Instrucción de tareas a un robot con retroalimentación en línea proporcionada por voz

por

Ana Cecilia Tenorio González

Tesis sometida como requisito parcial para obtener el grado de

Maestro en Ciencias en el Área de Ciencias Computacionales en el

Instituto Nacional de Astrofísica, Óptica y Electrónica

Supervisada por:

Dr. Eduardo F. Morales Manzanares, INAOE

©INAOE 2010
El autor otorga al INAOE el permiso de reproducir y distribuir copias
en su totalidad o en partes de esta tesis

Instrucción de tareas a un robot con retroalimentación en línea proporcionada por voz

Tesis de Maestría

Ana Cecilia Tenorio González

Asesor:

Eduardo F. Morales Manzanares

Instituto Nacional de Astrofísica, Óptica y Electrónica

Coordinación de Ciencias Computacionales

Diciembre 2010

Resumen

Los robots se encuentran inmersos en nuestra vida diaria cada vez más y por consiguiente necesitan estar capacitados para cumplir satisfactoriamente tareas comunes en ambientes compartidos con humanos. Específicamente, los robots de servicio requieren adaptarse a los ambientes cambiantes donde operan, interactuar de forma natural con humanos inexpertos (sin conocimientos de robótica) y trabajar con restricciones de tiempo. Tratando de resolver estas necesidades han surgido diversos métodos, entre ellos, el aprendizaje por refuerzo y el aprendizaje por demostración. Estos métodos han sido ampliamente usados y aunque han dado buenos resultados, tienen algunos problemas que deben ser resueltos. En aprendizaje por refuerzo se tienen largos tiempos de entrenamiento y problemas con los métodos que trabajan con espacios continuos, que suelen requerir mucha experiencia y por lo tanto consumir mucho tiempo en el entrenamiento, e incluso pueden no llegar a converger. *Reward shaping* ha sido usado en los algoritmos de aprendizaje por refuerzo para acelerar el aprendizaje, sin embargo, requiere conocimiento *a priori* y sus funciones son estáticas porque no son ajustables durante el proceso de aprendizaje. Por otro lado, el éxito del aprendizaje por demostración es dependiente de las habilidades del usuario que proporciona los ejemplos de la tarea al robot, y no cubre todo el espacio de posibilidades dentro del dominio de la tarea. Abordando esta problemática, en esta tesis se presenta un algoritmo de aprendizaje por refuerzo basado en Sarsa(λ) que incluye demostración inicial de la tarea proporcionada por voz, además de recibir los refuerzos usuales de los algoritmos tradicionales incluye retroalimentación en línea a través de comandos y calificativos proporcionados verbalmente por un usuario. La voz proporciona un medio de instrucción natural, asequible para usuarios inexpertos, y su inclusión funciona como una técnica de *reward shaping* sobre el algoritmo de aprendizaje. Pero a diferencia de los enfoques de *reward shaping* más usados, la retroalimentación por voz puede variar en el tiempo, por lo cual funciona como una técnica dinámica que no requiere conocimiento ni diseños previos. Al mismo tiempo se propone una representación sencilla y novedosa para espacios continuos que puede ser construida en línea. Los experimentos realizados en tareas de navegación y en una tarea de manipulación con el algoritmo propuesto muestran que éste es capaz de trabajar con espacios continuos y con retroalimentación en línea, y que su uso puede reducir significativamente el tiempo de aprendizaje en comparación con algoritmos de refuerzo tradicionales obteniendo políticas que resuelven las tareas de manera muy similar a éstos últimos.

Abstract

Robots are increasingly common in our daily lives and therefore they need to work in environments shared with humans. In service robotics they need to adapt to changing environments, interact naturally with non-expert users and also work with time restrictions. Trying to solve these needs, some methods have been proposed to program robots for those situations, between them, reinforcement learning and learning by demonstration. These methods have been widely used and although good results have been obtained with them, they have some problems that should be solved. Reinforcement learning has long times for training and some problems with methods working in continuous spaces, which require a lot of experience and therefore spend so long, and sometimes they do not converge. Reward shaping has been used in reinforcement learning algorithms to accelerate learning, however, it requires *a priori* domain knowledge and therefore, it is static because it can not be adjusted during the learning process. On the other hand, the success of learning by demonstration is based on the knowledge and the abilities of the user who provides examples to the robot, and also this learning does not cover all the space of possibilities in the task domain. Addressing these problems, this thesis presents an algorithm of reinforcement learning based on Sarsa(λ), with initial task demonstration by voice, and includes additional on-line feedback to the traditional reinforcements, feedback is provided through commands and qualifiers by voice. Speech provides a way of natural instruction, accessible to non-expert users, and its inclusion works as a reward shaping method in the learning algorithm. Unless the most widely used reward shaping approaches, additional feedback provided by voice is variable along time, so it works as a dynamic method of reward shaping that does not need a prior knowledge or designs (of functions). At the same time a new simple representation to work on-line with continuous spaces is proposed. Experiments done with navigation tasks and one handling task show how the proposed algorithm works with continuous spaces and on-line feedback, and how learning time can be reduced significantly compared to traditional reinforcement learning algorithms, obtaining very similar policies.

Índice general

Tabla de Contenido VII

Índice de Figuras XI

Índice de Tablas XIII

Índice de Algoritmos XV

1. Introducción 1
- 1.1. Antecedentes . . . 1
- 1.2. Problemática . . . 2
- 1.3. Motivación . . . 6
- 1.4. Objetivos . . . 7
- 1.5. Alcances y limitaciones . . . 7
- 1.6. Investigación realizada . . . 8
- 1.7. Guía del documento . . . 9

2. Marco Teórico 11
- 2.1. Aprendizaje por Refuerzo . . . 11
  - 2.1.1. Procesos de Decisión de Markov . . . 13
  - 2.1.2. Algoritmos de aprendizaje por refuerzo basados en diferencias temporales (TD) . . . 16
  - 2.1.3. *Reward Shaping* . . . 19
  - 2.1.4. Generalización y aproximación de funciones . . . 22
- 2.2. Aprendizaje por demostración . . . 24
- 2.3. Reconocimiento del habla . . . 26
  - 2.3.1. Modelo del lenguaje . . . 27
  - 2.3.2. Modelo acústico . . . 28
  - 2.3.3. Léxico y diccionario de pronunciación . . . 28
- 2.4. Síntesis . . . 29

3. Estado del Arte 31
- 3.1. Aprendizaje por Refuerzo con estados y acciones continuos . . . 32

3.2. Aprendizaje por refuerzo con retroalimentación 40
3.3. Aprendizaje por refuerzo con retroalimentación por voz 42
3.4. *Reward Shaping* 44
3.5. Síntesis 46

4. Método Propuesto 49
4.1. Algoritmo de aprendizaje con retroalimentación proporcionada por voz 49
4.2. Retroalimentación obtenida por voz 51
4.3. Reconocimiento e interpretación 52
4.4. Aprendizaje por Refuerzo 53
4.4.1. Representación para espacios continuos 53
4.4.2. Representación y tratamiento de los estados continuos 56
4.4.3. Representación y creación de las acciones continuas . . 64
4.4.4. Diseño de la función de recompensa 66
4.4.5. Diseño de la función de valor 67
4.5. Síntesis 68

5. Experimentos 71
5.1. Características de los robots 73
5.2. Descripción de los ambientes 74
5.3. Descripción de las tareas 75
5.4. Pruebas y resultados 78
5.4.1. Descripción general de las pruebas 78
5.4.2. Tareas de navegación 83
5.4.3. Tarea de manipulación 96
5.5. Discusión de los resultados 96
5.5.1. Tareas de navegación 96
5.5.2. Tarea de manipulación 103
5.6. Conclusiones finales 103

6. Conclusiones, contribuciones y trabajo futuro 105
6.1. Conclusiones 105
6.2. Contribuciones 107
6.3. Trabajo Futuro 107

Bibliografía 109

Referencias en línea 115

A. Listado del vocabulario 119

B. Tabla de notación **123**

C. Información técnica **127**
C.1. Player Stage 127
C.2. Pioneer 2 128
C.3. Harmonic arm 6m 135

Índice de figuras

3.1. Redes Neuronales 33
3.2. Procesos Gaussianos 33
3.3. Codificación gruesa 34
3.4. *Tile coding* o Codificación en teja 34
3.5. Regresiones 35
3.6. Árboles 35
3.7. Campos receptivos 36
3.8. Funciones Gaussianas 36

4.1. Esquema general del algoritmo 50
4.2. Representación para estados y acciones continuas 57

5.1. Robots utilizados en los experimentos 74
5.2. Tareas de navegación aprendidas 76
5.3. Brazo robótico y objeto 77
5.4. Número de episodios y tiempo de aprendizaje con las variantes del algoritmo para navegación 85
5.5. Número de intervenciones por episodio en experimentos con retroalimentación, en navegación 87
5.6. Comparación de trazas de acciones continuas y discretas para dos tareas de navegación 89
5.7. Número de episodios y tiempo con acciones continuas y discretas para dos tareas de navegación 90
5.8. Ejemplos de trazas obtenidas con acciones continuas en tareas de navegación 91
5.9. Número de episodios y tiempo, y número de intervenciones, con y sin retroalimentación perfecta para una tarea de navegación 93
5.10. Número de episodios, tiempo y número de intervenciones con retroalimentación ruidosa (con errores) y sin retroalimentación para la tarea 2 de navegación 95
5.11. Secuencia de la tarea de manipulación 97
5.12. Número de episodios, tiempo y número de intervenciones en la tarea de manipulación 98

Índice de Tablas

4.1. Ejemplos de vocabulario para indicar acciones al robot 54
4.2. Ejemplos de vocabulario para evaluar las acciones del robot con su respectivo valor de recompensa 55

5.1. Experimentos realizados 81
5.2. Número de episodios y tiempos de los diferentes experimentos por tarea de navegación 86
5.3. Número de intervenciones del usuario en los diferentes experimentos por tarea de navegación 88
5.4. Tiempos y número de intervenciones para la tarea 2 de navegación con retroalimentación con y sin retroalimentación perfecta 92
5.5. Tiempos y número de intervenciones para la tarea 2 de navegación, sin retroalimentación y con 50 % de retroalimentación ruidosa (con errores) 94
5.6. Número de episodios, tiempo y número de intervenciones para la tarea de manipulación 99

B.1. Tabla de notación 124
B.2. Tabla de notación 125

C.1. Robots compatibles con Player 129
C.2. *Hardware* adicional compatible con Player 130
C.3. *Hardware* adicional compatible con Player (segunda parte) . . 131
C.4. *Software* adicional compatible con Player 132
C.5. Algoritmos compatibles con Player 133
C.6. Algoritmos compatibles con Player (segunda parte) 134
C.7. Simuladores compatibles con Player 134
C.8. Especificaciones del Pioneer 2 135
C.9. Especificaciones del Harmonic arm 6m 136

Índice de Algoritmos

2.1. Q-learning 17
2.2. Sarsa(λ) 20
4.1. Recepción de trazas iniciales 58
4.2. Elección de una acción o creación de una acción continua . . . 59
4.3. Sarsa(λ) con trazas iniciales y retroalimentación dinámica. . . 60
4.4. Prueba de política 61

Capítulo 1

Introducción

1.1. Antecedentes

En la actualidad, la variedad de las tareas realizadas por los robots es cada vez más amplia y un gran número de estas tareas está relacionado con humanos dentro de ambientes diversos [Bekey, 2005]. Dentro de la robótica, un área que se encuentra en crecimiento y que en un futuro cercano será común en la vida cotidiana es la robótica de servicio. En ella es fundamental la interacción con personas de todo tipo, que en su mayoría no están familiarizadas con los mecanismos de funcionamiento de los robots.

De manera general, los robots requieren de mecanismos de programación cada vez más sofisticados que les permitan lidiar con incertidumbre durante la realización de sus funciones desenvolviéndose en ambientes cambiantes. Los paradigmas de control usados en robótica que basan su diseño en ambientes y acciones ideales, deterministas o estáticos [Muñoz & Morales, 2009], en la práctica presentan problemas y requieren de ajustes constantes fuera de línea o de un control preciso de todos los componentes involucrados para asegurar que todo ocurra como se espera.

Considerando lo anterior, el aprendizaje automático se ha empleado para

desarrollar la programación de los robots. Dos tipos de este aprendizaje que son muy usados en los robots de servicio, por sus características, son el aprendizaje por refuerzo y el aprendizaje por demostración. El aprendizaje por refuerzo [Sutton & Barto, 1998] permite a un robot solucionar un problema de manera exitosa con base en su propia experiencia en el ambiente. La experiencia puede ser acumulada a partir de la interacción con el ambiente real y mediante simulación. Con el aprendizaje por demostración [Bekey, 2005] el robot puede conocer los pasos de ejecución de una tarea enseñados por un usuario o tutor. El uso de técnicas de aprendizaje, como las mencionadas, puede proporcionar a los robots la capacidad de aprender tareas nuevas cuando ya se encuentran en su ambiente de trabajo.

1.2. Problemática

Los robots están siendo usados cada vez más en una amplia variedad de tareas, muchas de ellas involucrando la interacción directa con humanos. Específicamente en el caso de los robots de servicio, éstos no pueden llevar de antemano programadas todas las actividades que van a realizar y las situaciones que van a enfrentar en su área final de trabajo. Para tratar de solucionarlo, se han desarrollado técnicas para que los robots aprendan tareas; sin embargo, poco se ha considerado la colaboración con humanos inexpertos quienes seguramente tendrán restricciones no sólo de conocimientos respecto a la robótica, sino también en sus formas de expresión y comunicación e incluso, en el tiempo del que disponen para el proceso de aprendizaje del robot. El problema de lidiar con los humanos inexpertos se suma a los provocados por los componentes de los robots en general. Los sensores y los accionadores que componen a un robot no responden de igual manera ni en tiempo ni en forma cuando se repite una misma tarea. Estos factores son los generadores de incertidumbre en la realización de las labores de los robots. Además, en los robots

existen claras restricciones de tiempo por la duración de la batería. Aunado a los elementos anteriores, el propio ambiente donde se desenvuelven los robots tiene factores dinámicos e inesperados. Así, la suma del factor humano, las restricciones de los robots y la influencia del ambiente, hacen que los procesos de aprendizaje en entornos sociales requieran de periodos limitados (tiempo restringido), adaptación y una comunicación natural entre los humanos y los robots. Por estas condiciones descritas, en la robótica de servicio no son efectivos los mecanismos de control de enfoques deterministas, ya que no es posible mantener un control preciso de los ambientes dinámicos y la constante relación con personas. En la interacción del robot con humanos, como en muchos de los escenarios del mundo donde los humanos se desenvuelven, son necesarios ajustes en línea (en el momento en que se detectan cambios) de los controles y por lo tanto de las acciones que realizan los robots.

El uso de aprendizaje por demostración y el aprendizaje por refuerzo, actualmente está proporcionando adaptabilidad y potenciando el diseño de mecanismos de control. Las características de estas áreas del aprendizaje computacional proporcionan elementos útiles para los robots en ambientes cambiantes; sin embargo, tienen aún algunos problemas como se detalla a continuación. En el aprendizaje por refuerzo, de manera general, se pueden identificar tres componentes principales que son de especial interés para esta investigación:

- Un conjunto de estados. Un estado está definido por características o circunstancias específicas que indican en dónde y cómo se encuentra el robot en el ambiente.

- Un conjunto de acciones que el agente (robot) puede realizar.

- Una función de recompensa que especifica lo deseable de un estado.

Generalmente estos componentes necesarios para el aprendizaje por refuerzo son establecidos por el programador en un diseño elaborado previo a la puesta

en funcionamiento del robot, y se usan sin modificarse durante el entrenamiento del robot en el ambiente. Sin embargo, en muchas de sus aplicaciones el modelo de sus componentes requiere adaptaciones durante el entrenamiento. Además, este tipo de aprendizaje se caracteriza por requerir de largos tiempos de entrenamiento para acumular la experiencia necesaria para obtener una solución. Aunado a lo anterior, si se usan representaciones discretas del ambiente, como es usual, no es posible trabajar en dominios con espacios muy grandes de estados y acciones. Aunque han sido propuestos varios métodos con representaciones que pueden trabajar con espacios muy grandes e incluso con valores continuos [Sutton & Barto, 1998, Sierra, 2006], estos métodos requieren grandes cantidades de memoria y de tiempo de exploración, e incluso, pueden no llegar a encontrar una solución debido a la naturaleza propia del método. Considerando estos problemas existentes en el aprendizaje por refuerzo, es en esta área donde existe la mayor contribución de esta investigación, pues se propone el empleo del habla no sólo como un elemento para facilitar la interacción natural en los robots con los humanos, sino también, como un medio para acelerar el aprendizaje proporcionando retroalimentación adicional a la tradicional de los algoritmos de aprendizaje.

El aprendizaje por demostración está basado en la enseñanza mediante ejemplos y puede usarse para mostrar paso a paso cómo realizar una tarea; sin embargo, está restringido por el número de ejemplos de una tarea que se pueden mostrar durante un periodo de aprendizaje y por la calidad de los ejemplos que son proporcionados, hechos que lo hacen dependiente del usuario que está enseñando la tarea.

Sin embargo, con la demostración y la colaboración del aprendizaje por refuerzo, como se propone en esta tesis, el robot puede obtener experiencia del ambiente real y de la simulación que le ayude a obtener posibles soluciones ante situaciones no previstas o enseñadas, e incluso mejorar las soluciones que le fueron mostradas. No obstante, en el aprendizaje para robots integrando la

colaboración e influencia de los humanos, se deben considerar los siguientes factores:

- La adaptación de la representación (de estados y acciones) y de las funciones (de valor y recompensa) usadas en los algoritmos de aprendizaje. Respecto a la comunicación (incluyendo aquí el concepto de retroalimentación) entre el humano y el robot se debe considerar qué será dado (instrucciones, observaciones), cómo será dado (considerando a un humano inexperto) y cuándo será dado durante el proceso de aprendizaje. También, se deben identificar los componentes precisos en el aprendizaje que corresponden a los elementos de comunicación, qué se está criticando, qué se está aprobando y qué se espera como reacción del robot ante una observación o instrucción otorgada. Y finalmente, pero lo más importante, debe diseñarse la forma en la cual internamente los algoritmos deben representar y manejar todas los elementos mencionados.

- Las características (medios y formas de comunicación, limitaciones) del aprendiz (en este caso el robot) y el usuario (un humano), que determinan la forma en la cual debe y puede realizarse la interacción entre un humano y un robot [Lockerd et al., 2005]. Además de las condiciones de los medios de interacción. Aun con palabras simples, los sistemas de reconocimiento pueden tener problemas y tener errores de interpretación que afectan el aprendizaje, por lo que es importante saber sobre qué situaciones el aprendizaje es posible aún con errores. Además, el usuario puede equivocarse y/o proporcionar fuera de tiempo la retroalimentación, un factor que también puede influir en el aprendizaje. Es posible también que el usuario no provea retroalimentación constante, la cantidad y frencuencia de su participación puede variar en el tiempo, pudiendo ser más abundante al principio por ejemplo y disminuir conforme el proceso de aprendizaje avanza.

Una vez descrita la problemática sobre la que esta investigación se enfoca, en esta tesis se presentará la combinación de aprendizaje por demostración y por refuerzo mediante un algoritmo de aprendizaje con representaciones novedosas para espacios continuos de estados y acciones, en el que es posible otorgar ejemplos y retroalimentación mediante voz y de este modo enseñar tareas a un robot.

1.3. Motivación

La idea de este trabajo surge de la necesidad de diseñar técnicas eficientes que permitan a los robots aprender tareas nuevas en ambientes dinámicos y con interacción con usuarios que no tienen conocimientos en robótica, como es el caso de la robótica de servicio. Se combinan las técnicas de aprendizaje por refuerzo y por demostración para capacitar a un robot con las ventajas de ambos: el aprovechamiento de la propia experiencia del robot, el aprendizaje en línea, la ayuda de un usuario y los medios de comunicación e interacción entre usuario y robot. Tales características pueden capacitar a un robot para desenvolverse mejor en los ambientes de servicios.

Un elemento en la enseñanza que muchas personas comúnmente emplean de manera natural es el lenguaje verbal, de ahí que se considere el empleo del elemento de voz en el sistema de aprendizaje propuesto. Si bien es cierto que los humanos emplean una combinación de lenguaje verbal y visual, los elementos visuales presentan mayor complejidad (que el lenguaje verbal) de interpretación tanto para humanos como para los robots, y generalmente van acompañados de elementos del lenguaje hablado o escrito para clarificar. Aunque el lenguaje verbal puede llegar a presentar dificultades de interpretación, en contextos comunes (en situaciones donde el hablante y el oyente tienen conocimientos compartidos acerca del tema del que se habla) es fácilmente entendible y manejable. Así, en esta tesis se supone que cuando un humano está

aprendiendo algo (especialmente con un maestro), una forma en que recibe las primeras instrucciones u observaciones en la mayor parte de las ocasiones es a través del lenguaje verbal. De ahí que en esta tesis se haya elegido como medio de retroalimentación.

1.4. Objetivos

El **objetivo general** de esta tesis es desarrollar un algoritmo nuevo para instruir tareas a un robot mediante aprendizaje por refuerzo y retroalimentación verbal en línea.

Los **objetivos específicos** derivados del objetivo general de esta tesis son:

- Combinar técnicas de aprendizaje por demostración y por refuerzo para que un robot aprenda una tarea.
- Diseñar un algoritmo de aprendizaje por refuerzo con retroalimentación verbal en línea.
- Diseñar al algoritmo de aprendizaje por refuerzo para estados y acciones continuas.
- Definir un vocabulario representativo para las tareas a enseñar, que pueda ser interpretado y emplearlo en el proceso de aprendizaje en un robot.
- Probar los algoritmos sobre un robot en simulación y real.

1.5. Alcances y limitaciones

El trabajo de investigación estará caracterizado y delimitado de la siguiente forma:

- El algoritmo desarrollado incluirá aprendizaje por refuerzo y usará trazas de demostración, trabajará con espacios continuos y con retroalimentación por voz.

- Se utilizará un vocabulario restringido al dominio de las tareas y un reconocedor de voz existente (implementado en el software Sphinx 3 [CMU Sphinx] usando parte del reconocedor del proyecto DIMEx100 [Pineda, 2010]).

- Se probará el algoritmo en tareas simples de navegación y manipulación, en simulación y en un robot real.

- Se supondrá que el usuario colabora con el robot para la realización de la tarea y que las condiciones del entorno de aprendizaje están controladas dentro de un rango respecto al nivel de ruido e iluminación.

1.6. Investigación realizada

En esta tesis se diseñó un algoritmo de aprendizaje por refuerzo basado en Sarsa(λ) que incluye retroalimentación proporcionada por voz. El algoritmo consta de tres fases principales, al inicio el usuario proporciona con voz ejemplos de la tarea, en la segunda etapa el robot se entrena usando al algoritmo Sarsa(λ) modificado que incluye la retroalimentación por voz del usuario y en la tercera etapa, la etapa final, se refina, de ser necesario, la política aprendida por el robot mediante retroalimentación por voz. La retroalimentación fue proporcionada con comandos de acciones y calificativos mediante palabras y frases cortas. El algoritmo se propuso para trabajar con espacios continuos por lo que se diseñó una representación similar a los campos receptivos para la identificación de estados basada en dos funciones de similitud usando el coeficiente de Pearson y las funciones gaussianas. Para la creación de accio-

nes continuas se diseño un método simple que combina un conjunto básico de acciones discretas.

Los experimentos fueron hechos para dos tipos de tareas, de navegación y de manipulación. Las tareas de navegación fueron hechas en simulación y la de manipulación en un brazo robótico real. Los experimentos fueron diseñados para probar la influencia en el aprendizaje de las trazas iniciales y la retroalimentación de manera independiente y también de manera conjunta, además para observar la influencia en el aprendizaje de los errores producidos en el reconocimiento de voz y por el usuario.

Conforme a los resultados obtenidos, las trazas iniciales y la retroalimentación usadas de manera independiente reducen el tiempo de aprendizaje. Sin embargo, los mejores resultados fueron obtenidos cuando se utilizó el algoritmo completo, con trazas iniciales y retroalimentación, comparado contra las variaciones del algoritmo y contra un algoritmo tradicional de aprendizaje por refuerzo. Se mostró también que el algoritmo puede ser tolerante a variaciones y errores en el reconocimiento de voz o en la retroalimentación dada por el usuario.

1.7. Guía del documento

En el siguiente capítulo se presenta el marco teórico de aprendizaje por refuerzo, aprendizaje por demostración y los conceptos y definiciones básicos del reconocimiento de voz. En el tercer capítulo se presenta el estado del arte con los trabajos más relacionados con la investigación realizada, los trabajos se presentan organizados por temática de aprendizaje por refuerzo: para espacios continuos, con retroalimentación por dispositivos, con retroalimentación por voz, y los que se enfocan en *reward shaping*. En el capítulo cuatro se detalla el algoritmo propuesto, se da una descripción general del algoritmo y luego se abordan cada uno de sus elementos, la retroalimentación por voz y el proceso

de reconocimiento e interpretación, las representaciones usadas para espacios continuos y finalmente, la función de recompensa y las funciones de valor como elementos clave del algoritmo. En el penúltimo capítulo se presentan los resultados experimentales hechos en simulación y con un robot real. Se proporcionan los detalles de los experimentos, las características de los robots, los ambientes de prueba y las tareas realizadas. Y se muestran los resultados de comparar al algoritmo propuesto contra un algoritmo tradicional de aprendizaje por refuerzo y contra variantes de sí mismo. Al final de este capítulo se presenta una discusión de los resultados. En el capítulo final se presentan las conclusiones, las mayores aportaciones de esta tesis y el trabajo futuro.

Capítulo 2

Marco Teórico

En este capítulo se presentan las definiciones generales y fundamentos teóricos relacionados con los componentes del algoritmo propuesto en la tesis. Las primeras secciones presentan los fundamentos y teoría general del aprendizaje por refuerzo, la siguiente sección presenta los conceptos básicos del aprendizaje por demostración y finalmente se detallan los principales componentes del reconocimiento del habla.

2.1. Aprendizaje por Refuerzo

El aprendizaje automático o computacional es un área de la computación que trata de dotar a las máquinas con programas con la capacidad de mejorar/adaptarse automáticamente para solucionar y/o presentar resultados de manera eficiente [Sierra, 2006]. El aprendizaje utiliza la experiencia con ejemplos vistos o con información de los resultados obtenidos anteriormente ante circunstancias similares para aplicarlos en nuevas situaciones. Un tipo de aprendizaje computacional que ha resultado especialmente útil en áreas como la robótica [Bekey, 2005] es el aprendizaje por refuerzo. En el aprendizaje por refuerzo [Sutton & Barto, 1998, Sierra, 2006] se tiene a un agente o aprendiz que tomará decisiones en diferentes circunstancias, el conjunto de acciones re-

sultantes de esas decisiones definirá su comportamiento. El agente se encuentra inmerso en un ambiente con el cuál interactuará, para que aprenda una tarea se le otorgan refuerzos positivos o negativos de acuerdo a las acciones hechas y la contribución de éstas para completar una tarea meta. En este tipo de aprendizaje el agente usa su propia experiencia al explorar el ambiente para encontrar al conjunto de acciones que lo lleven a su meta. El conjunto de acciones resultantes con las cuales completa su meta define la llamada política de comportamiento del agente. Así, un agente explora el ambiente e identifica una determinada parte de éste como un estado s, realiza una acción a y recibe un refuerzo positivo o negativo r que evalúa esa acción. El objetivo del agente es obtener la política de comportamiento (conjunto de acciones) que maximice la recompensa a lo largo del tiempo, este objetivo es un criterio de optimalidad, en los algoritmos fundamentales del aprendizaje por refuerzo, dicho criterio está basado en el principio del horizonte infinito descontado. El horizonte infinito precisamente maximiza la suma de refuerzos en el tiempo utilizando un parámetro γ de descuento que minimiza la influencia de los refuerzos recibidos a futuro, como se muestra en la ecuación (2.1).

$$\Sigma_{k=0}^{\infty}\gamma^k r^k$$
$$0 \leq \gamma \leq 1 \tag{2.1}$$

donde k es el índice que indica el instante de tiempo en el cual es recibida la recompensa.

La definición formal de los fundamentos de los algoritmos de aprendizaje por refuerzo se encuentra basada en la propiedad de Markov y la forma completa en que se realiza el aprendizaje se basa en los procesos de decisión de Markov o MDPs por sus siglas en inglés, como será visto con más detalle en la siguiente sección.

Existen varios algoritmos y adaptaciones de los mismos en el aprendizaje

por refuerzo. Por su funcionamiento general los algoritmos de aprendizaje por refuerzo se dividen en algoritmos basados en un modelo y algoritmos libres de modelo. Los algoritmos basados en un modelo [Sutton & Barto, 1998, Sierra, 2006] usan un modelo de la dinámica del comportamiento durante el proceso de aprendizaje, generalmente usan técnicas de programación dinámica para encontrar la política óptima después de aprender el modelo. Entre estos algoritmos están *Dyna-Q* [Sutton, 1990], *Prioritized Sweeping* [Moore & Atkeson, 1993], *Queue-Dyna* [Peng & Williams, 1993], entre otros. Los algoritmos libres de modelo [Sutton & Barto, 1998, Sierra, 2006], como su nombre lo indica, no usan ningún modelo de comportamiento para aprender a realizar una tarea, sólo conocen los estados y acciones pero no los resultados de realizar las acciones en los estados. Entre estos algoritmos se encuentran los basados en Monte Carlo y en diferencias temporales [Sutton & Barto, 1998]. Dentro de los algoritmos de diferencias temporales [Sutton & Barto, 1998] los más comunes incluyen a TD(0), Q-learning y Sarsa.

2.1.1. Procesos de Decisión de Markov

En aprendizaje por refuerzo una vez identificado un estado debe realizarse una acción, de entre un conjunto de acciones posibles, con la cual se maximice el refuerzo [Sierra, 2006]. Las decisiones se toman de acuerdo al estado actual en el que se encuentra el agente, lo cual cumple con la propiedad de Markov. Esta propiedad se define matemáticamente en la ecuación (2.2).

$$Pr\left\{s_{t+1}, r_{t+1} | s_t, a_t\right\} = Pr\left\{s_{t+1}, r_{t+1} | s_t, a_t, r_t, s_{t-1}, a_{t-1}, ...r_1, s_0, a_0\right\} \quad (2.2)$$

donde Pr indica la probabilidad de transición de pasar a un estado determinado y recibir una recompensa determinada después de realizar una acción en el estado actual, s es un estado, a es una acción, r una recompensa y t indica el instante de tiempo.

La propiedad establece que el estado actual s_t no depende de los estados y las acciones pasados $(s_{t-1}, a_{t-1}...s_0, a_0)$, sino resume completamente a los estados que le antecedieron, siendo así ese único estado actual s_t y la acción a_t suficientes para tomar decisiones futuras s_{t+1} con su respectiva r_{t+1}.

Cumpliendo dicha propiedad, el proceso de decisión puede ser modelado y resuelto mediante los procesos de decisión de Markov. Así, los problemas de aprendizaje por refuerzo pueden ser modelados por los procesos de decisión de Markov [Sierra, 2006]. Un MDP se caracteriza por estar formado por una tupla (S,A,δ,R), donde:

- S es un conjunto de estados,
- A un conjunto de acciones,
- $\delta : S \times A \rightarrow S$ una función de transición que mapea desde un estado s realizando una acción a hacia un estado s',
- $R : S \times A \times S \rightarrow \Re$ una función de recompensa que califica la acción tomada en un estado s de acuerdo al estado s' resultante de hacer dicha acción.

El objetivo, o criterio de optimalidad [Sutton & Barto, 1998, Sierra, 2006], consiste en maximizar la suma estimada de los refuerzos que se obtendrían siguiendo la política, es decir eligiendo en cada estado una acción de acuerdo a la política. Los criterios de optimalidad pueden variar de problema a problema, por ejemplo el criterio de horizonte finito y el de horizonte infinito descontado.

En el horizonte finito la sumatoria de recompensas en el tiempo está limitada a un determinado número de acciones K preestablecido. El criterio de horizonte infinito descontado igualmente busca maximizar la suma de recompensas r_k obtenidas en el tiempo pero cuando k tiende a infinito, utilizando un parámetro de descuento γ. Este parámetro reduce la influencia de las recompensas recibidas en el futuro, esto es porque γ^k tiende a 0 cuando k tiende

a infinito, porque $0 \leq \gamma \leq 1$. El criterio de horizonte infinito queda expresado en la ecuación (2.3).

$$\Sigma_{k=0}^{\infty} = \gamma^k r_k \tag{2.3}$$

La mayoría de los algoritmos de aprendizaje por refuerzo buscan aproximar una función de valor. La función de valor [Sutton & Barto, 1998, Sierra, 2006] estima qué tan bueno es para un agente encontrarse en un estado determinado o qué tan bueno es hacer una acción en ese estado, siguiendo en ambos casos una política de comportamiento π. La función de valor basada sólo en el estado es denotada con $V^{\pi}(s)$ y está representada por la ecuación (2.4).

$$V^{\pi}(s) = E_{\pi}\left\{\Sigma_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t = s\right\} \tag{2.4}$$

donde se expresa el valor esperado de la recompensa acumulada a partir de un estado s y siguiendo una política π, suponiendo el criterio de horizonte infinito descontado.

La función de valor basada en el estado s y la acción a realizar a está denotada por $Q^{\pi}(s,a)$ y se define matemáticamente en la ecuación (2.5).

$$Q^{\pi}(s,a) = E_{\pi}\left\{\Sigma_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t = s, a_t = a\right\} \tag{2.5}$$

Generalmente se busca obtener las mejores políticas de entre todas las posibles que se denominan como óptimas, denotadas como π^*. Estas políticas se expresan en las ecuaciones, para cada caso de función de valor:

$$V^*(s) = max_{\pi}V^{\pi}(s), \forall s \in S \tag{2.6}$$

$$Q^*(s,a) = max_{\pi}Q^{\pi}(s,a), \forall s \in S, \forall a \in A \tag{2.7}$$

Para la solución de MDPs se han propuesto diferentes métodos [Sutton & Barto, 1998], agrupados en los que usan Programación Dinámica, los basados

en Monte Carlo y los basados en Diferencias Temporales, algunos de estos últimos serán abordados en esta tesis.

2.1.2. Algoritmos de aprendizaje por refuerzo basados en diferencias temporales (TD)

Estos algoritmos combinan parte de los métodos de Monte Carlo con programación dinámica [Sutton & Barto, 1998]. Utilizan la experiencia sin un modelo de la dinámica del ambiente como en Monte Carlo y hacen estimaciones en función de estimaciones pasadas como en Programación Dinámica. Entre los algoritmos más populares de este tipo [Sutton & Barto, 1998] se encuentran los métodos de Actor-Crítico, Q-Learning y Sarsa. Los métodos de Actor-Crítico mantienen dos estructuras, la primera de ellas, el actor, es la encargada de decidir qué acciones hacer de acuerdo a una política almacenada; la segunda, el crítico, representa a la función de valor que califica el desempeño del actor. El aprendizaje con este método es de acuerdo a la política almacenada que se va ajustando. A diferencia de este método, Q-learning y Sarsa(λ) aprenden funciones de acción-valor y determinan la política con base en estas estimaciones. Estos algoritmos se abordarán a continuación con detalle puesto que el algoritmo propuesto en esta tesis sigue los principios de estos últimos algoritmos.

Q-learning

Es el algoritmo [Sutton & Barto, 1998] más popular de aprendizaje por refuerzo y se encuentra dentro del grupo de algoritmos basados en diferencias temporales. Se encuentra dentro de los algoritmos *off-policy* o fuera de política porque selecciona una acción para actualizar la función de valor que no depende de la política, y estima la función de valor $Q(s, a)$.

El algoritmo 2.1 muestra a Q-learning, donde α es el parámetro de aprendi-

Algoritmo 2.1 Q-learning

Variables de entrada: α coeficiente de aprendizaje (número real), γ factor de descuento (número real)
Variables de salida: Q estructura (ej. tabla de números reales donde s, a son los índices que indican la fila y la columna) que representa la función de valor
Variables locales: s, s' estados (números enteros identificadores de estados, $s, s' \in S$, S conjunto de estados), a, a' acciones (números enteros identificadores de acciones, $a, a' \in A$, A conjunto de acciones), r recompensa (número real)
Inicio
Inicializar $Q(s, a)$ arbitrariamente (valores numéricos reales)
repeat
  Inicializar s
  repeat
    Seleccionar una acción a a partir de s usando una política derivada de Q
    Ejecutar la acción a y observar el refuerzo recibido r y el siguiente estado s'
    Actualizar $Q(s, a)$ con la ecuación:
    $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma max Q(s', a') - Q(s, a)]$
    $s \leftarrow s'$
  until s sea un estado final
until que se alcance el episodio de entrenamiento final
Fin

zaje que determina qué tan importantes son las nuevas actualizaciones respecto a las anteriores. Con un valor de 0 el agente no aprende nada, con un valor de 1 el agente sólo aprende los datos más recientes, y el parámetro de descuento γ determina la importancia de las futuras recompensas, con un 0 sólo se considera la recompensa inmediata, con un valor cercano 1 se consideran las recompensas a largo plazo.

Sarsa(λ)

Es uno de los algoritmos de aprendizaje por refuerzo similar en funcionamiento a Q-learning, basado también en diferencias temporales, pero éste es *on-policy*, es decir sigue una política [Sutton & Barto, 1998]. A diferencia de otros algoritmos que sólo propagan los refuerzos recibidos a un solo paso (estado-acción), el algoritmo de Sarsa(λ) propaga el refuerzo a n pasos (que conforman una traza) siguiendo un mecanismo denominado trazas de elegibilidad.

Las trazas son una serie de acciones que son comandadas por el usuario y que ejemplifican soluciones de la tarea. Las trazas de elegibilidad son un mecanismo para identificar una serie de estados y acciones que contribuyeron a alcanzar un estado determinado y su respectiva recompensa positiva o negativa. Cuando el ajuste de estados y acciones en los algoritmos de diferencia temporal se hace a más de un paso, se está utilizando el mecanismo de trazas de elegibilidad. La traza de elegibilidad funciona de manera similar a una memoria que mantiene el registro de los estados visitados y de las acciones realizadas hasta llegar a cierto estado, la longitud de la traza depende del parámetro λ. Así, la diferencia de actualización con Q-learning y los algoritmos TD(λ) radica en mantener la información de la traza $e(s, a)$ además de la función de valor Q, lo cual hace que se afecte el valor Q de todos los pares s, a contenidos en la traza en mayor o menor medida según sea la degradación producida por el largo de la traza. Existen dos tipos de algoritmos de trazas de

elegibilidad con y sin reemplazo, su diferencia radica en la forma en la cual es actualizada la información $e(s, a)$ de la traza. Cuando se realiza sin reemplazo se utiliza el valor acumulado de e. Cuando se realiza con reemplazo en lugar de usar el valor acumulado de $e(s, a)$ y agregarle 1 para el e del estado-acción dentro de la traza, se reemplaza sólo por 1 ($e(s, a) = 1$). Los algoritmos con reemplazo pueden agilizar el aprendizaje cuando se usan sobre dominios de alta dimensionalidad y con aproximadores que generalicen los estados lo suficiente, de otro modo con ellos se obtienen los mismos resultados que con un algoritmo sin reemplazo.

El algoritmo Sarsa(λ) se muestra en algoritmo 2.2 donde los parámetros de α y γ tienen el mismo efecto que en Q-learning.

En esta tesis se utilizan como base los principios de este algoritmo principalmente por la propagación de las recompensas en el tiempo y a lo largo de la traza que proporcionan las trazas de elegibilidad, y por la agilización que este tipo de mecanismos proporcionan a los algoritmos de aprendizaje.

2.1.3. *Reward Shaping*

Esta técnica consiste en agregar recompensas durante el proceso de aprendizaje para guiar al agente de acuerdo al conocimiento *a priori* del dominio, con ella se agiliza el aprendizaje al incluir ese conocimiento en los refuerzos adicionales [Ng et al., 1999, Laud, 2004]. Se basa en una función de recompensa F que es similar a la usada típicamente R en el modelo del aprendizaje por refuerzo. Al igual que la función de recompensa tradicional otorga un refuerzo de acuerdo a la transición producida por una acción tomada en un determinado estado.

Siguiendo con la notación de un MDP, para *reward shaping* se tiene un modelo $(S, A, \delta, R + F)$, que puede ser resuelto por cualquier algoritmo de aprendizaje por refuerzo, al igual que un MDP con una función de recompensa

Algoritmo 2.2 Sarsa(λ)

Variables de entrada: α coeficiente de aprendizaje (número real), γ factor de descuento (número real), λ factor de descuento de la traza (número real)
Variables de salida: Q estructura (ej. tabla de números reales donde s, a son los índices que indican la fila y la columna) que representa la función de valor
Variables locales: s, s' estados (números enteros identificadores de estados, $s, s' \in S$, S conjunto de estados), a, a' acción (números enteros identificadores de acciones, $a, a' \in A$, A conjunto de acciones), r recompensa (número entero), δ variable auxiliar para la actualización de la función de valor (número real), e estructura (ej. tabla de números reales donde s, a son los índices que indican la fila y la columna) que representa la información de la traza
Inicio
Inicializar $Q(s, a)$ arbitrariamente (valores numéricos reales) y $e(s, a) = 0, \forall s, a$
repeat
  Inicializar s, a
  repeat
    Seleccionar una acción a a partir de s, ejecutar la acción a y observar el refuerzo recibido r y el siguiente estado s'
    Seleccionar a' de s' usando una política derivada de Q (e.g. $e - greedy$)
    Actualizar:
    $\delta \leftarrow r + \gamma Q(s', a') - Q(s, a)$
    $e(s, a) \leftarrow e(s, a) + 1$
    for each s, a **do**
      $Q(s, a) \leftarrow Q(s, a) + \alpha \delta e(s, a)$
      $e(s, a) \leftarrow \gamma \lambda e(s, a)$
    end for
    $s \leftarrow s'$
    $a \leftarrow a'$
  until s sea un estado final
until que se alcance el episodio de entrenamiento final
Fin

R solamente. Esto es debido a que en el modelo con *reward shaping* se añade al refuerzo común el refuerzo de la función de *reward shaping*. Un proceso de Markov usando *reward shaping*, con la función $R = R + F$, es equivalente a uno estándar con la función R, lo que permite la solución de ambos de manera similar y la garantía en la obtención de políticas óptimas.

Cuando no se tiene suficiente conocimiento *a priori* del problema y no se puede diseñar directamente una función de refuerzo basada en *reward shaping*, se puede utilizar un método llamado *dynamic shaping* [Laud, 2004] para aprender la función de *reward shaping*. En *dynamic shaping* se busca maximizar la recompensa considerando los refuerzos tradicionales y los obtenidos por *reward shaping*, y el método es considerado dinámico porque durante el proceso de aprendizaje la función de *reward shaping* se va ajustando con base en la experiencia. En *dynamic shaping* se realiza un ajuste de parámetros de la función de *reward shaping* y se ajusta la forma en que se integra la recompensa del *reward shaping* al refuerzo tradicional.

Algunas de las funciones que pueden utilizarse para modelar una función de *reward shaping* cuando se tiene suficiente conocimiento *a priori* son las basadas en potenciales [Asmuth et al., 2008]. Las funciones de este tipo establecen que el refuerzo obtenido en una transición es producido por la diferencia de potenciales $F(s, a, s') = \gamma\phi(s') - \phi(s)$, los potenciales expresan un valor de cercanía de un estado con la meta. Otro de los tipos de funciones comúnmente usadas para *reward shaping* son las basadas en estimadores de progreso, con los cuales se califica el aprendizaje en cada paso, determinando qué tanto se ha avanzado para alcanzar la meta.

Siguiendo con la idea de *reward shaping* como un mecanismo que contribuye al aprendizaje, para el algoritmo que se propone en esta tesis se ha retomado la idea de adicionar recompensas al refuerzo tradicional, pero a diferencia de los métodos de *reward shaping* tradicionales se hace de manera dinámica con ayuda de la voz, construyendo un *reward shaping* dinámico como se verá en el

capítulo 4 en la sección 4.4.3.

2.1.4. Generalización y aproximación de funciones

Las funciones de valor en los algoritmos de aprendizaje por refuerzo pueden representarse de forma tabular cuando el número de estados y acciones son discretos y finitos [Sierra, 2006]. Pero en muchos de los dominios donde es aplicado el aprendizaje por refuerzo esto no sucede, porque se tienen dominios continuos en estados, acciones o en ambos. Incluso en ocasiones aunque se tiene un dominio discreto y finito, la cantidad de estados y/o acciones impide su tratamiento en tablas por cuestiones de memoria y tiempo de procesamiento. Para manejar ese problema se ha optado por utilizar técnicas de generalización y aproximación de funciones [Sierra, 2006], y con ellas diseñar representaciones manejables de los estados y las acciones. Con la aproximación de funciones se busca diseñar una función con sus parámetros para representar la función de valor del algoritmo de aprendizaje. Con la generalización se identifican características relevantes de los estados y se agrupan de acuerdo a ellas, los estados se transforman de un espacio continuo e infinito a representaciones reducidas y finitas.

Entre los métodos usados para este propósito [Sierra, 2006] se encuentran: (i) la codificación gruesa (*coarse coding*) [Sierra, 2006, Sutton & Barto, 1998] basada en la identificación de características de un estado dentro de un espacio dividido en regiones, (ii) la codificación en teja (*tile coding*) [Sierra, 2006, Sutton & Barto, 1998, Whiteson et al., 2007] donde el espacio de características se divide en particiones o tejas, (iii) el vecino más cercano [Sutton & Barto, 1998] dónde se utilizan prototipos y medidas de distancia para identificar las regiones que dividen el espacio de características, (iv) los árboles de decisión [Sierra, 2006, Zhang & Weng, 2002] donde cada hoja representa una parte del espacio, (v) las redes neuronales [Sierra, 2006, GroBmann & Poli, 1997,

Coulom, 2002], formadas por unidades de procesamiento conectadas entre sí con conexiones que tienen pesos asociados, ajustan sus parámetros para aproximar funciones con aprendizaje supervisado (vi) las regresiones [Sierra, 2006, Hernández, 2010], que buscan a partir de los datos de entrada una descripción funcional con la cual es posible predecir otros datos (vii) los procesos gaussianos [Rottmann et al., 2007, Rasmussen, 2004], similares a las regresiones, pero que tienen un número finito de variables aleatorias que tienen distribuciones gaussianas conjuntas, son una generalización de esas distribuciones (viii) la aproximación de Euler [Doya, 2000], que aproxima (la función de valor del aprendizaje por refuerzo) basándose en el valor de una función en un momento en el tiempo, para obtener otro posterior utilizando derivación y (ix) las funciones de base radial [Tamosiunaite et al., 2009], entre otros.

Los métodos basados en funciones de base radial [Sierra, 2006] son una generalización de los métodos basados en codificación gruesa que otorga a las características valores entre 0 y 1, ese valor representa el grado de pertenencia de la característica. Una de las funciones de base radial más populares es la función gaussiana, su ecuación se muestra en la ecuación (2.8).

$$\phi = e^{-(||s-\mu||)^2/2\sigma^2} \tag{2.8}$$

Donde s es la descripción del estado, μ, el centro o la media de la gaussiana, σ la desviación estándar. Los métodos basados en funciones radiales [Sierra, 2006] son ampliamente usados por la capacidad de las funciones gaussianas para estimar cualquier tipo de función. Las funciones gaussianas son muy utilizadas en los métodos basados en *kernels*. En ellos se utiliza una función que representa a un *kernel*, que recibe dos elementos del dominio y obtiene un valor real por su producto escalar, ese valor resultante puede expresar la similitud existente entre ambos elementos de entrada. Los campos receptivos [Sutton & Barto, 1998, Tamosiunaite et al., 2009] utilizados como representaciones para

espacios continuos son conjuntos de funciones *kernel* distribuidas en el espacio del dominio, usualmente con funciones gaussianas como *kernels*.

Respecto a los espacios continuos de las acciones, es posible también aplicar aproximaciones (similares a las usadas para los estados) o reducciones del espacio usando sólo conjuntos limitados o representativos de acciones en lugar de todas las posibles.

La representación diseñada para el espacio continuo de los estados del algoritmo propuesto en esta tesis se hace siguiendo los principios presentados en esta sección, de manera similar a los campos receptivos con funciones *kernel*, incluyendo dos tipos de funciones para medir la similitud (coeficiente de Pearson y funciones gaussianas). Para el diseño de acciones continuas se usó una mezcla de acciones discretas.

2.2. Aprendizaje por demostración

El aprendizaje por demostración [Bekey, 2005] es una de las áreas que ha tenido un mayor desarrollo en la robótica y que se relaciona con otras áreas como son aprendizaje automático, interacción humano-robot, visión y control. El aprendizaje por demostración proporciona la capacidad de enseñar nuevas tareas a un robot sin tener que hacer programación explícita y compleja, además de reducir el espacio de búsqueda para el aprendizaje. La idea general es que un usuario muestra al robot cómo hacer una tarea, proporcionando ejemplos buenos y malos dirige los esfuerzos del robot hacia espacios específicos de búsqueda donde se encuentran las soluciones óptimas o subóptimas y evita que el robot explore innecesariamente todo el espacio donde pueden hallarse o no las posibles soluciones. El aprendizaje por demostración representa también un medio natural para entrenar una máquina por un humano. Aunque poco se ha hecho, la combinación de aprendizaje por demostración y aprendizaje por refuerzo ha sido provechosa sobretodo en situaciones donde el robot se enfrenta

a situaciones nuevas a las que le fueron enseñadas en su entrenamiento.

En el aprendizaje por demostración pueden identificarse dos enfoques para la representación de una habilidad o tarea que se aprenderá. El primer enfoque considera la correspondencia entre los sensores y la información que se envía a los motores, lo que es llamado codificación de trayectorias. El segundo enfoque es de alto nivel y considera la relación de unidades de percepción y acción, llamada codificación simbólica.

El aprendizaje imitando los movimientos de otro ser es fundamental en los humanos y los animales [Bekey, 2005]. Se ha descubierto que en el cerebro hay neuronas "reflejo" que se activan cuando se hacen acciones aprendidas por imitación y cuando se observa a otro individuo hacer esas mismas acciones que se están aprendiendo. En los humanos incluso se ha llegado a sugerir una estrecha relación entre la imitación de movimientos y el desarrollo del lenguaje hablado, debido a la cercanía entre sí de las neuronas encargadas de esas funciones en el cerebro. La observación de animales vertebrados usando programas motores almacenados (primitivas motoras) para representar movimientos más complejos que son aprendidos por imitación, ha inspirado algunos trabajos y llevado a sugerir diversas arquitecturas para implementar un sistema de aprendizaje por demostración (similar al de los animales) en un robot. Una de las arquitecturas pioneras es la que supone que el aprendizaje de este tipo puede llevarse a cabo a través de un proceso jerárquico en el cual un robot primero aprende los aspectos básicos o primitivos de una acción y a partir de éstos crea combinaciones que lo llevan a poder hacer acciones más complejas (que las previamente aprendidas) e incluso secuencias de ellas. Por otra parte, existe otra corriente que sugiere que el aprendizaje por demostración debe hacerse usando modelos matemáticos, donde el objetivo es que el robot aprenda una política de control, es decir, una representación matemática de la tarea que se quiere que aprenda. Así, a partir de unas pocos ejemplos de la tarea, el robot deberá ser capaz de replicar lo que un usuario le enseña.

Finalmente, de manera general, para que un usuario muestre a un robot cómo hacer una tarea se han usado diversos dispositivos de *hardware* que faciliten la demostración (transmisión) por parte del usuario e identificación (correspondencia) por parte del robot de los elementos que conforman las acciones de la tarea que está siendo enseñada. Entre los elementos más comunes que se han usado se encuentran periféricos como el ratón, el *joystick*, el teclado, cámaras, guantes con sensores, sensores (por ejemplo de movimiento o posición) y elementos del lenguaje como los gestos y el habla.

En esta tesis se usa la combinación de aprendizaje por refuerzo y por demostración, específicamente se usa aprendizaje por demostración sin usar *hardware* especializado. Entendiéndose por *hardware* especializado al conjunto de periféricos de computadora (ya mencionados) que comúnmente se usan en aprendizaje por demostración y que no son del dominio de personas inexpertas en el área. Así, en esta tesis se propone un enfoque utilizando voz para proporcionar ejemplos y para colaborar en el aprendizaje por refuerzo.

2.3. Reconocimiento del habla

Un sistema de reconocimiento del habla convierte un registro de audio digital PCM (*Pulse Code Modulation*) en el discurso textual del lenguaje reconocido [Mansoor, 2001] . Algunos conceptos básicos para entender el funcionamiento del reconocimiento son:

- Vocabulario: conjunto de palabras que conforman el lenguaje o el dominio del lenguaje que se está analizando.
- Gramática: conjunto de reglas de formación para formar cadenas de caracteres a partir de un alfabeto dado (conjunto de símbolos).
- Fonema: unidad acústica mínima que forma parte de la pronunciación completa de las palabras dentro del lenguaje.

- Característica relevante (*feature*) de una señal de audio: representación numérica que identifica partes acústicas dentro de un registro de audio que se encuentra en el rango (de sonido) que puede ser emitido por un humano. Son obtenidas durante la transformación del audio al dominio de la frecuencia mediante un análisis de la señal empleando la transformada de Fourier.

Una vez establecidos esos conceptos se puede describir de manera general al reconocedor del habla. Básicamente:

1. El reconocedor convierte el registro de audio PCM en una representación acústica, transformando la señal al dominio de la frecuencia mediante una transformada de Fourier y extrayendo las características relevantes (*features*) del audio para ser analizadas

2. Después, el reconocedor aplica una gramática de fonemas conocidos sobre la representación acústica para identificarlos

3. Y finalmente el reconocedor convierte los fonemas identificados en palabras del lenguaje reconocido.

Los elementos principales del reconocedor usados en este proceso son el modelo acústico, el modelo del lenguaje, el diccionario y el léxico de pronunciación que se describen brevemente a continuación.

2.3.1. Modelo del lenguaje

El modelo del lenguaje [CMU Sphinx] asigna una probabilidad a una palabra, esa probabilidad expresa qué tan posible es que la palabra aparezca en dónde se le identifica dentro de las oraciones. Se basa en un conjunto finito de palabras (vocabulario) dado previamente al reconocedor, y que fue analizado durante una fase de entrenamiento. En el vocabulario se encuentran palabras

y frases del dominio que está siendo reconocido. Durante el entrenamiento el reconocedor calcula las probabilidades de cada palabra en función de su posición de aparición dentro de las oraciones. La probabilidad calculada puede corresponder a la palabra sola, a la probabilidad de aparición de una palabra dada otra (antes o después) o a la probabilidad de aparición dadas otras dos palabras (antes o después). En muchos reconocedores estas probabilidades son calculadas basándose en modelos ocultos de Markov o HMM por sus siglas en inglés.

2.3.2. Modelo acústico

El modelo acústico [CMU Sphinx] relaciona las características acústicas (*features*) a secuencias de fonemas, comúnmente usando modelos ocultos de Markov, y es independiente del vocabulario. Para este modelo se hace un proceso similar al del modelo del lenguaje, pero a nivel de fonemas en lugar de palabras.

2.3.3. Léxico y diccionario de pronunciación

El léxico de pronunciación [CMU Sphinx] asigna una o más cadenas de fonemas a una palabra dentro del vocabulario del lenguaje reconocido. Es establecido con anterioridad al proceso de reconocimiento y resulta del análisis del lenguaje.

Para la inclusión del habla en el algoritmo propuesto de esta investigación, el reconocedor usado está basado en los modelos ocultos de Markov tanto para los modelos acústicos como para el modelo del lenguaje. El léxico y el diccionario de pronunciaciones utilizados pertenecen a un vocabulario en español construido previamente.

2.4. Síntesis

En este capítulo se presentaron los principales conceptos de aprendizaje por refuerzo, por demostración y de reconocimiento del habla. De aprendizaje por refuerzo se presentó cómo pueden ser modelados los problemas en este tipo de aprendizaje mediante los procesos de decisión de Markov, los fundamentos de los algoritmos con los que se pueden resolver estos procesos de decisión y se hizo especial énfasis en dos de los algoritmos más populares, Q-learning y Sarsa. También se presentaron los principales conceptos de aprendizaje por demostración, respecto al tratamiento de los datos, las arquitecturas y el *hardware* usados en los sistemas de aprendizaje por demostración. Finalmente se describieron los componentes principales y el funcionamiento de un reconocedor de voz.

Estos fundamentos son la base del algoritmo propuesto en esta tesis. Principalmente para presentar al algoritmo propuesto se retomará el funcionamiento del algoritmo de aprendizaje por refuerzo Sarsa(λ). De aprendizaje por demostración se seguirá un enfoque de representación basado en trayectorias y encaminado a funcionar como una arquitectura jerárquica, donde las tareas aprendidas parten de acciones básicas para aprender acciones más complejas. Además, se incluirá un reconocedor de voz para proporcionar ejemplos y retroalimentación al robot durante el aprendizaje sin usar *hardware* especializado.

En el siguiente capítulo se presenta el trabajo relacionado con los temas de esta investigación, incluyendo aprendizaje por refuerzo para espacios continuos, con inclusión de retroalimentación y con algunas técnicas de demostración, y usando *reward shaping*.

Capítulo 3

Estado del Arte

En la tesis se propone el desarrollo de algoritmos de aprendizaje por refuerzo y demostración utilizando retroalimentación en línea proveniente del usuario mediante voz. Para abordar el trabajo más relevante relacionado con la investigación, el estado del arte se ha dividido en cuatro secciones. En la primera sección se presentan trabajos referentes al diseño e implementación de representaciones de espacios continuos para estados y acciones en aprendizaje por refuerzo. La segunda sección contiene los trabajos que incluyen aprendizaje por refuerzo con algún tipo de retroalimentación proveniente del usuario dada a través de fuentes diferentes a la voz, en su mayoría se trata de dispositivos periféricos de una computadora o de un robot. En la tercera sección se presentan los trabajos encontrados que usan retroalimentación con voz con aprendizaje por refuerzo en un robot. Finalmente, en la última sección se presentan trabajos que usan *reward shaping*.

3.1. Aprendizaje por Refuerzo con estados y acciones continuos

Para el tratamiento de estados y acciones continuas han sido utilizados diferentes métodos que usan aproximadores de funciones. De acuerdo a los trabajos existentes en la literatura, los trabajos de aprendizaje por refuerzo con estados y acciones continuas se pueden agrupar por el tipo de aproximación que usan. Los métodos de aproximación encontrados con más frecuencia son las redes neuronales artificiales, figura 3.1, como las presentadas en [GroBmann & Poli, 1997, Coulom, 2002, Dongli et al., 2005, Chohra et al., 2008, Tsubone & K.~Sugiyama, 2007, Van~Hasselt & Wiering, 2009]; los procesos gaussianos, figura 3.2, como los usados en [Rottmann et al., 2007, Rasmussen, 2004]; *tile coding*, figura 3.3 y figura 3.4, como el detallado en [Whiteson et al., 2007]; regresiones, figura 3.5 como las utilizadas en [Smart & Pack~Kaelbling, 2002, Melo & Lopes, 2008, Guenter et al., 2007, Zhang & Weng, 2002]; árboles, figura 3.6, como el presentado en [Zhang & Weng, 2002]; *kernels* y campos receptivos, figura 3.7, como los usados en [Tamosiunaite et al., 2009] y aproximaciones basadas en funciones gaussianas, figura 3.8, como se muestra en [Kimura et al., 2001]; aproximaciones basadas en la ecuación del horizonte infinito de Hamilton-Jacobi-Bellman como las usadas en [Doya, 2000, Morimoto & Doya, 2000] y la aproximación de Euler usada en [Doya, 2000], entre otros menos frecuentes como los presentados en [Millán et al., 2002, Kollar & Roy, 2006, Van~Hasselt & Wiering, 2007, Lazaric et al., 2008, Hernández, 2010]. A continuación se describen algunos de los trabajos más representativos de estos métodos y más relacionados con esta investigación.

En [Doya, 2000] se presenta un conjunto de algoritmos para tratar con estados y acciones continuas, basados en TD(0) y TD(λ), con el esquema de Actor-Crítico y el gradiente de la función de valor del algoritmo de refuerzo calculado con redes neuronales con retropropagación (considerando en un es-

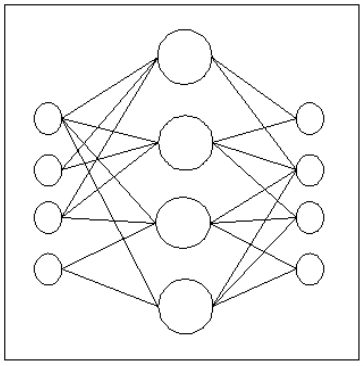


Figura 3.1 Redes Neuronales. Formadas por un conjunto de neuronas de procesamiento interconectadas entre sí, organizadas en capas. Las neuronas de las capas internas usan funciones de activación que reaccionan de acuerdo a los datos de entrada, la última capa, la de salida, combina las salidas de las capas internas. Tratan de aproximar sus salidas a las salidas deseadas proporcionadas durante un entrenamiento, para después ser utilizadas con datos nuevos [Sierra, 2006].

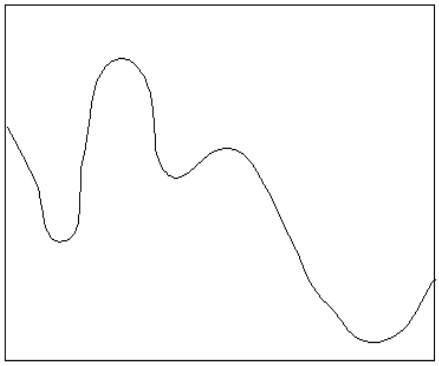


Figura 3.2 Procesos Gaussianos. Similares a las regresiones, pero tienen un número finito de variables aleatorias que tienen distribuciones gaussianas conjuntas [Rasmussen, 2004]

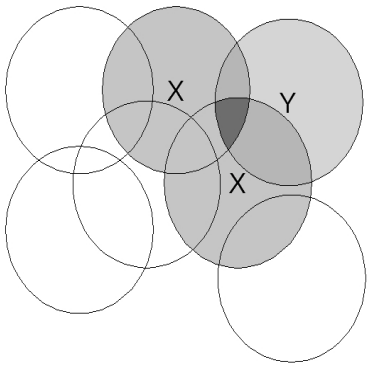


Figura 3.3 Codificación gruesa. La generalización de un estado a otro estado depende de número de características que en los campos receptivos se sobrelapen. En la figura se muestra que estos estados X y Y tienen una característica en común [Sutton & Barto, 1998].

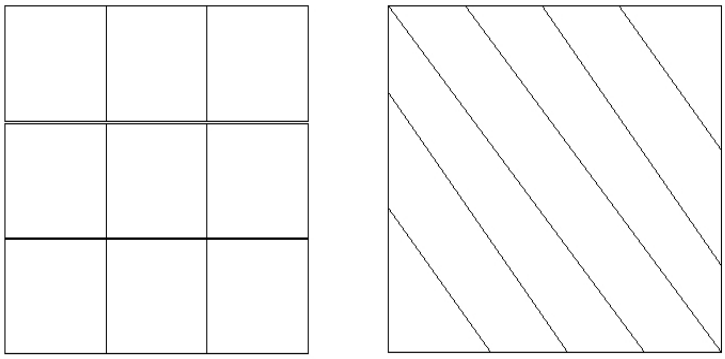


Figura 3.4 Codificación en teja. Similar a la codificación gruesa pero agrupando los campos receptivos de las características en regiones del espacio [Sutton & Barto, 1998].

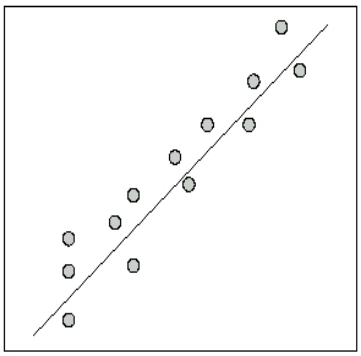


Figura 3.5 Regresiones. Son generalizaciones que tratan de diseñar una función que ajuste los datos, en la figura se muestra el caso más simple, una regresión lineal. Otro tipo frecuentemente usado son las regresiones localmente pesadas que hacen ajustes sobre datos dentro de una vecindad, con pesos basados en la distancia entre los datos del vecindario y el dato que se está localizando para el ajuste. Para modelarlas se pueden utilizar funciones lineales e incluso redes neuronales [Mitchell, 1997].

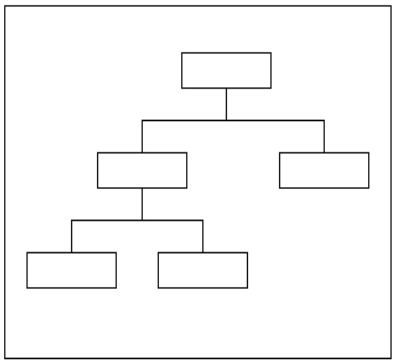


Figura 3.6 Árboles. Cada hoja representa una parte del espacio utilizando alguna función, por ejemplo usando las funciones gaussianas que representan a un grupo de datos [Sierra, 2006, Zhang & Weng, 2002].

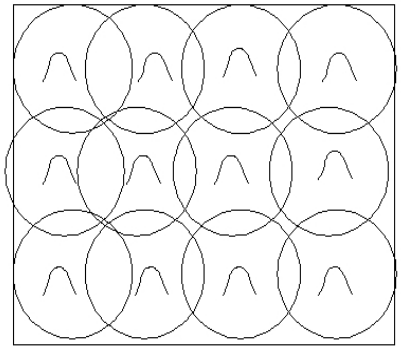


Figura 3.7 Campos receptivos. Formados por conjuntos de funciones *kernel* distribuidas en el espacio del dominio, en la figura se muestran los *kernels* usando funciones gaussianas [Sutton & Barto, 1998, Tamosiunaite et al., 2009].

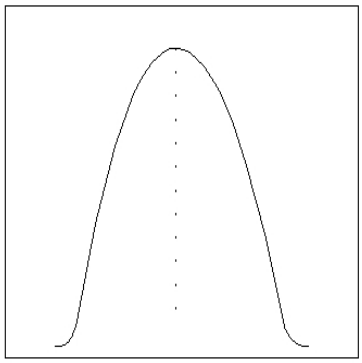


Figura 3.8 Funciones Gaussianas. Los métodos usando funciones radiales son una generalización de los métodos basados en codificación gruesa. Las funciones gaussianas son muy utilizadas en los métodos basados en *kernels*, [Sierra, 2006].

pacio continuo el costo de hacer una acción contra la recompensa recibida y su repercusión en el valor resultante de la función de valor). Los dos últimos tipos son usados para mejorar políticas aprendidas y fueron usados en el problema de balanceo de un péndulo invertido en simulación. En los experimentos se resalta la disminución de intentos tomados en el proceso de aprendizaje al usar una representación continua en comparación con una discreta. Se muestra que el método del gradiente de la función de valor con un modelo dinámico conocido o aprendido presenta mejores resultados que el esquema de Actor-Crítico, y que el uso de trazas de elegibilidad resulta más eficiente que los métodos basados en la aproximación de Euler. Para algoritmos que usan el gradiente es necesario el empleo de información *a priori* o aprender el modelo de la dinámica del sistema físico.

Para el procesamiento de los datos que definen el estado y las acciones a tomar por un robot (específicamente el tiempo y la velocidad) en [Guenter et al., 2007] se utiliza una regresión con mezcla de gaussianas y un modelo de mezcla de gaussianas, GMR y GMM por sus siglas en inglés respectivamente. Aquí se usa un módulo de aprendizaje por refuerzo en conjunción con las mezclas gaussianas y un modelo de la dinámica del movimiento para controlar los movimientos de un robot humanoide, HOAP3. El módulo de aprendizaje por refuerzo es utilizado para aprender nuevos conjuntos de medias (los parámetros de las funciones gaussianas) del GMM mediante exploración del espacio, usando una variante del esquema Actor-Crítico. Para la aproximación se usa al algoritmo de diferencia temporal con mínimos cuadrados LSTD-Q(λ), para el actor se usa un método de gradiente descendiente modificado (con una matriz de información de Fisher). El entrenamiento del robot se llevó a cabo en simulación y fue probado en un robot real. La matriz de covarianzas necesarias para el módulo de aprendizaje por refuerzo es obtenida a partir de una sesión de demostración realizada por un usuario.

En [Millán et al., 2002] se presenta un trabajo de aprendizaje por refuerzo

aplicado a un robot autónomo que construye un mapa de manera incremental mediante el método ITPM (*Incremental Topology Preserving Map*) para modelar el espacio de los estados. El ITPM está formado por unidades que representan una parte del espacio (de los estados), cada unidad está relacionada a un conjunto de valores Q correspodiente a un grupo de acciones discretas que se pueden hacer en cada estado. Así, para las acciones continuas se propone emplear el promedio de los valores Q de las acciones discretas de las unidades del mapa descrito. Se utilizó al algoritmo TD(λ) para el aprendizaje por refuerzo.

En [Hernández, 2010] se presenta un método para tareas de navegación y seguimiento consistente en dos fases, en la primera parte para manejar el espacio continuo de los estados, los datos son convertidos en representaciones relacionales de acuerdo al dominio en el trabajo, puertas, pasillos, habitaciones y paredes. Con esta primera fase se reduce el espacio de los estados. Con esta representación se aplica clonación de comportamiento a partir de un grupo de trazas proporcionadas por el usuario para encontrar una política con acciones discretas. En la segunda fase se aplica una regresión localmente pesada para convertir las acciones discretas en acciones continuas.

En [Tamosiunaite et al., 2009] se presenta un trabajo para manejar estados y acciones continuas basándose en el uso de campos receptivos de funciones gaussianas para determinar los movimientos que debe seguir un brazo robótico para alcanzar un objeto. El método propuesto se diseñó de manera similar a los algoritmos de aprendizaje por refuerzo Q-learning y Sarsa (dos de los algoritmos más utilizados y populares en aprendizaje por refuerzo). Para el diseño del campo receptivo se sigue una distribución normal para colocar los *kernels* de funciones gaussianas en un espacio de acción de tres dimensiones. El espacio está delimitado a las posiciones alcanzables por el brazo. Para la identificación de un estado se utiliza un promedio de los *kernels* (del campo) más cercanos a la posición del brazo analizada, luego se relaciona al estado

promediado (dado por los kernels) con conexiones pesadas a cada una de las acciones posibles del brazo. Así, se combina en una sumatoria un promedio de las funciones gaussianas del campo receptivo considerando sus relaciones pesadas con las acciones, para obtener una acción resultante en determinada posición del brazo robótico en el espacio.

Los diferentes tipos de aproximadores presentados tienen diversos problemas, siendo los más notables los problemas de convergencia (redes neuronales) y costo computacional por el tiempo de procesamiento necesario para explorar el espacio u obtener los datos necesarios para ser entrenados (redes neuronales y procesos, modelos y mezclas gaussianos), incoveniencia que no les permite trabajar en línea. En otros métodos de generalización se necesitan realizar algunos pasos previos sobre los datos lo que puede impedir que estos métodos trabajen en línea (representaciones relacionales con regresiones). Los métodos basados en regresiones, *tile coding* y campos receptivos presentan similitudes con los modelos discretos pero ofrecen la posibilidad de trabajar en línea, por no ser tan costosos computacionalmente.

El algoritmo que se propone en esta tesis usa una representación para la creación de los estados que sigue la idea de los campos receptivos pero que es incremental y construida en línea basándose en dos funciones, el coeficiente de Pearson y las funciones gaussianas. Para la creación de acciones continuas es usado un método novedoso que en esta investigación se presenta y que se basa en la combinación de acciones básicas discretas. Ambas representaciones (de estados y acciones) permiten que el algoritmo trabaje en línea puesto que son ajustadas durante el proceso de aprendizaje.

3.2. Aprendizaje por refuerzo con retroalimentación

Existen trabajos como los presentados en [Wang et al., 2003, Lockerd et al., 2005, Iida et al., 1998, Lockerd & Breazeal, 2006, Lockerd et al., 2006, Hernández, 2010] que proporcionan retroalimentación o guías del usuario a través de periféricos de *hardware* como *joystick*, teclado, ratón, cámaras, entre otros, y combinan esos datos externos con los algoritmos de aprendizaje por refuerzo. A continuación se describen dos de los trabajos más relevantes para la investigación de esta tesis.

En [Wang et al., 2003] se presenta un trabajo aplicado a un robot autónomo usando aprendizaje por refuerzo más comandos dados por el usuario. Para el aprendizaje por refuerzo se utiliza el algoritmo Q-learning, para la combinación de ese algoritmo y los comandos provenientes del usuario se usan procesos de decisión Semi-Markov, SMDP por sus siglas en inglés. El robot puede trabajar solamente con el algoritmo de aprendizaje por refuerzo o permitir al usuario momentos de mando a través de los comandos, que pueden ser usados para calificar o resaltar las preferencias de los usuarios para la realización de una tarea. La colaboración del usuario para guiar al robot es usada en el SMDP como submetas o acciones específicas a realizar y también es usada para entrenar al algoritmo de aprendizaje. Para la retroalimentación dada por el usuario se utilizaron como medios de comunicación con el robot, el *joystick* y el teclado. Se usaron representaciones discretas para el aprendizaje y marcas artificiales en el ambiente. La prueba del método se hizo con los robots *Pionner* y un perro *Astro*.

En [Lockerd et al., 2005] se presenta una revisión de varias preguntas de investigación y factores (como el caso de la asignación temporal, la frecuencia y el alcance de la retroalimentación, la interacción con diferentes usuarios, la transparencia en el proceso de aprendizaje del robot para el usuario, en-

tre otros) que deben considerarse en la interacción de un usuario inexperto que proporciona retroalimentación a un robot mientras éste aprende una tarea. Para estudiar y probar el efecto de la retroalimentación con algoritmos de aprendizaje por refuerzo, se presenta una plataforma de simulación experimental en la cual se enseña a un robot a cocinar. El espacio está modelado como un MDP de estado-acción basado en objetos y se definió un conjunto de acciones básicas. La retroalimentación es dada por el usuario a través del ratón y consiste en otorgar una recompensa por las acciones tomadas por el robot. Para las pruebas del aprendizaje se usó el algoritmo Q-learning. Se partió de un espacio discreto de estados y acciones, y las pruebas fueron realizadas sólo en simulación.

En estos trabajos se usa *hardware* que en la mayoría de las ocasiones no es del dominio de usuarios inexpertos lo que limita su utilización. También, en los algoritmos de aprendizaje por refuerzo utilizados se usan representaciones de estados y acciones discretas que no pueden trabajar en dominios con espacios continuos. Además, la colaboración del usuario puede reducirse a ejemplos de la tarea únicamente. Y en otros casos la retroalimentación con refuerzos y mandatos de acciones específicas no son contemplados o no aparecen en todas las etapas del proceso de aprendizaje. Entre las diferencias del método propuesto en esta tesis con estos trabajos está la inclusión de la voz como un elemento natural (sin *hardware* especial como sensores, *joystick*, ratón, cámaras, guantes, entre otros) para proporcionar retroalimentación, que además es de dos tipos, tanto de refuerzos como de acciones. Y la colaboración del usuario incluye trazas iniciales de ejemplo e intervenciones que pueden ser provistas en diferentes etapas del aprendizaje. Además, es usada una representación que puede trabajar con espacios continuos.

3.3. Aprendizaje por refuerzo con retroalimentación por voz

En [Rybski et al., 2007] se presenta un método de aprendizaje por demostración para un robot. El robot aprende a realizar una tarea mientras observa a un humano y además recibe instrucción verbal. El método de aprendizaje se basa en la descripción de comportamientos, un comportamiento es una acción que el robot puede realizar. Las tareas a su vez se definen como un conjunto de comportamientos y están descritas con las condiciones que deben encontrarse para desencadenar la tarea y las acciones a realizar dadas esas condiciones. Las tareas se relacionan en un grafo acíclico dirigido, donde se representan las transiciones entre una tarea y otra. Entre las tareas predefinidas se encuentra la tarea de aprender y dónde se espera la interacción con un ser humano mediante el lenguaje verbal. Un humano muestra cómo realizar una tarea a un robot mediante instrucciones parecidas a las que delimitan los bloques de control usados en lenguajes de programación, por ejemplo las de la estructura de decisión *if-then-else*. La demostración verbal debe estar estructurada para que, a partir de ella, el robot construya al grafo de tareas. El ambiente donde se encuentra el robot es proporcionado en un mapa para otorgar a las ubicaciones un sentido semántico. El método fue probado en un ambiente real empleando un robot autónomo móvil CMAssist. Aunque se emplea la voz durante el proceso de aprendizaje, la formación que debe seguir el lenguaje para que sea reconocido por el robot es inadecuada para usuarios inexpertos, pues son muy similares a estructuras de lenguajes de programación. Además se trata de un método de aprendizaje supervisado que depende completamente de un usuario durante el entrenamiento.

Un trabajo empleando estados y acciones continuas más voz es presentado en [Zhang & Weng, 2002], donde se muestra un sistema de aprendizaje por refuerzo formado por dos partes, cada una de ellas basada en el algoritmo

de IHDR (*Hierarchical Discriminant Regression*). Con ese método se trató de lidiar con la alta dimensionalidad en las entradas y salidas del sistema de aprendizaje (la descripción de estados y acciones). A partir de los datos obtenidos de las lecturas crudas de los sensores se crean grupos de datos que luego se reducen en funciones gaussianas representadas con su respectiva media y matriz de covarianzas. Las funciones gaussianas forman parte de los nodos internos de un árbol que crece o disminuye según sean la representación y utilidad de las funciones respecto a los datos entrantes. El árbol mantiene como sus hojas a conjuntos de prototipos de los datos de entrada (lecturas crudas de los sensores). El aprendizaje por refuerzo se basa en el algoritmo Q-learning modificado por un parámetro de amnesia, que permite 'olvidar' estados o acciones según el refuerzo recibido. Cada parte del sistema completo contiene lo descrito anteriormente, y cada parte funciona como un elemento del modelo de aprendizaje Actor-Crítico (funcionando como crítico o como actor). Para el entrenamiento del robot se empleó interacción física, un ser humano cambiaba la posición del brazo del robot indicándole verbalmente con un comando o identificador (una palabra) cómo se denominaba esa posición. Posteriormente se reutilizaron las posiciones aprendidas como componentes de un movimiento complejo que fue enseñado verbalmente con la secuencia de comandos previamente aprendidos asociados a posiciones básicas. El método se probó en un ambiente real con un robot autónomo móvil, Robot SAIL. En este trabajo aunque se enseña verbalmente, se emplean comandos simples formados por una única palabra. Respecto al sistema de aprendizaje, se usa una representación de los estados que aunque puede manejar datos crudos directamente de los sensores, puede presentar problemas de almacenamiento y tiempo de respuesta por un crecimiento desmedido del árbol de la representación.

En [Boulebteche et al., 1997] se presenta un sistema de guía por voz para robots autónomos. Se usa el algoritmo Q-learning para el diseño del sistema de aprendizaje por refuerzo que a su vez recibe como entrada comandos de

voz. El reconocimiento de comandos de voz se realiza mediante un microcontrolador especializado. Además, se propone la arquitectura pero no se realizó ninguna implementación del método en un robot real y solamente fue probado el reconocedor de voz.

Pocos trabajos fueron encontrados que utilicen la combinación de voz como elemento de demostración y recompensa con aprendizaje por refuerzo. Los que fueron hallados usan comandos simples o lenguaje que no está orientado al entrenamiento de un robot por parte de un usuario inexperto. Además, la retroalimentación es dada durante el proceso de entrenamiento o sólo durante la fase de demostración. Los algoritmos de aprendizaje están basados en Q-learning y modelos de Actor-Crítico sin considerar los beneficios que puede traer el uso de los algoritmos basados en trazas de elegibilidad para disminuir el tiempo de aprendizaje y la propagación del refuerzo, que en estos casos también es proveniente del exterior.

En esta investigación se propone un algoritmo combinando aprendizaje por demostración y por refuerzo usando trazas de elegibilidad, representaciones continuas y voz. Especialmente a diferencia de los trabajos relacionados, se propone usar la voz como el medio, con comandos y frases cortas, para proporcionar las trazas iniciales de ejemplo de la tarea al robot. Y del mismo modo usar a la voz para recompensar, corregir e indicar comandos de acciones durante todo el proceso de aprendizaje, retroalimentando así al robot.

3.4. *Reward Shaping*

Reward shaping ha sido utilizado para acelerar el aprendizaje por refuerzo como se presenta en [Ng et al., 1999, Laud, 2004]. Existen varias formas en las cuales se ha implementado esta técnica de aceleración, algunos trabajos usan métodos para aprender una función de *reward shaping* como en [Abbeel & Ng, 2004, Konidaris & Barto, 2006, Laud & DeJong, 2002, Grzes & Kudenko,

2009, Dorigo & Colombetti, 1994, Marthi, 2007], en otros más se usan diseños de *reward shaping* estructural en el que se descompone una tarea compleja en tareas simples u objetivos escalados como en [Gullapalli, 1992, Mataric, 1994, Ng et al., 2004, Konidaris & Barto, 2006, Asmuth et al., 2008, Randlov & Alstrom, 1998], y con funciones de *reward shaping* que incrementan en complejidad a través el tiempo como en los trabajos presentados en [Gullapalli, 1992, Mataric, 1994].

Generalmente, todos esos métodos usan el conocimiento previo del dominio para diseñar sus funciones o tratar de aprender las funciones con experiencia. Estas condiciones hacen de estos métodos dependientes del conocimiento *a priori* y por lo tanto de los diseños estáticos de sus funciones de *reward shaping* que permanecen igual a lo largo del entrenamiento. Y en el caso de aprendizaje de las funciones de recompensa es necesario acumular experiencia practicando la tarea, un proceso que puede llevar mucho tiempo. Otros trabajos que tratan de lidiar con los problemas mencionados han incluído retroalimentación de un usuario para modelar la función de *reward shaping* durante el proceso de aprendizaje, como los presentados en [Abbeel & Ng, 2004, Knox & Stone, 2010]. En el primer trabajo un experto proporciona una política de ejemplo y el método propuesto trata de obtener políticas parecidas a ella. Sin embargo, la intervención del experto es solamente a través del ejemplo proporcionado, el resto del método consiste en hacer aprendizaje inverso a partir del ejemplo dado, así el experto no colabora directamente durante el proceso de aprendizaje. En el segundo trabajo, primero es usado un *framework* (un conjunto de programas y librerías) llamado TAMER que se usa para modelar una función de *reward shaping* de acuerdo al refuerzo recibido de un humano, posteriormente, el resultado de este método es añadido a un algoritmo usual de aprendizaje por refuerzo para encontrar la política.

De manera similar a los dos últimos métodos mencionados, el algoritmo propuesto en esta tesis requiere de un usuario para obtener información para

modelar la función de *reward shaping*. A diferencia de los enfoques mencionados, la intervención del usuario incluye refuerzos y comandos de acciones, y es incluída directamente en el algoritmo de aprendizaje, en ejemplos iniciales de la tarea y durante el entrenamiento. De este modo, el algoritmo propuesto recibe retroalimentación dinámica que adapta la función de *reward shaping* durante el proceso de aprendizaje de acuerdo a la evaluación del comportamiento del robot que hace el usuario cuando le enseña la tarea. Así el robot logra aprender cómo realizar una tarea cuando el usuario da retroalimentación correcta o encaminada al objetivo la mayor parte del tiempo, pero también tolerando errores ocasionales de retroalimentación.

3.5. Síntesis

Se han descrito los trabajos más relevantes relacionados con la investigación hecha en esta tesis. Como se presentó se han diseñado diversos métodos para el tratamiento de espacios continuos en los algoritmos de aprendizaje por refuerzo, que trabajan con conocimiento *a priori* del dominio, que pueden o no trabajar en línea y que están basados en generalización y aproximación de funciones. Sin embargo, varios de ellos tienen problemas relacionados con el conocimiento previo necesario para su diseño y el costo computacional (como es el caso de las redes neuronales y los procesos gaussianos que son de los más usados). Estos métodos pueden trabajar cuando se pueden tener largos tiempos de entrenamiento e información suficiente del dominio de la tarea, a diferencia de las codificaciones gruesas, en teja, los campos receptivos y las regresiones que pueden trabajar en línea y requieren menores tiempos de entrenamiento.

Además, aunque se han hecho trabajos que han incluído retroalimentación adicional en los algoritmos de aprendizaje, estos trabajos han utilizado *hardware* adicional y en ocasiones de uso no común (como cámaras, *joystick*, guantes, teclado, ratón, sensores, entre otros), o incluso utilizado la voz

como medio para proporcionar instrucciones y comandos pero que no están orientados a usuarios inexpertos en robótica y programación. Asimismo, en los enfoques que han incluído la voz no solo para retroalimentar, sino también para modelar las funciones de *reward shaping*, la retroalimentación no se ha incluído directamente durante el entrenamiento en los diseños del algoritmo de aprendizaje por refuerzo, sino en fases adicionales y previas al entrenamiento. De ahí que en este trabajo se retomen algunas de las ideas de los métodos para espacios continuos que trabajan en línea (específicamente los campos receptivos, que pueden trabajar directamente sobre los datos continuos y que pueden ser incrementales), y se agregue retroalimentación con voz (con demostración pero sin *hardware* adicional), tratando de usar un enfoque con el habla más natural para la intervención del usuario. Además, se usan las intervenciones del usuario para modelar un método de *reward shaping* dinámico, que se modele y cambie directamente durante el entrenamiento, evitando así la necesidad de tener etapas previas al entrenamiento.

En el siguiente capítulo se detalla el algoritmo propuesto de aprendizaje por refuerzo con retroalimentación por voz, se describen cada una de sus partes resaltando la inclusión de las trazas iniciales y la intervención del usuario.

Capítulo 4

Método Propuesto

En este capítulo se describe el algoritmo propuesto. En la primera sección se da una explicación general del algoritmo y en las secciones posteriores se describen a detalle cada una de sus partes.

4.1. Algoritmo de aprendizaje con retroalimentación proporcionada por voz

El algoritmo está compuesto por tres fases que conforman el entrenamiento completo: demostración con trazas iniciales, entrenamiento con exploración y refinamiento de la política de comportamiento aprendida. En la Figura 4.1 se muestra un esquema del funcionamiento general del algoritmo y a continuación se describen cada una de las etapas de manera general.

En la primera parte, un usuario proporciona trazas mediante comandos de voz que le muestran al robot paso a paso cómo realizar una tarea. Las trazas están formadas por acciones que son comandadas por el usuario y que ejemplifican soluciones de la tarea, como se definió en la sección 2.1.2 en el algoritmo de Sarsa(λ).

En la segunda fase, el robot explora el ambiente como tradicionalmente

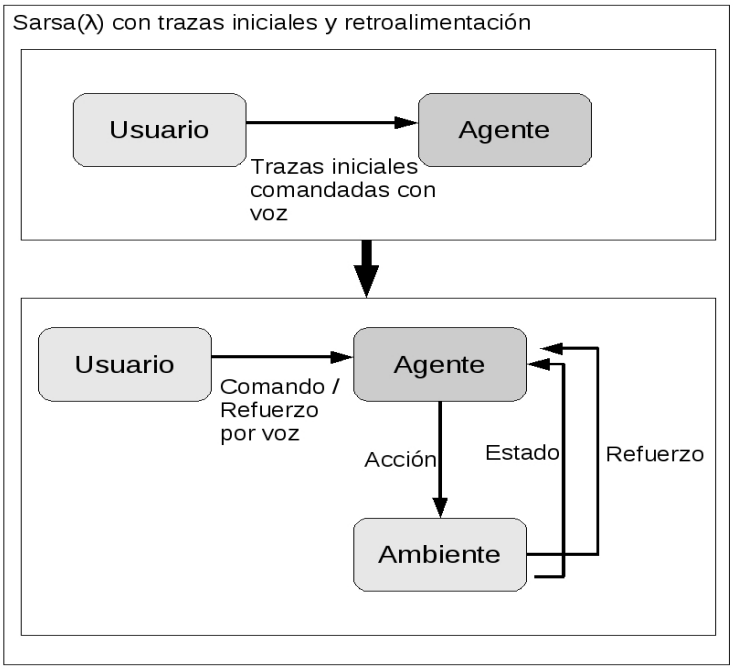


Figura 4.1 Esquema general del algoritmo. En la figura se muestran las fases principales que conforman el algoritmo. En el recuadro superior se muestra la primera fase donde el usuario proporciona mediante voz las trazas iniciales al agente. En el recuadro inferior se muestra el entrenamiento con exploración, donde además de seguir el esquema de un algoritmo tradicional de aprendizaje por refuerzo, identificando estados, haciendo acciones y recibiendo recompensas, también se ilustra la colaboración del usuario proporcionando retroalimentación con voz.

se haría con un algoritmo de aprendizaje por refuerzo, de manera independiente y recibiendo los refuerzos tradicionales por la percepción que tiene del ambiente. Sin embargo, en esta etapa en el algoritmo propuesto también se recibe la retroalimentación de comandos y calificativos mediante voz por parte de un usuario que observa el entrenamiento del robot. Mientras no reciba retroalimentación el algoritmo de aprendizaje se comporta como un Sarsa(λ) tradicional. Cuando recibe calificativos, se modifica la función de recompensa de acuerdo a la interpretación del vocabulario. Si se reciben comandos de acciones, se realiza el comando (la acción), indicado por el usuario, para después continuar con la elección de acciones de manera usual de acuerdo al algoritmo de aprendizaje. Como es usual en estos algoritmos, el entrenamiento es hecho realizando varios episodios o periodos de entrenamiento, cada episodio inicia en un determinado estado inicial en el ambiente y termina cuando se alcanza el estado objetivo (cuando se logra terminar la tarea).

Finalmente, en la última etapa se prueba la política de comportamiento que ha sido aprendida por el robot, y de ser necesario el usuario puede mejorar la política con comandos de voz nuevamente.

De este modo, la intervención del usuario puede ocurrir en cualquier momento, e incluso puede ser errónea en cierta cantidad (por fallas en el reconocimiento de voz o humanas) como se verá a detalle en las siguientes secciones.

4.2. Retroalimentación obtenida por voz

Además de utilizar los refuerzos tradicionales de un algoritmo de aprendizaje por refuerzo, la función de recompensa y el aprendizaje se modifican usando retroalimentación proveniente de un usuario proporcionada verbalmente. La mayoría de los trabajos que se han hecho con interacción con voz en robótica han sido realizados en el idioma inglés, poco se ha desarrollado el uso de otros idiomas para usuarios que no dominen el inglés. Siendo el español uno de los

idiomas más usados y aprovechando el desarrollo de proyectos de reconocimiento que se han estado realizando actualmente para este idioma, se eligió al español como el idioma de comunicación entre el usuario y el robot.

La retroalimentación puede ser de dos tipos: comandos y calificativos. Ambos pueden darse a través de palabras o frases cortas que puedan ser interpretadas de acuerdo al vocabulario del dominio de la tarea. Los comandos de voz son palabras que indican alguna acción básica que el robot es capaz de realizar dentro del dominio de la tarea, de acuerdo al vocabulario especificado. Los calificativos son palabras que evalúan el desempeño del robot durante el proceso de aprendizaje. El usuario puede intervenir en cualquier momento durante la fase de entrenamiento con cualquier tipo de retroalimentación, e incluso, puede cometer ocasionalmente errores proporcionando evaluaciones o comandos incorrectos.

4.3. Reconocimiento e interpretación

El reconocimiento de voz se hizo con un reconocedor existente del proyecto DIMEx [Pineda, 2010], solamente se definió el vocabulario de acuerdo al dominio de la tarea y a partir de él se hicieron los modelos del lenguaje. Se recabaron un conjunto de frases de ejemplo, para comandar y evaluar al robot, en un grupo de usuarios potenciales (10 usuarios). A partir de ellas se extrajo el vocabulario del reconocedor. Posteriormente con estas frases se alimentó al sistema (del software de reconocimiento de voz *Sphinx CMU*, basado en modelos ocultos de Markov) para crear los modelos del lenguaje. Para el proceso de interpretación, se diseñó un intérprete para proporcionar la información útil del lenguaje reconocido al algoritmo de aprendizaje.

En el proceso de reconocimiento se utilizaron los modelos acústicos generados por [Pineda, 2010] para el idioma español. Los modelos del lenguaje se generaron de acuerdo al vocabulario que fue definido dentro del dominio de

las tareas.

Se diseñó al intérprete para la traducción de las frases reconocidas que contienen los comandos y/o calificativos. El intérprete localiza las palabras clave o de interés en cada etapa del aprendizaje del robot y de acuerdo a su significado indica si se trata de un calificativo y cuál es éste, o si es un comando y la acción que se está indicando, esta información es enviada al algoritmo de aprendizaje para evaluar el comportamiento o cambiar las acciones que se están realizando.

El vocabulario está formado por adjetivos, verbos y conectores (preposiciones, artículos, conjunciones), que pueden usarse individualmente o combinadas para formar frases cortas que expresan acciones y evaluación (aproximadamente 200 elementos en el vocabulario). En las tablas 4.1 y 4.2 se presentan algunos ejemplos de uso del vocabulario del reconocedor. En la primera se muestran ejemplos para indicar acciones, en la segunda ejemplos para realizar evaluaciones de las acciones. En el apéndice A puede encontrarse un listado de todo el vocabulario usado en el reconocedor.

4.4. Aprendizaje por Refuerzo

El algoritmo propuesto es una modificación del Sarsa(λ), mostrado en el algoritmo 4.3. En las siguientes secciones se detallan sus componentes.

4.4.1. Representación para espacios continuos

El algoritmo fue diseñado para tratar con espacios continuos tanto de estados como de acciones. La representación diseñada es similar a los campos receptivos y se crea en línea de forma incremental de acuerdo a la exploración del ambiente. Los estados son identificados usando una función de similitud que trabaja como una función de un *kernel*, las acciones continuas se crean al

Tabla 4.1 Ejemplos de vocabulario para indicar acciones al robot

Izquierda	Derecha
Gira a la izquierda / derecha	Es para el otro lado
Gira hacia la izquierda / derecha	Avanza
Hacia tu izquierda / derecha	Adelante
A tu izquierda / derecha	Sigue avanzando
A la izquierda / derecha	Sigue caminando
Hacia la izquierda / derecha	Sigue derecho
Da vuelta a la izquierda / derecha	Camina derecho
Ve a la izquierda / derecha	Ve todo derecho
A mano izquierda / derecha	Ve derecho
Vuelta a la izquierda / derecha	Ahí derecho
Ve hacia la izquierda / derecha	Todo derecho
Dobla a la izquierda / derecha	Síguele
Dobla hacia la izquierda / derecha	Atrás
Hacia atrás	Hacia adelante
Para atrás	Para adelante
Hey regresa	Mejor regresa
Regresa	

Tabla 4.2 Ejemplos de vocabulario para evaluar las acciones del robot con su respectivo valor de recompensa

Hasta ahí +100	Ya hasta ahí +100
Hasta ahí nada mas +100	Para ahí +100
Excelente +50	Bien +10
Así como vas +10	Como vas +10
Tu síguele +10	Así sigue +10
Sigue así +10	Sí así +10
Muy bien +50	Bien hecho +50
Buen trabajo +50	Vas bien +10
Mal -10	Terrible -50
Así no -10	Muy mal -50
Estás mal -10	Hey para allá no -10
Hacia allá no -10	Para allá no -10
Allá no -10	Ya no hagas eso -50
Que no -50	Que eso no -50
Que ahí no -50	Que así no -50
Por ahí no -10	Ya te equivocaste -10
Ya la regaste -50	No te vayas por ahí -10
No era por ahí -50	Por ahí no era -50

combinar acciones discretas simples que son definidas de acuerdo al dominio de la tarea. La representación almacena vectores de datos de los sensores que identifican un área del ambiente (para representar los estados) y que sirven para realizar las comparaciones con las funciones de similitud, y a cada estado (vector de datos) lo relaciona con cada una de las acciones discretas (pertenecientes a un conjunto básico predefinido) con conexiones con pesos asociados. Los pesos asociados corresponden a valores Q del algoritmo de aprendizaje por refuerzo.

En la figura 4.2 se muestra un ejemplo de la representación diseñada donde se obtienen las lecturas de los sensores, después se hace el análisis para identificar el estado, posteriormente se elige una acción o se calcula una combinando acciones discretas, y finalmente la acción se realiza.

El espacio del ambiente que es cubierto o conocido por el robot y representado, depende de los sitios y circunstancias recorridos durante la demostración y por indicaciones del usuario, y de la exploración independiente hecha por el robot.

La representación diseñada trabaja con estados y acciones que puedan ser representados con números reales (un conjunto de ellos en el caso de representación de un estado) debido a la naturaleza de las funciones de similitud empleadas y al método de combinación de acciones discretas, si los datos fueran de otro tipo sería necesario definir otras funciones de similitud y seguir la idea de combinación de acciones discretas pero de acuerdo al nuevo tipo de datos.

4.4.2. Representación y tratamiento de los estados continuos

En este apartado se describe el proceso de creación e identificación de estados en un dominio continuo.

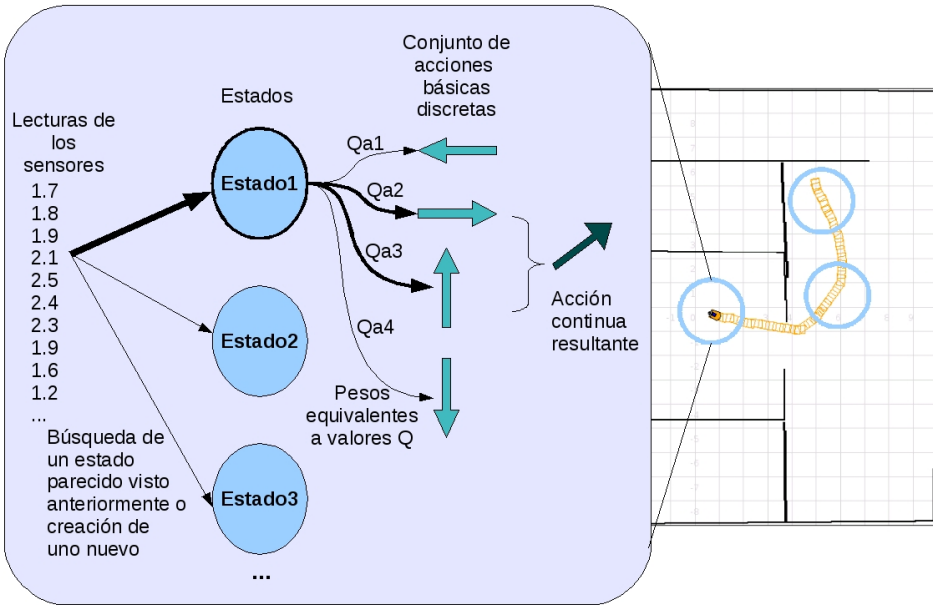


Figura 4.2 Representación para estados y acciones continuas. Se muestra un ejemplo de un robot en un ambiente realizando una tarea de navegación. De izquierda a derecha, se muestra un arreglo de datos que provienen de los sensores, luego estos datos son comparados con datos que representan estados almacenados. Una vez que se identifica un estado al que es similar, se muestran las conexiones que éste tiene con un conjunto de acciones discretas y sus pesos asociados (valores Q). Luego se ilustra la elección de dos acciones con mayor peso en su conexión, y cómo éstas se combinan para crear una acción resultante continua.

Algoritmo 4.1 Recibir trazas iniciales

Variables de entrada: α coeficiente de aprendizaje (número real), γ factor de descuento (número real), λ factor de descuento de la traza (número real)
Variables de salida: Q estructura dinámica (ej. lista con números reales donde s, a identifican el estado y la acción a los que corresponde cada número de la lista) que representa la función de valor, e estructura dinámica (ej. lista con números reales donde s, a identifican el estado y la acción a los que corresponde cada número de la lista) que representa la información de la traza
Variables locales: s, s' estados (números enteros que identifican estados, $s, s' \in S$, S conjunto de estados), a acción (número entero que identifica una acción, $a \in A$, A conjunto de acciones), r recompensa (número real)

```
Inicio
Inicializar e(s,a) = 0 para todos los s y a
repeat
  repeat
    if s no es parte del conjunto S then
      Agregar s a S y sus respectivas conexiones pesadas, con su Q(s,a),
      a cada acción del conjunto A de acciones básicas discretas
    end if
    Realizar a provista por el usuario y observar r, s'
    Actualizar Q(s,a) y e(s,a) para la traza como en Sarsa(λ)
  until s sea un estado terminal indicado por el usuario
until la última traza sea proporcionada
Fin
```

Algoritmo 4.2 Elegir una acción o crear una acción continua

Variables de entrada: s, s' estados (números enteros que identifican estados, $s, s' \in S$, S conjunto de estados), a, a_1, a_2 acciones (números enteros que identifican acciones, $a \in A$, A conjunto de acciones), Q estructura dinámica (ej. lista con números reales donde s, a identifican el estado y la acción a los que corresponde cada número de la lista) que representa la función de valor
Variables de salida: s, s' estados (números enteros que identifican estados, $s \in S$, S conjunto de estados), a acción (número entero que identifica una acción, $a \in A$, A conjunto de acciones), Q estructura dinámica (ej. lista con números reales donde s, a identifican el estado y la acción a los que corresponde cada número de la lista) que representa la función de valor
Variables locales: va_r valor (número real) de la acción continua resultante, va_1, va_2 valores (números reales) de dos acciones discretas, a_r acción continua resultante (número entero que identifica una acción)

Inicio
if s no es parte del conjunto S conocido **then**
  Agregar s a S y sus respectivas conexiones pesadas ($Q(s, a)$) hacia el conjunto A de acciones básicas discretas
  if usuario no selecciona una acción a en s **then**
    Seleccionar a en s aleatoriamente dentro de la acciones básicas en A
  end if
else
  if usuario no selecciona una acción a en s **then**
    Elegir las dos a_1, a_2 con mayor valor Q asociado a sus conexiones con s
    if Todas las acciones, del conjunto A asociadas al estado, tienen igual valor Q **then**
      Elegir a aleatoriamente de entre todas ellas en A
    else
      if a_1, a_2 tienen igual valor Q y son contrarias) **then**
        Elegir aleatoriamente como a a una de las accciones a_1, a_2
      else
        $va_r(s) = (Q(s, a_1)/Q(s, a_2)) * va_1 + (1 - Q(s, a_1)/Q(s, a_2)) * va_2$
        (acción continua)
        $a \leftarrow a_r$
      end if
    end if
  end if
end if
Fin

Algoritmo 4.3 Sarsa(λ) con trazas iniciales y retroalimentación dinámica.

Variables de entrada: α coeficiente de aprendizaje (número real), γ factor de descuento (número real), λ factor de descuento de la traza (número real)
Variables de salida: Q estructura dinámica (ej. lista con números reales donde s, a identifican el estado y la acción a los que corresponde cada número de la lista) que representa la función de valor, e estructura dinámica (ej. lista con números reales donde s, a identifican el estado y la acción a los que corresponde cada número de la lista) que representa la información de la traza
Variables locales: s, s' estados (números enteros que identifican estados, $s, s' \in S$, S conjunto de estados), a, a' acciones (números enteros que identifican acciones, $a, a' \in A$, A conjunto de acciones), r recompensa (número real), δ variable auxiliar para el cálculo de la variación en la función de valor, t tiempo (número entero)

```
Inicio
Recibir trazas iniciales
for each episodio de entrenamiento do
  repeat
    Elegir una acción o crear una acción continua
    if usuario selecciona una acción a en s then
      a es la acción indicada por el usuario
      Realizar a y observar s'
      r ← g(r, a) {donde g(r, a) es una función acotada que depende de la acción
      seleccionada}
    else
      Realizar a y observar r, s'
    end if
    if usuario provee retroalimentación f en s' then
      r ← h(r, f) {donde h(r, f) es una función acotada que depende de la
      retroalimentación provista}
    end if
    Seleccionar a' de s' usando una política derivada de Q (e.g. e − greedy)
    δ = r + γQ(s', a') − Q(s, a)
    if a es continua then
      e_t(s, a_1) = γλe_{t−1}(s, a_1) + 1
      e_t(s, a_2) = γλe_{t−1}(s, a_2) + 1
    else
      e_t(s, a) = γλe_{t−1}(s, a) + 1
    end if
    for each s, a do
      Q(s, a) = Q(s, a) + αδe(s, a)
      e_t(s, a) = γλe_{t−1}(s, a)
    end for
    s ← s', a ← a'
  until s sea un estado terminal
end for
Probar política
Fin
```

Algoritmo 4.4 Probar política

Variables de entrada: α coeficiente de aprendizaje (número real), γ factor de descuento (número real), λ factor de descuento de la traza (número real), Q estructura dinámica (ej. lista con números reales donde s, a identifican el estado y la acción a los que corresponde cada número de la lista) que representa la función de valor, e estructura dinámica (ej. lista con números reales donde s, a identifican el estado y la acción a los que corresponde cada número de la lista) que representa la información de la traza
Variables de salida: Q estructura dinámica (ej. lista con números reales donde s, a identifican el estado y la acción a los que corresponde cada número de la lista) que representa la función de valor, e estructura dinámica (ej. lista con números reales donde s, a identifican el estado y la acción a los que corresponde cada número de la lista) que representa la información de la traza
Variables locales: s estado (número entero que identifica un estado, $s \in S$, S conjunto de estados), a acción (número entero que identifica una acción, $a \in A$, A conjunto de acciones), r recompensa (número real)
Inicio
Usar la política aprendida en $Q(s, a)$
if usuario selecciona una acción a en s **then**
  a es la acción indicada por el usuario
  Realizar a y observar s'
  $r \leftarrow g(r, a)$
end if
if usuario provee retroalimentación f en s' **then**
  $r \leftarrow h(r, f)$
end if
Actualizar $Q(s, a)$ y $e(s, a)$ para la traza como en Sarsa(λ)
Fin

Desde que inicia el entrenamiento del robot con las trazas iniciales y durante el resto del proceso de aprendizaje, se analizan los datos provenientes de los sensores para la identificación de estados que definen la situación del robot en el ambiente.

La representación diseñada para los estados es incremental, conforme el robot va explorando el ambiente, se van obteniendo datos de los sensores. Las nuevas lecturas obtenidas se comparan contra datos existentes almacenados (que representan a un estado), si es que existen, identificándose así a un estado conocido; de no existir datos almacenados, se crea un nuevo estado.

Un arreglo de lecturas provenientes de los sensores está conformado por una serie de números reales, por ejemplo, si los sensores fueran un cinturón de láseres cubriendo 180 °, el arreglo estaría conformado por 180 lecturas de números reales:

$x = [3{,}16 \quad 4{,}21 \quad 4{,}25 \quad 2{,}19 \quad 2{,}18 \quad 1{,}52 \quad 1{,}65 \quad 1{,}78 \quad 1{,}289... \quad 3{,}4]$

La creación de un estado nuevo consiste en almacenar tal y como se obtienen las lecturas de los sensores. La idea es que, con el uso de una función de similaridad, un estado esté identificado por unas lecturas representativas de una zona o situación del ambiente que no son parecidas a algo visto anteriormente y que sean lo suficientemente representativas para que cubran una zona del espacio (a un grupo de lecturas similares).

Si existe parecido de los datos nuevos con las lecturas que representan a un estado registrado anteriormente, las lecturas nuevas no son consideradas, y la información necesaria para el algoritmo de aprendizaje es tomada del estado almacenado.

La comparación de los datos nuevos de los sensores contra los almacenados se realiza mediante una función de similitud considerando un umbral de semejanza entre ellos. Se utilizaron dos funciones de similitud, el coeficiente de correlación de Pearson y la función gaussiana. Se usan dos funciones distintas, porque aunque la primera propuesta fue usar solamente la función gaussiana,

ésta no presentaba buenos resultados para los dos tipos de tareas aprendidas debido a la diferencia entre los datos que representan a los estados.

Usando el coeficiente de correlación de Pearson se tiene la ecuación (4.1).

$$r = \frac{N\Sigma xy - \Sigma x \Sigma y}{(\sqrt{N\Sigma x^2 - (\Sigma x)^2})(\sqrt{N\Sigma y^2 - (\Sigma y)^2})} \tag{4.1}$$

donde x, y son respectivamente las variables independiente y dependiente del coeficiente de correlación, que para el caso de identificación de estados son los arreglos de datos correspondientes a las lecturas obtenidas de los sensores, uno conteniendo las lecturas almacenadas que caracterizan a un estado registrado y el otro conteniendo nuevas lecturas, y N el número de datos de cada arreglo de datos de las variables.

Y usando una función gaussiana se tiene la ecuación (4.2).

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)/(2\sigma^2)} \tag{4.2}$$

donde μ es un arreglo que contiene las lecturas de los sensores que representa a un estado registrado, π el número pi (3.141516), σ la desviación estándar y x un arreglo de lecturas nuevas de los sensores.

Cabe aclarar que los vectores de lecturas que representan a los estados y que se van almacenando no se modifican durante el resto del aprendizaje. Estos vectores no son ajustados con futuras lecturas considerando en parte que la exploración del espacio es guiada durante el entrenamiento por el usuario enfocándose en las zonas de interés o necesarias para la tarea. Por otro lado, el ajuste de estos vectores podría requerir una exploración más amplia del espacio (tal vez innecesaria para realizar la tarea) para obtener más información del ambiente.

Cuando se crea un estado nuevo, éste se relaciona con un conjunto de acciones básicas discretas estas conexiones que se establecen tienen asignado un peso, el conjunto de acciones está definido de antemano de acuerdo al dominio de la tarea. Los pesos equivalen a valores Q como será visto más

adelante. En los algoritmos 4.1 y 4.2 se muestran los pasos para la creación de estados según la representación, y en la figura 4.2 ilustra la representación diseñada desde la obtención de los datos de los sensores, las relaciones entre estados y acciones, hasta la creación de acciones continuas.

4.4.3. Representación y creación de las acciones continuas

El diseño de las acciones continuas parte de un conjunto básico de acciones discretas. De acuerdo al dominio de la tarea se define un número pequeño de acciones básicas discretas con las cuales el robot puede explorar el ambiente inicialmente. Conforme se realiza en el entrenamiento las acciones se van ajustando de acuerdo a los refuerzos y retroalimentación recibidos.

Cuando se crea un estado nuevo durante la segunda fase del algoritmo, durante la exploración, la acción a realizar se elige aleatoriamente dentro del conjunto básico de acciones discretas iniciales. Si el estado nuevo es creado durante la fase inicial cuando el usuario está proporcionando las trazas de ejemplo, la acción no es elegida aleatoriamente, sino que la acción a realizar es la que el usuario indica.

Si el estado ya se encuentra almacenado la acción que se realiza resulta de la combinación de dos acciones discretas. El par de acciones electo se forma por aquellas acciones discretas con el mayor peso (valor Q) en su conexión con el estado. Si las conexiones con mayor peso corresponden a dos acciones inversas de acuerdo al dominio de la tarea, se elige una de ellas al azar. Si todos los pesos de las conexiones a las acciones discretas tienen igual peso dentro de un umbral predefinido, se elige una de ellas al azar. Cuando el usuario comanda en la segunda fase de entrenamiento, la acción que se realiza es la que el usuario indica.

Se eligieron solamente dos acciones para crear una acción continua por el

reducido número de acciones discretas que formaban el conjunto de acciones discretas definidas en las tareas, y porque las pruebas realizadas con combinaciones con diferente número de acciones no cambiaban el desempeño del robot. Dependiendo del dominio de la tarea el número de acciones que se combinan podría ser ajustado.

Como ya se dijo, cuando puede obtenerse, una acción continua resulta de combinar dos acciones discretas. La combinación que resulta en la magnitud de la acción a realizarse se hace con los pesos (valores Q) de las conexiones entre el estado y el par de acciones elegido, como se muestra en la ecuación (4.3).

Si $Q(s, a_1) < Q(s, a_2)$

$$va_r\,(s) = (Q\,(s, a_1)\,/Q\,(s, a_2)) * va_1 + (1 - Q\,(s, a_1)\,/Q\,(s, a_2)) * va_2 \qquad (4.3)$$

Donde los valores (representados por números reales) de las acciones discretas que se combinan son va_1, va_2 y va_r es el valor de la acción resultante de combinarlas. Por ejemplo, una acción discreta para navegación puede estar representada por un sentido y un valor angular que indique dirección (si no se ajusta otra magnitud), así se podrían tener las acciones 'Avanzar hacia la derecha a 90 °' y 'Avanzar hacia el frente (0 °)', suponiendo que la primera acción tenga mayor peso respecto a la segunda, una acción resultante de combinar las dos mencionadas podría ser 'Avanzar a la derecha, 70 °'

La creación de acciones continuas se detalla en el algoritmo 4.2 y se ilustra en la figura 4.2.

Las acciones elegidas o creadas en un determinado estado se continúan realizando hasta que se detecte un cambio de estado o hasta que se encuentre en alguna zona de penalización máxima (cerca de la pared por ejemplo). Durante el entrenamiento, cuando el usuario indica una acción a realizar (dentro del conjunto básico de acciones), se realiza la acción comandada y después de ella, el algoritmo continúa con su funcionamiento como se ha descrito.

4.4.4. Diseño de la función de recompensa

La función de recompensa incluída en el algoritmo 4.3 devuelve un refuerzo compuesto R por una recompensa R_t de los algoritmos tradicionales de aprendizaje por refuerzo más el refuerzo R_e obtenido del usuario mediante la retroalimentación verbal.

$$R = R_t + R_e \tag{4.4}$$

La función de recompensa sigue el estilo de *reward shaping*; sin embargo, el diseño de la función de *reward shaping* no es estático ni la función es aprendida durante el entrenamiento, sino que es modificado dinámicamente con la retroalimentación proporcionada por el usuario. Se dice que la función de *reward shaping* es dinámica porque las intervenciones no son programadas, suceden cuando el usuario observa un comportamiento en el que desea intervenir y puede hacerlo expresando diferentes intensidades de evaluación de acuerdo al vocabulario que utilice. Esto es, que a lo largo del entrenamiento, en cualquier momento puede haber retroalimentación, puede ser variable (comandos y/o calificativos) y puede beneficiar o no al aprendizaje (por errores humanos o de reconocimiento de voz).

En el esquema propuesto de *reward shaping* se consideran tres casos de retroalimentación: permanente, esporádica y ruidosa. En la retroalimentación permanente se espera que el usuario la proporcione constantemente, en los mismos estados y de la misma forma, y que además esté encaminada hacia la meta. De esta forma la función de *reward shaping* trabaja igual que las funciones basadas en potenciales. Con la retroalimentación esporádica el usuario proporciona comandos y calificativos en diferentes estados y acciones, y con diferente frecuencia. Sin embargo, si el usuario proporciona recompensas con la suficiente persistencia en ciertos estados y con ciertas acciones, es posible que con ello se establezcan nuevas submetas que finalmente en conjunto lleven a la

meta final. La función modelada de esta forma funcionaría de manera similar a una función basada en potenciales también. Por último, la retroalimentación puede ser ruidosa debido a errores en el reconocimiento de voz o a errores del usuario (como atrasos o comandos de acciones incorrectos), y por lo tanto llevar a metas incorrectas. Aun así, si el robot puede explorar lo suficiente y/o el usuario proporciona retroalimentación adicional correcta, todavía será posible corregir el aprendizaje y alcanzar la meta. En este último caso es posible que el aprendizaje sea aún más tardado que con un algoritmo sin *reward shaping*, sin embargo es posible converger a la solución también.

Así, al usar la retroalimentación del usuario, no es necesaria la acumulación de experiencia para modificar la función de recompensa y por lo tanto el comportamiento del robot, y se tiene una función de refuerzo que cambia de acuerdo al progreso en el aprendizaje.

4.4.5. Diseño de la función de valor

El peso de las conexiones entre un estado y las acciones básicas discretas es equivalente a un valor Q y se actualiza siguiendo los principios de una función de valor Q considerando trazas de elegibilidad sin reemplazo, similar a Sarsa(λ). Sólo que para el algoritmo 4.3 aquí presentado, se actualizan los valores Q de las dos acciones básicas que se combinaron para dar lugar a la acción (continua) que se ejecuta realmente, a menos que se realice una acción discreta con lo cual sólo se actualiza el valor de ésta. La actualización se hace como se muestra en la ecuación (4.5).

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha\delta_t e_t(s,a) \tag{4.5}$$

La actualización de (4.5) de hace para todo s, a, donde δ se actualiza de acuerdo a (4.6).

$$\delta_t = r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t) \tag{4.6}$$

Y si $s = s_t$ y $a = a_{1t}$ o $a = a_{2t}$, entonces se actualiza de acuerdo a la ecuación (4.7).

$$e_t(s, a) = \gamma\lambda e_{t-1}(s, a) + 1 \tag{4.7}$$

Si no se cumple la condición anterior para actualizar siguiendo (4.7), se actualiza de acuerdo a la ecuación (4.8).

$$e_t(s, a) = \gamma\lambda e_{t-1}(s, a) \tag{4.8}$$

Donde s es un estado, a corresponde a una acción, a_{1t} y a_{2t} son las dos acciones discretas con conexiones con mayor valor Q, α el coeficiente de aprendizaje, γ es un factor de descuento y λ es un parámetro de descuento de la traza. La actualización de dos valores Q de dos acciones permite la creación de acciones continuas como ya se mostró anteriormente.

Se debe resaltar la utilización de trazas de elegibilidad, cuyo uso reduce la cantidad de exploración necesaria en los algoritmos de aprendizaje por refuerzo. Y en el caso del algoritmo propuesto, también ayuda a la propagación de la retroalimentación pues ésta puede en ocasiones llegar desfasada entre el estado en el cual el usuario deseaba intervenir y el estado en el cual se encuentra el robot al recibir la retroalimentación. Debido a que se trata de aprendizaje en línea, se están constantemente realizando acciones que pueden llevar a otros estados antes de recibir la intervención oportuna del usuario, para ello se usa el mecanismo de trazas de elegibilidad que afecta no sólo a un único estado, sino también a los estados contenidos a lo largo de la traza.

4.5. Síntesis

En este capítulo se presentó un algoritmo novedoso de aprendizaje por refuerzo con retroalimentación proporcionada por voz. El algoritmo incluye varios aspectos novedosos:

- Aprendizaje en línea incluyendo intervención del usuario.

- Una representación de estados y acciones para espacios continuos, con nuevo esquema para diseñar acciones continuas, basado en un conjunto básico de acciones discretas.

- La inclusión del habla como medio de intervención del usuario.

- Un esquema nuevo de *reward shaping* que es dinámico gracias a la intervención del usuario.

En el siguiente capítulo se presentan los experimentos realizados con el algoritmo propuesto y variantes de él respecto a la intervención del usuario. Se muestra el desempeño del algoritmo al utilizarlo para el aprendizaje de algunas tareas de navegación y en una tarea de manipulación.

Capítulo 5

Experimentos

En los experimentos que se presentan en este capítulo un robot móvil y un brazo robótico aprenden tareas de navegación y de manipulación. Estas tareas son útiles en robots de servicio que son móviles y/o que tienen manipuladores, por ejemplo, los robots guía, los robots que entregan y recogen objetos en diferentes ambientes (oficinas, hogares, hospitales, entre otros), los robots que recogen objetos haciendo limpieza u organizando objetos. Los robots móviles de servicio como los que entregan y recogen objetos o recados, en oficinas, casas, o que son guías, muchas veces requieren aprender nuevas rutas para realizar sus recorridos y/o ajustar las rutas que ya conocen por situaciones inesperadas que suceden en el ambiente (como cuando aparecen obstrucciones inesperadas), e incluso necesitan aprender nuevas rutas hacia destinos nuevos. De ahí que sea útil para los robots móviles de servicio desarrollar sus capacidades de navegación, para que éstos puedan aprender durante la ejecución de sus tareas nuevos caminos o ajustar sus rutas ante eventualidades que puedan ocurrir en los ambientes donde éstos operan. Además, varios de los robots de servicio usan manipuladores para realizar sus tareas (como los robots de limpieza que recogen objetos o los que recogen y entregan objetos), sin embargo, existe una gran posibilidad de que los objetos que deban tomar, mover o reco-

ger sean de distintas formas y estén en distintas posiciones, de ahí que llegue a ser necesario aprender a manipular distintos objetos en ambientes diversos.

Los experimentos fueron hechos con los dos robots para aprender cuatro tareas de navegación y una tarea de manipulación. Estos experimentos se hicieron para probar al algoritmo en dos dominios de tareas distintos, además de en condiciones simuladas y reales. Las primeras tareas, las de navegación, se hicieron en simulación para experimentar de manera flexible al incrementar la dificultad de las tareas y la variedad de los ambientes. Posteriormente se hicieron los experimentos en una tarea de manipulación para probar el algoritmo en condiciones reales, con un robot haciendo una tarea en un ambiente real.

Los experimentos fueron hechos principalmente para observar el efecto favorecedor en el aprendizaje de la inclusión de la retroalimentación por voz en línea. Como se esperaba la agilización en el proceso de aprendizaje, se muestra el tiempo de entrenamiento usado para enseñar al robot cinco tareas en diferentes condiciones. También, se muestra el número de intervenciones realizadas en cada experimento que incluyó retroalimentación, para observar la variación en la cantidad necesaria de retroalimentación durante el aprendizaje además del comportamiento del usuario. Además, se buscó observar el funcionamiento de la representación diseñada para estados y acciones, y el efecto en el aprendizaje de los errores en la retroalimentación por el uso de la voz.

Los primeros experimentos fueron hechos para observar y comparar el comportamiento del robot usando la representación diseñada para espacios continuos y usando una representación discreta tradicional, ambas con el algoritmo tradicional Sarsa(λ). En estos experimentos se esperaba mostrar el suavizado en los movimientos del robot y la reducción en distancia recorrida al usar la representación continua propuesta. Además se buscaba probar que la representación funcionaba construyéndose en línea en un algoritmo tradicional de aprendizaje por refuerzo.

Posteriormente se hicieron experimentos enfocados en probar el algoritmo Sarsa(λ) gradualmente modificado. Se buscó probar que tan favorecedor para el aprendizaje era el uso de las trazas iniciales solamente, y luego el uso de la retroalimentación también de manera independiente. Con estos experimentos se pudo determinar el grado de contribución para la agilización del aprendizaje de cada uno de estos componentes de manera independiente. Después se probó el algoritmo completo Sarsa(λ) modificado como es propuesto, usando las trazas iniciales y la retroalimentación de manera conjunta, para observar el efecto final en el aprendizaje.

El resto de los experimentos estuvo enfocado en observar el efecto en el aprendizaje de los problemas derivados del reconocimiento de voz cuando se proporciona retroalimentación. Se probó el caso con retroalimentación perfecta (sin ningún tipo de error) y un caso controlado donde la retroalimentación fuera 50 % ruidosa (con errores) y 50 % perfecta (sin ningún tipo de error). En el primer caso se esperaba que la retroalimentación perfecta agilizara aun más el aprendizaje por la ausencia de errores de interpretación y humanos. En el segundo caso se buscaba observar que tan perjudicial era para el aprendizaje (la tolerancia del algoritmo) cuando la retroalimentación tenía más errores de los habituales.

A continuación se describen las condiciones en las cuales fueron realizados los experimentos y los resultados obtenidos.

5.1. Características de los robots

Se utilizaron dos robots, un robot autónomo móvil *Pioneer* 2 simulado [Pioneer_MobileRobots, Player_Stage] y un brazo robótico *Harmonic Arm 6M* [HarmonicArm_AAICanadaInc]. El *Pioneer* equipado con un cinturón de láseres frontales cubriendo 180 ° y un cinturón de sonares traseros, como sensores. El brazo robótico de 6 grados de libertad incluyendo un *gripper* en

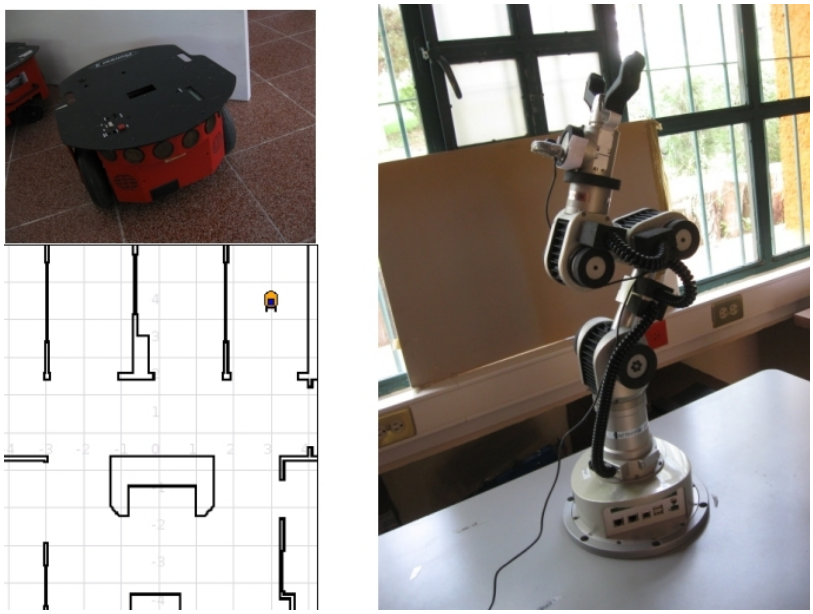


Figura 5.1 Robots utilizados en los experimentos. Se muestra a la izquierda al robot *Pioneer* real y su aspecto en simulación en 2D, a la derecha se muestra el brazo robótico.

su extremo, fue equipado con una webcam adicional colocada en el extremo del brazo junto al *gripper*. En la figura 5.1 se muestran los robots empleados.

5.2. Descripción de los ambientes

En los experimentos usando el robot móvil se aprendieron tareas de navegación en cuatro diferentes mapas que se muestran en la figura 5.2. Cada mapa fue elegido considerando la variabilidad en las características físicas de la estructura. En los diferentes mapas, hay paredes irregulares (no completamente rectas), las habitaciones y puertas o accesos tienen diferentes dimensiones. Los diseños de los mapas fueron elegidos para tareas de navegación con un nivel de dificultad moderado, puesto que el principal objetivo era observar los efectos de la inclusión en el algoritmo de las trazas iniciales y la retroalimentación por voz en el aprendizaje, sin enfocarse directamente en la solución de problemas

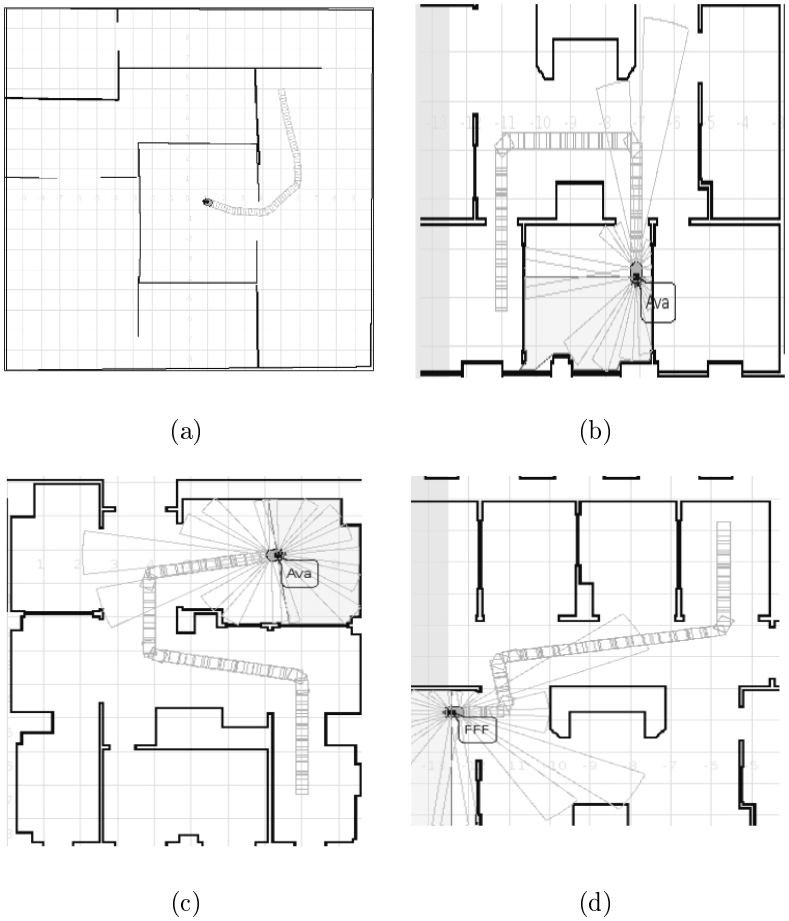
de navegación. En segundo término se quería observar el funcionamiento de la representación propuesta en estos ambientes, principalmente la capacidad de ésta para trabajar en línea.

Con los mismos objetivos de los experimentos hechos en navegación pero buscando probar el algoritmo de manera real se usó el brazo robótico. Además se aprovechó para probar la funcionalidad del algoritmo propuesto con otro tipo de tarea. Los experimentos fueron hechos con el brazo colocado sobre una mesa y con un objeto rectangular de color blanco marcado cruces negras para su fácil identificación, como se muestra en la figura 5.3.

5.3. Descripción de las tareas

En los experimentos de navegación se aprendieron cuatro tareas distintas (respecto al recorrido y distancia recorrida, y al ambiente en el que se realizaron) mostradas en la figura 5.2. Las tareas que le fueron enseñadas al robot cambiaban respecto a los niveles de complejidad, incrementando la distancia a recorrer, el número de accesos o puertas a atravesar, y el número de pasillos. En la primera tarea el robot solamente debía entrar a una habitación a partir de un salón. La segunda tarea requería que el robot saliera de una habitación, cruzara un pasillo con cuatro accesos y entrara a un cuarto. En la tercera tarea, el robot debía salir de una habitación, ir a través de un salón con tres accesos, entrar a un salón con dos accesos y entrar por uno de ellos. Finalmente, en la última tarea el robot debía ir a través de un pasillo con cinco accesos, ir a través de uno de ellos, avanzar por un pasillo y luego entrar a una habitación. Un estado estuvo representado por un vector de 180 lecturas correspondientes a los grados de cobertura del sensor láser. Las acciones básicas discretas usadas en el algoritmo en esta tarea fueron avanzar, retroceder, ir a la izquierda e ir a la derecha.

En la tarea de manipulación el robot debía tomar un objeto colocado sobre



(a) (b)

(c) (d)

Figura 5.2 Las tareas de navegación aprendidas en los distintos mapas. Se muestran las trayectorias de cada tarea. Las tareas difieren en la distancia que debe ser recorrida y en las características de los mapas (paredes, número de entradas y habitaciones, pasillos). En la primera tarea (a), el robot solo debía entrar a una habitación desde un salón. En la segunda tarea (b), el robot debía salir de una habitación antes de entrar a otra. La tercera tarea (c) consistía en hacer un recorrido más largo saliendo de una habitación pasando por un salón con múltiples entradas y entrar a un salón final. En la cuarta tarea (d) el robot debía salir de un cuarto, recorrer un pasillo con múltiples habitaciones y luego desviarse para entrar a una habitación.

Figura 5.3 El brazo robótico con la cámara web junto al *gripper* y el objeto rectangular en el ambiente de prueba.

la mesa. El objeto se encontraba marcado con cruces, el robot se colocaba en una posición inicial determinada y con ayuda de la cámara y los sensores en el *gripper* debía localizar al objeto y tomarlo. La tarea terminaba cuando el robot tomaba el objeto y lo levantaba. En los experimentos solo había un único objeto sobre la mesa, y el robot se entrenó para tomar a ese objeto específicamente. Ver la figura 5.3. Un estado estuvo representado para esta tarea, por un vector de los valores angulares de la posición de cada una de las 4 articulaciones que se controlaban del brazo, más el estado del *gripper* (abierto o cerrado, conteniendo algo o no) y un valor que indicaba si el objeto era visible o no desde esa posición. Las acciones básicas discretas usadas en el algoritmo para esta tarea de manipulación fueron adelante, atrás, abrir y cerrar (el *gripper*), cabe aclarar que solo las primeras dos acciones se combinaban para crear acciones continuas. Las acciones se comandaban indicando, mediante un número, la parte del brazo que debía moverse más la acción a realizar.

Se pueden reutilizar en tareas nuevas las acciones aprendidas para esta-

dos que sean similares a los vistos en las tareas aprendidas previamente. En navegación podrían reutilizarse acciones como pasar a través de una puerta o avanzar por un pasillo. Igualmente, en el caso del brazo las acciones que se podrían reutilizar dependerían de los estados vistos y las acciones aprendidas con anterioridad y la coincidencia de esos estados o situaciones en las tareas futuras que se quisieran aprender. En cada nueva tarea se pueden retomar las acciones que fueron aprendidas en tareas anteriores, como por ejemplo, acercarse a tomar el objeto a partir de una posición conocida desde donde es posible verlo con la cámara.

Cabe destacar que para ninguna tarea se le proporcionó información *a priori* al robot. El robot no tenía información de los mapas ni de los recorridos que debía realizar, hasta empezar el entrenamiento, solamente era capaz de identificar el estado final (meta). Y para el caso de la tarea que requería tomar un objeto, el robot no tenía indicios de donde estaba el objeto al inicio.

5.4. Pruebas y resultados

En esta sección se describen a detalle los experimentos realizados y los resultados obtenidos.

5.4.1. Descripción general de las pruebas

El reconocedor de voz usado está implementado en *Sphinx 3* [CMU Sphinx] y se usaron los modelos acústicos del proyecto DIMEx100 de [Pineda, 2010], los modelos del lenguaje y el vocabulario fueron hechos específicamente para los dominios de las tareas. Se utilizó una diadema con audífonos y micrófono para las intervenciones hechas por el usuario, y los experimentos fueron hechos en condiciones de ruido mínimo (casi en silencio total).

En los experimentos se usaron un refuerzo máximo positivo cuando se al-

canzaba la meta de $+100$, y uno negativo mínimo para situaciones críticas de -100, como parte de los refuerzos internos del algoritmo tradicional. Como situaciones críticas se especificaron la cercanía con la pared en el caso de navegación (debía mantenerse a no menos de 30 cm de la pared) y los choques o pérdida de vista del objeto en caso de la tarea de manipulación (si se detectaba una colisión o se perdía de vista al objeto, se daba la penalización máxima). Los refuerzos externos (provenientes de la retroalimentación verbal) se graduaron en tres intensidades para los refuerzos positivos y dos intensidades para los refuerzos negativos de acuerdo al significado del vocabulario usado para retroalimentar. La intensidad correspondía al valor numérico con el cual contribuían los refuerzos externos a la recompensa (positivos: $+100, +50, +10$ y negativos: $-50, -10$). Toda acción recibía un -1 por omisión a menos que se encontrara en un estado que ameritara una de las otras recompensas o castigos mencionados, como parte de los refuerzos tradicionales del algoritmo. Por ejemplo, la palabras clave 'objetivo' equivalía a $+100$, 'excelente' a $+50$, 'bien' a $+10$, 'terrible' a -50 y 'mal' a -10. Para el resto del vocabulario son empleadas recompensas similares a las de las palabras clave. En el apéndice se enlista el resto del vocabulario usado y sus correspondientes valores de refuerzo.

En los primeros experimentos hechos para navegación se probó a la función gaussiana como función de similitud con la cual se tuvieron problemas por el excesivo número estados creados (de 200 a 500) y por la confusión al identificarlos. Es por eso que para los experimentos de navegación finales que aquí se presentan se usó al coeficiente de Pearson, con el cual se logró mejorar la identificación del ambiente y reducir el número de estados (de 20 a 140 según la tarea). No obstante, en la tarea de manipulación, la función gaussiana fue la adecuada como función de similitud para el algoritmo propuesto (con 20 estados aproximadamente). Se usan dos diferentes funciones de similaridad debido a las diferencias en la información que representaban los valores de los estados de las dos tareas. Para la tarea de navegación era necesaria una

medida de similitud independiente de la escala (para patrones similares pero de diferentes proporciones) que identificara la correlación entre los valores que representaban a los estados. Por ejemplo, lecturas que definen patrones de puertas pueden ser de diferentes magnitudes (grandes y pequeñas). Para la tarea de manipulación era necesaria una medida que solo identificara la similitud entre los valores de los estados con una medida de distancia con un cierto rango de variación, dado que los estados principalmente están definidos con valores angulares de las partes del brazo.

Los parámetros que se usaron en el algoritmo de aprendizaje fueron: un coeficiente de aprendizaje (α) de 0.5 y 0.1 (navegación y manipulación respectivamente), un parámetro de descuento de la traza (λ) de 0.6 y 0.9 (navegación y manipulación respectivamente), y un factor de descuento (γ) de 0.9. Los valores de estos parámetros fueron establecidos de acuerdo a los datos experimentales y a las recomendaciones presentados en [Singh & Sutton, 1996] para trazas de elegibilidad sin reemplazo, en ese trabajo se recomiendan combinaciones de valores del coeficiente de aprendizaje y del parámetro de descuento de la traza que agilizan el aprendizaje con base en sus experimentos (por ejemplo, en una tarea de navegación se recomienda $\lambda = 0.6$ y $\alpha = 0.5$). Estos valores fueron puestos tal y como se recomiendan en la literatura en los primeros experimentos y posteriormente ajustados manualmente en esos experimentos observando su efecto en el aprendizaje hasta llegar a los valores finales que aquí se presentan. Para la función de similitud, usando el coeficiente de Pearson se consideró a dos estados similares si el valor del coeficiente era mayor a 0.5 y 0.6 dependiendo de la complejidad de la tarea, estos parámetros fueron ajustados de acuerdo a la experimentación, de manera similar a los parámetros del algoritmo de aprendizaje.

Cada uno de los experimentos se hizo tres veces y los promedios son mostrados en los resultados, las características que se probaron en cada experimento se muestran en la tabla 5.1.

Tabla 5.1 Experimentos realizados con el algoritmo propuesto basado en Sarsa(λ) (RD = Representación con acciones discretas, RC = Representación con acciones continuas, T = Trazas iniciales, F = Retroalimentación normal del usuario, PF = Retroalimentación perfecta, (50-50)RPF = Retroalimentación 50 % perfecta y 50 % ruidosa, NavS = Navegación en simulación, ManR = Manipulación en brazo robótico real).

Experimentos								
Experimento	Variante del algoritmo de aprendizaje						Tarea	
	RD	RC	T	F	PF	(50-50)RPF	NavS	ManR
Exp1		X					X	X
Exp2	X						X	
Exp3		X	X				X	
Exp4		X		X			X	X
Exp5		X	X	X			X	X
Exp6		X			X		X	
Exp7		X				X	X	

Con la finalidad de probar solamente la contribución y el funcionamiento de la representación diseñada para espacios continuos de estados y acciones se realizaron los experimentos 1 y 2. Para probar la contribución de cada una de los elementos del algoritmo propuesto, primero se probó la inclusión solamente de trazas iniciales de demostración en el experimento 3. Posteriormente, en el experimento 4 se probó la influencia de la retroalimentación solamente. En los experimentos 6 y 7 se probó la influencia en el aprendizaje de los errores en la retroalimentación (simulando errores en el reconocimiento de voz y humanos). Finalmente, en el experimento 5 se probó al algoritmo propuesto completo.

Todos los experimentos a excepción del segundo y los dos últimos fueron hechos para las cuatro tareas de navegación. Los dos últimos fueron hechos solo para la tarea de navegación 2 y el segundo para las tareas 2 y 4. Los experimentos 1, 4 y 5 fueron hechos para la tarea de manipulación del brazo robótico también. Los experimentos principales para probar los efectos de las trazas iniciales y la retroalimentación provista por voz fueron hechos para los 4 tareas, y debido a la consistencia en los resultados obtenidos en estos experimentos, los casos de experimentación con la retroalimentación en diferentes condiciones se hicieron solo en algunas tareas, suponiendo el mismo comportamiento mostrado en estos resultados en el resto de las tareas.

En los experimentos que incluían trazas iniciales se proporcionaron tres trazas. Las acciones comandadas para mostrar las trazas y para guiar en el entrenamiento fueron dadas indicando las acciones directamente en el caso de la navegación e indicando la sección del brazo que debía moverse y la acción para el caso del brazo robótico. En ambos casos usando el vocabulario definido en el reconocedor.

En los experimentos generales usando retroalimentación un mismo y único usuario participó en todos los experimentos, y la retroalimentación se usó tal y como provenía del reconocedor de voz, con errores de reconocimiento y posibles errores provocados sin intención por el usuario (aproximadamente 20 %

de la retroalimentación era errónea). Para la retroalimentación perfecta se simularon frases textuales (como si provinieran del reconocedor de voz) usando al teclado como medio para retroalimentar para no tener problemas derivados del reconocimiento de voz. Para insertar un porcentaje controlado de retroalimentación ruidosa se hizo de manera similar a la perfecta pero modificando intencionalmente parte de las frases para simular errores en el reconocimiento de voz o humanos. Se hizo este experimento con un 50 % de retroalimentación ruidosa porque se buscaba probar la tolerancia del algoritmo a una cantidad importante (la mitad de la cantidad total) de errores.

El entrenamiento del robot en los experimentos finalizaba cuando en los episodios (de entrenamiento) se obtenían trazas de solución de la tarea que alcanzaban ciertas características. Las trazas debían ser similares, en el recorrido hecho y la distancia recorrida, a la traza obtenida en los primeros experimentos hechos con el algoritmo tradicional de aprendizaje.

A continuación se presentan las comparaciones entre los diferentes experimentos respecto al tiempo para aprender las tareas, al número de episodios, a las políticas aprendidas y al número de intervenciones del usuario para la retroalimentación.

5.4.2. Tareas de navegación

Las pruebas para éstas tareas se hicieron en simulación utilizando el software *Player/Stage* [Player_Stage] y con el robot *Pioneer* [Pioneer_MobileRobots].

En la figura 5.4 se graficaron el número de episodios y el tiempo tomado por episodio de entrenamiento para cada tarea de los experimentos 1 (estados-acciones continuos), 3 (estados-acciones continuos y trazas), 4 (estados-acciones continuos y retroalimentación) y 5 (estados-acciones continuos, trazas y retroalimentación) en navegación. En la tabla 5.2 se muestra el número total de episodios y el tiempo total tomado para aprender cada una de las tareas de

navegación para los experimentos 1 (estados-acciones continuos), 3 (estados-acciones continuos y trazas), 4 (estados-acciones continuos y retroalimentación) y 5 (estados-acciones continuos, trazas y retroalimentación). Es posible apreciar como los menores tiempos de entrenamiento fueron obtenidos cuando se usa el algoritmo completo con retroalimentación y trazas iniciales. Y aunque el número de episodios no cambió significativamente entre los diferentes experimentos, sí lo hizo el tiempo tomado en cada uno de ellos, disminuyendo desde la inclusión de las trazas hasta usar retroalimentación y trazas.

En la figura 5.5 se muestra gráficamente el número intervenciones hechas por el usuario por episodio de entrenamiento para cada tarea de los experimentos 4 (estados-acciones continuos y retroalimentación) y 5 (estados-acciones continuos, trazas y retroalimentación) en navegación . En la tabla 5.3 se muestran el número total de intervenciones hechas para los mismos experimentos por tarea. En estos resultados puede observarse la disminución de la cantidad de retroalimentación a menos de la mitad en 3 de 4 tareas cuando se incluyen trazas iniciales de ejemplo. Además se puede observar que es necesaria menos retroalimentación conforme el entrenamiento va llegando a los episodios finales del aprendizaje. Del total de la retroalimentación proporcionada, la mayoría fue de calificativos solos o acompañados de comandos (40 % calificativos solos, 40 % comandos más calificativos, y 20 % comandos solos).

Representación discreta contra continua

En la figura 5.6 se muestra la comparación de trazas obtenidas con los experimentos 1 (estados-acciones continuos) y 2 (estados continuos, acciones discretas) para las tareas 2 y 4 de navegación. Y en la figura 5.7 se muestran el número de episodios y el tiempo por episodio para ambas tareas. Aunque el tiempo utilizado para aprender las tareas y el número de episodios son similares con ambos tipos de acciones (por usar la misma representación para los estados), la distancia es menor en las trazas obtenidas con las acciones

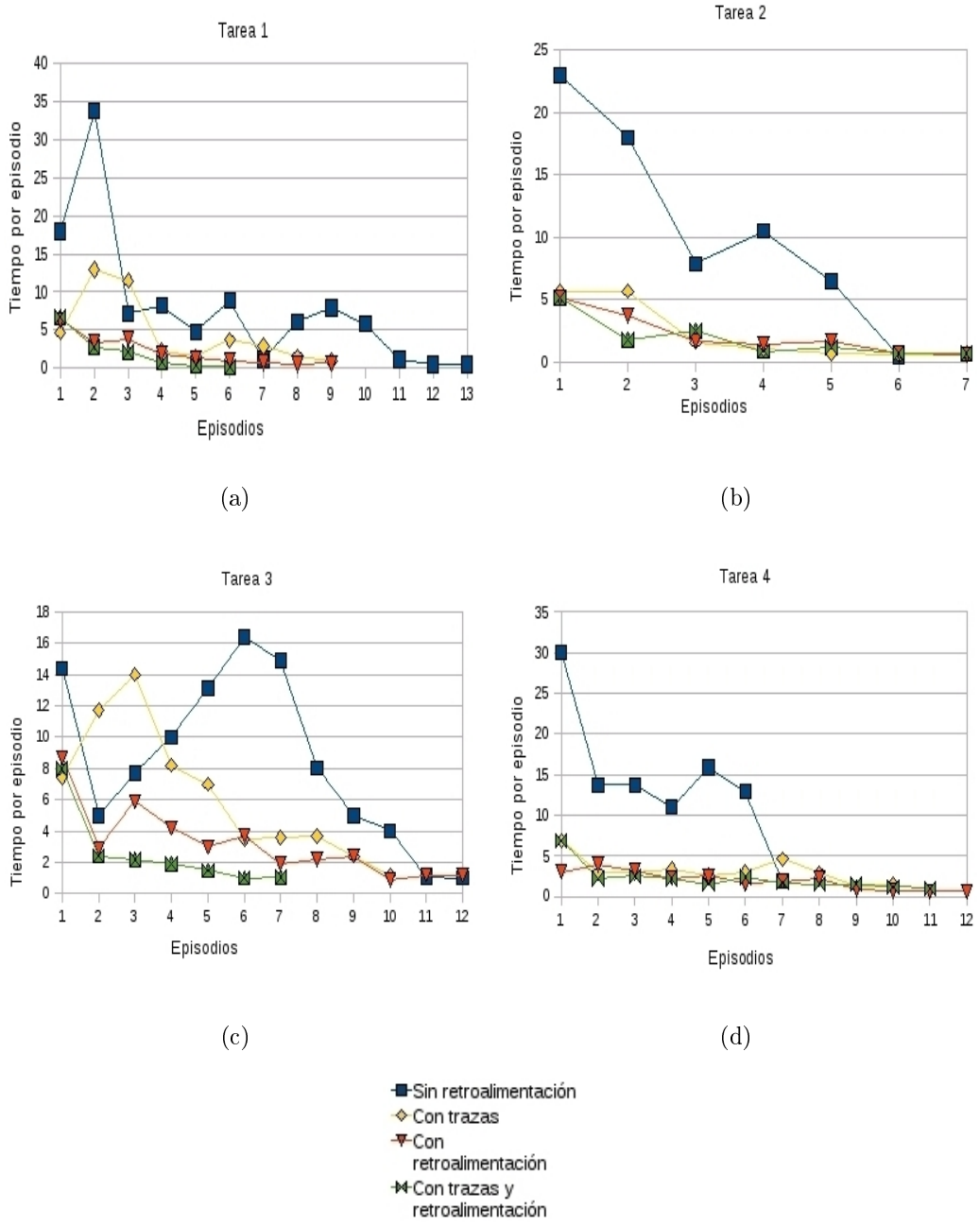


Figura 5.4 Número de episodios y tiempo (minutos) por episodio de aprendizaje de las variantes del algoritmo para navegación. Se muestran las 4 gráficas correspondientes a las 4 tareas de navegación (tarea 1 (a), tarea 2 (b), tarea 3 (c) y tarea 4 (d)), con la comparación de los resultados obtenidos en los experimentos usando retroalimentación solamente (triángulo), trazas iniciales solamente (rombo), con trazas y retroalimentación (moño), y sin retroalimentación ni trazas (cuadrado).

Tabla 5.2 Número total de episodios y tiempos totales (minutos) de los diferentes experimentos por tarea de navegación (RL = aprendizaje por refuerzo con representación para espacios continuos, RL + T= RL con trazas iniciales, RL + F = RL con retroalimentación del usuario, y RL + T + F = RL con trazas iniciales y con retroalimentación provistas por el usuario).

Número de episodios

Tarea	Variante del algoritmo de aprendizaje			
	RL	RL + T	RL + F	RL + T + F
T1	13	9	9	6
T2	6	7	7	7
T3	12	10	12	7
T4	7	10	12	11
Prom	9.5	9	10	7.75

Tiempo (min)

Tarea	Variante del algoritmo de aprendizaje			
	RL	RL + T	RL + F	RL + T + F
T1	103.93	41.80	19.59	12.85
T2	66.40	16.07	15.10	13.00
T3	100.65	62.66	38.20	18.09
T4	99.10	31.90	23.43	24.61
Prom	92.54	38.11	24.08	17.13

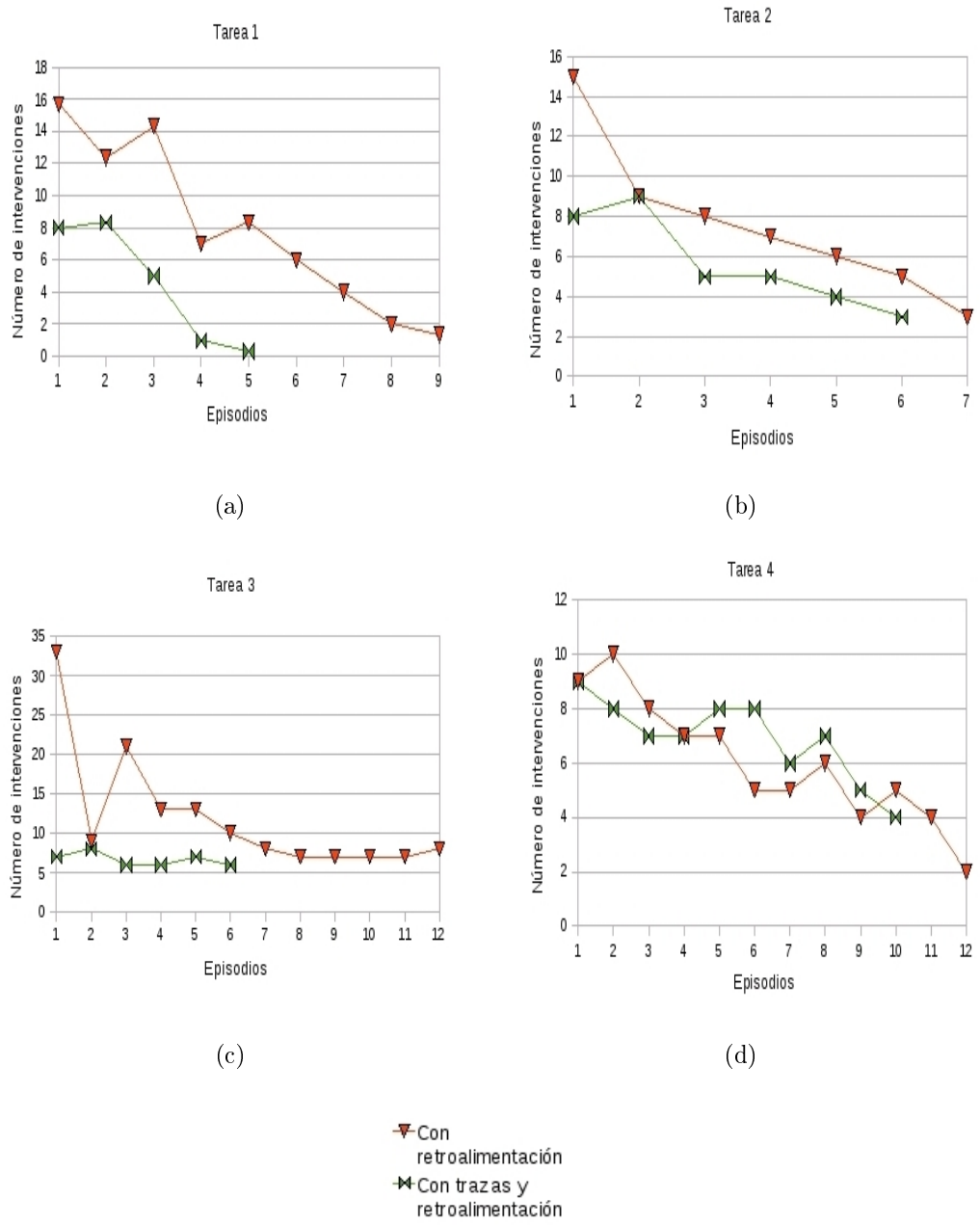


Figura 5.5 Número de intervenciones hechas por el usuario por episodio, en tareas de navegación. Se muestran las 4 gráficas correspondientes a las 4 tareas de navegación (tarea 1 (a), tarea 2 (b), tarea 3 (c) y tarea 4 (d)) comparando los dos experimentos que incluyen retroalimentación, con retroalimentación solamente (triángulo), y con trazas y retroalimentación (moño)

Tabla 5.3 Número de intervenciones del usuario en los diferentes experimentos por tarea de navegación (RL + F = RL con retroalimentación del usuario y RL + T + F = RL + F con trazas iniciales provistas por el usuario), a la izquierda.

Número de Intervenciones		
Tarea	Variante del algoritmo de aprendizaje	
	RL + F	RL + T + F
T1	71	23
T2	53	34
T3	143	40
T4	69	72
Prom	84	42

continuas (en casi 1m de diferencia, de 9 a 8m y de 12 a 11m). En la figura 5.8 se muestran ejemplos de trazas de las cuatro tareas obtenidas usando acciones continuas. Las trazas mostradas con las políticas obtenidas con acciones continuas recorren menor distancia y hacen movimientos más suaves (se observaron cambios de dirección menos bruscos con ángulos menores o mayores a 90 ° con los que se disminuye la distancia recorrida, a diferencia de como se hacían con las acciones discretas), y aunque pueden llegar a tener ciertas desviaciones producto de la identificación de estados, el robot es capaz de orientarse de nuevo para llegar a su meta.

Retroalimentación perfecta y ruidosa

La comparación de los experimentos 4 (estados-acciones continuos y retroalimentación) y 6 (estados-acciones continuos y retroalimentación perfecta), el número de episodios y el tiempo por episodio para la tarea 2 de navegación son mostrados en la figura 5.9. Además se muestra el número de intervencio-

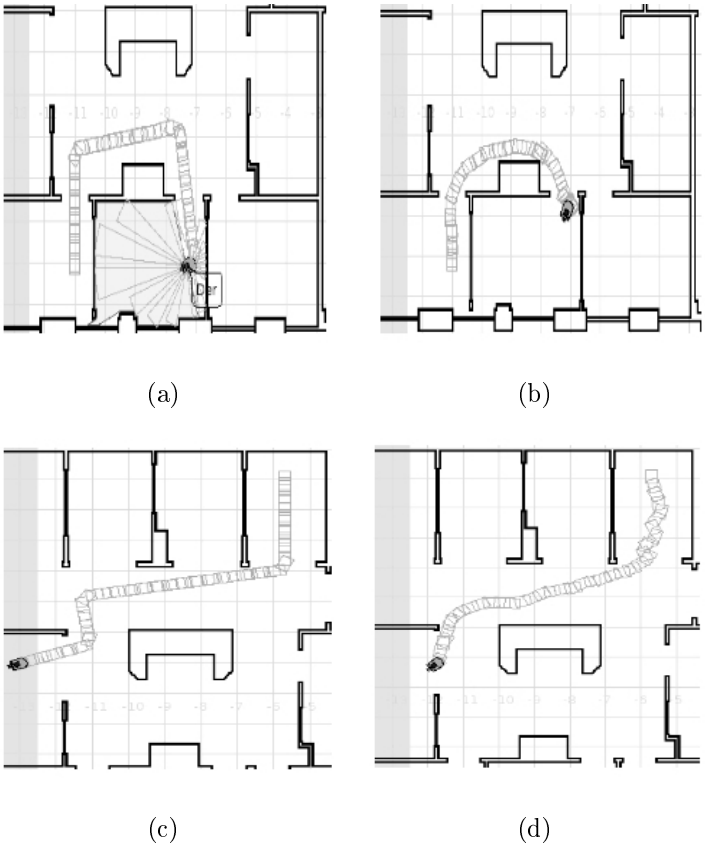


Figura 5.6 Comparación entre las trazas obtenidas con el algoritmo usando acciones discretas y acciones continuas en dos tareas de navegación. Se muestran la tarea 2 en (a) y (b). y la tarea 4 en (c) y (d), del lado izquierdo, en (a) y (c), se muestran las trazas obtenidas con acciones discretas y del lado derecho, (b) y (d), las trazas obtenidas usando las acciones continuas de la representación propuesta.

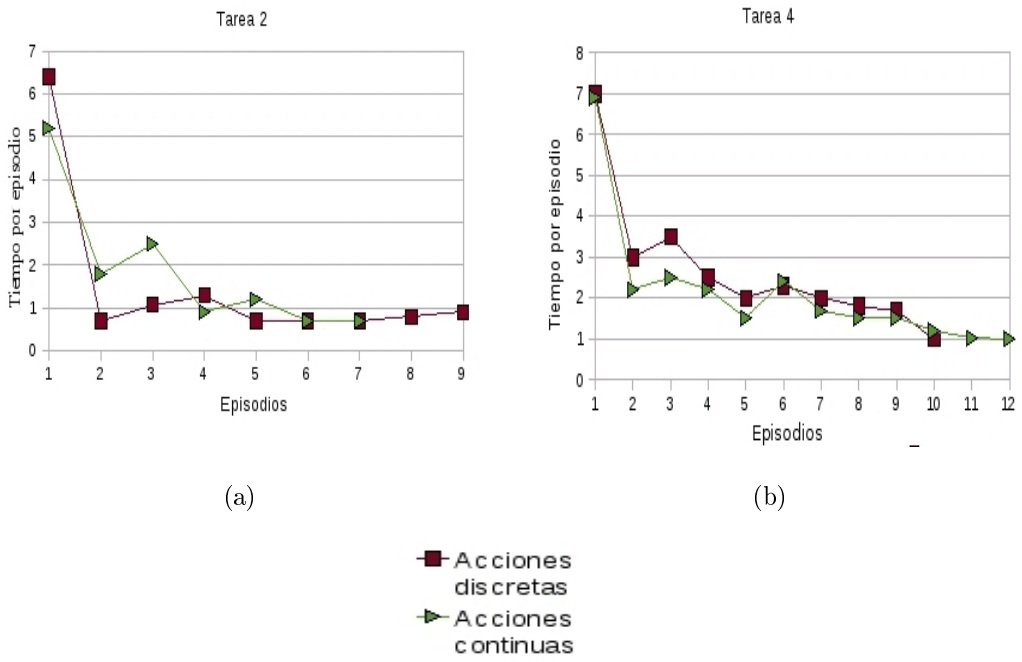


Figura 5.7 Número de episodios y tiempo (minutos) tomados por episodio con el algoritmo usando acciones discretas y acciones continuas para dos tareas de navegación. Se muestran las gráficas de la tarea 2 en (a) y de la tarea 4 en (b), comparando los experimentos usando acciones discretas (cuadrado) y acciones continuas (triángulo).

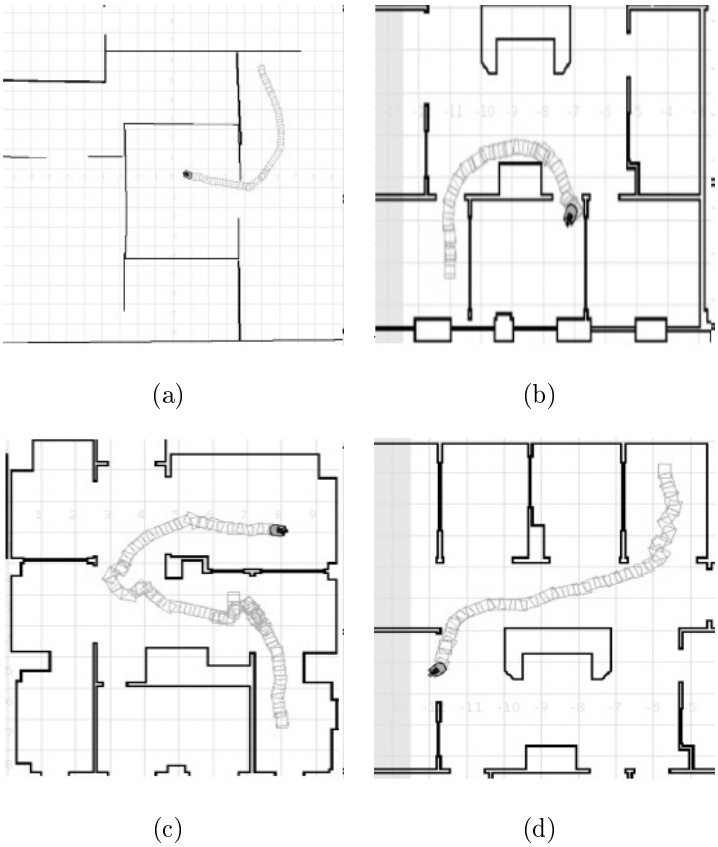


Figura 5.8 Ejemplos de trazas obtenidas con acciones continuas en tareas de navegación. Se muestra un ejemplo de una traza obtenida con el algoritmo propuesto para realizar cada una de las 4 tareas de navegación (tarea 1 (a), tarea 2 (b), tarea 3 (c) y tarea 4 (d)).

nes hechas por el usuario en cada episodio para ambos experimentos. En la tabla 5.4 se muestran el número total de intervenciones y los tiempos totales de entrenamiento para los mismos experimentos. Cuando la retroalimentación es perfecta, el tiempo de entrenamiento puede llegar a ser considerablemente menor al usado con la retroalimentación normal (con errores del reconocedor o del usuario), en este experimento reducido hasta la mitad. Aunque el número de intervenciones en ambos experimentos es casi el mismo. En el caso de la retroalimentación perfecta, en los últimos episodios de los experimentos la retroalimentación es abundante pero en su mayoría (hasta el 80 %) de calificativos positivos. Ocurre una reacción en el usuario que califica positivamente al robot porque observa un comportamiento correcto. Además, el aumento en la retroalimentación positiva se vió favorecido porque el usuario notaba que no había problemas en la interpretación que pudieran perjudicar el aprendizaje, como en el caso de los otros experimentos donde se usaba un reconocedor normal de voz en el que ocurrían errores. En los experimentos con retroalimentación normal un número similar de intervenciones (comparado con los experimentos con retroalimentación perfecta) es realizado pero de una forma más balanceada, tanto positivas como negativas en cuanto a los calificativos. Del total de la retroalimentación proporcionada de forma perfecta, la mayoría fue de calificativos solos y acompañados de comandos (50 % calificativos solos, 45 % comandos más calificativos, y 5 % comandos solos).

La figura 5.10 muestra la comparación entre los experimentos 1 (estados-acciones continuos) y 7 (estados-acciones continuos con retroalimentación 50 % correcta-50 % incorrecta), el número de episodios y el tiempo por episodio para la tarea 2 de navegación son presentados. También se muestran el número de intervenciones necesarias en el caso del experimento 7. En la tabla 5.5 se muestra el tiempo total y el número de intervenciones para los mismos experimentos. En estos experimentos se muestra que la retroalimentación ruidosa (50 %) afecta los tiempos de entrenamiento, sin embargo, estos tiempos aun

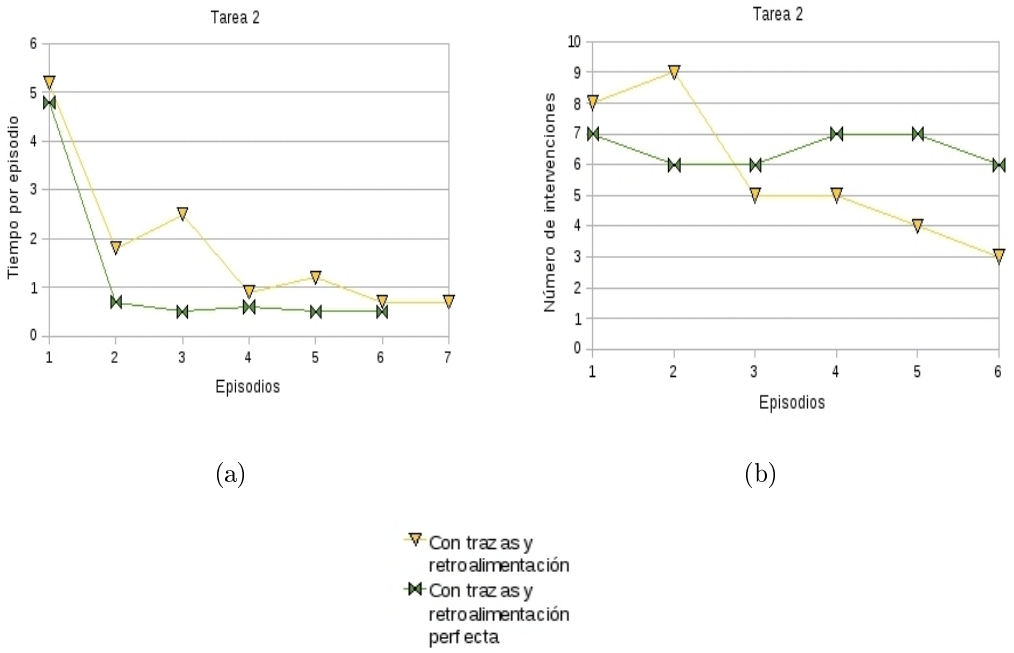


Figura 5.9 Número de episodios y tiempo (minutos) por episodio, y número de intervenciones del usuario para la tarea 2 de navegación usando al algoritmo con y sin retroalimentación perfecta. Se muestran las gráficas para la tarea 2 de los experimentos usando retroalimentación normal (triángulo) y retroalimentación perfecta (moño). En (a) se muestra los episodios y los tiempos, y en (b) el número de intervenciones.

Tabla 5.4 Tiempos totales (minutos) y número de intervenciones para la tarea 2 de navegación primero con retroalimentación normal (con posibles errores) y segundo con retroalimentación perfecta (PF).

Tiempo e intervenciones en la tarea 2 con y sin retroalimentación perfecta

Tarea	Tiempo (min)	Número de intervenciones
T2	13.00	34
T2 PF	7.60	39

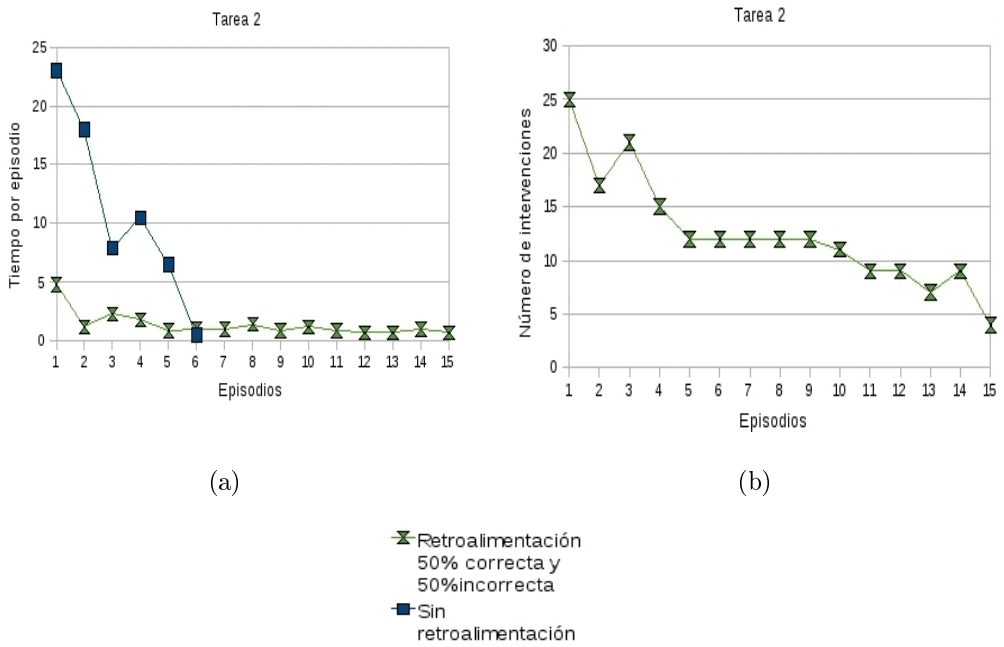


Figura 5.10 Número de episodios, tiempo (minutos) y número de intervenciones para la tarea 2 de navegación del experimento sin retroalimentación ni trazas contra el experimento con 50 % de retroalimentación ruidosa (con errores de reconocimiento de voz y humanos, con cuadrados en la gráfica) y 50 % de retroalimentación perfecta (moño). Se muestra en (a) la gráfica con los episosios y el tiempo por episodio, y en la gráfica en (b) el número de intervenciones del experimento que sí incluye retroalimentación.

son menores a los obtenidos con un algoritmo de aprendizaje por refuerzo tradicional. Además el número de intervenciones hechas por el usuario aumenta en comparación con la cantidad usada en experimentos anteriores con retroalimentación menos ruidosa. Del total de la retroalimentación proporcionada, la mayoría fue de calificativos solos o acompañados de comandos (30 % calificativos solos, 50 % comandos más calificativos, y 20 % comandos solos).

Tabla 5.5 Tiempos totales (minutos) y número de intervenciones para la tarea 2 de navegación, el algoritmo sin retroalimentación ni trazas contra el algoritmo con 50 % de retroalimentación ruidosa (con errores de reconocimiento de voz y humanos) y 50 % de retroalimentación perfecta.

Tiempo e intervenciones en la tarea 2
sin retroalimentación y con 50 % de retroalimentación ruidosa

Tarea	Tiempo (min)	Número de intervenciones
T2	66.40	-
T2 50-50	20.51	187

5.4.3. Tarea de manipulación

Las pruebas fueron programadas usando las librerías de Katana Native Interface para el brazo robótico. Para el reconocimiento de las marcas (cruces) del objeto en las imágenes obtenidas con la cámara del brazo robótico se utilizó una adaptación del algoritmo *FastMatchTemplate* [Georgiou, 2005] basado en OpenCV.

En la figura 5.12 se muestran los experimentos 1 (estados-acciones continuos), 4 (estados-acciones continuos y retroalimentación) y 5 (estados-acciones continuos, trazas y retroalimentación) realizados con el brazo robótico, el número de episodios y el tiempo por episodio además del número de intervenciones son presentados. En la figura 5.11 se muestra una secuencia de ejemplo de la tarea de manipulación. En la tabla 5.6 se presenta el número total de episodios y el tiempo total para los experimentos mencionados con el brazo robótico. Como en los experimentos anteriores, en estos experimentos también los mejores tiempos y el menor número de episodios fueron obtenidos cuando se incluyeron las trazas y la retroalimentación del usuario. Del total de la retroalimentación proporcionada, la mayoría fue de calificativos solos o acompañados

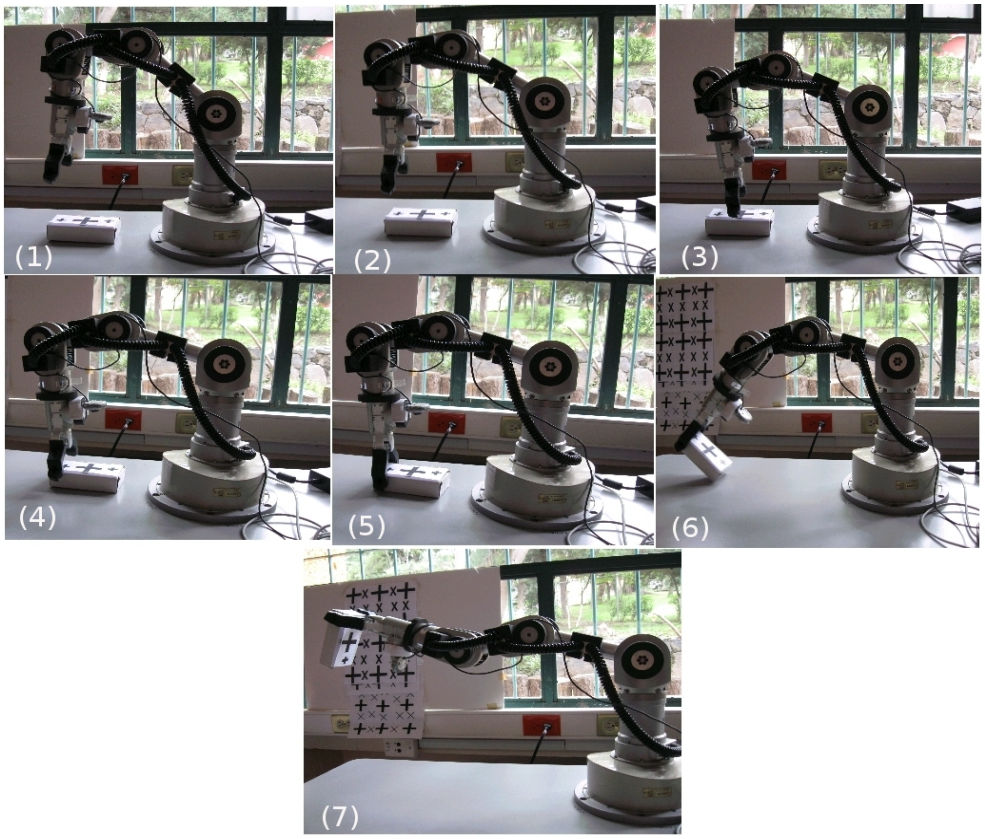


Figura 5.11 Secuencia de la tarea de manipulación. En las imágenes se muestra una secuencia que ilustra la tarea. El brazo robótico localiza el objeto con la cámara y se acerca para tomarlo de la mesa para finalmente levantarlo.

de comandos (30 % calificativos solos, 30 % comandos más calificativos, y 40 % comandos solos). En ambos casos la reducción fue de casi la mitad comparado con el algoritmo tradicional.

5.5. Discusión de los resultados

5.5.1. Tareas de navegación

Los experimentos muestran que el menor tiempo de aprendizaje es alcanzado cuando se utiliza al algoritmo propuesto completo, es decir, usando trazas

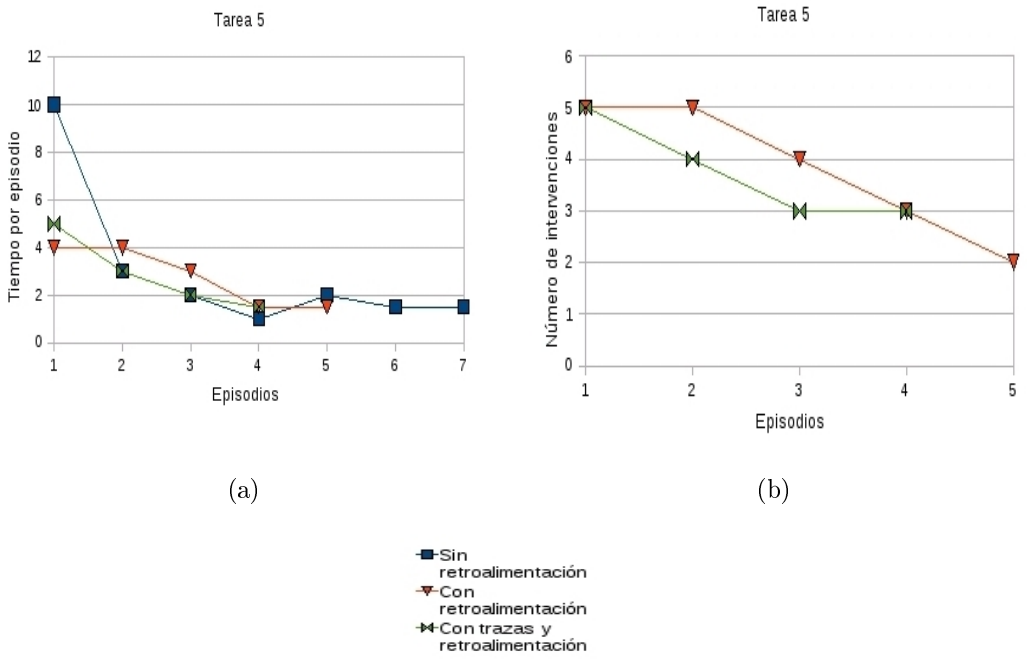


Figura 5.12 Número de episodios, tiempo (minutos) por episodio y número de intervenciones usando las diferentes variaciones del algoritmo para la tarea de manipulación. En la gráfica en (a) se muestran los episodios y el tiempo para los experimentos sin retroaliemntación (cuadrado), con retroalimentación (triángulo), con trazas y retroalimentación (moño). En la gráfica en (b) se muestra el número de intervenciones para los experimentos que incluyen retroalimentación.

Tabla 5.6 Número de episodios, tiempo total (minutos) y número de intervenciones para la tarea de manipulación con las variaciones del algoritmo. (RL = aprendizaje por refuerzo con representación para espacios continuos, RL + F = RL con retroalimentación del usuario, y RL + T + F = RL + F con trazas iniciales y con retroalimentación provistas por el usuario)

Número de Episodios

Tarea	Variante del algoritmo de aprendizaje		
	RL	RL + F	RL + T + F
T5	7	5	4

Tiempo (min)

Tarea	Variante del algoritmo de aprendizaje		
	RL	RL + F	RL + T + F
T5	21.00	14.00	11.50

Número de Intervenciones

Tarea	Variante del algoritmo de aprendizaje	
	RL + F	RL + T + F
T5	19	15

iniciales y retroalimentación con voz por parte de un usuario. Y en la mayoría de los casos también usando al algoritmo completo se tiene el menor número de episodios de entrenamiento. También, aún considerando errores en el reconocedor de voz en la retroalimentación, el algoritmo es capaz de converger con una mayor rapidez en comparación de cómo lo haría sin incluir retroalimentación con un algoritmo de aprendizaje por refuerzo tradicional. En promedio las 4 tareas de navegación pueden ser aprendidas en cerca de 10 episodios usando al algoritmo de aprendizaje tradicional o agregando trazas iniciales o retroalimentación de manera exclusiva; sin embargo, el menor número de episodios es obtenido cuando se usa al algoritmo propuesto completo usando las trazas iniciales y retroalimentación, con una diferencia de 2.5 episodios aproximadamente (de 9 y 10 episodios a 7.75). Aunque esa diferencia no parece sustancial, cuando se analizan los tiempos totales utilizados en entrenamiento con cada variación del algoritmo contra el tiempo total tomado por el algoritmo tradicional, se tiene que a partir de la inclusión de trazas se consigue una reducción en tiempo hasta de la mitad en promedio, y al añadir retroalimentación el tiempo se reduce hasta casi 4 veces.

Finalmente usando el algoritmo completo con trazas y retroalimentación, la reducción en tiempo es mucho mayor llegando a ser un tiempo total 5 veces menor comparado contra el tiempo que se tomaría al utilizar un algoritmo tradicional de aprendizaje por refuerzo. En general, cabe resaltar que es posible enseñar aproximadamente en 20 minutos en promedio una tarea de navegación similar a las mostradas en los experimentos (en mapas de aproximadamente 10 $\times$ 10m con condiciones estructurales como las de los mapas ya mostrados, con trayectorias de 10-16m). Y las políticas obtenidas en los diferentes experimentos son similares en distancia y ruta recorrida, a excepción de las obtenidas con acciones discretas.

Comparando los beneficios de la inclusión por separado al algoritmo de las trazas y de la retroalimentación, se tiene que la mayor contribución en

reducción de tiempo la hace la retroalimentación. Las trazas proporcionan información útil que aunque también reduce significativamente el tiempo de entrenamiento no lo favorece tanto como lo hace la retroalimentación.

Con respecto a la retroalimentación, el número de intervenciones necesarias disminuye al agregar las trazas iniciales en comparación con usar al algoritmo solamente con retroalimentación, lo que señala la importancia de la información que proporcionan las trazas iniciales de ejemplo. Además, en los episodios finales el número de intervenciones disminuye, como era de esperarse cuando el robot ya está aprendiendo la tarea. Cuando se añaden las trazas iniciales al algoritmo además de la retroalimentación, en 3 de 4 tareas de navegación se tiene una reducción en promedio a la mitad del número total de intervenciones hechas cuando sólo se usa retroalimentación.

Representación discreta contra continua

Al comparar las trazas obtenidas con las acciones continuas contra las trazas obtenidas con acciones discretas, es posible notar el afinamiento en los movimientos del robot (el ajuste de dirección se hace con ángulos menores/mayores a los hechos con acciones discretas) y la reducción en la distancia recorrida al utilizar el diseño con acciones continuas. Para la tarea 2, la diferencia de distancia recorrida usando acciones continuas contra discretas fue de 1m., para la tarea 4, la diferencia fue cerca de 1m también. Al utilizar la misma representación para el tratamiento de estados continuos, el tiempo de entrenamiento es similar; sin embargo las trazas obtenidas en la política final son significativamente distintas (se pueden observar trayectorias de movimientos más suaves que reducen la distancia a recorrida y que permiten moverse con más facilidad en ambientes complicados).

Retroalimentación perfecta y ruidosa

Con los experimentos con retroalimentación perfecta y, con la proporción 50 % retroalimentación perfecta y 50 % ruidosa, se observó el efecto de los errores en el reconocimiento de voz y de los posibles errores humanos al retroalimentar durante el proceso de aprendizaje usando al algoritmo propuesto. Para el primer caso se observó cómo la retroalimentación perfecta reduce aún más el tiempo de aprendizaje, a casi la mitad comparado con el tiempo usado con retroalimentación normal (que puede tener errores). Cuando se compara el número de intervenciones de la retroalimentación perfecta contra el usado con retroalimentación normal (posiblemente ruidosa) se tiene que son casi equivalentes. Sin embargo, esto ocurrió porque la retroalimentación perfecta en los episodios finales es abundante pero en su mayoría con calificativos positivos, fue una reacción del usuario ante el comportamiento correcto del robot y ante la ausencia de problemas en la interpretación en el reconocedor de voz. En los otros experimentos que incluyen retroalimentación estas condiciones no ocurren, porque hay problemas de reconocimiento de voz y el robot comete más errores afectándo el aprendizaje, situaciones que son percibidas por el usuario y que determinan su reacción. En el caso de la retroalimentación normal es parecida en cantidad en los episodios finales pero es tanto de calificativos positivos como negativos.

Con el segundo caso, con la proporcion 50-50 en la calidad de retroalimentación, se observa cómo aún con errores, el algoritmo es capaz de converger aunque requiere más tiempo (20 min) y episodios (15) comparado contra el algoritmo con retroalimentación normal (15.1 min y 7 episodios). Es importante notar que aún en estas condiciones es significativamente más rápido aprender con retroalimentación que sin usar retroalimentación (66.4 min y 6 episodios). De acuerdo a los resultados obtenidos en este experimento, suponiendo un comportamiento similar al que se ha obtenido, se estima que aproximadamen-

te con cerca de 80 % o más de retroalimentación ruidosa los beneficios de usar retroalimentación desaparecen y el aprendizaje llega a ser tan o más costoso que si se realizara con un algoritmo tradicional. Algo que sí es notable en este experimento es el aumento en el número de intervenciones por las correcciones que deben darse ante los errores.

5.5.2. Tarea de manipulación

En el experimento realizado con el brazo robótico se puede notar un patrón similar de comportamiento del algoritmo como en las tareas de navegación. La reducción en tiempo de entrenamiento es casi de la mitad y el número de episodios también disminuye. Para este experimento el tiempo necesario para que el brazo aprenda a tomar un objeto en una posición visible es de 12 min aproximadamente. En este caso la retroalimentación se mantiene casi igual al incluir las trazas iniciales, posiblemente debido a la interacción directa que representa el uso del robot real y a la naturaleza de la tarea. Además, la utilización del brazo robótico demuestra que es posible adaptar al algoritmo propuesto a otros dominios y tareas obteniendo las mismas ventajas ya mostradas.

5.6. Conclusiones finales

La utilización de trazas representa una inicialización de la representación de la función de valor para el algoritmo de refuerzo, y la retroalimentación representa un *reward shaping* dinámico, cualquiera de los dos usados de forma exclusiva en el algoritmo de aprendizaje disminuye el tiempo y el número de episodios necesarios en el entrenamiento. Y cuando se usa el algoritmo propuesto, incluyendo trazas y retroalimentación, se tiene una reducción aún mayor, que representa una importante agilización en el proceso de aprendizaje de tareas para un robot (5 veces más rápido, del orden de minutos). Además,

a pesar de los errores humanos y los producidos por el reconocedor de voz, es posible aprender más rápidamente con el algoritmo propuesto que con un algoritmo tradicional de aprendizaje por refuerzo.

Capítulo 6

Conclusiones, contribuciones y trabajo futuro

6.1. Conclusiones

Los robots de servicio requieren desarrollar sus capacidades cada día más por su creciente interacción con humanos inexpertos en ambientes dinámicos. Varias técnicas han sido sugeridas para proporcionar estas capacidades a los robots, la mayoría de ellas basadas en aprendizaje computacional. Específicamente el aprendizaje por refuerzo y el aprendizaje por demostración han sido utilizados con éxito para el aprendizaje en robots. Con el aprendizaje por demostración puede ser mostrado paso a paso cómo hacer una tarea, y con el aprendizaje por refuerzo se aprovecha la propia experiencia del robot de interactuar con el ambiente además del uso de refuerzos que guían el comportamiento para aprender más fácilmente. Sin embargo, aunque varios trabajos se han hecho con estos métodos, se han identificado desventajas en ellos, como es el uso de dispositivos difíciles de manejar por personas inexpertas para el caso de la demostración, y en aprendizaje por refuerzo, el tratamiento de espacios continuos puede requerir largos tiempos de entrenamiento para el robot,

tiempos que no son factibles en los ambientes de servicio.

De ahí que en esta tesis se haya propuesto un algoritmo de aprendizaje por refuerzo con una representación que puede manejar espacios continuos incluyendo además los beneficios de la demostración mediante ejemplos y retroalimentación, a través de una interfaz fácilmente utilizable por personas inexpertas en robótica. Con este algoritmo propuesto es posible instruir verbalmente robots en menor tiempo (del orden de minutos) en comparación con otros algoritmos clásicos de aprendizaje por refuerzo. Se puede resaltar cómo el uso de retroalimentación ayuda notablemente al proceso de aprendizaje, y que mediante elementos de interacción natural como lo es la voz, resulta más accesible para las personas instruir tareas y para los robots aprenderlas en ambientes como los de servicio, en comparación de usar *hardware* especializado (guantes, cámaras, *joystick*, entre otros) como medio para instruir. Se ha mostrado cómo un diseño sencillo puede manejar espacios continuos de estados basándose en el coeficiente de Pearson y la función gaussiana como funciones de similitud. Además, cómo un conjunto pequeño y básico de acciones discretas relacionadas con el dominio de una tarea puede dar lugar a acciones continuas aprovechando la información obtenida por las funciones de valor del aprendizaje por refuerzo, creando acciones resultantes más adecuadas para realizar la tarea que las acciones discretas estáticas.

Finalmente, los experimentos realizados con el algoritmo propuesto muestran que la retroalimentación y las trazas iniciales ayudan a reducir considerablemente el tiempo de aprendizaje. Aunque ambos componentes, trazas y retroalimentación, usados de manera independiente, reducen el tiempo de aprendizaje, la mayor reducción la hace la retroalimentación. El uso de las trazas reduce el número de intervenciones del usuario aparte de ayudar a reducir el tiempo de entrenamiento. Los experimentos también muestran que aun con errores en la retroalimentación es posible obtener tiempos de entrenamiento menores comparados con los obtenidos por un algoritmo tradicional

de aprendizaje por refuerzo. Así, se muestra que el menor tiempo de entrenamiento es obtenido usando al algoritmo propuesto comparado con un algoritmo tradicional de aprendizaje.

6.2. Contribuciones

Las principales contribuciones de este trabajo de investigación son, en primer lugar, la inclusión de retroalimentación dinámica externa adicional a los refuerzos tradicionales en un algoritmo de aprendizaje por refuerzo modificado Sarsa(λ), que maneja la retroalimentación espontánea, correcta y ocasionalmente incorrecta, y de diferentes tipos (comandos y calificativos) e intensidades (graduaciones en la evaluación). En segundo lugar, el diseño de una representación que trabaja en línea para espacios continuos de estados y acciones basada en los valores Q. En tercer lugar, la inclusión de la voz, con palabras y frases cortas en una forma natural, como medio para retroalimentar y proporcionar ejemplos iniciales que incluye la definición e identificación de vocabulario relacionado con los dominios de las tareas.

Finalmente, derivados del diseño de los componentes novedosos que conforman el algoritmo se obtuvo un algoritmo que trabaja con tiempos competitivos para la instrucción de tareas, mismo que fue probado en dos robots con dominios diferentes de trabajo mostrando una reducción significativa de tiempo de entrenamiento.

6.3. Trabajo Futuro

Entre las ideas de trabajo futuro se encuentra considerar la inclusión de un mecanismo de "olvido" tanto de acciones como de estados en el algoritmo de aprendizaje por refuerzo, para quitar información innecesaria y agilizar el aprendizaje. Se considera también el aumento en el vocabulario para incre-

mentar la naturalidad de interacción y ampliar el uso del método hacia más y diferentes dominios de trabajo. También, experimentar con otro tipo de representación para el espacio de estados y acciones. Además, la prueba formal de la convergencia del algoritmo y la descripción de las condiciones en las que ésta se da, deben ser estudiadas. Finalmente, es importante realizar pruebas con usuarios finales inexpertos en el área.

Bibliografía

P.Abbeel & A.Y.Ng. Apprenticeship learning via inverse reinforcement learning. En *21th International Conference on Machine Learning*, 2004.

J.Asmuth, M.Littman, & R.Zinkov. Potential-based shaping in model-based reinforcement learning. En *23rd AAAI Conference on Artificial Intelligence*, 2008.

G. A.Bekey. *Autonomous Robots: From Biological Inspiration to Implementation and Control.* MIT Press, 2005.

B.Boulebteche, M.Fezari, & M.Bedda. A voice guiding system for autonomous robots. *Asian Journal for Information Technology, Medwell Online*, 4(12): 1201–1207, 1997.

A.Chohra, K.Madani, & D.Kanzari. Reinforcement Q-learning and neural networks to acquire negotiation behaviors. En *Studies in Computational Intelligence (SCI). The Special Session New Challenges in Applied Intelligence Technologies (NCAIT)*, Springer Series. Springer-Verlag, 2008.

R.Coulom. Feedforward neural networks in reinforcement learning applied to high-dimensional motor control. En *13th International Conference on Algorithmic Learning Theory. Lecture Notes In Computer Science*, vol. 2533. Springer-Verlag, 2002.

W.Dongli, G.Yang, & Y.Pei. Applying neural network to reinforcement learning in continuous space. En *International Symposium on Neural Networks, ISNN*, 2005.

M.Dorigo & M.Colombetti. Robots shaping: developing autonomous agents through learning. *Artificial Intelligence*, 71(2), 1994.

K.Doya. Reinforcement learning in continuos time and space. *Neural Computation*, 12(1):219–245, 2000.

T.Georgiou. Algoritmo fast match template desarrollado con las librerías de open computer vision, 2005.

A.GroBmann & R.Poli. Continual robot learning with constructive neural networks. En *Sixth European Workshop (EWLR), Lecture Notes in Artificial Intelligence*, 1997.

M.Grzes & D.Kudenko. Learning shaping rewards in model-based reinforcement learning. En *Workshop on Adaptive Learning Agents ALA-AAMAS*, 2009.

F.Guenter, M.Hersch, S.Calinon, & A.Billard. Reinforcement learning for imitating constrained reaching movements. *Advanced Robotics*, 21(13):1521–1544, 2007.

V.Gullapalli. *Reinforcement Learning and its Application to Control.* Tesis de Doctorado, Departamento de Computación y Ciencias de la Información, University of Masachussetts, 1992.

J.C.Hernández. Aprendizaje por refuerzo relacional con acciones continuas. Tesis de Maestria, Coordinación de Ciencias Computacionales, Instituto Nacional de Astrofísica, Óptica y Electrónica, 2010.

F.Iida, M.Tabata, & F.Hara. Generating personality character in a face robot through interaction with human. En *7th IEEE International Workshop on Robot and Human Communication*, 1998.

H.Kimura, T.Yamashita, & S.Kobayashi. Reinforcement learning of walking behavior for a four-legged robot. En *40th IEEE Conference on Decisions and Control*, 2001.

W. B.Knox & P.Stone. Combining manual feedback with subsequent MDP reward signals for reinforcement learning. En *9th International Conference Autonomous Agents and Multiagent Systems*, 2010.

T.Kollar & N.Roy. Using reinforcement learning to improve exploration trajectories for error minimization. En *IEEE International Conference on Robotics and Automation, ICRA*, 2006.

G.Konidaris & A.Barto. Autonomous shaping: Knowledge transfer in reinforcement learning. En *23rd International Conference on Machine Learning*, 2006.

A.Laud. *Theory and Application of Reward Shaping in Reinforcement Learning.* Tesis de Doctorado, Departamento de Ciencias Computacionales, Universidad de Illinois, 2004.

A.Laud & G.DeJong. Reinforcement learning and shaping: Encouraging intended behaviors. En *19th International Conference on Machine Learning*, 2002.

A.Lazaric, M.Restelli, & A.Bonarini. Reinforcement learning in continuous action spaces through sequential Monte Carlo methods. *Advances in Neural Information Processing Systems*, (20):833–840, 2008.

T. A.Lockerd & C.Breazeal. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. En *21st National Conference on Artificial Intelligence*, vol. 1, 2006.

T. A.Lockerd, G.Hoffman, & C.Breazeal. Real-time interactive reinforcement learning for robots. En *AAAI Workshop on Human Comprehensible Machine Learning*, 2005.

T. A.Lockerd, G.Hoffman, & C.Breazeal. Reinforcement learning with human teachers: Understanding how people want to teach robots. En *15th IEEE International Symposium on Robot and Human Interactive Communication*, 2006.

A.Mansoor. *How Speech Recognition Works.* University of Engineering and Technology Lahore Pakistan, 2001.

B.Marthi. Automatic shaping and decomposition of reward functions. En *24th International Conference on Machine Learning*, 2007.

M.J.Mataric. Reward functions for accelerated learning. En *Eleventh International Conference on Machine Learning*, 1994.

F. S.Melo & M.Lopes. Fitted natural actor-critic: A new algorithm for continuous state-action MDPs. En *European Conference on Machine Learning and Knowledge Discovery in Databases - Part II. Lecture Notes In Artificial Intelligence*, vol. 5212, 2008.

J.Millán, D.Posenato, & E.Dedieu. Continuous-Action Q-learning. 49, 2002.

T.Mitchell. *Machine Learning.* McGraw-Hill, 1997.

A.Moore & C.Atkeson. Prioritized sweeping: reinforcement learning with less data and less real time. 13, 1993.

J.Morimoto & K.Doya. Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning. En *Seventeenth International Conference on Machine Learning*, 2000.

A.Muñoz & E.Morales. *Notas de curso: Introducción a la Robótica.* Coordinación de Ciencias Computacionales, Instituto Nacional de Astrofísica, Óptica y Electrónica, 2009.

A.Y.Ng, J.Kim, M.Jordan, & S.Sastry. Autonomous helicopter flight via reinforcement learning. En *Advances in Neural Information Processing Systems*, pp. 799–806. MIT Press, 2004.

Y. A.Ng, D.Harada, & S.Russel. Policy invariance under reward transformations: Theory and application of reward shaping. En *International Conference in Machine Learning*, 1999.

J.Peng & R.Williams. Efficient learning and planning with the dyna framework. 1, 1993.

L.Pineda. Corpus dimex100 (level t22) dime project. computer sciences department. IIMAS, UNAM, 2010.

J.Randlov & P.Alstrom. Learning to drive a bicyle using reinforcement learning and shaping. En *Fifteenth Intenational Conference on Machine Learning*, 1998.

C. E.Rasmussen. Gaussian processes in machine learning. En *Advanced Lectures on Machine Learning, ML Summer Schools 2003. Lecture Notes in Computer Science.* Springer, 2004.

A.Rottmann, C.Plagemann, P.Hilgers, & W.Burgard. Autonomous blimp control using model-free reinforcement learning in a continuous state and action space. En *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2007.

P. E.Rybski, K.Yoon, J.Stolarz, & M.Veloso. Interactive robot task training through dialog and demonstration. En *ACM/IEEE International Conference on Human Robot Interaction, HRI*, 2007.

BasilioSierra. *Aprendizaje Automático: conceptos básicos y avanzados.* Pearson Prentice Hall, 2006.

S. P.Singh & R.Sutton. Reinforcement learning with replacing elegibility traces. *Machine Learning*, (22):123–158, 1996.

W. D.Smart & L.Pack Kaelbling. Effective reinforcement learning for mobile robots. En *IEEE International Conference on Robotics and Automation, ICRA*, 2002.

R.Sutton. Integrated arquitectures for learning, planning, and reacting based on aproximating dynamic programming. En *Seventh International Conference on Machine Learning*, 1990.

R.Sutton & A.Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998.

M.Tamosiunaite, T.Asfour, & F.Worgotter. Learning to reach by reinforcement learning using a receptive field based function approximation approach with continuous actions. *Biological Cybernetics*, 100, 2009.

T.Tsubone & W. YasuhiroK. Sugiyama. Robot task learning based on reinforcement learning in virtual space. En *Neural Information Processing - Letters and Reviews*, pp. 165–174, 2007.

H.Van Hasselt & M.Wiering. Using continuous action spaces to solve discrete problems. En *Proceedings of the International Joint Conference on Neural Networks, IJCNN*, 2009.

H.Van Hasselt & M. A.Wiering. Reinforcement learning in continuous action spaces. En *IEEE International Symposium on In Approximate Dynamic Programming and Reinforcement Learning*, 2007.

Y.Wang, M.Huber, V.N.Papudesi, & D.J.Cook. User-guided reinforcement learning of robot assistive tasks for an intelligent environment. En *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

S.Whiteson, M. E.Taylor, & P.Stone. Adaptive tile coding for value function approximation. Reporte Técnico AI-TR-07-339, Departamento de Ciencias Computacionales, University of Texas, 2007.

Y.Zhang & J.Weng. Action chaining by a developmental robot with a value system. En *2nd International Conference on Development and Learning*, 2002.

Referencias en línea

[HarmonicArm_AAICanadaInc] Harmonic Arm 6M. AAI Canada Inc.

URL: *http://www.aai.ca/robots/h_arm.html*

Fecha de consulta: 5 de marzo del 2010

[Pioneer_MobileRobots] Pioneer 2. Mobile Robots.

URL: *http://www.mobilerobots.com/ResearchRobots/ResearchRobots.aspx*

Fecha de consulta: 2 de noviembre del 2010

[Player_Stage] Player/Stage.

URL: *http://playerstage.sourceforge.net*

Fecha de consulta: 3 de noviembre del 2009

Apéndice A

Listado del vocabulario

Se presenta la lista de palabras que conforman el vocabulario del reconocedor de voz, y en el caso de palabras que involucran una evaluación, se acompañan de su valor de refuerzo. Para indicar acciones:

avanza
avance
avanzar
avanzando
caminando
camina
sigue
adelante
llega
ir
dirigir
dirigirse
ve
ven
gira
vuelta
girar
dobla
antes
atras
comenzar
comienza
iniciar
inicio
empezar
empieza
vuelve
regresa
regresar
retrocede

espere
detente
detenerse
dar
para
vienes
vas
estas
hagas
vayas
era
acaba
acabar
accede
acceda
acceder
accesar
actuar
alcanza
alcanzar
siguele
sigue
continua
continuar
efectuar
efectue
entra
entrar
espera
explorar
salir
terminar
termino

Calificativos:

bien 10
mal -10
excelente 50
terrible -50
peor -50
regaste -50
equivocaste -10
error -10
aprobado 10
asqueroso -50
aturdido -10
atrocidad -50
autorizado 10
bravo 100
buenisimo 100
bueno 10
buena 10
completado 100
completo 100
hecho 100

condenado -100
suficiente
despistado -50
efectivo 10
excelso 100
extraordinario 100
fallaste -50
fallo -50
felicidades 100
grandioso 100
hermoso 100
horrible -100
incorrecto -50
listo 100
malo -10
mejor 10
mejoras 10
super 50
muy 50
buen 50
objetivo 100

Vocabulario adicional:

a
hacia
para
por
ahi
aquí
cerca
hasta
allá
acá
através
aquello
aquellos
aquella
aquellas
el
la
tu
te
es
uno
dos
tres
cuatro
cinco
seis
directo
derecho
izquierda
derecha
fin
final

mano	así
lado	como
parte	hecho
otro	trabajo
todo	eso
hey	que
mejor	ahora
ya	alto
no	lejos
si	lento
mas	rápido
casi	más

Apéndice B

Tabla de notación

A continuación se presenta una tabla con la notación usada a lo largo de esta tesis.

Tabla B.1 Tabla de notación

Algoritmo de Aprendizaje	
α	coeficiente de aprendizaje
γ	parámetro de descuento
λ	factor de descuento de la traza
r	recompensa
S	conjunto de estados
s	estado
A	conjunto de acciones
a	acción
Pr	función de probabilidad
δ	función de transición
$V^{\pi}(s)$	función de valor basada en estados siguiendo la política π
E_{π}	valor esperado basado en la política π
$Q^{\pi}(s,a)$	función de valor basada en estado-acción siguiendo en la política π
$e(s,s)$	función badada en estado-acción que mantiene información de la traza
max_{π}	función que devuelve un máximo valor de acuerdo a la política π
va	valor (magnitud) de una acción
f	retroalimentación del usuario (evaluando)
$g(r,a)$	función de recompensa adicional cuando el usuario comanda una acción
$h(r,f)$	función de recompensa adicional cuando el usuario retroalimenta (evaluando)
R	recompensa total acumulada
R_t	recompensa tradicional del algoritmo
R_e	recompensa adicional provista por el usuario

Tabla B.2 Tabla de notación

Coeficiente de Pearson	
x	vector de números reales que representan a un estado
y	vector de números reales que representan a un estado
N	número de datos de cada vector
r	coeficiente de Pearson

Función gaussiana	
ϕ	función gaussiana
e	constante matemática de valor real 2,7182818284590452354...
μ	media de la función gaussiana (vector de números reales)
σ	desviación estándar de la función gaussiana
x	vector de números reales que representan a un estado
π	número *pi* 3,14159265358979323846...

Símbolos matemáticos varios	
t	tiempo
$\Re$	conjunto de números reales
k	índice indicando un momento en el tiempo
Σ	sumatoria
$\forall$	cuantificador universal *para todo*
$\in$	símbolo de relación *pertenece a*
∞	infinito

Apéndice C

Información técnica

C.1. Player Stage

Player v2.1.1 es un servidor que trabaja sobre una red para controlar un robot. Provee una interfaz para trabajar con los accionadores y los sensores a través de una red IP. Un programa cliente se conecta con Player a través de TCP para leer datos de los sensores, mandar comandos a los accionadores y configurar los equipos de *hardware*. Fue creado originalmente para trabajar con los robots de ActiveMedia Pioneer 2 pero son soportados actualmente otro tipo de robots.

El servidor Player:

- Es compatible con Linux (en PC o en computadoras embebidas), Solaris y BSD.
- Es independiente del lenguaje o plataforma (librerías para el programa cliente libplayerc++, libplayerc_py, libplayerc, disponibles en varios lenguajes).
- Cuenta con utilidades para programas cliente en C++, Tcl, Java y Phyton (playernav, playervcr, player print, playerwritemap, playerv, dgps_server,

player server, playerjoy, playercam).

- Puede trabajar con programas de un solo hilo o multihilo.
- Sus interfaces de los dispositivos de *hardware* funcionan para varios robots.
- Soporta virtualmente cualquier número de clientes.
- Puede trabajar con simuladores para robóts.

Stage v2.1.1 es un simulador 2D para multiples robots. El *software* simula objetos, robots, sensores en ambientes en 2 dimensiones.

El simulador Stage:

- Provee varios modelos para sensores y accionadores, como sonares, sensores infrarrojos, láser, odometría y localización global.
- Soporta sistemas multiagente.
- Compatible con el servidor Player.

Las tablas C.1, C.2, C.4, C.5, C.7 muestran el *software* y *hardware* compatible con Player.

C.2. Pioneer 2

El robot móvil Pioneer 2 de MobileRobots (ActivMedia) es un robot autónomo que puede navegar fácilmente en espacios reducidos y saturados de objetos. El robot Pioneer 2:

1. Cuenta con una computadora interna.
2. Puede trabajar de forma completamente autónoma.

Tabla C.1 Robots compatibles con Player

Fabricante	Equipo	Controlador
Acroname	Garcia	garcia
Botrics	Obot d100	obot
Evolution Robotics	ER1 and ERSDK robots	er1
iRobot	Roomba vaccuming robot	roomba
K-Team	Robotics Extension Board (REB) en el Kameleon 376BC	reb
K-Team	Khephera	khepera
MobileRobots (ActivMedia)	Robots basados en PSOS/P2OS/AROS (e.g., Pioneer, AmigoBot) y accesorios integrados, incluyendo una cámara CMUcam conectada al puerto AUX.	p2os
Nomadics	Robots móviles NOMAD200	nomad
RWI/iRobot	Robots basados en RFLEX (e.g., B21r, ATRV Jr) y sus accesorios integrados.	rflex
Segway	Plataforma de mobilidad robótica (RMP), una versión modificada por el cliente del Transporte Humano (HT)	segwayrmp
UPenn GRASP	Clodbuster	clodbuster
Videre Design	Plataforma de robots móviles ERRATIC	erratic
White Box Robotics	914 PC-BOT	wbr914

Tabla C.2 *Hardware* adicional compatible con Player

Fabricante	Equipo	Controlador
Amtec	PowerCube pan-tilt unit amtecpowercube	driver_amtecM5
Ascension	Flock of Birds position tracker	flockofbirds
Canon	Cámara pan-tilt-zoom VC-C4	driver_canonvcc4
Crossbow	Nodos de sensores Mica2 / Mica2DOT	mica2
DirectedPerception	Unidad pan-tilt PTU-D46	ptu46
Garmin	Receptor GPS Geko 201 (y la mayoría de las unidades GPS compatible con NMEA)	garminnmea
Hokuyo	URG telémetro láser	urglaser
Inside Contactless	Lector M300/R300 2G RFID (13.56MHz)	insideM300
InterSense	InertiaCube2 IMU	inertiacube2
MicroStrain	3DM-G IMU	microstrain3dmg
Logitech	Cámara pan-tilt-zoom QuickCam Sphere	sphere
SICK	LMS 200 telémetro láser (y posiblemente otros láseres SICK)	sicklms200, driver_sickpls, sicks3000, sick-nav200
SkyeTek	Lectores M1/M1-mini RFID (13.56MHz)	skyetekM1
Sony	Cámaras pan-tilt-zoom EVID30 y EVID100	sonyevid30
TeCo	Particle Router Core (XBridge) nodo de sensores	rcore_xbridge
XSens	Mtx / Mti Unidad de medida inercial	xsensmt
Various (CMU)	Cámara pan-tilt blob-tracking CMUcam2	cmucam2

Tabla C.3 *Hardware* adicional compatible con Player (segunda parte)

Fabricante	Equipo	Controlador
Various	Generic Linux sound hardware. Realiza la Transformada rápida de Fourier sobre los datos entrantes de audio y emite varios tonos.	driver_acoustics
Various	Generic Linux sound hardware. Provee acceso para controles mezcladores (e.g., volumen).	driver_mixer
Various	Generic Linux sound hardware. Provee la habilidad de transmitir y recibir señales de audio raw.	driver_waveaudio
Various	Generic Linux wireless hardware. Provee acceso, via /proc, a información acerca de el *hardware* de red (e.g., calidad de la señal).	linuxwifi
Various	Generic Linux wireless hardware. Provee acceso, via iwspy, a información acerca del *hardware* de red inalámbrico (e.g., calidad de la señal).	iwspy
Various	Cámaras IEEE1394 (aka Firewire)	camera1394
Various	Cámaras compatibles con Video4Linux	camerav4l
Various	Generic Linux joystick. Permite acceso al eje y al botón del estado	linuxjoystick

Tabla C.4 *Software* adicional compatible con Player

Autor	Equipo	Controlador
Jim Bruce	Software blob-tracking Color Machine Vision (CMVision)	cmvision
MobileRobots (formerly ActivMedia); escrito pot Paul Rybski	Software blob-tracking ActivMedia Color Tracking System (ACTS)	acts
MIT / LIRA-Lab	YARP image capture	yarpimage
The Centre for Speech Technology Research, Universidad de Edinburgh	Festival speech-synthesis software	festival
Speech de CMU	Sphinx2 speech-recognition software	sphinx2
Porchdog Software	Librería de redes Howl Zeroconf	service_adv_mdns

Tabla C.5 Algoritmos compatibles con Player

Algoritmo	Descripción	Referencias	Controlador
Nearness Diagram (ND) Navigation	Búsqueda de un objetivo y evasión de obstáculos, especialmente para robots no-holonomicos, robots no circulares y robots que trabajen en espacios reducidos	Minguez & Montano (2004), Nearness Diagram Navigation (ND): Collision Avoidance in Troublesome Scenarios Minguez, Osuna, & Montano (2004), A Divide and Conquer Strategy based on Situations to Achieve Reactive Collision Avoidance in Troublesome Scenarios.	nd
Configuration space	Modula los datos del láser raw de acuerdo a la configuración de las dimensiones del robot para presentar el área vacía en el área del espacio de configuración.		lasercspace
Vector Field Histogram (VFH+)	Algoritmo de búsqueda de un objetivo evasión de obstáculos	Ulrich & Borenstein (1998), "VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots"	vfh

Tabla C.6 Algoritmos compatibles con Player (segunda parte)

Algoritmo	Descripción	Referencias	Controlador
Adaptive Monte Carlo Localization (AMCL)	Realiza localización probabilística para un robot móvil equipado con un mapa y los sensores apropiados	Fox (2001), KLD-Sampling: Adaptive Particle Filters	amcl
Wavefront propagation planner	Realiza planeación de trayectorias en el espacio de configuración para un robot via descomposición de céldas y propagación *wavefront*.	Latombe (1991), Robot Motion Planning	wavefront

Tabla C.7 Simuladores compatibles con Player

Proyecto	Nombre	Descripción	Controlador
The Player / Stage / Gazebo project	Gazebo	Un simulador 3-D multi-robot	gazebo
The Player / Stage / Gazebo project	Stage	Un simulador 2-dimensional multi-robot	stage

Tabla C.8 Especificaciones del Pioneer 2

Característica	Pioneer 2
Motores	DC-Motores con codificadores
Altura	21.5 cm
Ancho	38 cm
Construcción	Aluminio
Peso	9 Kg
Peso de la carga	23 Kg
Voltaje y corriente	12V 7A
Velocidad	1.6 m/s

3. Cuenta con el *software* servidor P2OS que trabaja con el *software* cliente de ActivMedia Robotics, con los cuales es capaz de hacer mapas del ambiente y ubicarse, además de ejecutar tareas de planeación.

4. Cuenta con carcasa y consola, nariz, cuerpo y paneles de accesorios, un arreglo de sonares, motores y codificadores, baterías y toma de alimentación, dos ruedas.

En la tabla C.8 se muestran las especificaciones físicas del robot móvil.

C.3. Harmonic arm 6m

El brazo robótico Harmonic arm 6m de AAI Canada Inc. es un manipulador con diferentes sensores y 6 grados de libertad. El Harmonic arm 6m:

1. Puede ser usado principalmente en aplicaciones de investigación y educación.

Tabla C.9 Especificaciones del Harmonic arm 6m

Característica	Harmonic Arm 6m
Motores	DC-Motores con codificadores digitales de posición integrados
Precisión	+-0.1 mm (precisión de repetición)
Altura de la base	188 mm
Grados de libertad	5 or 6 (6 Motores)
Radio de trabajo	60 cm
Construcción	Aluminio anodizado
Peso	4.1 kg
Peso de la carga	500 grs
Voltaje y corriente	12V 3.5A
Velocidad	Promedio de 90 grados en 1 segundo (todas las uniones se mueven en paralelo)

2. Puede trabajar en tareas de manipulación en la industria con objetos de hasta 500 grs.

3. Puede trabajar solo o sobre un robot móvil.

4. Puede ser controlado directamente a través de una interfaz serial con una PC o usando un controlador lógico programable (PLC), o puede ser controlado a través de una interfaz especial de un robot móvil.

5. La interfaz serial de comunicación puede trabajar en modo asíncrono o síncrono.

En la tabla C.9 se muestran las especificaciones físicas del brazo robótico.