ángulo el que no varíe (ver figura 3.7).

Las funciones de refuerzo utilizadas en cada subcomportamiento se derivan de la función presentada en la sección anterior para la tarea de seguimiento que, como vimos, asigna un refuerzo positivo en aquellos estados que hacen que el robot cumpla con el predicado de seguimiento. De esta manera, la función de refuerzo para *mantener-distancia* es:

$$md(s_0..s_n) = \begin{cases} 1 & \text{si } |s_f - 0{,}5| < \sigma_1 \\ 0 & \text{si } \sigma_1 \leq_| s_f - 0{,}5| \leq \sigma_2 \\ -1 & \text{si } \sigma_2 < |s_f - 0{,}5| \end{cases} , \tag{3.17}$$

y la función de refuerzo para *mantener-ángulo* es:

$$ma(s_0..s_n) = \begin{cases} 1 & \text{si } i^* = i_f \\ 0 & \text{si } i^* = i_f - 1 \vee i^* = i_f + 1 \\ -1 & \text{en caso contrario} \end{cases} , \tag{3.18}$$

donde $s_0..s_n$ son los valores de los sensores en un instante dado ($0 \leq s_i \leq 1$),

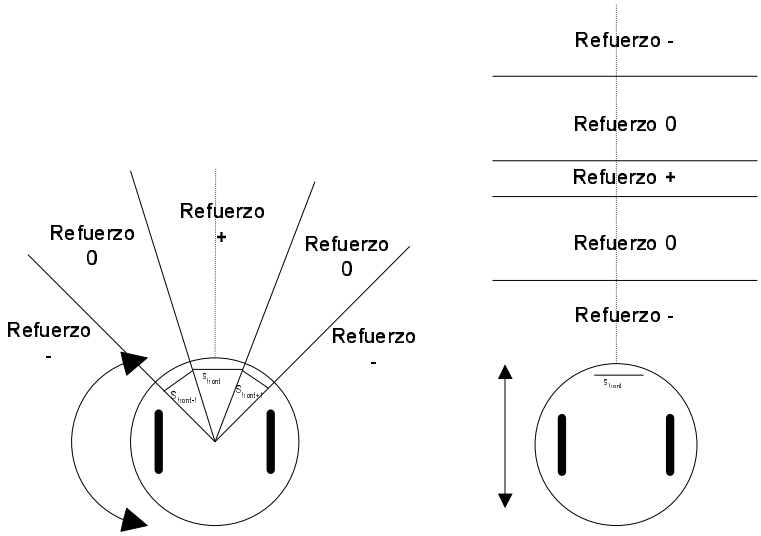


Figura 3.8: Representación de las funciones de refuerzo para el aprendizaje de *mantener-distancia* y *mantener-ángulo*.

σ_1 y σ_2 son parámetros de la función de refuerzo ($0 \leq \sigma_1 \leq \sigma_2 \leq 1$), i_f es el índice del sensor ubicado en el frente del robot y i^* es el índice del sensor con el valor máximo ($i^* = \arg\max_{0 \leq i \leq n}\{s_i\}$). De existir más de un sensor frontal, pueden fusionarse en un solo sensor a los efectos del cálculo del refuerzo. La figura 3.8 muestra una representación gráfica de las dos funciones de refuerzo.

El valor absoluto de un sensor, si el objeto se encuentra perfectamente alineado con éste, es una estimación de la distancia. La función de refuerzo que define el comportamiento *mantener-distancia* delimita cinco zonas para el valor absoluto del sensor frontal. El robot recibe refuerzo positivo si el valor del sensor se encuentra en la zona central (el objetivo está a una distancia media), nulo si se encuentra cercano a esa zona, y negativo si el valor es muy alto (lo que representa una colisión) o muy bajo (el objetivo se encuentra demasiado lejos). El sensor con valor máximo representa siempre (salvo fallas en los sensores) la dirección más próxima al objetivo. La función de refuerzo para *mantener-ángulo* asigna un refuerzo positivo si este sensor es el frontal (lo que indica que el objetivo está alineado), neutro si es un sensor vecino

y negativo si se encuentra lejos del frente.

3.3.3. Representación del espacio de situaciones/acciones

Durante el aprendizaje, el espacio de situaciones, acciones y valores de utilidad de cada comportamiento es representado con una técnica de *clustering* desarrollada por Santos [37], que denominamos QRBF. La misma esta basada en las redes neuronales RBF (*radial basis functions*). Cada unidad representa una región del espacio de situaciones/acciones/valores q, definida por una función de activación gaussiana. Durante el entrenamiento, las unidades son desplazadas en función de las situaciones visitadas, de los pares situación/acción ejecutados y del refuerzo obtenido. Además, la red se construye en forma incremental: si ninguna unidad se encuentra suficientemente cerca del estado visitado, entonces una nueva unidad es creada. La función de activación de cada unidad, en su forma general, es:

$$y = e^{-((s-w_s)^t(s-w_s)+|(1-w_q)/2|^2+(a-w_a)^2)/\sigma^2}, \tag{3.19}$$

donde w_s, w_a y w_q son los pesos de la unidad en las dimensiones de situación, acción y valor q respectivamente. El uso que le damos a esta red es similar al de una memoria asociativa: la función de activación es aplicada en forma parcial, utilizando dos dimensiones como entrada y la tercera como salida. Por ejemplo, para seleccionar la acción óptima asociada con una determinada situación, la función de activación es aplicada sobre las dimensiones situación/valor q, utilizando como entrada la situación actual y el máximo valor q posible, y como salida la acción asociada a la unidad ganadora. Esta acción representa un compromiso entre la bondad de la acción para esa situación en particular (su valor q) y la diferencia entre la situación actual y la memorizada por la red. Para actualizar la red, la función de activación es aplicada, también en forma parcial, al par situación/acción ejecutado. La unidad ganadora es actualizada en sus tres componentes: situación y acción (con la regla de Hebb, es decir, desplazando el centroide de la unidad hacia los valores de la entrada)

Funcion MejorAccion
Entrada: s
Salida: a

Seleccionar la unidad ganadora i*, donde
$i^* \leftarrow \arg\ \text{máx}_i\{y_i = e^{-((s-ws_i)^t(s-ws_i)+|(1-wq_i)/2|^2)/\sigma^2}\}$

Si el nivel de activación (y_{i^*}) es mayor que *acceptance* entonces,
devolver wa_{i^*}

En caso contrario,
$a \leftarrow rand[0.,1]$
$RedQRBF \leftarrow RedQRBF \bigcup\{(s, a, 0)\}$
devolver a

Figura 3.9: Algoritmo QRBF. Obtención de la acción óptima asociada a una situación (pseoducódigo).

y valor q (mediante el algoritmo de Q-Learning). En las figuras 3.9 y 3.10 se describen los algoritmos de selección de la acción óptima y de actualización de la red respectivamente.

3.3.4. Ejecución del comportamiento

Para combinar los subcomportamientos aprendidos y obtener comportamientos de seguimiento, ambos son ejecutados en forma concurrente. Las acciones llevadas a cabo por el robot son combinaciones de las acciones individuales (un ángulo y una distancia) traducidas al espacio de acciones del robot (velocidades en cada rueda). Para poder combinar una distancia y un ángulo en una única acción, transformamos las acciones en velocidades lineales y angulares respectivamente.

$$\begin{aligned} v_{lin} &= \frac{d\ p}{t_l} \\ v_{ang} &= \frac{\phi\ \pi/180}{t_a}, \end{aligned} \tag{3.20}$$

Funcion Actualizar

Entrada: (s, a, s', r)
Salida: $-$

Calcular Q-Max (valor q de la nueva situación s'),
$j^* \leftarrow \text{argmax}_j\{y_j = e^{-((s'-ws_j)^t(s'-ws_j)+|(1-wq_j)/2|^2)/\sigma^2}\}$
$QMax \leftarrow wq_{j^*}$

Seleccionar la unidad más cercana al par situación inicial/acción (s, a),
$i^* \leftarrow \text{arg máx}_i\{y_i = e^{-((s-ws_i)^t(s-ws_i)+(a-wa_i)^2)/\sigma^2}\}$

Si el nivel de activación (y_{i^*}) es mayor que *acceptance* entonces,
actualización Q-Learning:
$wq_{i^*} \leftarrow wq_{i^*} + \eta_q(r + \gamma QMax - wq_{i^*})$
actualización Hebb:
$ws_{i^*} \leftarrow ws_{i^*} + \eta_s(s - ws_{i^*})$
si r es positivo:
$wa_{i^*} \leftarrow wa_{i^*} + \eta_s(a - wa_{i^*})$
en caso contrario,
$RedQRBF \leftarrow RedQRBF \bigcup \{(s, a, r)\}$

Figura 3.10: Algoritmo QRBF. Actualización de la red a partir de la ejecución de una acción y del refuerzo obtenido (pseudocódigo).

donde d es el resultado de *mantener-distancia*, ϕ es el resultado de *mantener-ángulo*, t_l y t_a es el tiempo necesario para la ejecución de las acciones de avanzar y girar respectivamente, y p la distancia (en mm) de una unidad del encoder de las ruedas. La transformación de (v_{lin}, v_{ang}) a (v_{izq}, v_{der}) puede ser derivada de la ecuación:

$$\begin{aligned} v_{izq} &= v_{lin} - L/2 \times v_{ang} \\ v_{der} &= v_{lin} + L/2 \times v_{ang}. \end{aligned} \tag{3.21}$$

En resumen, las acciones que ejecutará el robot se obtienen a partir de los resultados de los dos subcomportamientos según:

$$a = (v_{izq}, v_{der}) = \left(\frac{k_1\ d - k_2\ \phi}{v_p}, \frac{k_1\ d + k_2\ \phi}{v_p} \right), \tag{3.22}$$

donde $k_1 = \frac{p}{t}$, $k_2 = \frac{L}{2} \frac{\pi}{180} \frac{1}{t}$ y v_p es la unidad de valocidad del robot expresada en mm/s.

Finalmente, con el fin de aliviar los problemas que produce el hecho de aprender un comportamiento en entornos estáticos y ejecutarlo en entornos dinámicos, realizamos una superposición de los ciclos de control (sensado, obtención de la acción óptima, ejecución de la acción) de manera de abortar la ejecución de una acción y reemplazarla por una nueva si la acción óptima cambia. La figura 3.11 muestra en pseudocódigo el algoritmo que controla la ejecución del comportamiento final de seguimiento.

3.4. Análisis de la solución propuesta

El método presentado tiene varios puntos que merecen ser estudiados. Desde las ventajas y los posibles inconvenientes que se pueden generar, hasta algunas características emergentes de los comportamientos obtenidos. Comenzaremos por analizar de qué manera este método resuelve los problemas de RL para la tarea de seguimiento.

Constantes:

t_l, t_a	Tiempo utilizado para ejecutar una acción de avanzar y girar respectivamente
p	Cantidad de unidades del encoder de las ruedas en un milímetro
L	Diámetro del robot

Entrada:

$qrbf_d$, $qrbf_a$	Redes que representan la política óptima para los comportamientos *mantener-distancia* y *mantener-ángulo* respectivamente.
max_d, max_a	Valores máximos para las medidas de distancia y ángulo, expresadas en unidades de los encoders de las ruedas y grados respectivamente.

Repetir para siempre

Paso 1: Obtener situación actual

$s \leftarrow$ situación actual(), $s = s_0, s_1, ..., s_n$

Paso 2: Obtener la mejor acción para la situación actual en ambos subcomportamientos

$d \leftarrow$ qrbf-best-action($qrbf_d, s$)

$\phi \leftarrow$ qrbf-best-action($qrbf_a, s$)

Paso 3: Transformar las acciones en velocidades

$v_{lin} \leftarrow \frac{d\ p}{t_l}$

$v_{ang} \leftarrow \frac{\phi\ \frac{\pi}{180}}{t_a}$

Paso 4: Transformar velocidades lineal y angular en velocidades de las ruedas

$v_{izq} \leftarrow v_{lin} - L/2 \times v_{ang}$

$v_{der} \leftarrow v_{lin} + L/2 \times v_{ang}$

Paso 5: Si las nuevas velocidades son diferentes a las actuales, modificar la velocidad del robot

$(av_{izq}, av_{der}) \leftarrow$ velocidades-actuales()

Si $v_{izq} \neq av_{izq} \vee v_{der} \neq av_{der}$, entonces

asignar-velocidades(v_{izq},v_{der})

Paso 6: Esperar

Dormir(p), $p << t_l, t_a$

Figura 3.11: Algoritmo de control para la ejecución del comportamiento de seguimiento (pseudocódigo).

3.4.1. Ventajas

Se eliminan los problemas de exploración

Al aprender en entornos estáticos, se soluciona el principal problema del aprendizaje: el riesgo de perder de vista al objetivo producto del reducido alcance de los sensores. Se logra que el objetivo se mantenga todo el tiempo dentro del rango de sensado o, si se pierde de vista (por ejemplo, debido a un movimiento constante del robot hacia atrás) que pueda ser detectado nuevamente mediante la ejecución de una acción reflejo. El robot puede entonces ejecutar libremente su política de exploración sin riesgo de perder de vista al objetivo.

Además, en los nuevos espacios de situaciones/acciones y con el objetivo detenido, todas las situaciones tienen aproximadamente la misma probabilidad de ser visitadas mediante la ejecución de una política aleatoria, lo cuál permite la exploración completa de todo el espacio y la generación de comportamientos que funcionen en forma correcta para cualquier situación.

Reducción del espacio de situaciones

La reducción del espacio de situaciones que se produce al aprender en entornos estáticos hace que el problema tenga un tamaño razonable para ser resuelto con el algoritmo RL, comparable al de otros problemas clásicos de RL aplicado a robots reales, como por ejemplo evitar obstáculos. Esta reducción se logra sin perder la precisión de las mediciones ni la cantidad de acciones posibles.

Hemos visto que con el robot objetivo en movimiento, el valor de los sensores en un instante dado no alcanza para distinguir estados que requieren acciones distintas. Si el objetivo está detenido, estos valores sí son suficientes, ya que todos los demás datos (velocidad del robot perseguidor, diferencia de los sensores para mediciones consecutivas) no afectan la acción que debe tomar el robot en cada estado. Pasamos entonces de un espacio de estados de tamaño $4\ {s_{max}}^{2n}\ {v_{max}}^{2}$ a uno de sólo ${s_{max}}^{n}$.

Este número representa la cantidad total de situaciones que se pueden obtener

con el objetivo detenido. Debido a las limitaciones de movilidad de los dos subcomportamientos, la cantidad de situaciones visitadas en el aprendizaje es mucho menor: durante el aprendizaje de *mantener-distancia*, como los robots se mantienen siempre alineados, sólo los sensores frontales perciben valores distintos de cero, lo que produce un espacio de situaciones de tamaño ${s_{max}}^2$. Respecto de *mantener-ángulo*, las situaciones visitadas son solamente aquellas que representan estados con los robots a una distancia fija, ya que ésta se mantiene constante durante el aprendizaje.

Simplificación del espacio de acciones

El método de división de la tarea se basa en la simplificación del espacio de acciones: el espacio original, de dos dimensiones, se divide en dos espacios de una dimensión cada uno. Una primera ventaja de esto radica en el hecho de que en los nuevos espacios, las acciones similares producen resultados similares. Esto no es así en el espacio original, ya que el resultado de la acción $(1, 1)$ es más parecido al de la acción $(2, 2)$ que al de las acciones $(1, 2)$ y $(2, 1)$ ya que las dos primeras representan movimientos rectilíneos y las dos últimas describen arcos. Además, los nuevos espacios son crecientes o decrecientes en relación a las variables físicas: cuanto mayor es el movimiento hacia delante, mayor es el valor de los sensores de la situación resultante y menor es la distancia entre el robot y el objetivo. Por último, podemos decir que las acciones en los nuevos espacios son *conmutativas* y *asociativas*. Con la primera característica queremos indicar que el resultado obtenido no depende del orden de ejecución de las acciones. Con la segunda que, por ejemplo, un movimiento hacia delante de 5 mm seguido de un movimiento de 10 mm es, (salvo errores de medición), equivalente a un movimiento único de 15 mm.

Las características que acabamos de mencionar simplifican el problema, haciendo más fácil la definición de los experimentos, su ejecución y el análisis de los resultados. Pero además mejora sustancialmente la performance de cualquier algoritmo de clustering, en especial de aquellos basados en redes neuronales artificiales, como el utilizado por nosotros.

Respecto de la transformación de las acciones en movimientos discretos (distancias y ángulos), esto simplifica el proceso de aprendizaje en comparación con los movimientos expresados con velocidades. Gracias a sensores en las ruedas, el robot puede ejecutar las acciones en forma más precisa y más rápida, ajustando por sí mismo la velocidad en función de la distancia a recorrer. Al expresarse con velocidades, los movimientos son imprecisos y dependen de la duración del ciclo de control: si éste es demasiado largo, los comandos serán muy poco precisos y habrá situaciones en las cuales el robot *se pase* de la situación óptima. Si el ciclo es demasiado corto, en cambio, se requerirían velocidades demasiado altas para distancias o ángulos grandes. Al expresar las acciones con ángulos y distancias, todo esto no ocurre, y es posible además definirlas con una precisión mucho mayor. Con esta transformación evitamos también los problemas relacionados con el tiempo, que pueden afectar el resultado de las acciones (latencia en las comunicaciones, nivel de carga de la computadora, etc.) y aumentamos la confiabilidad de las mediciones de los sensores, porque las realizamos con el robot completamente detenido.

Mayor cantidad de refuerzos positivos y de refuerzos inmediatos

Con los nuevos espacios de situaciones y las nuevas funciones de refuerzo, vemos que la proporción de situaciones con refuerzo positivo respecto del total es mucho mayor: esto es natural si notamos que la cantidad de situaciones que representan una distancia acotada entre el robot y el objetivo, en un espacio de una dimensión es mucho mayor que la cantidad de situaciones que representan al robot alineado y a una cierta distancia, en un espacio de dos dimensiones (figura 3.12).

Con el entorno estático y los movimientos restringidos, es posible definir el rango de acciones para que sea posible, desde cualquier situación, obtener un refuerzo positivo con una o dos acciones solamente. Por ejemplo, si el rango de los sensores es de 5 mm, fijando en 2.5 mm la distancia máxima de movimiento existe una política para *mantener-distancia* que permite llegar desde cualquier situación a los estados objetivo en a lo sumo dos pasos. De esta manera podemos reemplazar los refuerzos demorados

por refuerzos inmediatos, y acelerar así el aprendizaje.

3.4.2. Características del comportamiento obtenido

La técnica de aprendizaje presentada en este trabajo hace que el comportamiento obtenido presente algunas características especiales. Analizaremos en particular el efecto que produce el hecho de aprender los subcomportamientos con el objetivo detenido y la falta de exploración de regiones grandes del espacio de situaciones.

Aprendizaje con el objetivo detenido y ejecución con el objetivo en movimiento

Una política aprendida con el objetivo detenido no es necesariamente la mejor política si se considera un objetivo móvil. Durante su ejecución, el resultado de las acciones será distinto que el obtenido en el aprendizaje, ya que mientras la acción es ejecutada el objetivo cambia de posición. Por ejemplo, si el objetivo se está acercando a velocidad alta en dirección al robot, pero se encuentra todavía demasiado lejos, la política aprendida por el robot haría que éste avance, probablemente también con alta velocidad, lo que produciría una colisión.

Nuestra propuesta de superponer los ciclos de control hace que una acción mal seleccionada sea inmediatamente corregida. Volviendo al ejemplo, vemos que el robot intentará moverse a toda velocidad, pero luego de unos pocos milisegundos (y probablemente, debido al tiempo que toma la aceleración, sin llegar a desarrollar esa velocidad), el objetivo estará ya más cerca y la mejor acción cambiará por un movimiento más lento, hasta terminar en un movimiento hacia atrás al cabo de muy poco tiempo. Pero además, esta técnica produce comportamientos emergentes interesantes: si el objetivo se mueve a velocidad constante, el robot, luego de unas pocas iteraciones, comenzará a moverse con la misma velocidad.

Para analizar estos efectos, consideraremos una simulación en la cuál el robot selecciona las acciones de avanzar y girar que lo llevan a una posición exacta en relación con el objetivo, a partir de la distancia y el ángulo al mismo. Por ejemplo, si

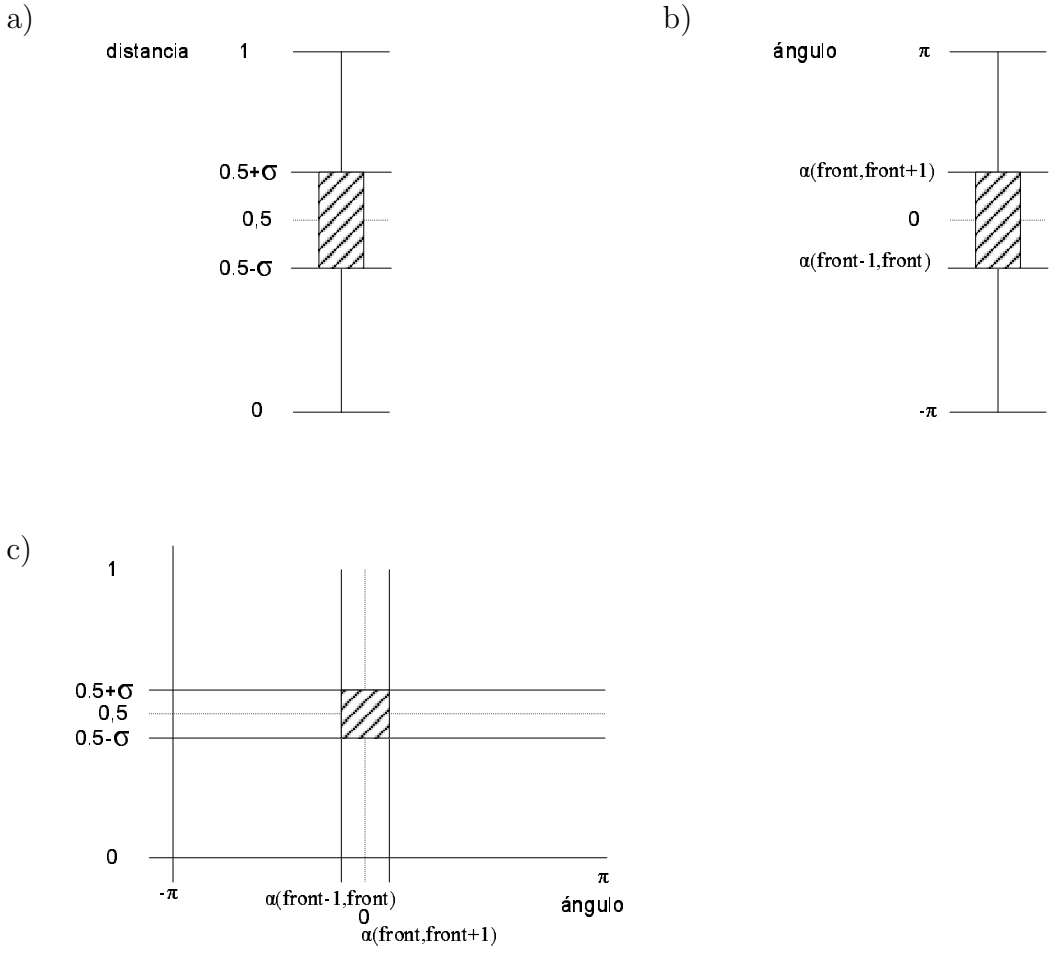


Figura 3.12: Situaciones que cumplen con cada subtarea, respecto del total de situaciones. a) *mantener-distancia*, b) *mantener-ángulo* y c) *seguimiento*.

la distancia deseada es de 20 mm y el objetivo está alineado y a 50 mm de distancia, entonces la acción será avanzar 30 mm.

Vamos a considerar tres escenarios: En el primero, a partir de una posición inicial con los robots alineados, el objetivo comienza a alejarse en línea recta con velocidad constante. En este escenario probaremos diferentes velocidades para el objetivo. En el segundo escenario, a partir de la misma posición inicial, el objetivo se acerca al robot. Probaremos aquí el resultado para diferentes valores del parámetro t (el tiempo del ciclo de control). Por último estudiaremos la trayectoria del robot para objetivos alejándose con diferentes ángulos.

Para el primer escenario, la distancia óptima fue fijada en 20 mm y la distancia inicial en 50 mm. El valor de t (tiempo del ciclo de control) respecto de t_l (tiempo para ejecutar una acción) es de 1/4. Esto significa que sólo se ejecuta un cuarto de la acción seleccionada. Por ejemplo, si la acción indica moverse 100 mm hacia delante, en el siguiente ciclo el robot se habrá movido 25 mm. El gráfico 3.13 muestra la distancia entre el robot y el objetivo a lo largo del tiempo para diferentes velocidades del objetivo.

En este gráfico puede verse que en una primera etapa la distancia varía gradualmente, y luego se estabiliza. Esto significa que el robot logra moverse a la misma velocidad que el objetivo. Este comportamiento es emergente, ya que la selección de la acción está basado exclusivamente en la distancia al objetivo en instantes de tiempo, y no en estimaciones de su velocidad. Es interesante notar también que la distancia de equilibrio entre el robot y el objetivo depende de la velocidad de este último: cuanto mayor es ésta, desde más lejos es perseguido por el robot. Esta distancia es siempre mayor a la distancia óptima, la cuál sólo se obtiene con el objetivo detenido (y que produce en el robot acciones nulas).

El siguiente gráfico muestra la influencia de la variable t en la trayectoria seguida por el robot, en situaciones en las que la acción aprendida es contraria a la que debería ejecutarse. En este caso, el objetivo se acerca al robot, partiendo de una distancia de 50 mm.

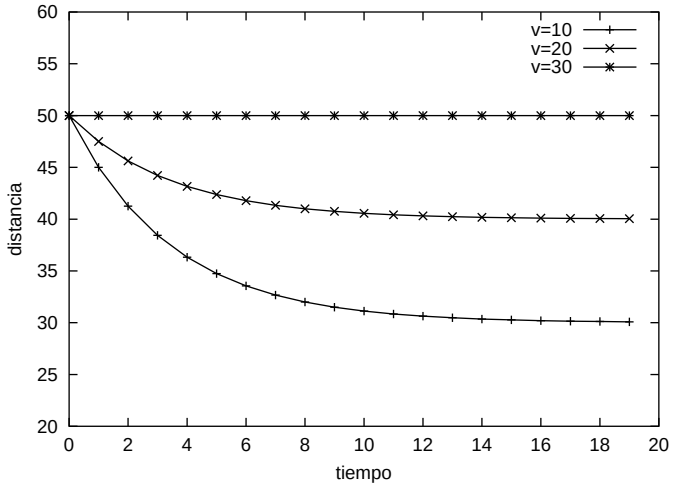


Figura 3.13: Simulación de la diferencia de distancia entre el robot y el objetivo, para movimientos rectilíneos a distintas velocidades. El eje y indica la distancia entre el robot y el objetivo, y el eje x representa el tiempo.

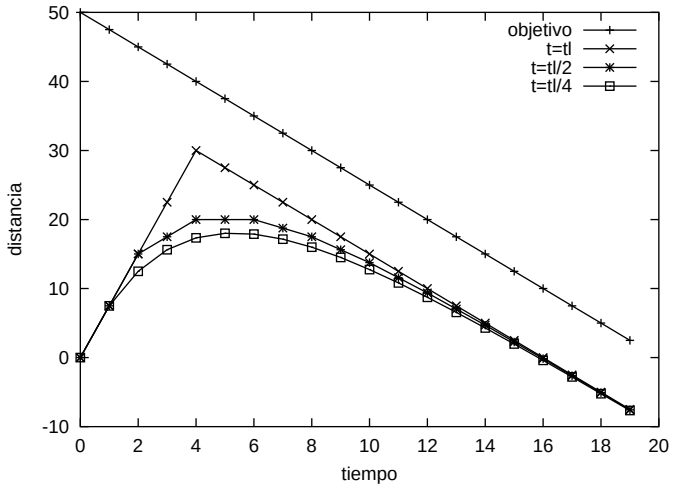


Figura 3.14: Simulación de la distancia entre el robot y el objetivo, para un objetivo que se acerca al robot en línea recta. Trayectoria del robot para distintos valores del parámetro t. El eje y representa la distancia entre el robot y el objetivo, considerando la posición inicial del robot como el origen de coordenadas. El eje x representa el tiempo.

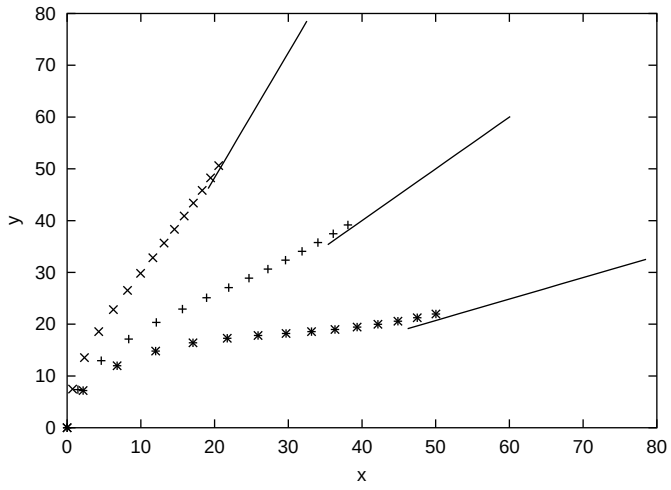


Figura 3.15: Trayectoria seguida por el robot para un objetivo que se aleja con trayectoria rectilínea, orientado a diferentes ángulos del robot. Ambos ejes del gráfico representan el plano, y cada punto indica la posición del robot en un instante de tiempo.

Las primeras acciones elegidas por el robot son inadecuadas: como el objetivo está todavía muy lejos, el robot avanza, porque no tiene manera de detectar que éste se está acercando. Pero luego, el robot comienza a retroceder con la misma velocidad que el objetivo. Podemos ver que cuanto menor es el valor de t (es decir, cuanto antes se corrigen las acciones ejecutadas) más suave es la trayectoria y menos graves las consecuencias de seleccionar una mala acción.

Por último, el gráfico 3.15 muestra diferentes trayectorias trazadas por el robot, para objetivos alejándose en línea recta, pero a diferentes ángulos respecto del robot. Puede verse en el gráfico la suavidad de las trayectorias obtenidas, producto de la ejecución parcial de las acciones.

Aprendizaje de una parte del espacio de situaciones únicamente

Los entornos estáticos y las limitaciones en los movimientos de las subtareas restringen el conjunto de situaciones que el robot puede experimentar durante el aprendizaje. Por ejemplo, al aprender *mantener-distancia*, el robot nunca experimentará situaciones que tengan valores muy altos en los sensores laterales, a pesar de que estas situaciones sí pueden ser obtenidas durante la ejecución de seguimiento. Asimismo, mientras se aprende *mantener-ángulo*, los sensores tendrán siempre valores moderados, a pesar de que estas son situaciones válidas.

Sin embargo, hay tres elementos que alivian los problemas que esto trae: 1) la técnica de clustering para la representación del espacio de situaciones, 2) el hecho de que el espacio de situaciones con el que se aprende cada comportamiento incluye los estados objetivo del otro comportamiento, y 3) la ejecución de la acción nula para ambos subcomportamientos si el robot se encuentra en un estado objetivo.

La técnica de clustering del espacio de situaciones favorece muy especialmente a la tarea *mantener-ángulo*, debido a que el cluster más cercano a una situación no explorada (con el objetivo a una distancia distinta a la utilizada en el aprendizaje) es en general aquél que representa una situación con un ángulo similar, porque en ambas la distribución de los valores de los sensores será igual, y sólo será distinto su valor absoluto. La única excepción es la situación nula (valor cero en todos los sensores), que no puede ser asimilada a ninguna otra. Para esa situación en particular, colocamos en el comportamiento un reflejo que indica no girar, por medio de la incorporación de un nuevo cluster.

En el comportamiento *mantener-distancia*, el clustering sólo influye favorablemente para los casos en los que el ángulo es muy cercano al óptimo, porque en todas las demás situaciones los sensores delanteros obtienen valores nulos. Esta situación produce un movimiento rápido hacia delante, indicado por la acción óptima asociada. Sin embargo, el hecho de que la región objetivo de un comportamiento coincida con con la región en la que se realizó el aprendizaje del otro hace en este caso que las acciones

del comportamiento *mantener-ángulo* lleven al robot lo antes posible (porque ese es su objetivo) a situaciones conocidas por el comportamiento *mantener-distancia*. Existe igualmente un conjunto de situaciones (especialmente aquellas en las que el objetivo se encuentra de costado y muy cerca) que hacen que un movimiento rápido hacia delante produzca la pérdida del objetivo. Sin embargo, durante la ejecución normal de seguimiento, el objetivo no puede aparecer en esas posiciones sin ser visto antes en otras, con más chances de ajustar adecuadamente el ángulo.

Por último, podemos mencionar que el hecho de que en ambos subcomportamientos las situaciones óptimas tengan asociada la acción nula, hace que el comportamiento de seguimiento resultante sea más estable y se logren movimientos muy precisos (por ejemplo, que el robot imite la velocidad del objetivo si se encuentran alineados y moviéndose en línea recta).

Capítulo 4

Resultados experimentales

En este capítulo describiremos las experiencias realizadas y los resultados obtenidos tanto para la síntesis de los subcomportamientos *mantener-distancia* y *mantener-ángulo* como para su composición y uso en seguimiento de objetivos móviles.

4.1. Equipamiento utilizado

El equipamiento utilizado para las experiencias está formado por dos robots Khepera, fabricados por la empresa K-Team [12] (ver figura 4.1). Estos son robots autónomos móviles para uso académico y de investigación. Su cuerpo es circular, de 55 mm de diámetro. Cuentan con un microcontrolador Motorola MC68331, 256K de memoria RAM, que permite almacenar programas escritos por el usuario y 512K de memoria ROM que contiene el software de control de bajo nivel (BIOS), y un sistema operativo multitarea. Los modos de operarlo son dos: mediante la transferencia de un programa completo (procesamiento dentro del robot) o a través de la transmisión de comandos (procesamiento en una PC). Los comandos de alto nivel permiten, entre otras cosas, leer el valor de los sensores, aplicar velocidades a las ruedas, leer los encoders de las ruedas y mover cada rueda una determinada distancia, en cuyo caso el software del robot define una función de aceleración para que el movimiento sea suave.

Los robots Khepera se comunican con una PC a través de una interface serie.

Alternativamente, los robots Khepera pueden comunicarse entre sí y con una PC a través de una interface de radio que funciona como un medio compartido de transmisión de paquetes. La alimentación de los Khepera puede realizarse mediante una batería interna, que tiene una autonomía de entre 20 y 30 minutos, o mediante una fuente externa.

Sensores

Los robots Khepera poseen 8 sensores infrarrojos, cada uno de ellos compuesto por un transmisor y un receptor. Estos sensores están distribuidos en todo el perímetro del robot, de manera no uniforme. La figura 4.2 muestra un esquema aproximado de la distribución de los mismos.

Los sensores pueden ser usados para detectar objetos próximos o el nivel de luminosidad del ambiente, en forma direccional. Tienen una resolución, para ambos casos, de 10 bits, es decir que toman valores entre 0 y 1023. Utilizados como sensores de proximidad, cada uno puede detectar obstáculos hasta una distancia de 50 mm y un ángulo entre el objeto detectado y el sensor que depende de la distancia pero que puede llegar a los 40 grados. La sensibilidad puede variar considerablemente para distintos sensores, y puede verse afectada por condiciones externas, como por ejemplo la diferencia de luminosidad o su orientación respecto del piso [18]. Una característica importante de los sensores infrarrojos es que el valor que estos devuelven, si bien depende de la distancia entre el sensor y el objeto, no es lineal respecto de la misma y su relación también es influenciable por variables externas. A modo de ejemplo, la figura 4.3 muestra el nivel de activación de los 8 sensores para un mismo objeto ubicado a una distancia fija (30 mm) y a diferentes ángulos respecto del frente del robot. En esta figura se puede ver la orientación de cada sensor (que corresponde con el ángulo en el que toma su valor máximo). También se observa que algunos sensores, en particular el número 3, tienen menor alcance, ya que en su pico obtiene un valor menor que el resto.

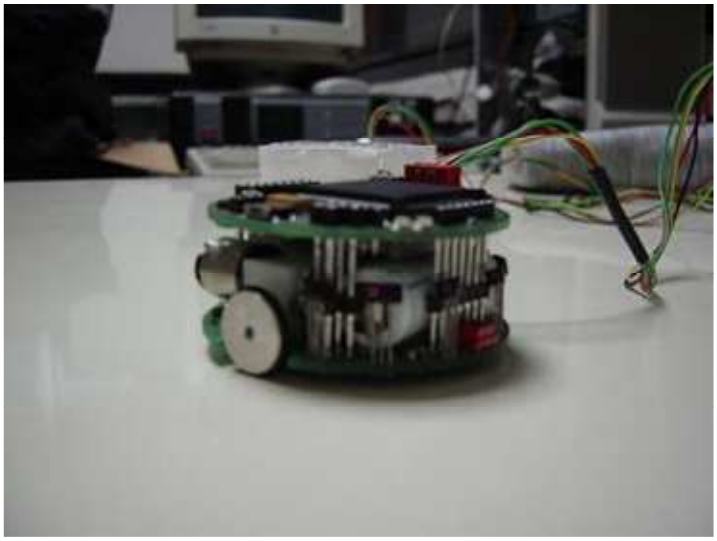


Figura 4.1: Fotos del robot Khepera.

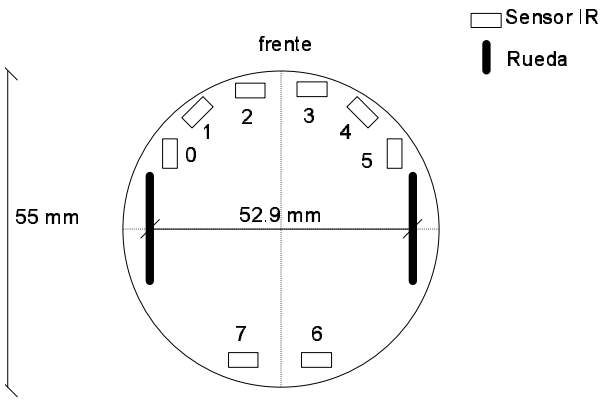


Figura 4.2: Esquema de los sensores y actuadores del robot Khepera.

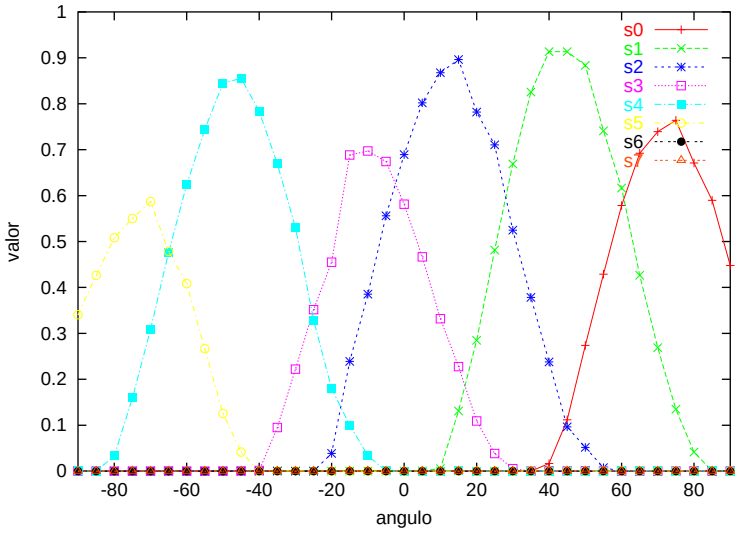


Figura 4.3: Nivel de activación de los sensores de Khepera ante un mismo objeto alineado a diferentes ángulos respecto del frente del robot.

Actuadores

Khepera utiliza para movilizarse dos ruedas independientes que se encuentran ubicadas sobre el perímetro externo del robot (ver figura 4.2). Estas ruedas son accionadas por motores que pueden moverse a velocidades de hasta 1 m/s, con una precisión de 8 mm/s (valores entre -127 y 127). Además las ruedas cuentan con encoders que permiten detectar con una precisión de 1/12 mm los movimientos realizados por las ruedas. Dadas las características de las ruedas, el robot puede moverse en la dirección adelante/atrás, describir arcos y girar en el lugar.

4.2. Aprendizaje de los subcomportamientos

4.2.1. Consideraciones generales

Los experimentos para el aprendizaje de los dos subcomportamientos pueden dividirse en tres etapas. La primera etapa consistió en la validación de la robustez del método mediante la repetición de los experimentos y en el ajuste de los parámetros. Para esto, definimos una función de utilidad aplicable a la política completa que nos permitió por un lado verificar el correcto funcionamiento del algoritmo de aprendizaje (comprobando que éste mejora la política con el tiempo), y por el otro comparar objetivamente políticas obtenidas con valores diferentes en los parámetros. También esta medida nos permitió validar la robustez del aprendizaje, verificando que diferentes experimentos con idénticas condiciones generan políticas con valores de utilidad similares. En la segunda etapa realizamos un análisis más exhaustivo de las mejores políticas aprendidas. Para ello estudiamos, para un muestreo del espacio de situaciones, la acción asociada a cada una de ellas, la situación que resulta de aplicar dicha acción y el agrupamiento de las situaciones producto del clustering. Esto nos permitió analizar si el comportamiento es adecuado en todo el espacio de situaciones. En la tercera etapa, analizamos el resultado de la ejecución de la política aprendida con el objetivo en movimiento y verificamos que el comportamiento obtenido es el esperado,

comprobando que el comportamiento *mantener-distancia* logra seguir al objetivo e imitar su velocidad, y que el comportamiento *mantener-ángulo* produce movimientos suaves al girar.

Medida de utilidad para las políticas

Para verificar la mejora del comportamiento durante el aprendizaje y para poder comparar los comportamientos obtenidos a partir de distintos valores de los parámetros, definimos la siguiente medida de utilidad:

$$K(\pi, S') = \sum_{s \in S'} R(T(s, \pi^*(s))), \tag{4.1}$$

donde S' es un muestreo del espacio de situaciones, $R(s)$ la función de refuerzo [1], $\pi^*(s)$ la política óptima y $T(s, a)$ la función de transición. Esto es, el valor de utilidad de la política π para el conjunto de estados S' es igual a la suma de los refuerzos inmediatos obtenidos al ejecutar la acción óptima en cada uno de los estados del conjunto S'.

Esta medida no tiene en cuenta la utilidad futura de las situaciones, implícita en los valores q. Esta elección se tomó debido a que los valores de refuerzo son exactos, mientras que los valores q son estimaciones cuyo error es muy variable en el espacio de estados y depende del grado de avance del aprendizaje y de la estrategia de exploración, entre otros factores. Debido a esto, una medida basada en valores q no podría ser utilizada al comenzar el aprendizaje ni para estudiar la política en todo el espacio de situaciones, porque las regiones poco exploradas tendrían valores q inadecuados.

Muestreo del espacio y modelo de interacción

Con el objeto de calcular la utilidad de la política durante el aprendizaje sin necesidad de detener el mismo y ejecutar exhaustivamente la política obtenida hasta el momento, generamos, en forma previa al aprendizaje y a partir de un muestreo

[1] Nótese que hemos definido la función de refuerzo a partir de la situación alcanzada, independientemente de la situación inicial y de la acción

del espacio de situaciones/acciones, un modelo de interacción aproximado. Para esto colocamos el objetivo alineado respecto del robot perseguidor y a una distancia apenas mayor que el rango de los sensores. Luego ejecutamos con el robot perseguidor un algoritmo que consiste en, alternativamente, avanzar 5 unidades y leer el valor de los sensores, hasta que ambos robots colisionen para un modelo del espacio de situaciones de *mantener-distancia*. Para el caso de *mantener-ángulo*, la única diferencia es que la acción es girar 5 grados (en lugar de avanzar) y parte con un ángulo inicial de 90°. El modelo de interacción se basa en suponer que el resultado de las acciones es acumulativo, es decir, que el resultado de avanzar dos veces 5 unidades es igual al resultado de avanzar 10 unidades en un solo movimiento (idem para las rotaciones). Si bien esto no es del todo correcto (debido al rozamiento de la superficie y a errores producidos durante el arranque y la detención, el resultado suele ser distinto), puede considerarse como una buena aproximación si lo que buscamos es comparar la calidad de las políticas.

Entonces, a partir del muestreo:

$$S' = (s_0, s_1, ..., s_n), T(s_i, 5) = s_{i+1}, \tag{4.2}$$

deducimos la función de transición:

$$T(s_i, a) = s_{i+|a/5|}. \tag{4.3}$$

4.2.2. Comportamiento *mantener-distancia*

El aprendizaje del comportamiento *mantener-distancia* se realizó con dos robots, uno cumpliendo el rol de objetivo y el otro de perseguidor. El robot objetivo fue recubierto con un papel blanco para aumentar el alcance y la performance de los sensores infrarrojos del perseguidor. Los parámetros fueron ajustados a mano, a partir de la función de evaluación de la política.

En todos los experimentos, el espacio de situaciones estuvo compuesto por el valor de los 8 sensores (con 1024 valores posibles cada uno). El movimiento máximo

permitido fue de 100 unidades (8 mm) hacia delante o hacia atrás.

Como el robot Khepera posee dos sensores ubicados en el frente y con la misma orientación (los sensores 2 y 3), se utilizó, para la función de refuerzo, el promedio entre ambos. Los parámetros de refuerzo utilizados fueron: $\sigma_1 = 0{,}1$ y $\sigma_2 = 0{,}3$. La función de refuerzo resulta entonces:

$$reinf(s_0..s_7) = \begin{cases} 1 & \text{si } |\frac{(s_2+s_3)}{2} - 0{,}5| \leq 0{,}1 \\ -1 & \text{si } |\frac{(s_2+s_3)}{2} - 0{,}5| > 0{,}3 \\ 0 & \text{en caso contrario} \end{cases} . \tag{4.4}$$

Con esta función, se obtiene un refuerzo positivo si el valor del nuevo sensor frontal se encuentra entre 0.4 y 0.6, refuerzo neutro para valores entre 0.2 y 0.4 y entre 0.6 y 0.8, y negativo si es menor que 0.2 o mayor que 0.8. Aplicando esta función de refuerzo al espacio muestreado, se obtiene un refuerzo positivo para el 12 % de las situaciones.

En cuanto a la exploración del espacio de situaciones y de acciones, ésta se logró eligiendo la acción a ser ejecutada por el robot en forma aleatoria dentro de un intervalo con centro en la acción indicada por la política. El tamaño del intervalo se redujo linealmente durante el aprendizaje, entre un valor máximo al inicio y cero al final. Por lo tanto, para la iteración i se utilizó como factor de exploración un número real aleatorio entre $E(i)$ y $-E(i)$, donde:

$$E(i) = \frac{1}{2} \times \text{max-random} \times \left(1 - \frac{i}{\text{max-iter}}\right). \tag{4.5}$$

El valor utilizado para *max-random* fue de 0.9. Este valor hace que el intervalo de exploración abarque casi el total del espacio de acciones al iniciarse el aprendizaje.

Por último, se incorporaron durante el aprendizaje dos reflejos para evitar que el robot se aleje demasiado de su objetivo y que se mantenga demasiado cerca por mucho tiempo:

- *reflejo-colisión*: si el sensor frontal produce un valor máximo durante más de 6 iteraciones seguidas, entonces retroceder lentamente hasta que el valor del

sensor sea menor que uno (el valor máximo).

- *reflejo-perdido*: si el sensor frontal produce un valor nulo durante más de 6 iteraciones seguidas, entonces avanzar lentamente hasta que el valor del sensor sea mayor que cero.

Resultados del aprendizaje

El primer conjunto de experiencias tiene por objeto ajustar los parámetros involucrados y verificar que el aprendizaje produce un mejoramiento de la política óptima a lo largo del tiempo.

Los parámetros η del algoritmo de clustering indican, de alguna manera, la magnitud de los ajustes en los clusters para cada dimensión del espacio (situaciones, acciones y valores q) y su adaptación a los nuevos valores observados. Un valor muy pequeño en estos parámetros demora el aprendizaje, y un valor muy elevado produce oscilaciones que impiden la convergencia. El ajuste de estos parámetros fue realizado a mano, a partir de la comparación de la performance obtenida con distintos valores. La tabla 4.1 muestra algunas experiencias realizadas, y los valores obtenidos al variar cada parámetro. Los parámetros que especifican el tamaño de los clusters y el criterio para agregar clusters nuevos (σ y *acceptance*) también fueron analizados y ajustados a mano.

Finalmente, los parámetros fueron ajustados en los siguientes valores: para el tamaño de los clusters: $\sigma = 0{,}25$, *acceptance* $= 0{,}5$; para la adaptabilidad de los clusters: $\eta_a = 0{,}1$, $\eta_s = 0{,}01$ y $\eta_q = 0{,}1$; para el algoritmo Q-Learning: $\gamma = 0{,}3$.

La figura 4.4 muestra la evolución de la medida de utilidad de la política aprendida para una instancia de aprendizaje de 250 iteraciones, con los parámetros mencionados. Se realizó una nueva medición de la utilidad de la política aprendida hasta el momento cada 5 iteraciones del algoritmo de aprendizaje, a partir del modelo aproximado. En el gráfico puede verse que a medida que avanza el aprendizaje, la medida de utilidad aumenta.

η_a	prom.	max
0.05	0.024	0.147
0.1	0.191	0.382
0.2	0.154	0.279
0.3	0.176	0.205

η_s	prom.	max
0.01	0.191	0.382
0.02	-0.110	-0.08
0.05	0.051	0.294

η_q	prom.	max
0.05	-0.058	0.044
0.1	0.191	0.382
0.2	0.142	0.264

Tabla 4.1: Variación de los parámetros η durante el aprendizaje de *mantener-distancia*.

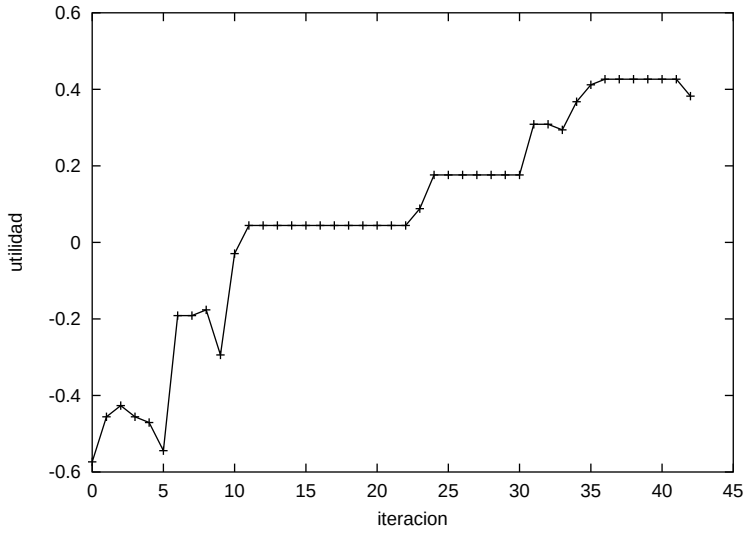


Figura 4.4: Evolución de la medida de utilidad durante el aprendizaje de *mantener distancia*. El eje x representa el número de iteración, y el eje y el valor de la función de utilidad para la política aprendida hasta el momento.

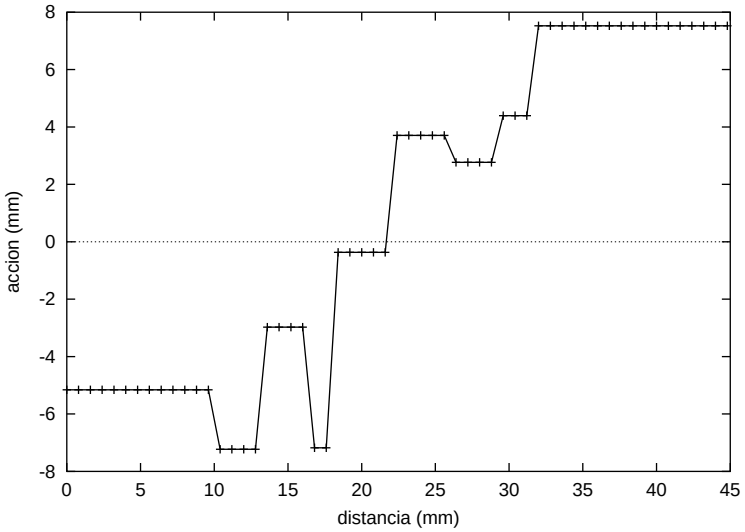


Figura 4.5: Resultado del aprendizaje de *mantener-distancia*. El eje x representa la distancia al objetivo y el eje y la acción indicada por la política óptima.

Validación de la política aprendida

La figura 4.5 muestra la acción asociada a cada situación del espacio de muestra para la política aprendida. El eje x indica la distancia entre el robot y el objetivo y el eje y la acción asignada, traducida a mm. Puede verse que, salvo excepciones, la función es decreciente respecto de la situación (cuanto más lejos están los robots, mayor es el valor de la acción asociada). Cada cluster está representado con un escalón o cambio de valor en la función.

La figura 4.6 muestra, para las mismas situaciones, el valor de los sensores frontales luego de aplicar un paso de la política óptima, también promediando las 20 iteraciones anteriores. Las líneas punteadas horizontales dividen el espacio entre las regiones de refuerzo positivo, neutro y negativo. Para el 34 % de los estados de la muestra, un sólo paso de la política óptima deja al robot dentro de la región de refuerzo positivo, y para el 37 % en la región de refuerzo neutro.

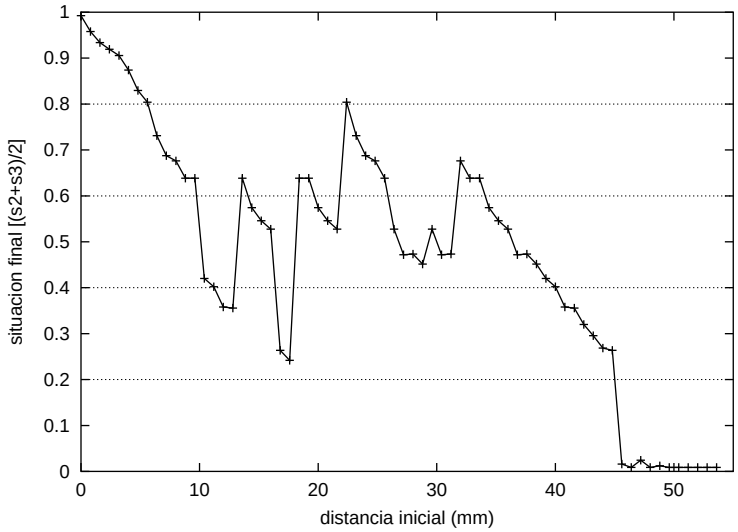


Figura 4.6: Resultado del aprendizaje de *mantener-distancia*. El eje x representa la distancia al objetivo, y el eje y el valor de los sensores frontales luego de ejecutar la acción indicada por la política.

Traducción de distancias en velocidades

Para probar el comportamiento aprendido con el objetivo en movimiento, tradujimos las distancias en velocidades y colocamos al robot objetivo siguiendo la trayectoria expresada con el siguiente algoritmo:

```
Para i entre 1 y 5, y entre -1 y -5
    Fijar velocidad en i
    Dormir(4 seg)
    Detener
    Dormir(2 seg)
```

La figura 4.7 muestra la velocidad del robot perseguidor en cada instante del experimento. Cada valor representa el promedio de velocidad de las 20 iteraciones anteriores (el intervalo de tiempo entre iteraciones fue de 60 ms aproximadamente).

Se puede ver que el robot reproduce satisfactoriamente la velocidad del objetivo para cualquier velocidad que éste tome, entre -5 y 5 unidades (hasta 40 mm/s, hacia

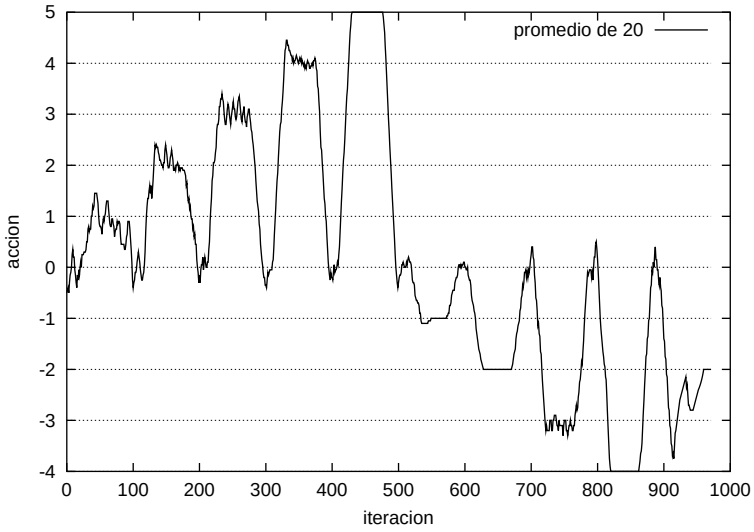


Figura 4.7: Acción ejecutada por el robot durante el experimento de *mantener-distancia* con el objetivo en movimiento.

delante y hacia atrás). Los movimientos, salvo muy pocas excepciones, son siempre en la misma dirección que el objetivo. Se puede ver también que la velocidad del robot es más estable si el objetivo se mueve a velocidades más altas. Esto puede deberse a que los movimientos demasiado lentos tienen mayor grado de error que aquellos con velocidades más altas.

La figura 4.8 muestra el valor de los sensores delanteros durante este experimento. La distancia que separa ambos robots está todo el tiempo acotada y se mantiene estable mientras el objetivo se mueve a velocidades constantes. Además, la distancia es mayor si el objetivo se mueve hacia delante con velocidades más altas, medio si el objetivo está detenido, y menor si el objetivo se mueve hacia atrás. Esto confirma el razonamiento realizado en el capítulo 3 respecto de la utilización de políticas aprendidas con el objetivo detenido en entornos con el objetivo en movimiento.

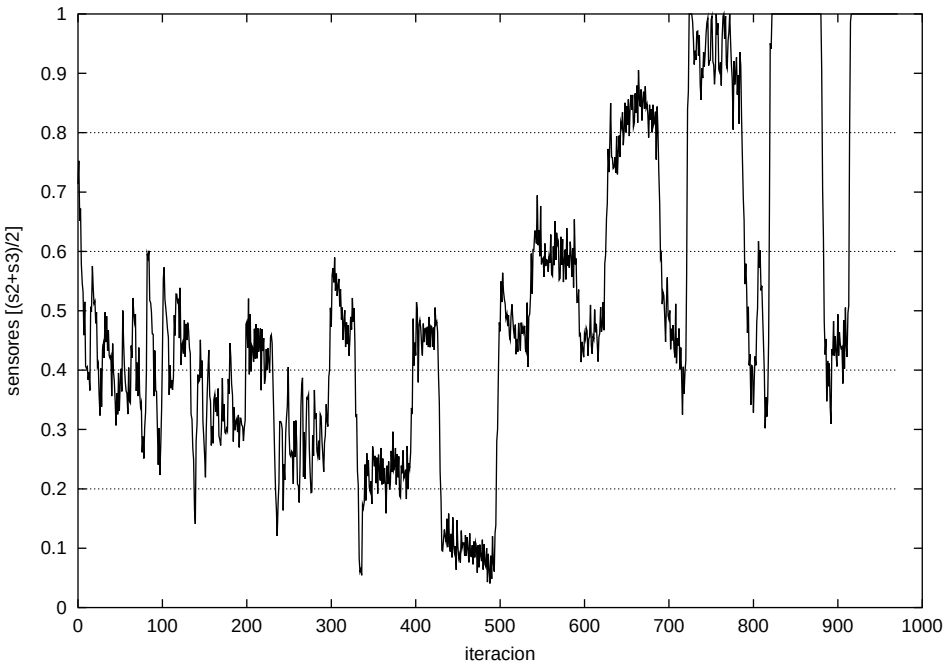


Figura 4.8: Valor de los sensores delanteros durante el experimento de *mantener-distancia* con el objetivo en movimiento.

4.2.3. Comportamiento *mantener-ángulo*

Para la definición del comportamiento *mantener-ángulo*, el espacio de situaciones utilizado fue el mismo que para el comportamiento anterior: el valor de los 8 sensores, transformado en números reales entre 0 y 1. Como posibles acciones, se consideraron ángulos de giro entre -90 y 90 grados.

La función de refuerzo empleada fue la descripta en el capítulo 2. Esta función, a diferencia de la utilizada para el comportamiento *mantener-distancia*, no tiene ningún parámetro ajustable. Con respecto al método de exploración, se utilizó el mismo mecanismo que en el otro comportamiento, con idéntico valor inicial. No se incorporó aquí ningún tipo de reflejo durante el aprendizaje.

Resultados del aprendizaje

La figura 4.9 muestra el resultado de la medida de utilidad, calculada sobre el modelo de interacción aproximado, durante el transcurso del aprendizaje. Los parámetros utilizados fueron: para el tamaño de los clusters: $\sigma = 0{,}3$, *acceptance* $= 0{,}1$; para la adaptabilidad de los clusters: $\eta_a = 0{,}1$, $\eta_s = 0{,}01$, $\eta_q = 0{,}7$; para el algoritmo Q-Learning: $\gamma = 0{,}5$.

En la figura se puede ver que la utilidad aumenta considerablemente en una primera etapa y luego se mantiene constante, e incluso se reduce ligeramente.

Validación de la política aprendida

Las figuras 4.10 y 4.11 muestran un análisis del resultado de la mejor política aprendida, a partir del modelo aproximado. La figura 4.10 muestra la acción asociada a cada situación, con ambas medidas expresadas en grados. La situación indica el ángulo del robot respecto del objetivo, y la acción, el ángulo que debe girar. Cada cluster está representado por una línea horizontal continua. Podemos ver que, en general, la acción es decreciente respecto del ángulo, y se asocian valores máximos para ángulos máximos, y valores mínimos para ángulos mínimos. Además, la acción aprendida para ángulos cercanos a cero es la acción nula. Puede verse que la acción

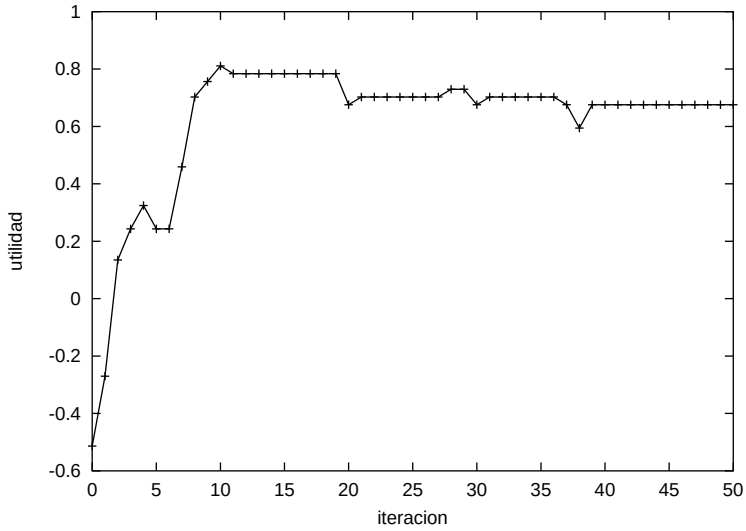


Figura 4.9: Evolución de la medida de utilidad durante el aprendizaje de *mantener-ángulo*. El eje x representa el número de iteración y el eje y el valor de la función de utilidad para la política aprendida hasta el momento.

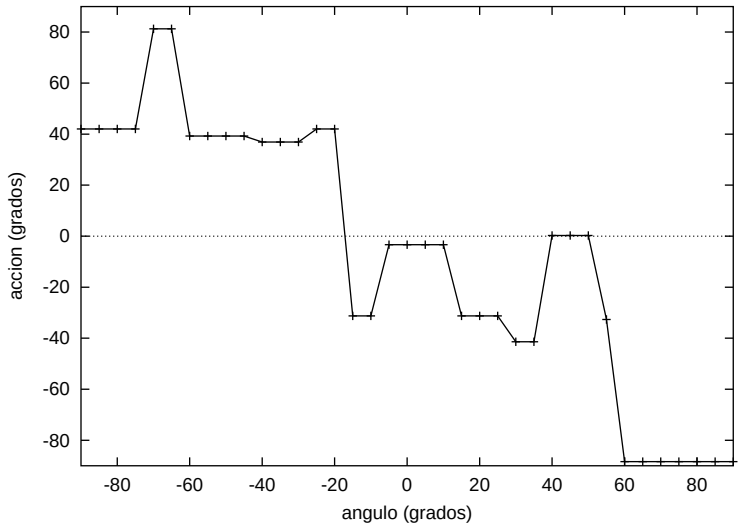


Figura 4.10: Resultados del aprendizaje de *mantener-ángulo*. El eje x representa el ángulo entre el frente del robot y el objetivo, y el eje y la acción indicada por la política óptima.

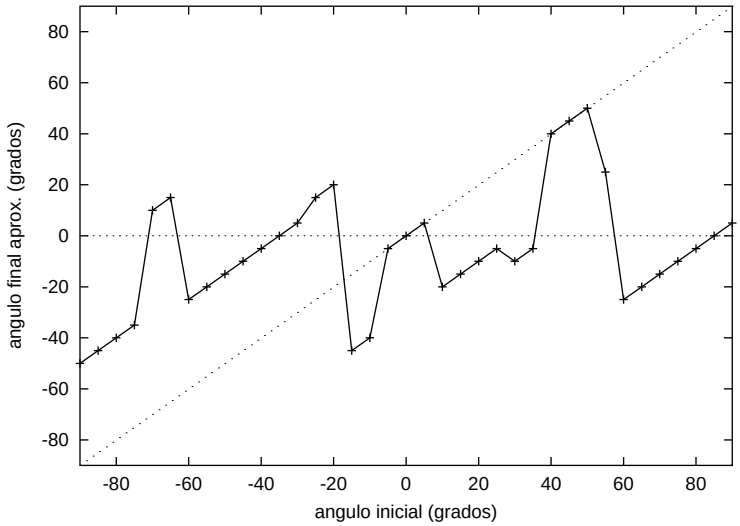


Figura 4.11: Resultados del aprendizaje de *mantener-ángulo*. Los ejes x e y representan el ángulo entre el frente del robot y el objetivo, antes y después de ejecutar la acción indicada por la política aprendida, respectivamente.

aprendida para ángulos de entre 40 y 50 grados es incorrecta (indica quedarse en el lugar), lo cuál demuestra que el comportamiento no fue aprendido correctamente en esa sección del espacio de situaciones. La figura 4.11 muestra la situación inicial y final, luego de ejecutar la acción indicada en la figura anterior. Cada línea continua con pendiente 1 representa un cluster. Se puede ver en esta figura que en casi todas las situaciones, la acción tiene la orientación correcta y, por lo tanto, acerca al robot al objetivo (ángulo nulo). También se puede ver que, siguiendo la política, el robot se mantiene detenido si ya se encuentra en la situación óptima (ángulo cero).

Traducción de distancias en velocidades

Al traducir las acciones discretas en continuas, en cada comportamiento se obtienen resultados distintos. Mientras que con *mantener-distancia* se logra imitar la velocidad lineal del objetivo, con *mantener-ángulo* se logran giros suaves: si el ángulo inicial es muy grande, la velocidad de giro es alta en un principio, pero se va

reduciendo a medida que el ángulo se achica.

Para probar esto último, ejecutamos la política traducida a velocidades con el objetivo ubicado en diferentes ángulos iniciales. La figura 4.12 muestra, para distintos ángulos, la velocidad que tomo el robot en cada instante de tiempo. La velocidad está expresada en unidades de velocidad del robot (8 mm/s aproximadamente) para la rueda izquierda. El tiempo está medido en intervalos de 60 ms (valor aproximado).

En todos los casos, el robot se detuvo al estar alineado. Para ángulos grandes, vemos que la velocidad del robot es máxima en un principio y luego se reduce escalonadamente. El robot en ningún caso se detuvo abruptamente, ni se detuvo sin estar alineado con el objetivo. Se ven en esta figura algunos casos en los cuales la velocidad aumenta en la mitad del trayecto.

4.3. Experimentos de seguimiento

Para probar la composición de los subcomportamientos y su eficacia para la tarea de seguimiento, ejecutamos la combinación de las mejores políticas aprendidas para *mantener-distancia* y *mantener-ángulo* con el robot objetivo siguiendo diferentes trayectorias a diferentes velocidades. Para medir y comparar la performance del perseguidor en cada experimento, utilizamos como medida el valor de los sensores frontales.

En cada experimento, colocamos inicialmente al robot perseguidor alineado detrás del objetivo, a una distancia de 30 mm. Las trayectorias del robot objetivo fueron: movimientos en círculos de diferentes radios, rectángulos y movimiento aleatorio.

En los dibujos de las trayectorias, los círculos representan la posición de los robots objetivo (color claro) y perseguidor (color oscuro) durante el experimento. Como elemento adicional, los segmentos que parten del centro de los círculos indican la orientación de los robots en cada momento.

Para analizar los comportamientos desde diferentes puntos de vista, vamos a estudiar, además del dibujo de las trayectorias, tres gráficos adicionales: el valor de los

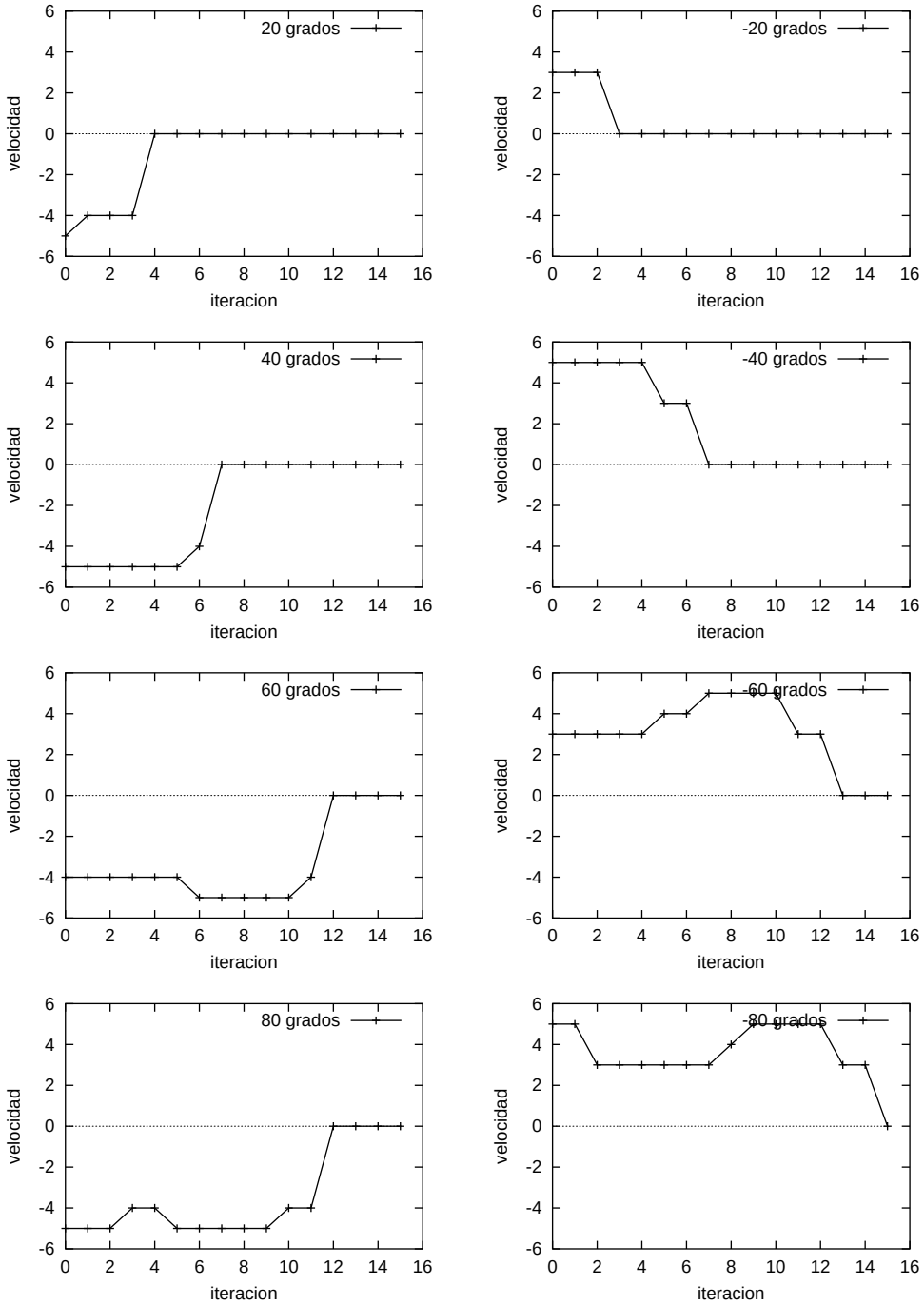


Figura 4.12: Resultados de *mantener-ángulo* traducido a velocidades angulares. Cambios de velocidad para objetivos con distintos ángulos iniciales.

sensores frontales, que nos permitirá verificar la correctitud del comportamiento, la orientación de ambos robots y la distancia recorrida por la rueda derecha, que ilustra el tipo de movimiento que los robots realizan.

La figura 4.13 muestra una trayectoria circular amplia, generada con velocidades ligeramente distintas para las dos ruedas del robot objetivo. En este caso, el robot perseguidor se mantuvo siempre detrás del objetivo, y reprodujo su trayectoria en forma exacta. Además de verificarse en el dibujo, este puede verse en el hecho de que tanto el ángulo como la distancia recorrida por las ruedas de ambos robots se mantienen en valores cercanos.

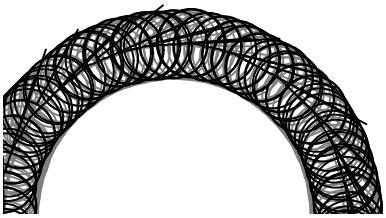
El segundo caso (figura 4.14) representa también una trayectoria circular, pero con un radio mucho menor. Aquí, el robot perseguidor pudo .ªcortar caminoz cumplir con la tarea de seguimiento recorriendo menores distancias que el objetivo. Este efecto se puede ver en el gráfico de ángulo y de contadores de las ruedas, los cuales demuestran que el perseguidor se movió menos que el objetivo pero manteniendo en todo momento su orientación similar a la de aquel.

La trayectoria rectangular se obtuvo con movimientos rectilíneos y giros de 90° en el lugar. En este experimento (figura 4.15) mientras el robot objetivo giraba, el robot perseguidor se detuvo por completo. Luego, mientras el objetivo avanzaba en línea recta en dirección perpendicular al perseguidor, éste comenzó a seguir una trayectoria circular amplia, formando un arco. Esto se puede ver claramente en el gráfico de ángulo (el objetivo cambia su ángulo en forma brusca, mientras que el perseguidor lo cambia en forma suave). La trayectoria experimentada es natural, y responde al hecho de que los movimientos del robot se basan sólo en la posición del objetivo, y no en la dirección de su movimiento. De esta manera, los cambios (graduales) de distancia y ángulo entre el objetivo y el perseguidor, son corregidos por movimientos (también graduales) del robot perseguidor. Los picos en los valores de los sensores que pueden verse a intervalos regulares se corresponden con los momentos en los que el objetivo se detuvo. El tiempo que tardó el robot en reaccionar produjo que la distancia entre

ambos disminuyera.

Por último, la figura 4.16 muestra porciones de trayectorias aleatorias, obtenidas con movimientos en línea recta seguidos por giros de ángulo aleatorio. Estas trayectorias resultan ilustrativas, porque en ellas pueden verse los resultados de las tres experiencias anteriores: imitación de la trayectoria y velocidad con movimientos rectilíneos del objetivo, giros suaves del perseguidor ante giros bruscos del objetivo y atajos para acortar camino. Se puede ver también que en cierto momento el objetivo realizó un giro de 180° y comenzó a avanzar hacia el perseguidor, disminuyendo la distancia entre ambos. La reacción del perseguidor fue moverse en línea recta hacia atrás, luego girar en el lugar, y finalmente, lograda la misma orientación del objetivo, avanzar nuevamente.

Para comparar la performance del comportamiento de seguimiento para diferentes trayectorias, en la tabla 4.2 se muestra el valor del sensor frontal clasificado en 5 intervalos (que representan valores muy bajos, bajos, medios, altos y muy altos) y se consigna, para cada experimento, el porcentaje de tiempo que el sensor frontal toma valores en cada uno de estos intervalos. Puede verse aquí que en todas las trayectorias, los sensores frontales se mantuvieron la mayor parte del tiempo en valores medios (entre 0.2 y 0.8). La mejor performance se obtuvo en las trayectorias circulares, y la peor en la trayectoria aleatoria. En los primeros dos experimentos, como el objetivo siempre se movió hacia delante, alejándose del robot, los sensores nunca devolvieron valores muy altos. En el experimento 3, los valores altos se obtuvieron cuando el objetivo se detuvo súbitamente, y en el número 4 cuando el objetivo giró más de 180 grados y comenzó a acercarse al robot.



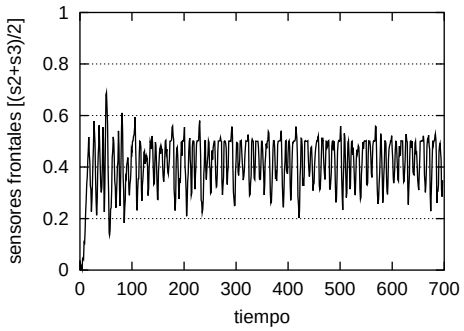


figura a)

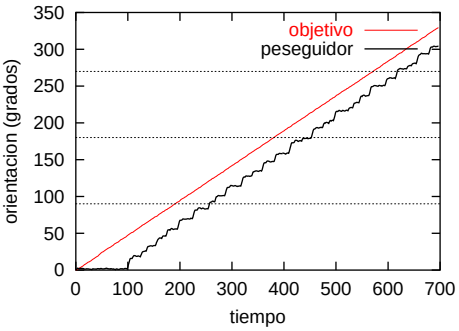


figura b)

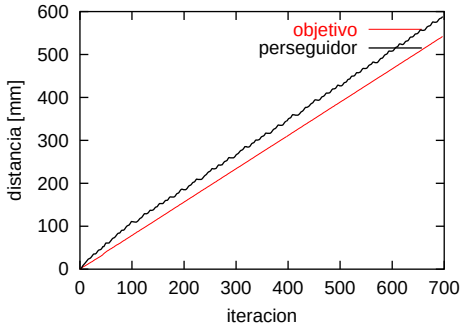
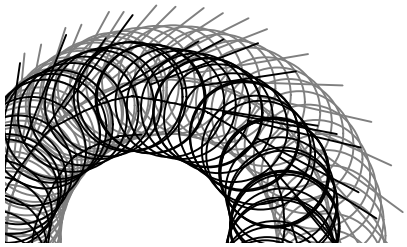


figura c)

Figura 4.13: Trayectoria de círculos amplios. a) Valor de los sensores frontales del robot perseguidor. b) Angulo de cada robot respecto del origen de coordenadas. c) Distancia recorrida por la rueda derecha de ambos robots.



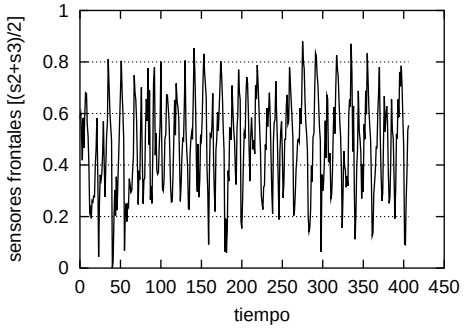


figura a)

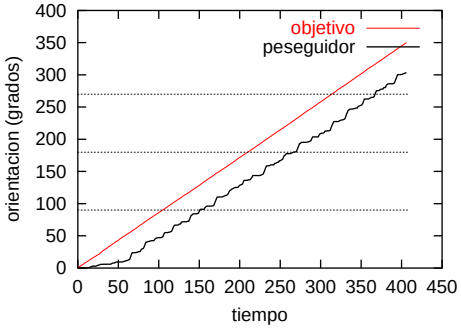


figura b)

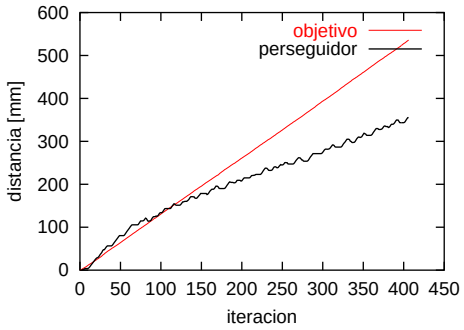
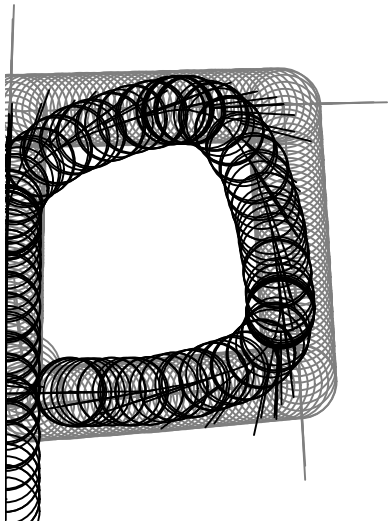


figura c)

Figura 4.14: Trayectoria de círculos cerrados. a) Valor de los sensores frontales del robot perseguidor. b) Angulo de ambos robots respecto del origen de coordenadas. c) Distancia recorrida por la rueda derecha de ambos robots.



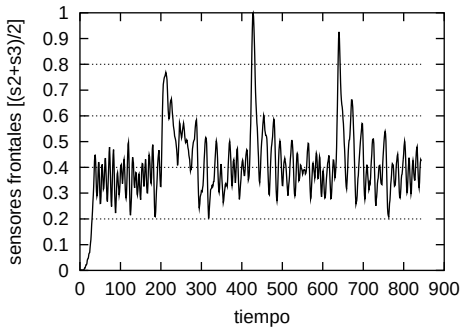


figura a)

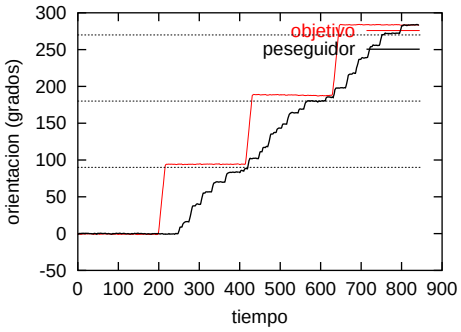


figura b)

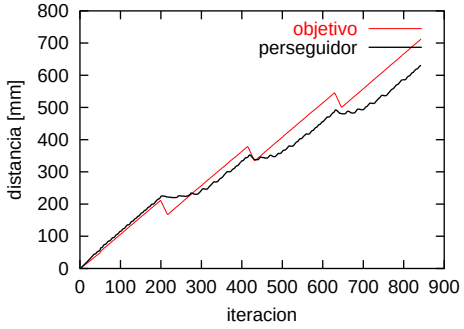
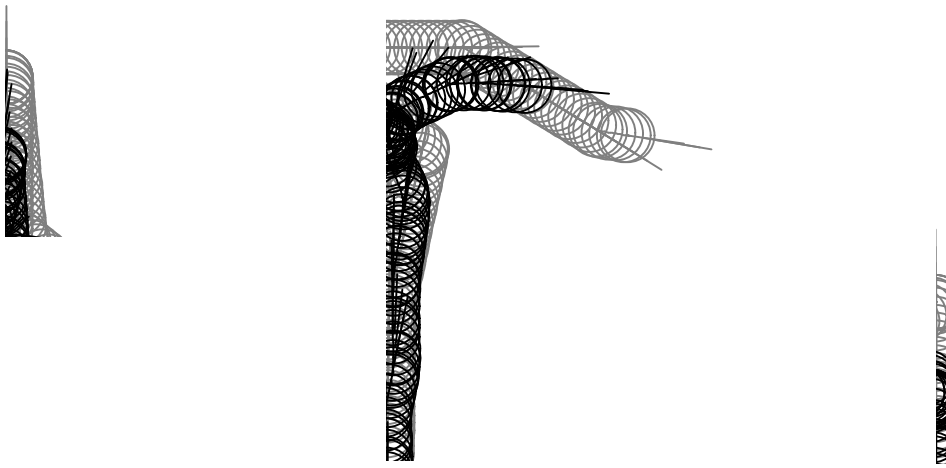


figura c)

Figura 4.15: Trayectoria rectangular. a) Valor de los sensores frontales del robot perseguidor. b) Angulo de ambos robots respecto del origen de coordenadas. c) Distancia recorrida por la rueda derecha de ambos robots.



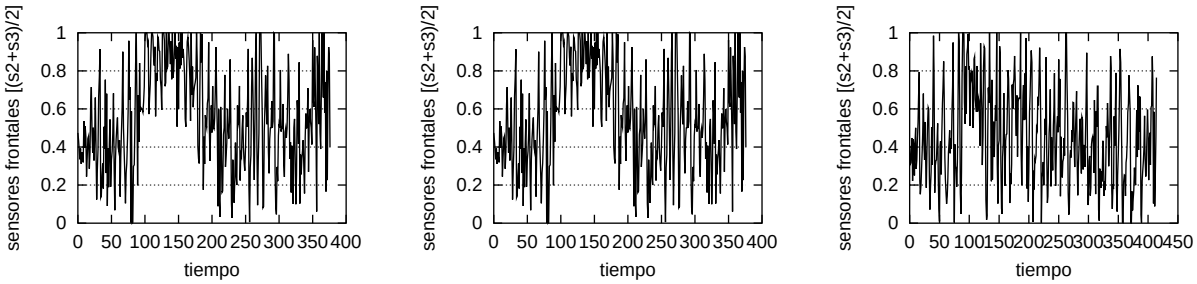


figura a)

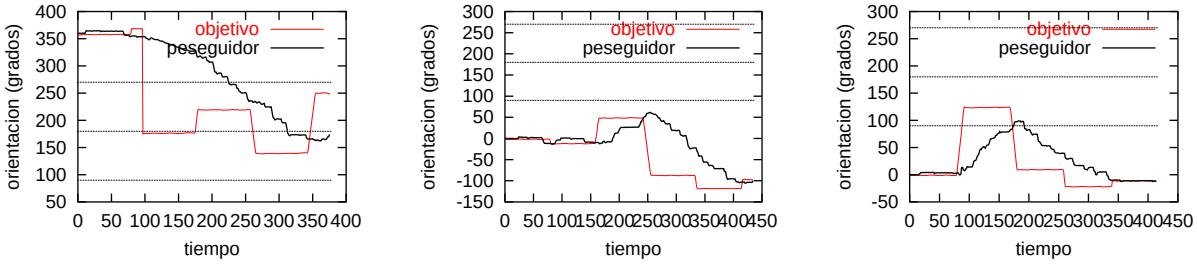


figura b)

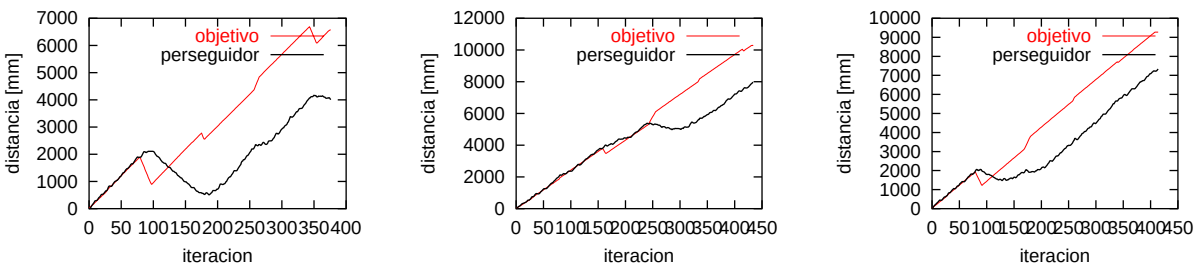


figura c)

Figura 4.16: Trayectoria aleatoria. a) Valor de los sensores frontales. b) ángulo de ambos robots respecto del origen de coordenadas. c) contador de la rueda derecha.

Prueba	Menos de 0.1	Entre 0.1 y 0.2	Entre 0.2 y 0.8	Entre 0.8 y 0.9	Más de 0.9
1 (circ. grande)	1 %	9 %	89 %	1 %	0 %
2 (circ. chico)	8 %	5 %	81 %	4 %	2 %
3 (cuadrado)	12 %	9 %	63 %	6 %	10 %
4.1 (aleatorio)	4 %	6 %	68 %	7 %	15 %
4.2 (aleatorio)	7 %	7 %	73 %	4 %	9 %
4.3 (aleatorio)	6 %	10 %	74 %	5 %	5 %

Tabla 4.2: Distribución de los valores de los sensores frontales para distintas trayectorias del objetivo.

Capítulo 5

Uso de seguimiento en formaciones de robots

En el capítulo 2, hemos mencionado el uso de seguimiento de objetivos móviles en tareas de grupos de robots y en algunas de sus aplicaciones. En este capítulo vamos a estudiar hasta qué punto los comportamientos de seguimiento obtenidos con el método propuesto en este trabajo pueden ser utilizados para la generación de formaciones de robots con diferentes formas geométricas.

En los últimos años, la investigación en el campo de los robots autónomos se ha orientado en parte hacia los sistemas multi-robot. Estos sistemas son atractivos por diversos motivos [43], entre ellos:

- Algunas tareas son demasiado complejas para ser resueltas por un único robot, o pueden ser resueltas más eficientemente con un grupo de robots.
- Desarrollar y usar muchos robots simples puede ser más sencillo, barato, flexible y con mayor tolerancia a fallas que el uso de robots más complejos.
- Los sistemas multi-robots permiten estudiar problemas presentes en ciencias sociales y biológicas.

El campo de los sistemas multi-robots ha derivado en diversas líneas de investigación y enfoques diferentes. Dudek, Jenkin y Millos [15] presentan una taxonomía de estos sistemas. Cao [43] enumera una cantidad importante de trabajos y los clasifica según la misma taxonomía, además de describir las arquitecturas más conocidas en forma completa. Parker [34] describe las líneas de investigación más importantes y menciona los avances y temas abiertos que se pueden observar en cada una de ellas.

Actualmente, existe cierto consenso en la utilización de algoritmos de control distribuido para el manejo de grupos de robots [43]. Sin embargo, hay un debate importante respecto de la información que debe manejar cada uno de los robots del grupo. Touzet [41] define una escala para el nivel de conocimiento (*robot awareness*) y hace un análisis del impacto que este nivel tiene en la dificultad para el aprendizaje de los comportamientos por medio de técnicas RL. Parker [30] estudia el balance entre información local y global que debe manejar cada robot y el impacto en la performance de las soluciones. Su conclusión es que la información local permite generar comportamientos reactivos y cumplir mejor los objetivos de corto plazo, mientras que información global mínima (especialmente en lo que respecta a la definición de los objetivos comunes) puede mejorar el cumplimiento a largo plazo de esos objetivos. Balch y Arkin [6] estudian el impacto que tiene el nivel de intercambio de información entre robots en la eficiencia de la solución obtenida para distintas tareas.

Nosotros creemos que un número importante de tareas colectivas pueden realizarse sin comunicación ni coordinación entre los robots y utilizando sensado local, con las ventajas de robustez y escalabilidad mencionadas en el capítulo 2. En ese marco, nos proponemos utilizar el comportamiento de seguimiento desarrollado en este trabajo para la generación de comportamientos grupales de formaciones.

5.1. Definición de formaciones

La tarea de formaciones ha sido muy estudiada en la literatura y es, además, la tarea grupal más relacionada con el comportamiento de seguimiento [7]. Esta tarea

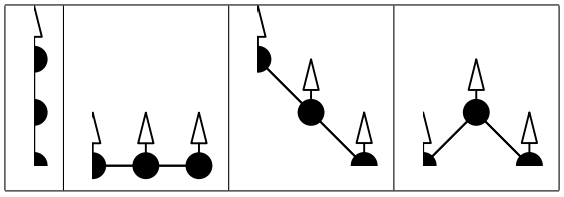


Figura 5.1: Configuraciones geométricas más utilizadas para formaciones de robots. De izquierda a derecha: fila, columna, diagonal y triángulo.

consiste en lograr que un grupo de robots establezcan entre ellos una forma geométrica predeterminada y la mantengan mientras se trasladan de un lugar a otro. Uno de los motivos por los cuales esta aplicación es atractiva para el estudio de algoritmos de control multi-robot es que la misma es fácil de especificar y verificar, pero a la vez admite distintas variantes y permite estudiar diferentes aspectos de los algoritmos de control. Por ejemplo, se puede variar las formas geométricas (desde columnas y líneas hasta estructuras complejas, pasando por formas triangulares y rectangulares), la cantidad de robots, se pueden incorporar comportamientos secundarios adicionales, como por ejemplo evitar obstáculos, o se puede requerir el cambio de la estructura en forma dinámica. Además, esta tarea requiere diferentes estrategias de control y representa desafíos diferentes en función del equipamiento disponible para sensado, comunicación y movilidad. A su vez, los comportamientos de formaciones puede ser utilizados para resolver tareas aún más complejas, como por ejemplo el traslado de objetos grandes mediante un grupo de robots o sensado distribuido de áreas amplias [43].

La tarea de formaciones para un grupo de robots autónomos consiste en lograr que éstos se ubiquen en función de una distribución espacial predeterminada y puedan mantenerla mientras se trasladan de un lugar a otro. Las formas geométricas más comúnmente utilizadas en la bibliografía son: columna, fila, diagonal triángulo (ver fig. 5.1).

A partir de la definición de seguimiento dada en el capítulo 3, las formaciones que surgen son de tipo fila, ya que el objetivo se mantiene alineado con el frente del robot.

Para obtener otro tipo de formaciones, vamos a dar una definición paramétrica del predicado de seguimiento que permita variar el ángulo con el que deben esta alineado el objetivo y que, al aplicarla a un grupo de robots, nos va a permitir obtener otro tipo de formaciones:

$$follow'(t) \Leftrightarrow (d - \epsilon_d \leq \lambda(t) \leq d + \epsilon_d) \wedge (\alpha - \epsilon_\alpha \leq \theta(t) \leq \alpha + \epsilon_\alpha), \qquad (5.1)$$

donde α es el ángulo en el cuál se debe mantener el objetivo respecto del perseguidor (este ángulo es cero en la definición de seguimiento que dimos antes). De esta manera, para obtener formaciones en diagonal, utilizaremos esta definición con $\alpha = 45°$ y para formaciones en columna, $\alpha = 90°$. Para formaciones en triángulo, se deberán utilizar parámetros distintos para aprender el comportamiento en cada robot: la mitad de ellos deberán utilizar $\alpha = 45°$ y la otra mitad $\alpha = -45°$.

Según nuestro enfoque, las formaciones pueden ser vistas como comportamientos emergentes, a partir de comportamientos individuales de seguimiento, o de variantes del mismo. Como ya mencionamos, en las formaciones en fila todos los robots deben ejecutar el comportamiento de seguimiento tal cuál fue definido en el capítulo 3. Para obtener otro tipo de formas, modificaremos las funciones de refuerzo para adaptarlas a la tarea que ahora deben cumplir. El único cambio que debemos realizar consiste en considerar para los refuerzos el o los sensores que deberían alinearse con el robot de adelante, dependiendo de la forma que se busque. Por ejemplo, para funciones en diagonal, las funciones de refuerzo son exactamente las mismas, pero considerando como sensor frontal el promedio entre los sensores 0 y 1:

$$mddiag(s_0..s_n) = \begin{cases} 1 & \text{si } |(s_0+s_1)/2 - 0{,}5| < \sigma_1 \\ 0 & \text{si } \sigma_1 \leq |(s_0+s_1)/2 - 0{,}5| \leq \sigma_2 \\ -1 & \text{si } \sigma_2 < |(s_0+s_1)/2 - 0{,}5| \end{cases}, \tag{5.2}$$

$$madiag(s_0..s_n) = \begin{cases} 1 & \text{si } i^* = 0 \vee i^* = 1 \\ 0 & \text{si } i^* = 2 \\ -1 & \text{en caso contrario} \end{cases},$$

5.2. Resultados experimentales

Para probar las formaciones, se realizaros dos tipos de experimentos. En el primero se utilizó el comportamiento de seguimiento original para la obtención de formaciones en fila. En el segundo se utilizó la definición modificada de seguimiento para obtener formaciones en diagonal. En todos los experimentos, cada robot realizó el aprendizaje en forma individual, para lograr un comportamiento que se adapte a las particularidades de sus sensores. En ambos casos se realizaron experimentos con grupos de 3 y 4 robots Khepera.

5.2.1. Formaciones en fila

Para los experimentos de formaciones en fila, los robots fueron colocados inicialmente alineados, aunque no a distancia óptima. En todos los robots excepto en el de adelante, que actuó como líder, se inició la ejecución del comportamiento de seguimiento, lo que produjo oscilaciones hacia delante y hacia atrás en los robots, hasta que todos ellos alcanzaron la distancia óptima, momento en el cuál todos se detuvieron por completo. Luego, se inició la ejecución en el líder de la trayectoria preestablecida. Las experimentos realizados incluyeron trayectorias de círculos amplios y movimiento en zigzag.

En todos los experimentos los robots lograron mantener la formación. La figura 5.2 muestra los movimientos de los robots para las dos trayectorias. Aquí puede

Trayectoria circular

Robot	Menor a 0.1	Entre 0.1 y 0.2	Entre 0.2 y 0.8	Entre 0.8 y 0.9	Mayor a 0.9
1	0 %	2 %	98 %	0 %	0 %
2	1 %	9 %	87 %	1 %	2 %
3	3 %	23 %	66 %	3 %	5 %

Trayectoria zigzag

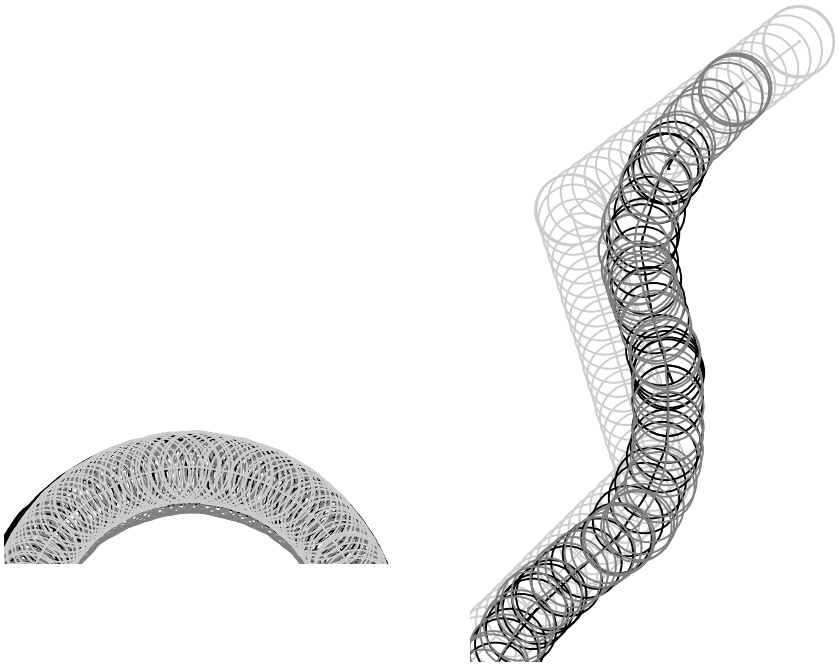
Robot	Menor a 0.1	Entre 0.1 y 0.2	Entre 0.2 y 0.8	Entre 0.8 y 0.9	Mayor a 0.9
1	8 %	9 %	71 %	4 %	8 %
2	11 %	12 %	57 %	5 %	15 %

Tabla 5.1: Distribución de los valores del sensor frontal en cada robot para formaciones en fila.

verse que ningún robot se apartó de la formación, y que todos los robots siguieron aproximadamente la misma trayectoria (que puede ser distinta a la del líder, como en el caso de movimientos en zigzag). La tabla 5.1 muestra el porcentaje de tiempo que los sensores delanteros del robot permanecen dentro de cada rango de valores. El robot número 1 es el primero de la formación, luego del líder. Nótese que, en forma sistemática, el porcentaje de tiempo que los sensores toman valores medios es decreciente con el número de robot, lo cuál indica que la performance del comportamiento se degrada con la cantidad de robots que separan al robot en cuestión del líder. Este fenómeno es razonable, y puede explicarse por el hecho de que los movimientos de un robot se ven amplificados por los robots que lo siguen, y cualquier variación mínima en la trayectoria de los robots delanteros producen cambios importantes en los de atrás.

5.2.2. Formaciones en diagonal

Para probar formaciones en diagonal, se utilizaron las funciones de refuerzo modificadas. Antes de verificar los resultados en formaciones, realizamos unas pequeñas



a) b)

Figura 5.2: Formación en columna. a) Trayectoria de círculos amplios y b) Trayectoria en zigzag.

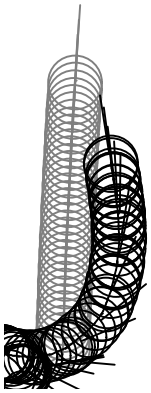


Figura 5.3: Comportamiento de seguimiento en diagonal. El robot logra alcanzar la posición óptima ubicado inicialmente con el objetivo del lado opuesto al deseado.

pruebas de este comportamiento en forma individual. En ellas el robot fue capaz de seguir en diagonal a un objetivo cuando éste se movía a baja velocidad y con trayectorias no demasiado bruscas, y fue capaz de alcanzar la posición correcta a partir de una variedad importante de posiciones iniciales (ver figura 5.3). A velocidades más altas o con trayectorias más complejas, el robot no logró evitar perder de vista al objetivo. Además, para las pruebas exitosas, la calidad del comportamiento fue menor respecto del seguimiento frontal, y el robot experimentó algunos movimientos oscilatorios y no fue capaz de imitar en todo momento la velocidad del objetivo.

Creemos que esto se debe a varios motivos, entre otros el hecho de que la capacidad de sensado de los robots es mucho mayor en el frente que en diagonal, y que la cantidad de movimientos del robot que hacen que el objetivo permanezca dentro del área de sensado es menor aún que en el caso de seguimiento en línea recta.

Para probar las formaciones en diagonal, realizamos dos tipos de experiencias. En las primeras, los robots partireron de una posición inicial de fila, con robot delantero actuando como líder y moviéndose lentamente en línea recta. En esta experiencia se verificó la capacidad de alcanzar la formación y manentenerla con una trayectoria

Movimiento rectilíneo a partir de formación en fila

Robot	Menor a 0.1	Entre 0.1 y 0.2	Entre 0.2 y 0.8	Entre 0.8 y 0.9	Mayor a 0.9
1	9/ %	11 %	79 %	0 %	1 %
2	8 %	15 %	67 %	7 %	3 %
3	20 %	14 %	53 %	4 %	9 %

Giro abrupto del líder

Robot	Menor a 0.1	Entre 0.1 y 0.2	Entre 0.2 y 0.8	Entre 0.8 y 0.9	Mayor a 0.9
1	0 %	2 %	93 %	3 %	4 %
2	9 %	9 %	69 %	6 %	7 %

Tabla 5.2: Distribución de los valores del sensor lateral en cada robot para formaciones en diagonal.

muy simple. El resultado de esta experiencia fue exitoso (ver figura 5.4). Las segundas experiencias tuvieron como objetivo verificar el comportamiento con una trayectoria más compleja. Para ello, los robots partireron de una posición inicial óptima (alineados en diagonal). La trayectoria del líder incluyó un giro de 90° luego de unos pocos segundos de comenzada. En este caso el resultado también fue exitoso: los robots que lo seguían no lo perdieron de vista y fueron capaces de acomodarse en diagonal con la nueva orientación (ver figura 5.5).

Al igual que en el caso de formaciones en fila, para comparar la performance de los robots, utilizamos como medida el porcentaje de tiempo que los sensores de cada robot reciben valores dentro de distintos rangos, sólo que considerando los sensores 0 y 1 en lugar de los frontales (tabla 5.2). Nótese que estos valores no pueden utilizarse para comparar el comportamiento de formaciones en diagonal respecto de las formaciones en fila, ya que los sensores diagonales representan una superficie y un ángulo de giro mucho mayor que los sensores frontales, cuyas areas de sensado se encuentran casi superpuestas. En esta tabla se verifica que, al igual que en las formaciones en file, la calidad del comportamiento se reduce sistemáticamente con la distancia al líder.

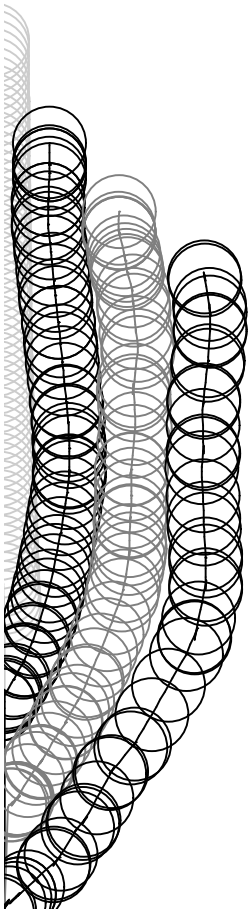


Figura 5.4: Formaciones en diagonal. Los robots parten de una formación en fila y alcanzar una formación en diagonal mientras avanzan hacia delante siguiento al líder.

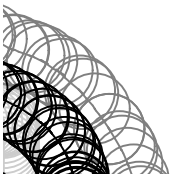


Figura 5.5: Formaciones en diagonal. El líder realiza un giro de 90° en el lugar y los demás robots logran mantener la formación.

Capítulo 6

Conclusiones y trabajo futuro

En este trabajo estudiamos la tarea de seguimiento de objetivos móviles en el contexto de robots autónomos de sensado local. La complejidad de esta tarea depende en gran medida de la capacidad de sensado del robot: en nuestro caso, trabajamos con robots dotados únicamente de sensores infrarrojos de muy corto alcance. Analizamos tres técnicas para la síntesis de comportamientos y vimos que todas son difíciles de aplicar a la tarea de seguimiento. Las dificultades más importantes son: el grado de error de los sensores es muy grande y depende de variables externas, no es sencillo generar un modelo del sistema a partir de los valores de los sensores, el espacio de situaciones y acciones es demasiado grande y su exploración se ve dificultada por el reducido alcance de los sensores.

Presentamos luego un método que nos permite aplicar técnicas de aprendizaje por refuerzo a este problema, y obtener comportamientos en tiempos de aprendizaje muy cortos. Este método está basado en: 1) la división de la tarea de seguimiento a partir de la transformación del espacio de acciones, 2) el aprendizaje de las subtareas con el objetivo detenido, 3) la ejecución simultánea de las acciones de ambos subcomportamientos a partir de su combinación en una única acción y 4) la superposición de los ciclos de control. Utilizamos también un método de clustering basado en redes neuronales artificiales, denominado QRBF, que permite representar, en una misma red y en forma uniforme, el espacio de situaciones, acciones y utilidad (valores

q). El método presentado soluciona las dificultades que surgen con el uso de RL en problemas de robots reales: los espacios de situaciones y acciones se ven reducidos y simplificados y los entornos estáticos nos permiten mejorar la exploración y obtener refuerzos inmediatos. La superposición de los ciclos de control hace que las acciones incorrectas sean abortadas y corregidas luego de un breve lapso de tiempo. La técnica de síntesis de comportamientos presentada en este trabajo puede ser aplicada en el desarrollo de otros comportamientos de robots autónomos, en particular para tareas de navegación.

Los comportamientos de seguimiento obtenidos fueron capaces de seguir a objetivos que se movían con diversas trayectorias y velocidades. Los movimientos de los robots fueron suaves, con giros y cambios de velocidad graduales. En las experiencias puede verse que cuando el objetivo se detiene, el robot también lo hace. Además, se ve cómo los movimientos suaves del robot hacen que éste realice .ªtajos"en su trayectoria cuando el objetivo realiza movimientos bruscos (por ejemplo, giros en el lugar). Además, a pesar de que el aprendizaje se realizó con el objetivo detenido, al ejecutar el comportamiento con el objetivo en movimiento, el robot logra imitar la velocidad del objetivo. Esta característica emergente fue analizada formalmente y verificada en los experimentos.

Luego, aplicando este comportamiento a un grupo de robots, obtuvimos formaciones de fila de manera emergente y sin ningún tipo de comunicación entre los robots, sin control central y sin sensores externos. Los robots fueron capaces de mantener la formación siguiendo trayectorias rectilíneas, circulares y en zigzag a diferentes velocidades, indicadas por movimientos en el robot delantero. Lograron también aquí imitar la velocidad del robot de adelante, y detenerse por completo (luego de un período de acomodamiento) cuando se detenía el líder. Puede notarse que la performance en los robots delanteros es superior a la de los robots de atrás. Esto se debe en parte a que los movimientos de un robot son amplificados por los robots que lo siguen, con lo que pequeñas oscilaciones o variaciones en el robot de adelante producen cambios importantes en los robots de atrás.

Utilizando el mismo método y variando la función de refuerzo, obtuvimos comportamientos de seguimiento en diagonal. Analizándolos en forma individual, estos comportamientos fueron menos estables y robustos que los de seguimiento frontal: en muchos casos los robots perdían de vista a su objetivo, realizaban movimientos incorrectos u oscilaban demasiado. Tampoco fueron capaces de imitar con exactitud la velocidad del objetivo. Esto posiblemente se deba en parte a que la capacidad de sensado de nuestros robots es mayor en la zona frontal, y es nula si el objetivo se encuentra an 90°, y a que para el seguimiento en diagonal es menor la cantidad de acciones adecuadas en cada situación. De todas maneras, aplicando el comportamiento en formaciones se logró que los robots alcancen la posición correcta y puedan mantenerla con el robot delantero avanzando en línea recta e incluso realizando giros bruscos.

Con los resultados obtenidos comprobamos la factibilidad de síntetizar comportamientos relativamente complejos con equipos de sensado limitados, y de obtener comportamientos colectivos emergentes sencillos a partir de comportamientos individuales. Creemos que esto es importante en el campo de robots cooperativos porque permite soluciones más flexibles, robustas y escalables que aquellas que se basan en la comunicación entre robots o en sensores compartidos. Creemos además que soluciones como la presentada en este trabajo pueden resultar también de utilidad como "ladrillos"para la construcción de tareas colectivas complejas, optimizando el uso de los recursos disponibles y liberando sensores sofisticados y ancho de banda de comunicación para subtareas de más alto nivel.

Hemos detectado diversas áreas para la continuación de este trabajo. Sería interesante aplicar el método presentado en otros problemas, tanto para tareas de un solo robot como para sistemas multi-robot. Las pruebas preliminares que hemos hecho permiten suponer que puede ser utilizado con éxito para seguimiento de paredes (*wall following*) y corredores, traslado de cajas (*box pushing*) y observación de múltiples objetivos (*CMOMMT*).

Respecto de la descomposición del espacio de acciones, sería interesante estudiar

de qué forma este mismo método puede aplicarse en acciones producidas por otro tipo de actuadores, como por ejemplo brazos mecánicos.

Vimos que los comportamientos obtenidos con esta técnica, aunque funcionan adecuadamente, no son los óptimos. Una idea interesante es utilizarlos para mantener controlado el entorno (en el caso de seguimiento, para no perder de vista al objetivo) durante el aprendizaje de la política óptima en el entorno real, o utilizarlos como base para la construcción de la política óptima en forma incremental a partir de éstos. Para completar la autonomía completa de los robots, sería también interesante incluir algún tipo de adaptación de la política óptima durante la ejecución, o de aprendizaje continuo para adaptarla a los cambios del entorno (por ejemplo, día/noche) o a fallas en los sensores.

Por último, respecto de la solución a la tarea de formaciones obtenida en este trabajo, todavía dista en eficiencia y capacidad de las soluciones que utilizan equipos más sofisticados. Sería interesante, para mejorar esta solución, la incorporación de un comportamiento de evitar obstáculos, alguna técnica de predicción de la posición del objetivo para los casos en los cuales éste es perdido de vista (por ejemplo, a partir de los últimos valores de los sensores o de las últimas acciones), la prueba de esta técnica en formaciones con otras formas geométricas y el cambio dinámico de la forma durante la ejecución. Como hemos verificado que la performance de los comportamientos se degrada con la cantidad de robots, otro estudio interesante resultaría de cuantificar esta degradación y estudiar si existe un límite en cuanto a la cantidad de robots para el uso de este método, y evaluar soluciones a este problema.

Referencias

[1] James Albus. A new approach to manipulator control: The cerebellar model articulation controller. *Transactions of the ASME*, pp. 220–227, 1975.

[2] C.W. Anderson. Q-learning with hidden-unit restarting. En *Advances in Neural Information Processing Systems 5*, pp. 81–88, 1993.

[3] Ronald Arkin. Motor schema based navigation for a mobile robot: An approach to programming by behavior. En *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 264–271, 1987.

[4] Ronald Arkin. *Behavior-Based Robotics*. MIT Press, 1998.

[5] Tucker Balch. Behavior-based formation control for multi-robot teams. *IEEE Transactions on Robotics and Automation*, 14(6):1–15, 1999.

[6] Tucker Balch and Ronald Arkin. Communication in reactive multiagent robotic systems. *Autonomous Robots*, 1:27–52, 1994.

[7] Diego Bendersky. Robot formations as an emergent collective task using target-following behavior. Argentinian Symposium on Artificial Intelligence (ASAI) 2002.

[8] Diego Bendersky and Juan Santos. *Reinforcement Learning for the Target Following Behavior Using Task Decomposition by Action*, En *Intelligent Robots: Vision, Learning and Interaction*. KAIST Press (en prensa), 2003.

[9] Andrea Bonarini and Filippo Basso. Learning to compose fuzzy behaviors for autonomous agents. *Int. Journal of Approximate Reasoning*, 17(4):409–432, 1997.

[10] Rodney Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23, 1986.

[11] J. del R. Millàn. Rapid, safe and incremental learning of navigation strategies. *IEEE Transactions on Systems, Man and Cybernetics. Special issue on Learning Autonomous Robots*, 26(3):408–420, 1996.

[12] E. Franzi F. Mondada and P. Ienne. Mobile robot miniaturization: A tool for investigation in control algorithms. En *Proceedings of the Third International Symposium on Experimental Robotics*, Kyoto, Japan, 1993.

[13] Jakob Fredslund and Maja Mataric. A general, local algorithm for robot formations. *IEEE Transactions on Robotics and Automation, special issue on Multi Robot Systems*, 2002.

[14] Bernd Fritzke. Growing cell structures – a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9):1441–1460, 1994.

[15] Michael Jenkin Gregory Dudek and Evangelos Milios. *Robot Teams: From Diversity to Polymorphism*, chapter 1: A Taxonomy of Multirobot Systems. A.K. Peters Ltd., 2002.

[16] A. Koch John Hertz and R. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley Publishing, 1991.

[17] David Jung. *An architecture for Cooperation among Autonomous Agents*. PhD thesis, Intelligent Robotics Laboratory, University of Wollongong, Australia, 1998.

[18] K-Team. *Khepera User Manual, version 5.02*, 1999.

[19] P. Kaelbling and A. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, pp. 237–285, 1996.

[20] Jonas Karlsson. *Learning to Solve Multiple Goals*. PhD thesis, University of Rochester, Rochester, New York, 1997.

[21] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. En *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 500–5005, 1985.

[22] Teuvo Kohonen. *Self-Organization and Associative Memory*. Springer Series in Information Sciences. Springer Verlag, 1988.

[23] R. Matthew Kretchmar and Charles W. Anderson. Comparison of cmacs and radial basis functions for local function approximators in reinforcement learning. En *Proceedings of the International Conference on Neural Networks*, 1997.

[24] Long-Ji Lin. *Reinforcement Learning for Robots Using Neural Networks*. PhD thesis, Carnegie Mellon University, Pittsburgh, 1993.

[25] Maja Mataric. Learning to behave socially. En *Proceedings, From Animals to Animats 3, Third International Conference on Simulation of Adaptive Behavior*, pp. 453–462. MIT Press, 1994.

[26] Maja Mataric. Reward functions for accelerated learning. En *Machine Learning: Proceedings of the Eleventh International Conference*. Morgan Kaufmann Publishers, 1994.

[27] Maja Mataric. Designing and understanding adaptive group behavior. *Adaptive Behavior*, 4(1):51–80, 1995.

[28] Roxana Matuk and Juan Santos. The clustering aliasing problem in reinforcement learning for robots. En *Proceedings of the Fifth European Workshop on Reinforcement Learning*, pp. 33–35, Utrecht, The Netherlands, 2001.

[29] John Moody and Christian Darken. Fast learning in networks of localy-tuned processing units. *Neural Computation*, 1:281–294, 1999.

[30] Lynne Parker. Designing control laws for cooperative agent teams. En *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, pp. 582–587, 1993.

[31] Lynne Parker. Cooperative robots, 2000. Escuela de Ciencias Informáticas, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires.

[32] Lynne E. Parker. *Behavior-based cooperative robotics applied to multi-target observation*, En *Intelligent robots: Sensing, modeling, and planning*, pp. 356–373. World Scientific, 1997.

[33] Lynne E. Parker. Alliance: An architecture for fault-tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240, 1998.

[34] Lynne E. Parker. Current state of the art in distributed autonomous mobile robotics. *Distributed Autonomous Robotic Systems*, 4:3–12, 2000.

[35] Lynne E. Parker and Claude Touzet. Multi-robot learning in a cooperative observation task. En *Proceedings of Fifth International Symposium on Distributed Autonomous Robotic Systems*, 2000.

[36] J. Rosenblatt. Damn: A distributed architecture for mobile navigation. *Working Notes AAAI 1995 Spring Symposium on Lessons Learned for Implemented Software Architectures for Physical Agents*, March 1995.

[37] Juan Miguel Santos. *Contribution to the study and the design of reinforcement functions*. PhD thesis, Universidad de Buenos Aires, Universite d'Aix-Marseille III, 1999.

[38] Sebastian Thrun. Efficient exploration in reinforcement learning. Technical Report CMU-CS-92-102, Carnegie-Mellon University, 1992.

[39] Claude Touzet. Neural reinforcement learning for behaviour synthesis. *Robotics and Autonomous Systems*, 22:251–281, 1997.

[40] Claude Touzet. Bias incorporation in robot learning. submitted for publication in Autonomous Robots, 1998.

[41] Claude Touzet. Robot awareness in cooperative mobile robot learning. *Autonomous Robots*, 8(1):87–97, 2000.

[42] C. J Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, University of Cambridge, 1989.

[43] Alex Fukunaga Y. Cao and Andrew Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4:1–23, 1997.

[44] Ping Zhang and Stphane Canu. Indirect adaptive exploration in entropy based reinforcement learning. En *Proceedings of the International Conference on Artificial Neural Networks*, pp. 354–359, 1995.