UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

TESIS DOCTORAL

Planificación óptima de movimiento y aprendizaje por refuerzo en vehículos móviles autónomos

Mariano Gómez Plaza
Ingeniero de Telecomunicación

2009

A mi esposa Marta e hijo Marcos por el tiempo que no he podido dedicarles debido a la realización de este trabajo, y por tanto, que les debo.

A mis padres y hermana, que después de algunos años, al fin pueden ver la culminación de esta obra.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABLAS

Resumen

El presente trabajo de investigación se ha centrado en la propuesta de un algoritmo, capaz de realizar una planificación óptima de movimiento en vehículos móviles autónomos, basándose en técnicas de control óptimo en lazo cerrado. Estos vehículos se caracterizan por estar dotados de cuatro ruedas, con dirección delantera y tracción delantera o trasera y, fundamentalmente, por ser sistemas dinámicos no lineales en los cuales la planificación de movimiento y su control, son tareas complejas, por ser sistemas no holónomos. Todos los estudios llevados a cabo se han realizado considerando los conceptos de *aprendizaje por refuerzo* y *espacio de estados celular*.

El nuevo algoritmo es muy robusto ante cambios en el entorno o en la estructura física y mecánica del propio vehículo, de manera que la generación del controlador óptimo vendrá dada tras la ejecución de una fase de aprendizaje que tiene en cuenta de forma implícita estos posibles cambios. El aprendizaje se orienta para alcanzar un objetivo desde cualquier origen, de acuerdo a un criterio de optimización (por ejemplo, tiempo mínimo). El vehículo aprende de su propia experiencia, la dinámica y cinemática sin necesidad de disponer de modelos matemáticos. Además, el tiempo de aprendizaje se reduce debido a la capacidad del algoritmo de extrapolar el conocimiento adquirido localmente en una zona del espacio, al resto del espacio sin necesidad de que el vehículo se mueva físicamente por todo el espacio considerado.

Una vez concluido el aprendizaje, la planificación se hace en lazo cerrado aplicando las acciones de control óptimas correspondientes al estado del vehículo en tiempo real.

Summary

This research work is focus on the proposal of an algorithm able to perform an optimal motion planning in two-wheel-drive (front or rear) and front steering autonomous mobile vehicles. The algorithm is based on optimal control techniques in closed loop applied to four-wheel vehicles. These vehicles are non-linear dynamic systems in which the motion planning and its control are difficult tasks due to they are non-holonomic systems. All studies done have been developed taking into account the *reinforcement learning* and *cell state space* concepts.

Possible changes in the environment or in the physical or mechanical structure of the vehicle do not affect the new algorithm. The optimal controller will be generated when the learning stage has been previously performed and therefore, these changes have been implicitly taken into account. The learning is oriented to reach a specific goal from each origin, according to an optimization criterion (e.g. minimum time). The vehicle learns its kinematics and dynamics from its own experience. There is no need to have any kind of mathematical model of the system. Also, the algorithm is able to extend the local knowledge acquired in a specific zone of the state space to the rest of the space without the vehicle moving physically to those zones.

Once the learning stage finishes, the planning is performed in closed loop applying the optimal control actions associated to the state of the vehicle in real time.

Capítulo 1

Sólo cabe progresar cuando se piensa en grande, sólo es posible avanzar cuando se mira lejos.

JOSÉ ORTEGA Y GASSET (1883-1955. Filósofo y ensayista español)

1. Introducción

1.1 Motivación de la tesis

Uno de los mayores retos de la comunidad científica dedicada a la investigación de técnicas y métodos para la planificación del movimiento en vehículos no holónomos autónomos, es proponer un algoritmo que permita al vehículo realizar dicha planificación en cualquier entorno. Al vehículo se le dota de sensores y sistemas de referencia, dando preferencia a que éstos sean internos al propio vehículo. La idea es conseguir una mayor autonomía prescindiendo de elementos de control externos y simplificando o reduciendo los internos.

La planificación de rutas y algoritmos de control óptimo para resolver problemas prácticos específicos, tales como generación de trayectorias en vehículos, se han utilizado en diversos proyectos de investigación. Los vehículos empleados son sistemas dinámicos cuyas leyes cinemáticas han sido ampliamente estudiadas [Ree-90] [Lat-91] y se suelen utilizar a menudo en la industria por su capacidad para

transportar objetos de gran peso. Sin embargo, la planificación de su movimiento y control, son tareas complejas debido a que son sistemas no holónomos. Estos trabajos orientados a la generación de trayectorias se basan en el conocimiento previo del entorno por el que se moverá el vehículo. Basados en un modo de funcionamiento *off-line*, se calcula la trayectoria, que posteriormente y en función de determinadas acciones de control, se aplicará al vehículo para su generación. El problema de estas técnicas es que el control aplicado se hace en lazo abierto mientras que en entornos reales, donde existen perturbaciones diversas, las soluciones en lazo cerrado se comportarán mejor.

Considerando un control en lazo cerrado, éste se podría llevar a cabo usando programación dinámica [Lat-91], pero las dificultades y desventajas de esta solución son el alto coste computacional y la necesidad de disponer de un modelo dinámico del vehículo.

La resolución del problema planteado anteriormente de generación de trayectorias en vehículos, en general tiene dos enfoques distintos: un enfoque cinemático, que estudia las leyes del movimiento del propio vehículo sin tener en cuenta las causas que lo producen y limitándose esencialmente al estudio de la trayectoria conforme transcurre el tiempo; y otro enfoque dinámico, que estudia también las trayectorias en función del tiempo, pero a la vez teniendo en cuenta las fuerzas y las alteraciones que éstas provocan en el movimiento del vehículo. El primer enfoque es el utilizado por los trabajos anteriores, mientras que en esta tesis, además de hacer un estudio cinemático del problema, el objetivo principal es proponer una solución que considerando la dinámica haga un control óptimo en la planificación del movimiento del vehículo.

Existen otros trabajos que, aún proponiendo soluciones en tiempo real y sin necesidad de disponer de un modelo dinámico matemático, son capaces de realizar un control autónomo e incluso navegar en un entorno sin colisionar con posibles obstáculos [Gas-00] [Thr-95] [Gas-99b] [Wet-99]. Por ejemplo, uno de los últimos avances e investigaciones en esta línea, es llevado a cabo por el Programa

DARPA Urban Challenge cuyo objetivo principal es la investigación y desarrollo de vehículos autónomos, centrándose en la búsqueda de rutas para dirigir a los vehículos por un camino determinado, basándose en la información recopilada de una red de carreteras urbanas a través de una serie de puntos de control [Thr-06] [Sta-07] [Mon-08]. También es capaz de tratar con posibles cambios dinámicos en la red de carreteras haciendo una re-planificación en tiempo real con la información actualizada.

Los trabajos fruto de los cuales ha surgido esta tesis están centrados en la investigación de una solución que, basada en control óptimo y en lazo cerrado, dote al vehículo del conocimiento del entorno por donde se mueve y que éste aprenda su propia dinámica interactuando con dicho entorno, de manera que sea lo más inmune a posibles perturbaciones externas. De esta manera y en tiempo real el vehículo adquiere el conocimiento necesario para la generación óptima de cualquier ruta.

Sin embargo, hay otras investigaciones que se centran en el problema de la navegación de robots para generar caminos óptimos, pero no tienen la capacidad de adquirir un conocimiento de su propia dinámica, sino que van generando sus trayectorias en tiempo real conforme se van desplazando y esquivando los posibles obstáculos que encuentran por el camino. En estos casos, se diferencian dos tipos de navegación: local, basada únicamente en la información instantánea del entorno cercano al robot; y global basada en un conocimiento a priori del escenario completo por el que se puede mover el robot. En el caso de una navegación global, el movimiento del robot se podría abordar mediante la técnica del campo de gradiente o *Gradient Path Planning* [Bor-91] [Min-08], que garantiza caminos óptimos. Esta técnica genera un campo de potencial, que se expande como una onda por los huecos libres del mapa, desde el objetivo hasta la posición del robot, asignando a cada punto del espacio valores crecientes. Para navegar hacia un objetivo, simplemente se siguen los valores de este campo en la dirección del gradiente. La navegación local complementaría a la técnica anterior,

haciéndola más robusta y dotándola de un comportamiento reactivo, para que pueda esquivar obstáculos inesperados que no aparecen inicialmente en el mapa. Para ello, se usa la técnica de los campos de fuerza virtuales o *Virtual Force Field (VFF)* [Kon-00]. En ella, el objetivo marcado por la navegación global ejerce una fuerza atractiva sobre el robot y los obstáculos detectados generan fuerzas repulsivas. El robot se dirigirá por la suma vectorial de ambas fuerzas consiguiendo una navegación completa.

1.2 Planteamiento del problema

Esta tesis está fundamentada en el estudio y análisis de la dinámica de un sistema continuo no lineal con objeto de poder realizar una planificación de movimiento y control óptimo sobre el mismo. El sistema considerado ha sido un vehículo no holónomo cuyas leyes de movimiento han sido ampliamente estudiadas en [Ree-90] [Lat-91] [Gu-02]. Sus características vienen expresadas en el Apéndice A. La complejidad de realizar tareas de planificación de movimiento y control óptimo en este tipo de vehículos (véase apartado 1.1), ha generado una motivación especial en esta tesis para investigar en la propuesta de un algoritmo capaz de realizar dichas tareas.

En un sistema dinámico las distintas variables que podemos asociar a sus partes sufren cambios a lo largo del tiempo como consecuencia de las interacciones que se producen entre ellas. Su comportamiento vendrá dado por el conjunto de trayectorias de todas las variables, que suministra algo así como una narración de lo acaecido en el sistema. Otros conceptos que conviene revisar en este momento son los de *experimento, modelo* y *simulación*. El término *experimento* se puede definir como el proceso de extraer datos de un sistema mediante la activación de sus entradas. Con el término *modelo*, nos referimos a un objeto que representa a otro. En este sentido una definición muy apropiada sería la siguiente: Para un observador O un objeto M es un modelo de un objeto S (un sistema) y un experimento E, si O se puede servir de M para aplicar E y responder a cuestiones que le importan con relación a S. Como se verá en el Capítulo 3, un aspecto

crucial en el trabajo llevado a cabo en esta tesis ha sido el establecimiento de un modelo, a partir del cual, poder generar, reduciendo tiempo de cómputo, el controlador óptimo que planificará el movimiento del vehículo de una forma óptima. En cuanto al término *simulación*, se podría definir como una técnica de desarrollo y ejecución de un *modelo* de un sistema dinámico real para estudiar su conducta sin irrumpir en el entorno del sistema real.

La complejidad del presente trabajo radica en el hecho de contemplar la dinámica intrínseca al propio sistema continuo y no lineal en el que se quiere aplicar el control óptimo. Para ello, se va a definir un espacio de estados formado por una serie de variables. La idea es que en el conjunto de los estados de este espacio esté incluido el comportamiento del sistema. Teniendo en cuenta todo lo anterior, existen dos formas de abordar el problema de control óptimo:

1) Considerar las ecuaciones diferenciales que rigen el comportamiento del sistema en el entorno de su aplicabilidad. En este caso el controlador puede ser generado a priori, es decir, bajo un entorno simulado utilizando dichas ecuaciones e interactuar con ellas en vez de con el entorno real. Una vez implementado el controlador utilizando alguno de los métodos o algoritmos expuestos a lo largo de los capítulos siguientes, éste podrá ser incorporado al sistema para su uso.

   Esta alternativa tiene la ventaja de obtener el controlador en un tiempo relativamente pequeño y en unas condiciones de operación más cómodas y sencillas que en un entorno real. Sin embargo, presenta los siguientes inconvenientes:

   - Grado de fidelidad de las ecuaciones diferenciales con respecto al comportamiento real.
   - Sensibilidad al cambio de cualquier parámetro (masa, dimensiones, ángulos, etc.).
   - Dificultad de encontrar o definir las ecuaciones diferenciales que caracterizan el comportamiento del sistema.

Los inconvenientes anteriores hacen que en el momento de aplicar el controlador generado sobre el sistema real, la planificación de movimiento para el vehículo no sea la esperada y difiera considerablemente.

2) Dotar al sistema de la capacidad de aprender su propia dinámica interactuando con el entorno real. Con esta segunda forma, la gran ventaja es que el controlador que se genere es fiel al entorno real por lo que a la hora de su aplicabilidad no deberá provocar comportamientos inesperados. Sin embargo, la fase de aprendizaje en la que se debe obtener el controlador, puede llegar a ser muy extensa e invertir un tiempo en algunos casos no aceptable. Para subsanar este problema, se ha propuesto una fase de aprendizaje que sea capaz de generar un *modelo* del comportamiento del sistema en un subconjunto de los estados que constituyen el sistema y el entorno reales. Este modelo servirá para extrapolarlo al resto de estados y así tener el conocimiento total del comportamiento del sistema.

1.3 Estructura de tesis

La presente memoria consta de seis capítulos y dos apéndices. El capítulo actual sirve de introducción y en él se reflejan de forma breve el contexto y los principales objetivos de investigación. En el capítulo dos se presentan los conceptos fundamentales sobre los que se basa la investigación y se realiza una revisión de las técnicas de control óptimo involucradas en el diseño del nuevo algoritmo.

En el capítulo tres se detallan los trabajos llevados a cabo durante el desarrollo de la investigación, se describe un método de evaluación basado en el concepto de controlabilidad y curva de conmutación para controladores generados con el nuevo algoritmo y se indica una nueva transformación de las transiciones para reducir el tiempo de aprendizaje del controlador. El capítulo cuatro presenta y analiza los

resultados que se han obtenido como consecuencia de la investigación realizada, además de evaluar el trabajo llevarlo a cabo.

En el capítulo cinco se presentan las conclusiones extraídas, las aportaciones originales de esta tesis y futuras líneas de investigación. En el capítulo seis, se enumera por un lado, la bibliografía utilizada en la investigación y que ha servido como referencia en los trabajos realizados y por otro, aquellas referencias que se han generado fruto de la tesis: [Gom-01] [Gom-02] [Gom-03] [Gom-07a] [Gom-07b] [Gom-08] [Gom-09a] [Gom-09b] [Gom-09c].

Por último, en el apéndice A se especifican los parámetros y los distintos modelos matemáticos con los que se pueden caracterizar los vehículos no holónomos; y en el apéndice B se describen las características del vehículo utilizado en esta tesis y con el que se han hecho las diferentes pruebas reflejadas en el capítulo cuatro.

1.4 Contribuciones

Durante la investigación realizada en la presente tesis, se han aportado las siguientes contribuciones científicas en la planificación del movimiento de vehículos autónomos no holónomos:

- Propuesta de un nuevo algoritmo de control óptimo basado en la combinación de técnicas de *Cell Mapping* y aprendizaje por refuerzo, para la generación de controladores aplicados a vehículos reales no holónomos.
- Reducción drástica del tiempo de aprendizaje del controlador, gracias a la introducción de las ecuaciones de transformación.
- Generación de un controlador óptimo basado en un espacio de estados que considera la dinámica del vehículo sin tener en cuenta modelo matemático alguno.

Capítulo 2

Las ciencias no tratan de explicar, incluso apenas tratan de interpretar, construyen modelos principalmente. Por modelo, se entiende una construcción matemática que, con la adición de ciertas interpretaciones verbales, describe los fenómenos observados. La justificación de tal construcción matemática es sólo y precisamente que se espera que funcione.

JOHN VON NEUMANN (1903-1957. Matemático)

2. Revisión histórica y contexto

2.1 Control óptimo de sistemas dinámicos

2.1.1 Introducción

La teoría de control óptimo se enmarca dentro de la teoría de control moderna que empieza a desarrollarse a comienzos de la década de los 60 para dar respuesta a la necesidad de controlar sistemas cada vez más complejos. Esta teoría permite abordar el estudio de sistemas con múltiples entradas y salidas relacionadas entre sí de forma, en muchos casos, no lineal y variante con el tiempo.

Los problemas de control óptimo han recibido una gran atención debido a la creciente demanda de sistemas de elevada eficiencia. Esas demandas pueden traducirse en especificaciones de funcionamiento o desempeño, tales como minimizar los gastos de combustibles y las distancias en sistemas de transporte, minimizar tiempos de operación, minimizar costes, maximizar ganancias, etc. En general, los sistemas y procesos reales que se encuentran en el área de la ingeniería presentan un comportamiento no lineal, muchos de los cuales son multivariables, y pueden variar en el tiempo algunas de sus características debido a factores externos que influyen sobre los mismos.

El objetivo central de la teoría de control es proporcionar estrategias para conducir un cierto proceso hacia un objetivo deseado. La teoría de control óptimo hace hincapié tanto en el estudio de condiciones necesarias y suficientes para la existencia y unicidad de un determinado control, como en el desarrollo de metodologías para su determinación y computación. Se entiende como control óptimo aplicado a un sistema dinámico, aquél donde es necesario especificar:

- Cómo se va a controlar el sistema, es decir, se debe especificar el conjunto de estrategias admisibles, también llamadas políticas de control o simplemente políticas o estrategias.
- Restricciones adicionales tanto en el estado del sistema como en las estrategias en caso de que las hubiere.
- Cómo se va a medir la respuesta del sistema ante cada una de las distintas políticas de control, es decir, se debe especificar la denominada función coste.

De acuerdo con la idea de control óptimo de [Bar-95], se podría afirmar que el problema del control óptimo es mejorar el resultado de una función coste sujeta a la evolución dinámica del sistema, políticas de control y restricciones. Teniendo en cuenta estas premisas, un problema típico de control óptimo requiere controlar

un sistema para ir desde un estado inicial a un estado final siguiendo una trayectoria que minimice su coste.

El estado de un sistema dinámico es el conjunto más pequeño de variables, denominadas variables de estado, tales que su conocimiento en $t=t_0$, junto con el de los valores de la entrada para $t \in [t_0, t_1]$, determinen totalmente el comportamiento del sistema en el intervalo $[t_0, t_1]$ [Oga-80].

Es práctica común representar el estado de un sistema, mediante un vector de dimensión igual al número de variables de estado de que consta el sistema dinámico. Este vector puede tomar valores en el llamado espacio de estados, y representa el estado en el cual se encuentra el sistema en cada instante de tiempo. Los sistemas que se van a considerar en todos los problemas planteados en el presente trabajo se van a regir por la siguiente ecuación diferencial vectorial:

$$\dot{x} = f(x, u, t), \quad F : \Re^n x \Re^m \Re^+ \rightarrow \Re^n, \quad F \in C^1, \qquad [1.1]$$

donde $x \in \Re^n$ es el vector de estado, $u \in \Re^m$ es el vector de control y f es una función vectorial, en general, no lineal y diferenciable con continuidad. La ecuación [1.1] define la evolución en el tiempo del estado del sistema. El conocimiento de esta evolución temporal que define el comportamiento de la dinámica del sistema, nos va a permitir aplicar técnicas de control (en este caso óptimo) sobre el propio sistema con objeto de conseguir optimizar cierto comportamiento del mismo. Con esta premisa, en el año 1957 Richard Bellman empezó a trabajar en la teoría de control óptimo. Para ello abordó su trabajo primero, haciendo uso del *cálculo variacional* y posteriormente de la *programación dinámica*. El problema era tratado como un problema de toma de decisión multietapa. En 1962 publicó una serie de programas de computador para la solución numérica de algunos problemas de optimización. La principal dificultad con la programación dinámica era (y sigue siendo) la complejidad de los sistemas y la necesidad de utilizar aproximaciones.

A continuación en los apartados siguientes se exponen cada una de estas formulaciones.

2.1.2 Cálculo variacional

En este apartado se va a exponer un tipo de formulación utilizada para resolver un problema de control óptimo caracterizado por las especificaciones expuestas en el apartado anterior. Este tipo de formulación se fundamenta en el cálculo de variaciones [Bar-95], proporcionando una ley óptima de control en lazo abierto, es decir, esta formulación no conduce habitualmente a una relación explícita entre el vector de control y el vector de estado.

La resolución del problema de control óptimo aplicando el método del cálculo variacional se va a describir teniendo en cuenta la idea de un planteamiento general [Bry-75] [Lew-86]: se considerará un sistema dinámico continuo definido por la ecuación [1.1] al cual se le asociará la siguiente función de coste.

$$J(x_0, t_0) = \phi\left(x(T), T\right) + \int_{t_0}^{T} L\left(x, u, t\right) dt, \qquad [1.2]$$

donde $[t_0, T]$ es el intervalo de interés. El objetivo del control óptimo es encontrar una ley o trayectoria de control en dicho intervalo que minimice el valor de la función de coste, cumpliendo además, la restricción de estado final x(T) y tiempo final T.

Observando la ecuación [1.2], se puede ver que la función de coste está formada por los siguientes dos términos:

- $\phi(x(T), T)$: se denomina función de coste terminal e indica el coste asociado al estado final, siendo deseable minimizar su valor en el instante de tiempo T;

- $\int_{t_0}^{T} L(x,u,t)dt$: se denomina función de coste de etapa y equivale a una expresión integral cuyo valor se desea minimizar en el intervalo $[t_0, T]$.

La Figura 2-1 muestra la contribución relativa de ambas componentes a la función de coste J. Como se puede observar, el término integral contribuye con un coste siempre creciente cuando $L \geq 0$, ya que para transitar de un estado a otro en un determinado intervalo de tiempo será necesario invertir algún tipo de esfuerzo. Cuando se alcanza el estado final, la curva J experimenta un crecimiento instantáneo, ya que el término ϕ está asociado al estado final.

A partir del Principio del Mínimo de Pontryagin [Pon-86] se puede obtener una ley de control óptimo bajo ciertas restricciones para un problema específico, donde las condiciones de primer orden establezcan la posibilidad de hallar un control óptimo admisible que minimice la función de coste y donde los estados inicial y final puedan ser fijados. En la Tabla 2-1 se indican las ecuaciones que debe satisfacer la solución de un problema de control óptimo[1], siendo H la función Hamiltoniana, definida como:

$$H(x,u,\lambda,t) = L(x,u,t) + \lambda^T f(x,u,t), \qquad [1.3]$$

donde λ es la variable de coestado continua en el intervalo $[t_0, T]$ y que hace máxima, en cada instante t, la función Hamiltoniana.

Conviene destacar que la condición de control especificada en la Tabla 2-1 y denominada *Principio del Mínimo de Pontryagin* [Pon-86], bajo la hipótesis de suavidad de la función H, constituye una condición necesaria y suficiente para obtener la solución al problema del control óptimo para una trayectoria dada en un sistema dinámico,

[1] En la Tabla 2-1, las funciones con asterisco son funciones que satisfacen el criterio de optimalidad.

f. Esta condición será aplicable siempre que el vector de control esté restringido, situación presente en todos los problemas prácticos de control óptimo.

En general, a partir de la formulación anterior tan sólo se pueden resolver analíticamente un reducido número de problemas de control óptimo. Como alternativa, se han desarrollado varios métodos numéricos que, si bien resuelven de forma aproximada problemas no abordables analíticamente, por el contrario, el coste computacional que requieren es enormemente elevado [Ste-86].

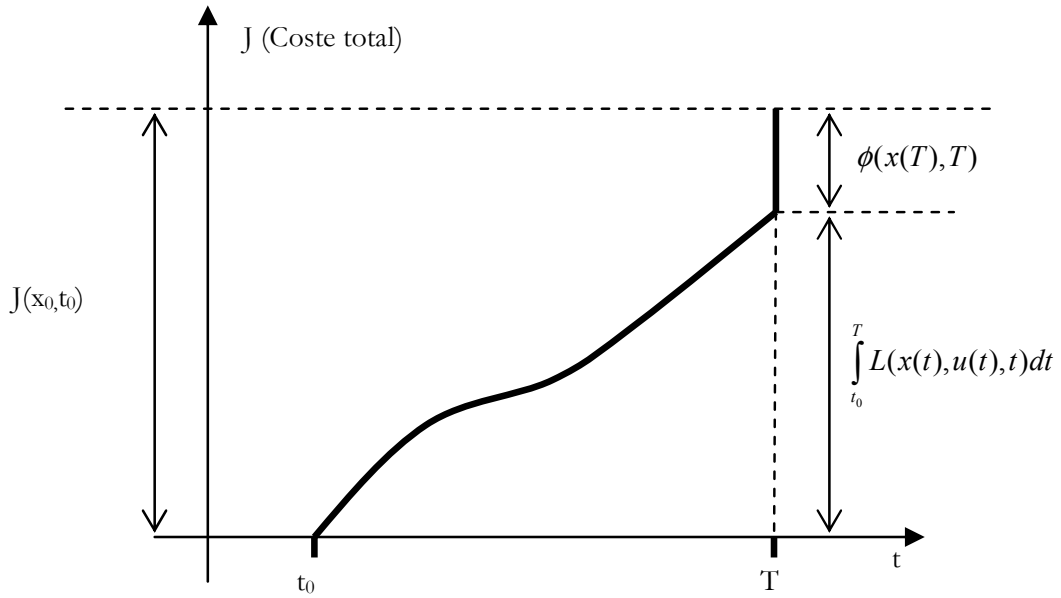


Figura 2-1: Contribución de los términos L y ϕ al coste total, $J(x_o,t)$

La siguiente condición de control, aparte de proporcionar únicamente condiciones necesarias, no es aplicable en general cuando el vector de control solamente puede tomar valores en un conjunto compacto:

$$\frac{\partial H}{\partial u} = 0$$
$$\frac{\partial^2 H}{\partial u^2} \geq 0 \qquad [1.4]$$

Tabla 2-1: Formulación de la ley de control óptimo para un sistema continuo no lineal

Ecuación de estado	$\dot{x} = \left.\frac{\partial H}{\partial \lambda}\right\|_{*} = f\left(x^{*}, u^{*}, t\right),\ t \geq t_0$
Ecuación de coestado	$-\dot{\lambda}^{*} = \left.\frac{\partial H}{\partial x}\right\|_{*} = \left.\frac{\partial L}{\partial x}\right\|_{*} + \left.\frac{\partial f^{T}}{\partial x}\right\|_{*} \lambda^{*},\ t \leq T$
Condición de control	$H(x^{*}, u^{*}, \lambda^{*}, t) = \min_{u \in U} H\left(x^{*}, u, \lambda^{*}, t\right)$
Condición de frontera	$\left(\phi_x + \psi_x^{T}\nu - \lambda^{*}\right)^{T}\Big\|_{T} dx(T) + \left(\phi_t + \psi_t^{T}\nu - \lambda^{*}\right)^{T}\Big\|_{T} dT = 0$

Problemas con ley de control restringida

Debido a que en gran variedad de aplicaciones, el vector de control debe mantenerse dentro de unos límites especificados, es necesario utilizar el *Principio del Mínimo de Pontryagin* [Pon-86] para obtener la ley de control óptima. Si se empleara la condición [1.4] podría ocurrir que la solución óptima no perteneciera al rango de valores permitidos del vector de control.

Dentro de los problemas con ley de control restringida, cabe destacar el control óptimo en tiempo mínimo. Este tipo de problemas reciben el nombre de control *bang-bang*, y se caracterizan porque las acciones de control están normalmente acotadas entre unos límites máximo y mínimo, y el control varía de un extremo a otro en determinados instantes de tiempo [Bry-75]. Por ejemplo, dado un sistema [Bry-75] [Mil-94]:

$$\ddot{s} = u(t) \qquad [1.5]$$

el objetivo será encontrar un control u(t) que pertenezca a un intervalo acotado y que transfiera al sistema desde una posición inicial s_0, a una posición final deseada s_f, en tiempo mínimo. En este caso, la entrada de control es la aceleración a lo largo del camino recorrido. Se puede deducir fácilmente que la solución al problema debe ser un control bang-bang en la variable de entrada u(t). Este es

un ejemplo sencillo donde tenemos una única acción de control y una variable de estado (velocidad). Toda solución basada en bang-bang conlleva un método de representación gráfica denominado *curva de conmutación* [Son-02], por el cual podemos saber el grado de optimalidad del controlador generado. Esta curva se suele denominar también *curva de conmutación en forma de 'S' (S-shaped switching curve)*.

Considérese la ecuación de estado lineal siguiente:

$$\dot{x} = Ax + Bu \qquad [1.6]$$

siendo A y B matrices de coeficientes constantes de dimensión $(n \times n)$ y $(n \times m)$, respectivamente. La función de coste es:

$$J(x_0, t_0) = \int_{t_0}^{T} 1 \, dt \qquad [1.7]$$

donde T es libre, y el estado final $x(T)$ fijo, satisfaciéndose $\Psi(x(T), T) = x(T) - x_T = 0$. Además, el control debe cumplir la restricción:

$$\|u(t)\|_\infty \leq u_{máx} \qquad [1.8]$$

para todo $t \in [t_0, T]$. La restricción anterior significa que el módulo de cada componente del vector u debe tener una magnitud inferior o igual que $u_{máx}$. Esta restricción normalmente se impone en los sistemas de control prácticos.

Entonces, el problema consiste en determinar la ley de control $u(t)$ que, cumpliendo la restricción [1.8], minimiza $J(x_0, t_0)$ satisfaciendo $x(T) - x_T = 0$, y conduce al sistema desde el estado $x(t_0)$ hasta $x(T)$. En estos casos la ley de control óptima siempre tiene la forma de una función constante a tramos, donde los valores posibles para dicha constante son $u_{máx}$ y $-u_{máx}$ [Bry-75] [Lew-86] [Ber-95].

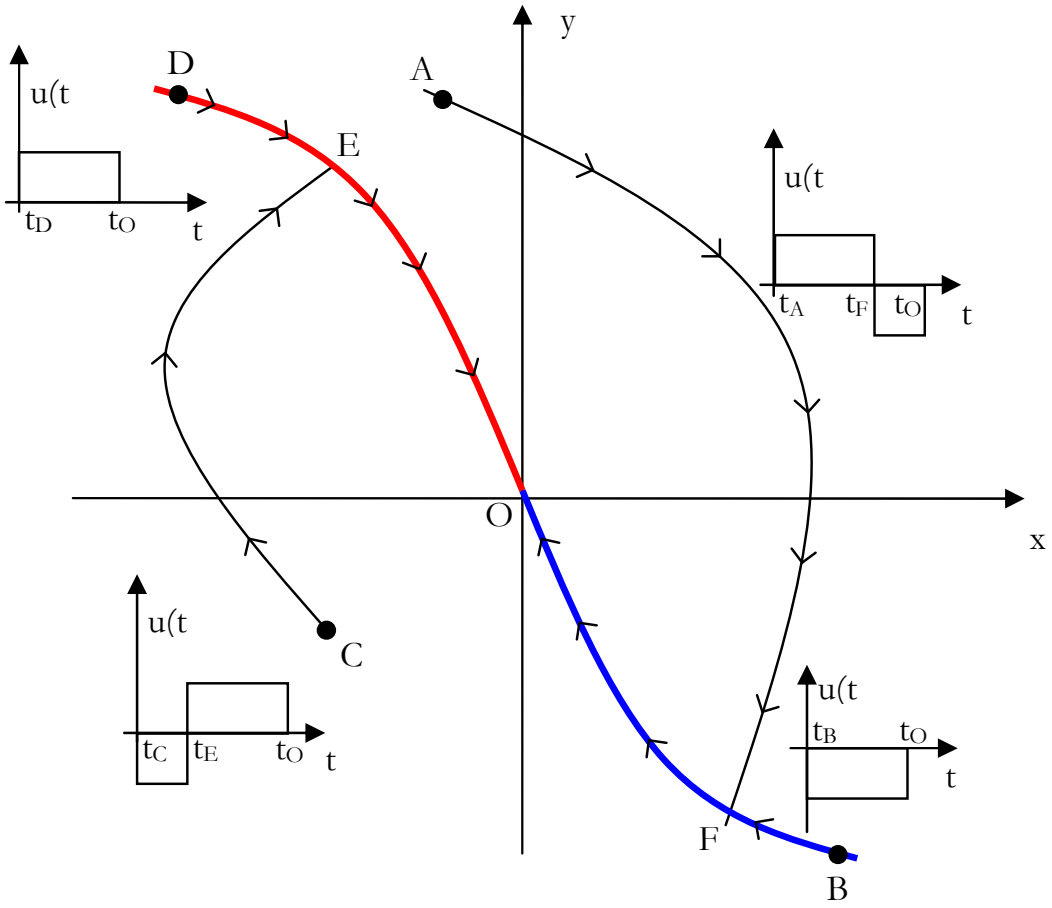


Figura 2-2: Trayectorias óptimas de estado y control para distintos orígenes, en un problema del tipo bang-bang

En la Figura 2-2 se ilustra gráficamente este tipo de control mediante un caso particular, en el cual el espacio de estados del sistema tiene dimensión dos $(n = 2)$, la variable de control u es escalar $(m = 1)$ y el estado final $x(T) = 0$ (origen del plano de estados). Se representan las trayectorias óptimas por las cuales el sistema se desplaza desde cualquier estado inicial hasta el origen del espacio de estados. La curva que contiene los puntos DEOFB representa la denominada curva conmutación indicada anteriormente. Esta curva divide el espacio de estados en dos zonas claramente diferenciadas. Tal y como se puede observar en dicha figura, se distinguen cuatro tipos de trayectorias posibles según el estado inicial. Si el estado inicial se

encuentra sobre la curva de conmutación, la trayectoria describe la curva sin producirse conmutación en la ley de control. Sin embargo, si el estado inicial se halla en una de las dos regiones separadas por la curva de conmutación, el controlador aplicará la ley de control máxima (o mínima, según la región donde esté situado el estado inicial), hasta que la trayectoria alcance la curva de conmutación (tramos AF o CE), instante en el cual la ley de control conmuta al valor mínimo (o máximo respectivamente), manteniendo dicho valor hasta que el sistema alcance el objetivo marcado (tramos FO o EO).

2.1.3 Programación dinámica

Existen problemas cuyas soluciones pueden ser expresadas recursivamente en términos matemáticos, y posiblemente la manera más natural de resolverlos es mediante un algoritmo recursivo [Bar-95]. Sin embargo, la ejecución de la solución recursiva, normalmente provoca una explosión combinatoria y por tanto impracticable. Esta dificultad puede mejorarse substancialmente mediante la programación dinámica.

A veces un problema se puede dividir en subproblemas independientes, que se resuelven de manera recursiva para combinar finalmente las soluciones y así resolver el problema original. El inconveniente se presenta cuando los subproblemas obtenidos no son independientes sino que existe solapamiento entre ellos; entonces es cuando una solución recursiva no resulta eficiente por la repetición de cálculos que conlleva. En estos casos es cuando la programación dinámica puede ofrecer una solución aceptable. La eficiencia de esta técnica consiste en resolver los subproblemas una sola vez, guardando sus soluciones en una tabla para su futura utilización. La programación dinámica no sólo tiene sentido aplicarla por razones de eficiencia, sino porque además presenta un método capaz de resolver de manera eficiente problemas cuya solución ha sido abordada por otras técnicas y ha fracasado. En este tipo de problemas se pueden presentar distintas soluciones, cada una con un valor, y lo que se

desea es encontrar la solución de valor óptimo (máximo o mínimo). La solución de problemas mediante esta técnica se basa en el llamado *Principio de Optimalidad* enunciado por Bellman en 1957 [Bel-57] y que dice:

> *"En una secuencia de decisiones óptima toda subsecuencia ha de ser también óptima".*

Es necesario observar que aunque este principio parece evidente no siempre es aplicable y por tanto es necesario verificar que se cumple para el problema en cuestión. Con esta formulación alternativa se pueden resolver problemas de control de sistemas no lineales de un modo especialmente adecuado para su programación mediante computador [Lew-86] [Ber-95].

Supongamos que un problema se resuelve tras tomar una secuencia $d_1, d_2, ..., d_n$ de decisiones. Si hay n opciones posibles para cada una de las decisiones, una técnica de fuerza bruta exploraría un total de d^n secuencias posibles de decisiones (explosión combinatoria). Sin embargo, mediante la técnica de programación dinámica se evita explorar todas las secuencias posibles por medio de la resolución de subproblemas de tamaño creciente y almacenamiento en una tabla de las soluciones óptimas de esos subproblemas para facilitar la solución de los problemas más grandes.

Más formalmente: Sea E_0 el estado inicial del problema, $D_1 = \{v_{11}, ..., v_{1n_1}\}$ el conjunto de valores posibles para la decisión d_1, E_{1i} el estado del problema tras la elección del valor $v_{1i}, 1 \le i \le n_1$ y S_{1i} una secuencia óptima de decisiones respecto al estado E_{1i}. De acuerdo a estas definiciones y respecto al Principio de Optimalidad de Bellman, una secuencia óptima de decisiones respecto a E_0 es la mejor de las secuencias de decisión $\{v_{1i}, S_{1i}\}, 1 \le i \le n_1$. El mismo razonamiento puede aplicarse a cualquier subsecuencia de decisiones $d_k, ..., d_l$, donde $1 \le k \le l \le n$, partiendo como estado inicial de E_{k-1}. Una solución dinámica para este problema, simbolizado como (k,l),

debe expresarse en términos de los valores de decisión existentes para la decisión d_k y el subproblema $(k+1,l)$, resultante de aplicar cada valor de decisión.

Principio de Optimalidad para sistemas discretos

En un sistema con un número de estados finito S, se puede emplear el Principio de Optimalidad de forma recursiva para resolver un problema de optimización. Para ello, dicho problema debe estar estructurado en un conjunto ordenado de $N+1$ etapas, donde la primera etapa representa el estado inicial del problema (etapa 0), y la última etapa el estado final deseado (etapa N). En general, el estado del sistema en la etapa k ($k=0,1,2,...,N-1$), puede transitar a cualquier estado (incluido él mismo) correspondiente a la etapa $k+1$. Cada transición entre etapas llevará asociado un determinado coste. De este modo, el problema queda estructurado como un grafo, en el cual los nodos y arcos se corresponden con estados y transiciones, respectivamente. Finalmente, el problema de optimización se resuelve mediante la exploración del grafo hacia atrás, comenzando por la última etapa. El Principio de Optimalidad establece la estrategia de exploración del grafo, y se enuncia [Oll-91]:

> *"Una ley de decisión óptima tiene la propiedad de que independientemente de las decisiones previas, las restantes decisiones constituyen una ley de decisión óptima con respecto a los estados resultantes de las decisiones previas".*

Para ilustrar el proceso de resolución basado en el Principio de Optimalidad se puede hacer uso de los denominados *grafos multietapa*. Se define un *grafo dirigido* como un conjunto de *vértices* (también llamados *nodos*) y un conjunto de *arcos*. Si el grafo es ponderado, habrá además un peso asociado a cada arco. Un grafo multietapa es un grafo en el que:

- El conjunto de vértices está particionado en K etapas.
- Todos los arcos se producen entre etapas adyacentes y en el mismo sentido.

- Existe un conjunto de vértices iniciales y un conjunto de vértices finales.

Un grafo multietapa $G(N, A)$ es un grafo dirigido en el que se puede hacer una partición del conjunto de nodos o vértices N en k ($k \geq 2$) conjuntos disjuntos C_i, $1 \leq i \leq k$; donde todo arco del grafo (n, m) es tal que $n \in C_i$ y $m \in C_{i+1}$ para algún $i, 1 \leq i \leq k-1$. El coste de cada arco (i, j) lo proporciona la función, c. Los conjuntos C_1 y C_k tienen un sólo nodo, llamándose al primero *nodo origen (o)* y al segundo *nodo destino (d)*. Si el número de conjuntos C_i es k se dice que es un grafo *k*-etapa. El problema consiste en encontrar un camino de coste mínimo que vaya desde el nodo origen, o, al nodo destino, d. Por las restricciones impuestas, un camino de coste mínimo tendría exactamente un sólo nodo en cada conjunto C_i, de ahí que cada C_i defina una etapa del grafo. También puede decirse que cada arco del camino une una etapa con la siguiente.

A continuación se va a aplicar la programación dinámica a esta situación: Se satisface el Principio de Optimalidad ya que el camino de coste mínimo debe contener subcaminos de coste mínimo; en caso contrario, podrían sustituirse dichos subcaminos por otros mejores, resultando un camino total de coste menor. Además, la solución final será una secuencia de decisiones, y cada decisión depende del resultado de otras. En general, la *i*-ésima decisión consiste en determinar, a partir de un nodo n_i de C_i, un arco que tenga a n_i como nodo origen y a algún n_{i+1} de C_{i+1} como nodo destino, siendo $1 \leq i \leq k-2$. El número mínimo de decisiones para un grafo *k*-etapa es $k-2$ ya que cada camino consta de $k-1$ arcos, y el último queda determinado por la decisión tomada en la etapa anterior. Si $CA(i, j)$ es un camino de coste mínimo, $Coste(i, j)$, desde el nodo origen o, hasta el nodo j de C_i, usando una formulación hacia atrás tenemos:

$$\text{Coste}(i,j)=\begin{cases} i=2\rightarrow \begin{cases} c(o,j), & si(o,j)\in A \\ \infty, & \text{otro caso} \end{cases} \\ 3\le i\le k\rightarrow \min_{l\in C_i-1,(l,j)\in A}\left(c(l,j)+\text{Coste}(i-1,l)\right) \end{cases}$$

De igual forma, podemos analizar el coste mediante una formulación hacia delante: si $CA(i,j)$ es un camino de coste mínimo $\text{Coste}(i,j)$ desde el nodo j, del conjunto C_i, hasta el nodo destino d, entonces:

$$\text{Coste}(i,j)=\begin{cases} i=k-1\rightarrow \begin{cases} c(j,d), & si(j,d)\in A \\ \infty, & \text{otro caso} \end{cases} \\ 1\le i\le k-2\rightarrow \min_{l\in C_i+1,(l,j)\in A}\left(c(l,j)+\text{Coste}(i+1,l)\right) \end{cases}$$

En general, se podrán utilizar indistintamente cualquiera de las dos formulaciones. En la Tabla 2-2 se indica el algoritmo que permite construir la trayectoria de coste mínimo con una formulación hacia atrás (se ha suprimido el primer índice de la variable Coste ya que no es trascendente para el algoritmo). Para estudiar la complejidad de este algoritmo hay que tener en cuenta que la complejidad del primer bucle es $O(n+a)$ siendo n el número de vértices y a, el número de aristas o arcos, y la del segundo $O(k)$; donde k siempre es menor que n. Por tanto, la complejidad total del algoritmo es $O(n+a)$.

Tabla 2-2: Algoritmo que construye la trayectoria de coste mínimo con una formulación hacia atrás

```
algoritmo multietapa (ENTRADA A: arcos; k, n: entero; C: vect[1..n, 1..n]; SALIDA P: vect[1..k] de 1..n)

{El algoritmo construye a partir de un grafo de k etapas, n vértices, conjunto de arcos A y costes C, la trayectoria P de coste mínimo}

variables
     D: vect[1..n] de real
     COSTE: vect[1..n] de real
     j, r: 1..n
principio {multietapa}
     COSTE[1] ← 0
     for (j=2; j<=n; ++j) {
       r ← vértice para el que (r, j) ∈ A y
       COSTE[r] + C[r, j] es mínimo
       if (COSTE[r] + C[r, j] es mínimo) {
          COSTE[j] ← COSTE[r] + C[r, j]
          D[j] ← r
       }
     }
     {Se almacena en P la trayectoria mínima}
     P[1] ← 1, P[k] ← n
     for (j=k-1; j>=2; --j) {
          P[j] ← D[P[j+1]]
     }
fin {multietapa}
```

En la Figura 2-3 se muestra un grafo de 5 etapas con los nodos numerados para ilustrar cómo calcular la trayectoria de coste mínimo haciendo uso del algoritmo anterior. Siguiendo el algoritmo se obtienen los siguientes valores para las variables Coste, D y P:

NODO	1	2	3	4	5	6	7	8	9	10	11	12
COSTE	0	9	7	3	2	9	11	10	15	14	16	16

NODO	2	3	4	5	6	7	8	9	10	11	12
D	1	1	1	1	3	2	2	6	6	8	10

NODO	1	2	3	4	5
P	1	3	6	10	12

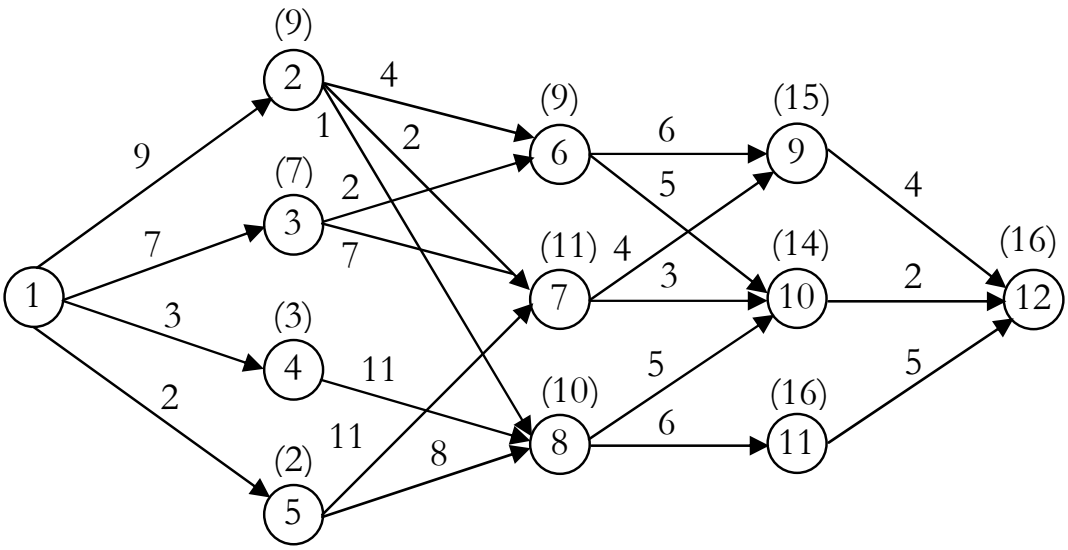


Figura 2-3: Grafo de 5 etapas con nodos numerados

Por tanto, la trayectoria de coste mínimo (16) que calcula el algoritmo para este grafo de 5 etapas está representado en la Figura 2-4. Aunque el algoritmo descrito es relativamente simple, la dificultad de la aplicación de la programación dinámica reside en la necesidad de estructurar el problema en etapas, de tal forma que todos los caminos o transiciones conecten etapas consecutivas (desde la etapa k hasta la etapa k+1). En muchos casos, la dimensión del grafo generado para cumplir la premisa anterior hace inviable el uso de esta técnica.

El Principio de Optimalidad también se puede aplicar cuando el problema está definido mediante un grafo general, a costa de introducir mayor sofisticación en el algoritmo de cálculo del coste óptimo de cada estado del sistema. Hay técnicas, por ejemplo las

basadas en *Cell Mapping* que se estudiarán en el apartado 2.2, que poseen esta característica: construyen un grafo dirigido y hacen uso del Principio de Optimalidad para determinar la ley de control óptima del sistema.

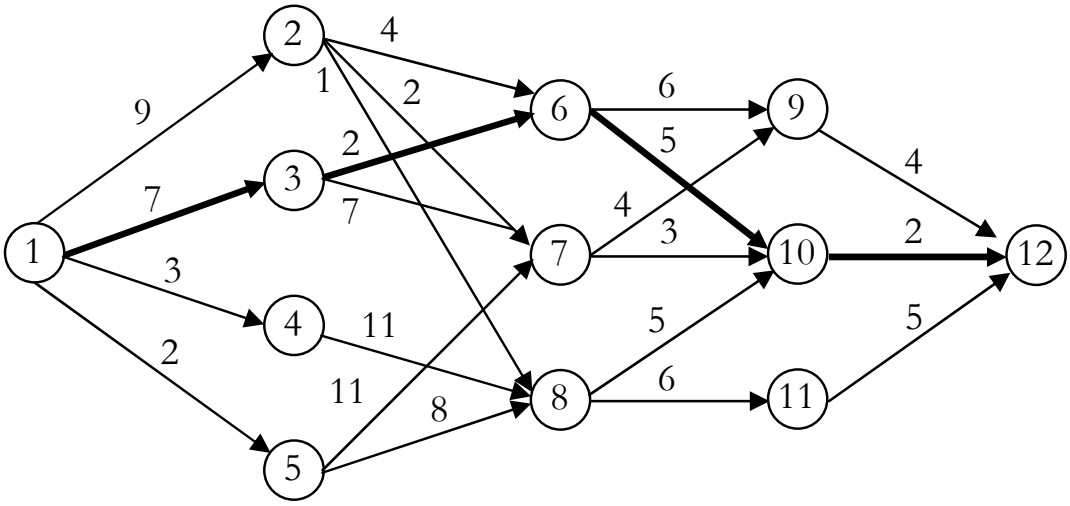


Figura 2-4: Grafo de 5 etapas con trayectoria óptima

Principio de Optimalidad para sistemas continuos

La aplicación de la programación dinámica a sistemas continuos exige la discretización de las ecuaciones diferenciales que modelan el proceso o sistema, así como la cuantificación de las variables de estado, de las acciones de control y del tiempo [Mar-99a] [Mar-99b].

El problema de optimización del comportamiento de sistemas continuos no lineales se basa en el Principio de Optimalidad, por lo que el coste óptimo $J^*(x,t)$, se calcula a partir de costes óptimos obtenidos en instantes de tiempo posteriores. Esto implica que el cálculo del control y coste óptimo se debe realizar operando hacia atrás en el tiempo [Mar-99b]. Para determinar el coste óptimo desde un punto (x, t) hasta el objetivo (x_f, t_f), tan solo hay que calcular dicho coste desde (x, t) hasta $(x+\Delta x, t+\Delta t)$ y sumarle $J^*(x+\Delta x, t+\Delta t)$, donde J^* se ha calculado en la etapa anterior $(t+\Delta t)$ del algoritmo [Mar-99b].

En el apartado 2.2 se describirán dos técnicas de Cell Mapping aplicadas al control óptimo que explotan el Principio de Optimalidad, y por lo tanto, son técnicas de programación dinámica, aunque con un enfoque distinto.

2.1.4 Conclusiones

Al aplicar los métodos variacionales a la resolución del problema del control óptimo, hay que tener presente que:

- Los métodos variacionales suministran los máximos y mínimos relativos de J(u) y no los absolutos.
- La obtención de soluciones de forma explícita es compleja.
- Los valores admisibles para las señales de control normalmente están acotados, lo que hace imposible la determinación de la señal de control óptimo por métodos variacionales.

Para obtener comportamientos óptimos con respecto a determinados criterios se requiere que se mantengan las señales de control en sus valores extremos. Esto sucede especialmente en los problemas de control en tiempo mínimo. El principio del mínimo de Pontryagin constituye una generalización de los resultados alcanzados con ayuda del cálculo variacional para resolver el problema del control óptimo. La diferencia esencial entre los resultados alcanzados con ayuda del cálculo variacional y aquellos que se obtienen con ayuda del principio del mínimo de Pontryagin, reside en que en este último caso se puede definir un espacio de funciones admisibles U(t) para las señales de control u(t). Al mismo tiempo las señales u(t) de control admisibles pueden presentar discontinuidades en un número finito de puntos; con ello se abre la posibilidad de estudiar el control bang-bang, que tanto interés tiene en determinadas aplicaciones prácticas.

De acuerdo con el principio de Pontryagin la elección del control óptimo u(t), debe seleccionarse de manera que garantice el mínimo posible de la Hamiltoniana H, teniendo en cuenta las restricciones

(limitaciones) impuestas sobre los valores admisibles de u(t). La función Hamiltoniana permite evaluar variaciones del criterio J debido a variaciones admisibles e infinitesimales de la señal de control u(t). Sin embargo, esto sólo puede realizarse analíticamente en determinados casos o ejemplos concretos, ya que en la mayoría de los casos prácticos y reales (por ejemplo, el trabajo planteado en esta tesis), no es fácilmente aplicable.

Por otro lado, mediante la programación dinámica, la solución al problema de control óptimo toma la forma de una ley de control, y no de una señal de control como sucedía en la solución del control óptimo mediante métodos variacionales. Sin embargo, no modela de forma adecuada restricciones temporales para problemas de tamaño realista [Hob-88], dando lugar a un tiempo de cálculo excesivo.

Otro aspecto que debe ser destacado a favor de la programación dinámica es su aplicabilidad para realizar un control en lazo cerrado, como consecuencia de disponer de una ley de control. Sin embargo, mediante los métodos variacionales el control que se realiza es en lazo abierto, precisamente por disponer de una señal de control y no de una ley.

2.2 Técnicas de *Cell Mapping*

2.2.1 Introducción

Las primeras técnicas de Cell Mapping fueron introducidas y desarrolladas por C. S. Hsu y R. S. Guttalu a principios de los años 80 [Hsu-80a] [Hsu-80b]. Dichas técnicas surgieron como necesidad de realizar el estudio del comportamiento global de sistemas dinámicos altamente no lineales, en los cuales los métodos analíticos son aplicables únicamente en algunos casos. La carencia de generalidad de los métodos analíticos, unida al extraordinario coste computacional de la utilización directa de métodos de simulación numérica, justifica la aparición de las técnicas de Cell Mapping, que facilitan el análisis eficiente de sistemas no lineales.

Estas técnicas definen un método de análisis computacional global de sistemas dinámicos no lineales [Hsu-80a] [Hsu-87] [Ton-88] [Gut-93] [Zuf-93] [Van-94], basado en la discretización del espacio de estados en *celdas*. El cómputo de un número finito de trayectorias posibilita la construcción de una aplicación celular, que se puede considerar una aproximación en tiempo y espacio discretos del flujo original. En problemas de tiempo continuo, este cómputo se lleva a cabo mediante integración numérica. La flexibilidad y el carácter global de los métodos celulares los hacen aptos para el análisis y la síntesis de diferentes sistemas, que incluyen diferentes tipos de controladores. Otras aplicaciones de estas técnicas se dan en campos como la mecánica [Gut-89] y en problemas de optimización y determinación de raíces [Zuf-90] [Gut-93]. Sin embargo, muchas de las aplicaciones basadas en Cell Mapping sufren la carencia de una caracterización formal de la relación entre las propiedades de la dinámica del sistema original y la de la aproximación celular. En las técnicas de Cell Mapping se toman en consideración los siguientes aspectos:

- *Discretización temporal:* la elección de un tiempo de integración, T, que convierta el flujo continuo en un sistema dinámico discreto es el punto de partida de los métodos

celulares para problemas de tiempo continuo. Posteriormente, el sistema discreto es sometido a un proceso de discretización espacial, que genera un sistema dinámico celular. Cuando los problemas se formulan en tiempo discreto [Dia-93], se omite la elección del tiempo de integración pasando directamente a la discretización espacial.

- *Discretización espacial:* las técnicas de discretización espacial y, en particular, de técnicas celulares van a estar orientadas a estudios globales. Como consecuencia de esta discretización se comete un error y por tanto, no se establece una relación formal rigurosa entre la dinámica celular y la del sistema original. El comportamiento de este error celular está descrito por el denominado *shadowing lemma* [Bow-70] [Bow-78], demostrándose su dependencia lineal con el tamaño de celda. Además, el criterio de elección del tiempo de integración debe garantizar la convergencia de esta aproximación celular.
- *Integración numérica:* cualquier método de integración numérica elegido para la realización de la discretización temporal del sistema continuo, deberá disponer de algún mecanismo de detección de singularidades. Además, la incidencia relativa en el error global de la discretización espacial y la integración numérica, asociadas respectivamente, al tamaño de celda y el paso de integración deberán también contemplarse en el modelo.

Además, el uso de métodos celulares para el análisis global de sistemas linealmente implícitos presenta dificultades específicas. En particular, los puntos de atracción dan lugar a soluciones definidas en un intervalo de tiempo finito, y el intento de integrar numéricamente estas trayectorias más allá del punto de terminación dará lugar a una solución espuria, haciendo inadecuada la técnica de *Simple Cell Mapping* (*SCM* en adelante) [Hsu-80a] [Hsu-87] para estos sistemas. Técnicas adaptativas como *Adjoining Cell Mapping* (*ACM* en adelante)

[Gut-93] [Zuf-93] muestran un mejor comportamiento en ciertos problemas implícitos.

2.2.2 Métodos celulares

Las técnicas celulares aproximan la dinámica de un sistema de acuerdo a la ecuación [1.1], donde la regularidad impuesta a F viene definida por los requisitos del método numérico empleado mediante un sistema dinámico celular

$$z^{k+1} = C\left(z^{(k)}\right), \qquad [1.9]$$

definido en un espacio de estados discreto **S**. Este espacio de estados celular se obtiene como $S = \varphi(\Omega)$. La aplicación φ identifica todos los puntos $x(x_1,...,x_n)$ de un rectángulo *n*-dimensional con un solo vector entero $z = (z_1,...,z_n)$ a través de la relación:

$$\left(z_i - \frac{1}{2}\right)l_i \le x_i - \tilde{x}_i < \left(z_i + \frac{1}{2}\right)l_i, \qquad i = 1,...,n. \qquad [1.10]$$

donde, los parámetros vectoriales $\tilde{x} = (\tilde{x}_1,...,\tilde{x}_n)$ y $l = (l_1,...,l_n)$ son el origen de la región de interés y el tamaño de celda, respectivamente.

De acuerdo a la ecuación [1.1], el espacio de estados S es un conjunto finito si el vector de estado x está acotado en un conjunto Ω. Normalmente, la región de interés de un sistema dinámico, generalmente, se restringe a un subconjunto del espacio de estados euclídeo *n*-dimensional que caracteriza a dicho sistema. Se denominará Ω a este subconjunto y, puesto que, por simplicidad, se establecerá un límite inferior ($x_{mín}$) y superior ($x_{máx}$) a cada variable de estado, Ω adoptará forma de rectángulo *n*-dimensional. Un modo sencillo de construir un espacio de estados celular es dividir Ω en celdas. Es recomendable utilizar también celdas con forma de rectángulo. La partición se realiza dividiendo el intervalo

$\left[x_{i_{mín}}, x_{i_{máx}} \right]$ correspondiente a cada variable de estado x_i (i=1,2,...,n) en M_i intervalos de tamaño l_i, donde:

$$l_i = \frac{x_{i_{máx}} - x_{i_{mín}}}{M_i}, i = 1,...,n \qquad [1.11]$$

Mediante la variable entera z_i, reflejada en la ecuación [1.10], se etiquetan los M_i intervalos a lo largo del eje x_i. En concreto, el intervalo etiquetado por un valor particular de z_i será aquel que contiene todas las x_i que satisfacen la ecuación [1.10].

Por definición, una *n*-tupla $z = [z_1, z_2, ..., z_n]$ se denomina vector celular. Un punto $x \in \Re^n$ con componentes x_i pertenece a la celda $z \in S$ con componentes z_i si, y sólo si, satisface la condición [1.10] para todo i. En sentido inverso, se emplea habitualmente el método del punto central, que asocia la celda z con su punto central x_z^c cuyas coordenadas son $x_i^c = \left(z_i + \frac{1}{2} \right) l_i + x_{i_{mín}}$. El número total de celdas contenidas en Ω, a las que se denominan celdas regulares, es:

$$M = \prod_{i=1}^{n} M_i, \qquad [1.12]$$

y, cada celda se etiqueta con un índice j (j=1,...,M). La región restante del espacio de estados $\Re^n \setminus \Omega$ se considerará como una sola celda, llamada *celda sumidero*, y se identificará con el índice 0.

2.2.3 Técnicas de análisis de sistemas dinámicos

En este apartado se revisarán las técnicas SCM y ACM, las cuales han motivado metodologías de diseño de controladores óptimos.

Técnica no adaptativa: SCM

El método SCM [Hsu-80a] [Hsu-87] se caracteriza por la elección de un tiempo de integración T_s igual para todas las celdas en S. Entre las propiedades principales de SCM se encuentra su naturaleza global y distribuida, así como la existencia de eficientes algoritmos de análisis de la dinámica celular [Hsu-87], particularmente para el cómputo de dominios de atracción invariantes asintóticamente estables (puntos de equilibrio, trayectoria periódica, etc.). En este sentido, es también destacable su capacidad de detectar puntos de equilibrio asintóticamente estables como celdas fijas o como núcleos celulares invariantes y de atracción. En particular, la elección de un tamaño de celda l, suficientemente pequeño y un T_s suficientemente grande garantiza que un punto de equilibrio asintóticamente estable genera un grupo celular invariante. Cuando este grupo se reduce a una única celda de atracción, la aplicación celular se denomina dinámicamente robusta [Dia-98].

Por otra parte, los métodos celulares presentan, en general, el inconveniente de que el coste computacional crece exponencialmente con la dimensión del problema. Las técnicas SCM tienen, además, algunas limitaciones específicas:

- *Invariantes espurios:* las celdas fijas y periódicas espurias son invariantes celulares que no corresponden a ningún punto de equilibrio ni solución periódica de la dinámica original [Hsu-87]. En particular, en regiones de dinámica lenta pueden aparecer celdas fijas espurias.
- *Trayectorias de tiempo finito:* Estas trayectorias son soluciones definidas en un intervalo (a,b), con $(b<\infty)$ y/o $(a>\infty)$. Cuando el intervalo está acotado por la derecha $(b<\infty)$, el intento de integrar numéricamente el sistema más allá del límite *b* generará necesariamente una solución espuria.

Adicionalmente, el error espacial introducido por la aproximación celular debe ser tenido en cuenta.

Técnica adaptativa: ACM

La técnica ACM [Zuf-93] define un criterio para la elección adaptativa del tiempo de integración T(z), asociado a cada celda en S. Este criterio utiliza la siguiente distancia celular:

$$d(z,z') = \max_{i=1,\ldots,n} \left\{ \left| z_i - z_i' \right| \right\}. \qquad [1.13]$$

Dos celdas z, z' se dicen adyacentes si $d(z,z')=1$, y un tiempo de integración adyacente, T, asociado a una celda $z \in S$ y a un instante t_0 es cualquier tiempo t_a tal que:

$$d\left(z(t_0+t_a), z(t_0)\right) = 1 \qquad [1.14]$$

Salvo que una celda contenga un invariante (por ejemplo, algún punto de equilibrio o trayectoria periódica) del sistema, siempre será posible determinar un tiempo de integración adyacente asociado a la misma. Así, partiendo de la celda $z(t_0)$, se puede demostrar que si en el instante t_1 se cumple $d\left(z(t_1), z(t_0)\right) = 0$ y en el instante t_2 $d\left(z(t_2), z(t_0)\right) > 1$, entonces existe al menos un intervalo $I_A \subset (t_1, t_2)$ y $d\left(z(t), z(t_0)\right) = 1 \forall t \in I_A$. Es decir, si la trayectoria del sistema se encuentra en un instante t_1 en la celda origen y en t_2 en una celda no adyacente a dicha celda, para todos los tiempos contenidos en I_A la trayectoria se encuentra en una celda adyacente.

Con objeto de cumplir la condición de adyacencia, el tiempo de integración adyacente se calcula para cada celda de forma adaptativa, al contrario que en el algoritmo SCM, donde el tiempo de integración T_s es fijo durante todo el proceso de integración. De esta manera, se resuelve el problema de determinar el tiempo de integración apropiado. Además, este método admite el análisis del sistema mediante un particionado recurrente del espacio de estados, con el cual se consigue una considerable reducción en el espacio de almacenamiento de memoria [Zuf-93].

En general, el esquema ACM mejora el comportamiento del SCM reduciendo considerablemente el número de celdas fijas espurias, mientras que el requisito de adyacencia limita los saltos asociados a trayectorias de tiempo finito. Sin embargo, ACM presenta un problema de convergencia provocando desviaciones en las trayectorias generadas con respecto a las trayectorias óptimas correspondientes [Mar-99b].

Comparación de las técnicas SCM y ACM

Tanto la técnica SCM como ACM extraen las características del comportamiento de un sistema dinámico. Sin embargo, ninguna está exenta de ciertas limitaciones relacionadas con el proceso de discretización llevado a cabo en el espacio de estados. La razones principales por las que se propuso la técnica ACM fueron las de mejorar las características de SCM con respecto a la elección del tiempo de integración y el espacio de memoria requerido durante el proceso de análisis.

Así, la elección de un tiempo de integración adecuado es muy importante en el caso de aplicar la técnica SCM. Un periodo muy pequeño podría provocar la aparición de falsos grupos periódicos. Además, para aquellos sistemas que por su dinámica sean relativamente rápidos, la técnica SCM podría generar resultados incorrectos independientemente del tiempo de integración empleado. Sin embargo, la técnica ACM resuelve los problemas de la aparición de falsos grupos periódicos y la elección de un tiempo de integración apropiado, imponiendo la condición de adyacencia. A cambio, el coste computacional en la obtención de cada transición se ve incrementado.

Finalmente, en las técnicas de Cell Mapping conviene tener en cuenta el análisis del error cometido al aproximar todos los estados contenidos en una celda mediante un único estado, correspondiente a su punto central. Es posible que en determinados problemas se detecten ciertos errores al aproximar trayectorias continuas mediante trayectorias celulares [Mar-99b]. En definitiva, el error en la

aproximación de trayectorias no tiende a cero cuando el tamaño de la celda se reduce arbitrariamente. Por lo tanto, se puede afirmar que los errores de aproximación pueden llegar a manifestarse de forma diferente tanto en la técnica SCM como ACM.

2.2.4 Técnicas de control de sistemas dinámicos

Las técnicas de Cell Mapping, además de ser muy útiles en el análisis global de sistemas no lineales, adquieren un gran protagonismo entre los métodos de control óptimo para dichos sistemas. Tales técnicas aplicadas en control combinan, por un lado, la aplicación eficiente de los métodos numéricos para integrar sistemas no lineales y, por otro, la potencia del Principio de Optimalidad para efectuar la búsqueda del control óptimo eficientemente. El resultado es un nuevo método de control óptimo, basado en el concepto de espacio de estados celular, capaz de diseñar controladores óptimos de sistemas altamente no lineales de forma muy eficiente.

Método CSCM

En 1985 se publicó el primer controlador óptimo basado en la técnica SCM [Hsu-85]. Posteriormente, se han abierto diversas líneas de investigación orientadas al diseño de controladores basándose en el controlador original de Hsu.

Este método denominado CSCM se basa en la técnica SCM introducida por Hsu. Utiliza un algoritmo de búsqueda basado en el Principio de Optimalidad de Bellman para encontrar el nivel de control óptimo correspondiente a cada celda. El elemento central del algoritmo es la función de coste, que precisa ser discreta para dirigir la búsqueda según riguroso orden en coste creciente.

El método de control óptimo de sistemas continuos no lineales basado en SCM (CSCM), utilizando un algoritmo de búsqueda como el expresado anteriormente, se desglosa en tres partes. En primer lugar, se discretizan las variables continuas del proceso, a continuación se genera un grafo explícito del sistema y, finalmente, se

ejecuta el algoritmo de búsqueda que explora el grafo e introduce en la tabla de control óptimo los intervalos de control óptimo encontrados. A continuación, se describe el método paso a paso:

1. Discretizar las variables de estado del sistema para obtener un espacio de estados celular. Establecer la región de interés y seleccionar el criterio de optimalidad (tiempo mínimo, distancia más corta, energía mínima, etc.).
2. Discretizar las acciones de control, u, normalmente en un conjunto de niveles equiespaciados.
3. Seleccionar un conjunto de intervalos de tiempo (τ).
4. A partir del centro de cada celda de la región de interés, realizar la integración numérica de las ecuaciones del sistema para todos los pares (u, τ), almacenando la celda imagen asociada a cada integración.
5. Partiendo de un conjunto de celdas objetivo, realizar una búsqueda exhaustiva de las trayectorias de control óptimo, siguiendo el Principio de Optimalidad de Bellman.
6. Construir la tabla de control óptimo discreta, asignando a cada celda el par acción de control e intervalo de tiempo óptimo, obtenido por el algoritmo de búsqueda.

La estrategia empleada por la técnica SCM consiste en generar, para cada celda, un conjunto de transiciones, entre las cuales se elegirá la transición óptima mediante un algoritmo de asignación ordenada de costes acumulados. Para asegurar una aproximación aceptable de la ley de control óptima, generalmente es necesario generar un elevado número de transiciones, ya que el conjunto de intervalos de control que genera tales transiciones se establece previamente, aplicándose dichos intervalos sin tener en cuenta el comportamiento dinámico local del sistema. Siguiendo este procedimiento, se genera gran cantidad de transiciones que no aportan información útil para la determinación del control óptimo, produciéndose una pérdida de eficiencia en el algoritmo SCM.

En relación con los problemas de tiempo final fijo, mediante la técnica SCM, la solución computacionalmente más eficiente es aquella en la que se amplía la dimensión del sistema con la variable tiempo, manteniendo el mismo algoritmo de búsqueda [Mar-99b].

Método CACM

Debido a las limitaciones presentes en los controladores CSCM, en 1999 se empezaron a estudiar y a diseñar los primeros controladores óptimos basados en la técnica adaptativa celular ACM [Mar-99a] [Mar-99b]. En 2003 se aplicó este tipo de controladores en un entorno de simulación para encontrar leyes del movimiento óptimo de un vehículo no holónomo [Mar-03].

Este método utiliza la propiedad de adyacencia, estableciendo que la distancia (d-*k*) entre una celda y su celda imagen sea un número entero *k* igual o mayor que 1. La distancia entre dos celdas z y z' se define como:

$$d_k(z,z') = \max_{i=1,\ldots,n} \left\{ \left| z_i - z_i' \right| \right\} = k \qquad [1.15]$$

Dos celdas z, z' se dicen adyacentes si, y solo si,

$$d_k(z,z') = k, \text{ con } k = 1,2,\ldots \qquad [1.16]$$

Una característica que diferencia este método del anterior y que supone una de sus principales ventajas, es que el método CACM efectúa la búsqueda del control óptimo y la integración numérica de las ecuaciones del sistema de forma simultánea. Esto es debido a la propiedad de adyacencia entre celdas y sus imágenes asociadas, la cual posibilita la implementación de un algoritmo de búsqueda altamente eficiente que, en sistemas de dimensión elevada, resulta imprescindible.

Además, el problema de convergencia intrínseco a la técnica ACM en cuanto a que las trayectorias celulares que se obtenían no convergían a las óptimas cuando el tamaño de celda tendía a cero, queda

solucionado con este método aumentando la longitud de las transiciones con respecto al tamaño de las celdas [Mar-99b].

La estrategia adoptada por la técnica CACM hace uso del comportamiento dinámico local del sistema para generar una transición por cada nivel del vector de control. Por la condición de adyacencia, únicamente se permiten transiciones entre celdas adyacentes. Como consecuencia, con CACM se determina el coste óptimo de una celda como suma de dos términos:

- coste incremental asociado a la transición con una celda adyacente.
- coste acumulado asignado a dicha celda adyacente.

Es decir, la asignación de costes óptimos empleando la técnica CACM se propaga, partiendo de un conjunto objetivo, a través de celdas adyacentes. Esta técnica utiliza las siguientes estructuras de datos para organizar la información disponible y almacenar resultados parciales del proceso de exploración:

- *Tabla de Control Óptimo (TCO):* contiene las celdas que pertenecen al conjunto objetivo extendido, es decir, las celdas a las cuales se les ha asignado un nivel de control óptimo. En cada iteración del algoritmo, se introduce una celda en la TCO. Cuando finaliza la búsqueda, la TCO contiene todas las celdas controlables del sistema (véase apartado 3.2.4), junto con sus correspondientes leyes de control óptimas.
- *Estructura de Celdas Frontera (ECF):* posee toda la información sobre controles y costes asociados a las celdas frontera contenidas en la TCO. Para cada celda frontera se almacena la siguiente información: celda imagen, nivel de control, tiempo de adyacencia correspondiente a cada transición y los resultados parciales del proceso de búsqueda.

- *Lista Candidatas (LC):* contiene las celdas origen de las transiciones cuyas celdas imagen pertenecen al conjunto objetivo extendido (TCO). Además, las celdas están ordenadas por coste acumulado, siendo la primera celda de la lista, la celda con menor coste acumulado. En cada iteración del algoritmo, la primera celda es la elegida para expandir sus nodos.

El procedimiento general para la implantación del método CACM es el siguiente:

1. Se incluyen las celdas del conjunto objetivo en la LC y en la ECF (estas celdas son las únicas con coste cero).
2. Se comprueba la condición de parada. El algoritmo termina cuando la LC está vacía, lo cual significa que no hay transiciones de celdas exteriores a celdas interiores de la TCO. En caso contrario, se extrae la primera celda de la LC, la cual se denominará celda en proceso (z_p).
3. Se introduce la celda z_p en la TCO junto con su control óptimo correspondiente.
4. Se localiza z_p en la ECF, identificando sus celdas adyacentes que no hayan sido analizadas todavía.
5. A partir del centro de cada celda sin analizar, se integran numéricamente las ecuaciones del sistema para todos los niveles de control, durante el intervalo de tiempo adyacente, siempre que sea posible. A continuación, se almacena en la ECF las celdas imagen y su información correspondiente.
6. Todas las celdas origen que tengan como imagen a z_p se insertan en la LC en orden creciente según el coste acumulado. Si una celda origen ya se encuentra en la LC, se comparan los costes derivados de los dos caminos posibles, actualizando el camino con menor coste.
7. Se elimina de la ECF, tanto la celda z_p como toda su información asociada y se salta al punto 2.

Una característica asociada a este método de control óptimo es la capacidad de funcionar, sin pérdida de optimalidad, a una frecuencia de muestreo fija, habitualmente impuesta por requisitos físicos en los sistemas de control digital.

Con la técnica CACM, los problemas de tiempo final fijo también son abordables aumentando, al igual que ocurría con la técnica CSCM, la dimensión del sistema con la variable tiempo [Mar-99b].

2.2.5 Ejecución de los controladores CSCM y CACM

Independientemente de la naturaleza del controlador, bien sea CSCM o CACM, el controlador actualiza la ley de control óptima en cada periodo de muestreo, a partir de un estado inicial x_0 del siguiente modo:

1. Encontrar la celda z_0 que contiene a x_0.
2. Extraer de la tabla de control óptimo el vector de control deseado.
3. Con x_0 como estado inicial y bajo la acción del vector de control extraído, el sistema se desplazará a x_1 cuando haya transcurrido el tiempo de muestreo.
4. Encontrar la celda z_1 correspondiente a x_1.
5. Si z_1 pertenece al conjunto objetivo el algoritmo finaliza. En caso contrario, se buscará en la tabla de control óptimo la acción de control correspondiente a z_1.
6. El proceso continuará hasta que el sistema alcance el conjunto objetivo, con lo cual el control óptimo del sistema habrá concluido.

2.2.6 Efectos de discretización: *uniforme vs. no uniforme*

Las técnicas de discretización [Kus-92] basadas en métodos de elementos-finitos (EF) o diferencias-finitas (DF) aplicadas a un espacio de estados con rejillas uniformes son ampliamente utilizadas, proporcionando buenos resultados de convergencia, bien usando soluciones analíticas [Cra-83] [Gba-91] [Cra-92] o probabilísticas [Kus-92] [Dup-98]. Sin embargo, con este tipo de discretización, cuando el tamaño de rejilla se hace muy pequeño, el coste computacional aumenta tanto que lo hace poco práctico, especialmente si el espacio de estados se caracteriza por ser de una alta dimensión (a partir de 4 variables de estado).

Por otro lado, existe la posibilidad de considerar una discretización no uniforme o variable [Mun-92] para aproximar la función de coste y el control óptimo. De este modo, se pueden comparar de una forma experimental los resultados obtenidos utilizando diversos criterios de división. En [Mun-92] se parte de una rejilla con un tamaño grueso y determinadas áreas del espacio de estados se va dividiendo en rejillas cada vez más finas de acuerdo a un criterio determinado. A veces, por ejemplo, un criterio puede aproximar de forma muy precisa la función de coste pero el algoritmo requeriría un coste computacional excesivamente alto cuando la función es discontinua. Además, normalmente las singularidades de la función de coste no están localizadas en las mismas áreas de aquellas del controlador óptimo, por lo que una buena aproximación de la función de coste en algunas áreas quizás no se necesite si esto no tiene ningún impacto en la calidad del controlador. En otras ocasiones, el criterio de división podría tener en cuenta la política y dividir siempre y cuando la política óptima pueda cambiar. En este caso el problema reside en que para efectuar las divisiones se consideran características de celdas desde un punto de vista individual y no global. Sin embargo, se pueden implementar criterios un poco más complejos que consideren la influencia de un estado a la función de coste en otros estados.

2.2.7 Discretización y distancia de transición

Aplicando la técnica ACM se crea un conjunto de transiciones exclusivamente entre celdas adyacentes [Zuf-93]. La duración del tiempo de integración de cada transición se determina adaptativamente con objeto de implementar la propiedad de adyacencia. Esta propiedad proporciona mejoras sustanciales con respecto a otras técnicas (por ejemplo CSCM) que aun basándose en Cell Mapping no tienen en cuenta ninguna condición de adyacencia.

Existe una limitación en cuanto a la selección de transiciones óptimas entre varias existentes con las mismas celdas origen e imagen. Esta limitación aparece en la técnica CSCM. En la técnica CACM, para minimizar los errores de discretización, se han seleccionado las transiciones que cubran una distancia similar, es decir, seleccionar aquellas que queden más cerca del punto central de las celdas adyacentes. Para ello, el tiempo de integración, T_i, se selecciona de acuerdo a:

$$1-\delta < d\left(x_i, \phi\left(x_i, T_i\right)\right) < 1+\delta \qquad [1.17]$$

donde:

$$d\left(a,b\right) = \max_j \frac{\left|a_j - b_j\right|}{h_j}, \qquad [1.18]$$

con $\delta > 0$ (típicamente, $\delta = 0.05$). El tamaño de celda es h, y $\phi\left(x_i, T_i\right)$ es el estado en el instante T_i de la transición originada en el estado x_i.

En general el tamaño de celda y la distancia de adyacencia que se considere son dos criterios de decisión que van a influir directamente en la controlabilidad del propio controlador [Mar-03] (véase apartado 3.2.4). Hay que tener en cuenta que si el número de celdas que se fije es bajo y por lo tanto el tamaño de celda por eje no es excesivamente pequeño, la distancia de adyacencia deber mantenerse en valores bajos (1, 2). Sin embargo, cuando se opta por tener una rejilla muy

fina con un número elevado de celdas, entonces para mantener una optimalidad y una controlabilidad alta, la distancia de adyacencia debe ser alta (3, 4, 5).

2.2.8 Ejemplos

Control del motor de tracción de un vehículo

Debido a que la aplicabilidad de cualquiera de las técnicas de Cell Mapping descritas en los apartados anteriores requiere un modelo matemático del sistema durante la fase de aprendizaje, para este ejemplo se ha procedido a caracterizar un motor de continua, como podría ser el instalado en la plataforma móvil utilizada en la investigación de esta tesis. Para ello, se propone un modelo basado en las siguientes ecuaciones diferenciales:

$$\theta' = \omega$$
$$\omega' = \frac{-\omega + (5.0 * V)}{\tau} \qquad [1.19]$$

donde, θ es el ángulo recorrido por el eje del motor en radianes y ω es la velocidad en rad/s del eje del motor. El tiempo de integración utilizado ha sido 3 ms. Los rangos de valores y discretización de las dos variables de estado empleadas son:

θ : Rango: -100 ≤ x_1 ≤ 100 rad ⇒ Nº de celdas : 101.

ω: Rango: -50 ≤ x_2 ≤ 50 rad/s ⇒ Nº de celdas : 101.

La velocidad angular máxima se corresponde a una velocidad lineal del coche igual 1.5 m/s. En [1.19] sólo se utiliza una acción de control que se corresponde con la tensión aplicada al motor.

V: Rango: -10 V. y 10 V. ⇒ Nº de acciones: 3 {-10, 0, 10}

En la Figura 2-5 se representan gráficamente las trayectorias que recorrería el motor para al alcanzar el objetivo $\{\theta = 80,\ \omega = 0\}$ desde

el estado origen $\{\theta = -80,\ \omega = -40\}$. Se han indicado tres curvas, una para cada constante de tiempo τ, del motor: negra ($\tau = 0.1$s.), azul ($\tau = 0.3$s.) y roja ($\tau = 0.6$s.).

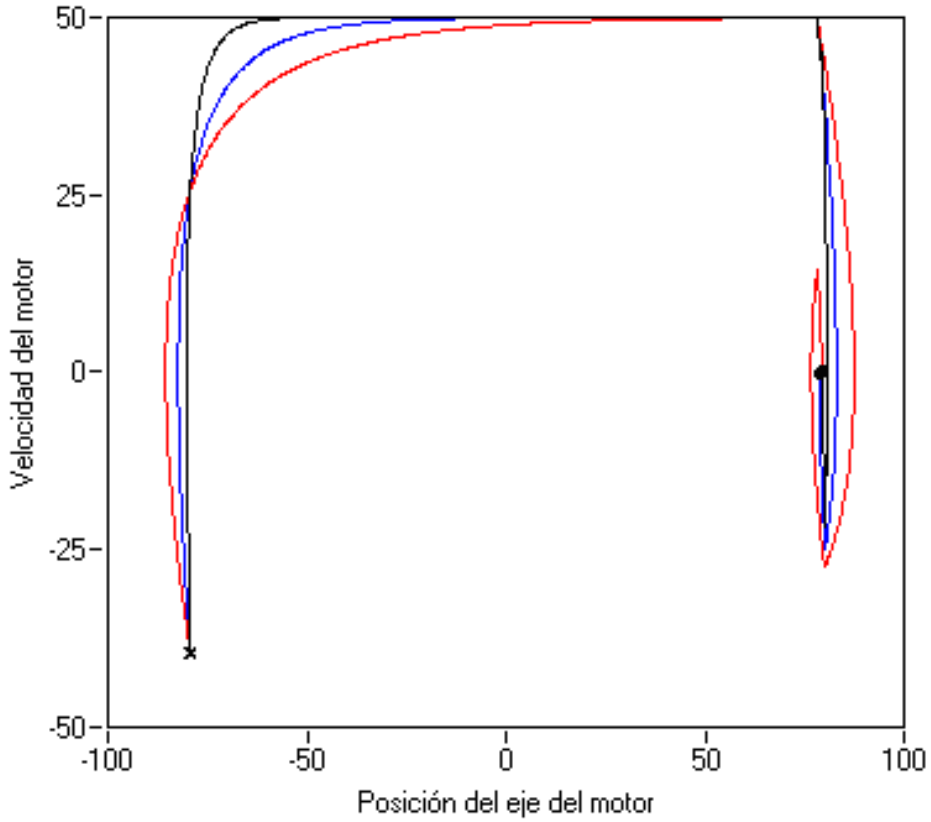


Figura 2-5: Trayectorias recorridas por el motor para diferentes constantes de tiempo

Control de un vehículo sobre la colina

La descripción detallada de la dinámica de este problema está descrita en [Moo-95]. Se trata de un problema de dos dimensiones, donde las variables de estado son la posición y velocidad del coche. La acción de control $\overline{F}$, a través de la cual se va a poder empujar el coche tendrá dos valores: uno positivo y otro negativo. En total se tendrán hasta 3 fuerzas ejercidas sobre el vehículo, además de $\overline{F}$: el propio peso, $\overline{P}$ y la resistencia, $\overline{R}$ sobre la colina ofrecida en el movimiento del coche.

Este problema se ha resuelto utilizando la técnica de Cell Mapping y aplicando la misma al siguiente modelo matemático, utilizando como tiempo de integración 40 ms:

$$x' = v$$

$$v' = \frac{F}{M\sqrt{\left(1+H'^2\right)}} - \frac{gH'}{(1+H'^2)} \qquad [1.20]$$

donde,

x, es la posición del coche.

v, es la velocidad del coche.

M, es la masa del coche. En este ejemplo se toma $M = 1$.

g, es la fuerza de gravedad.

H', es una función que depende de x:

$$H' = \begin{cases} x(x+1), \text{si } x < 0 \\ \dfrac{K_1 x}{\sqrt{1+K_2 x^2}}, \text{si } x \geq 0 \end{cases}$$

$$K_1 = 1 \qquad [1.21]$$

$$K_2 = 5$$

Los datos de entrada para la aplicabilidad de la técnica Cell Mapping son:

$$-1 < x < 1, \ n^o \text{ de celdas: } 131$$

$$-4 < v < 4, \ n^o \text{ de celdas: } 131 \qquad [1.22]$$

$$F \begin{cases} -4 \\ 4 \end{cases}$$

Observando la Figura 2-7 y viendo las trayectorias recorridas por el vehículo para alcanzar el objetivo deseado $(x = 1, v = 0)$, desde diferentes estados origen, se puede afirmar que los datos de entrada indicados en [1.22] son válidos para generar un control óptimo sobre el propio vehículo. En función del número de celdas asignadas a cada una de las dos variables, tendremos más o menos precisión en los movimientos del coche hacia el objetivo. Evidentemente, cuantas más celdas, mayor precisión pero también mayor tiempo de cómputo durante la fase de aprendizaje y una mayor necesidad de memoria para el almacenamiento de las transiciones del espacio de estados considerado. Al fin y al cabo, estas transiciones se corresponden con el conocimiento del sistema controlado (colina y vehículo). Por otro lado, y otro factor importante a la hora del aprendizaje y generación de trayectorias es el tiempo de muestreo. En este caso y con un valor de 40 ms. se obtiene un resultado que se considera aceptable debido a que la controlabilidad obtenida está por encima del 70% (véase apartado 3.2.4). En la misma figura se distinguen dos zonas claramente diferenciadas: una en color cian y otra de color amarillo. Cuando el coche se encuentra dentro de la primera, se aplica una acción de control F, negativa y positiva en el otro caso.

El coche debe alcanzar la cima de la colina tan rápido como sea posible y pararse allí. Se supone que el coche no puede subir la pendiente sin velocidad.

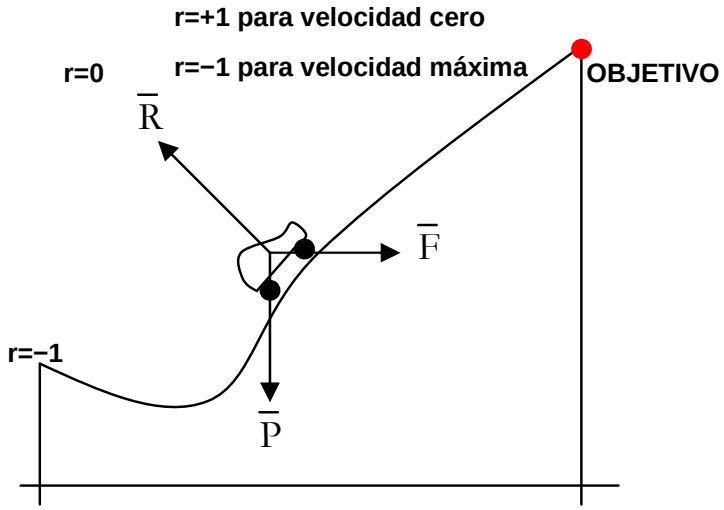


Figura 2-6: Control del vehículo sobre la colina.

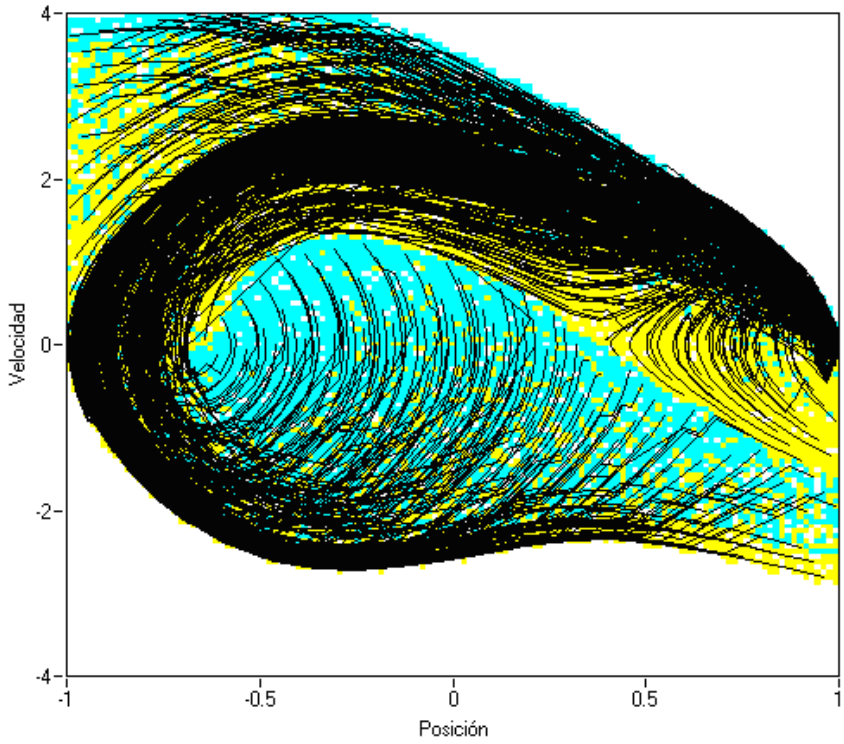


Figura 2-7: Solución al problema del vehículo sobre la colina. Se muestran diversas trayectorias generadas para alcanzar el objetivo situado en x=1 y v=0 desde diversos estados origen. Se observan dos zonas: una de color cian y otra de color amarillo. Cuando el coche se encuentra dentro de la primera, se aplica una acción de control, F, negativa y positiva en el otro caso.

2.2.9 Conclusiones

En este apartado 2.2 se han revisado las metodologías de diseño de controladores óptimos basados en las técnicas de Cell Mapping. Esto ha servido para encuadrar estas técnicas dentro del marco de las técnicas de optimización, y en particular, dentro de las técnicas derivadas de la Programación Dinámica.

La arbitrariedad que presenta la técnica CSCM en la selección del conjunto de tiempos de integración, sin un criterio adaptado al sistema que se desea controlar, se sustituye por una nueva metodología de diseño de controladores óptimos denominada CACM [Mar-99a] [Mar-99b]. Esta técnica impone la condición de adyacencia estableciendo un criterio para el ajuste automático de los tiempos de integración.

Con la introducción de la propiedad de adyacencia en CACM, la exploración del espacio de estados presenta dos mejoras con respecto a CSCM:

1. Se procesan exclusivamente las celdas situadas en la frontera del objetivo, mientras que en SCM se procesaría el volumen total de celdas contenidas en el objetivo.
2. Se permiten simultanear la obtención del Cell Mapping con el proceso de búsqueda.

El número de niveles de control sólo afecta en CACM al tiempo de cálculo, en cambio, en CSCM afecta tanto al tiempo de cálculo como al espacio de memoria requerido.

En espacio de estados continuos tanto el método CSCM como el CACM presentan una alternativa a la Programación Dinámica convencional, ya que consiguen una buena aproximación del comportamiento óptimo del sistema con un menor coste computacional.

En este capítulo se han presentado dos problemas de control óptimo a modo de ejemplo, que se han resuelto utilizando estas técnicas de

Cell Mapping particularizados para un tiempo de integración diferente para cada caso y donde los resultados obtenidos han sido los esperados. En el primer ejemplo del control del motor de tracción, dado que se ha hecho un control óptimo en tiempo mínimo, la gráfica representada debe mostrar como el motor tiene que ir acelerando hasta alcanzar su velocidad máxima para mantenerse ahí e ir reduciendo la misma hasta llegar al objetivo (típico problema de bang-bang). Se puede observar como a mayor valor de la constante de tiempo, al motor tarda más tiempo en alcanzar su velocidad máxima y pararse.

En el segundo ejemplo, la gráfica representada en la Figura 2-11 refleja mediante dos colores el signo que adquirirá la fuerza de empuje del vehículo para alcanzar el objetivo planteado. Además, se trazan algunas trayectorias desde distintos estados origen viendo como llegan al objetivo, aplicándose en cada estado la fuerza que corresponda de acuerdo a la tabla de control óptimo generada.

2.3 Aprendizaje por refuerzo

2.3.1 Introducción

En este apartado se aborda una solución desde el punto de vista computacional para que un agente[2] aprenda a través de una interacción con su propio entorno. Esta solución está centrada en un aprendizaje del entorno con el propósito de alcanzar un determinado objetivo respetando ciertos criterios para su control (por ejemplo, tiempo mínimo).

El denominado *aprendizaje por refuerzo* (RL en adelante) podría definirse como aprender lo que hacer en cada momento con el propósito de maximizar una función de recompensa. Al agente no se le dice qué acciones debe ejecutar sino que debe averiguarlas a partir de las recompensas recibidas del entorno en cada interacción con él. En la mayoría de casos, las acciones pueden afectar no sólo a la recompensa inmediata sino también al siguiente estado que pueda tomar el sistema. Una peculiaridad de RL es su capacidad para considerar globalmente un sistema que ha de ser dirigido hacia un objetivo concreto interactuando con un entorno en principio totalmente desconocido. Precisamente esta característica difiere considerablemente con la programación dinámica descrita en el apartado 2.1.3, donde el problema general se descomponía en subproblemas.

Existen dos tipos de aprendizajes: *supervisado* y *por refuerzo*. Mediante el primero se trata de aprender a partir de ejemplos proporcionados por algún supervisor externo que tenga conocimiento del entorno. Sin

[2] El término *agente* es uno de los más empleados en la actualidad. Existen múltiples definiciones para este término. Para esta tesis se ha escogido por su claridad y concisión la de Pattie Maes [Mae-95]. Según esta definición, un agente es un sistema computacional (controlador), que habita en un entorno complejo y dinámico, con la capacidad de percibir y actuar autónomamente sobre dicho entorno, y de esta manera es capaz de cumplir un conjunto de objetivos o de llevar a cabo ciertas tareas para las cuales fue diseñado.

embargo, carece de las propiedades o características necesarias para aprender a través de la interacción con el propio entorno. Hay que tener en cuenta que en los problemas interactivos no es práctico ni sencillo obtener muestras o ejemplos del comportamiento deseado que sean extrapolables a todas las situaciones en las que el agente tenga que actuar.

El RL puede ser visto como una metodología de aprendizaje no supervisado en la cual se asume que el agente no tiene conocimiento previo de su entorno. El agente explora el entorno y ajusta constantemente su política de acciones de acuerdo al objetivo planteado, asumiendo que dicho entorno es dinámico (aunque estacionario), por lo que la exploración nunca terminaría. En el transcurso del tiempo, el agente explota la política de acciones que ha aprendido para actuar de forma óptima. Sin embargo, debe seguir explorando nuevas acciones que pueden generar comportamientos óptimos. Por lo general, la curva de aprendizaje indica que el agente debe explorar más al principio y que con el transcurso del tiempo debe de explorar menos y explotar más la política de acciones aprendida. Ocurre que el intercambio óptimo entre exploración y explotación es una tarea difícil, ya que la exploración es la clave del aprendizaje de la política de acciones óptima y la explotación es la clave del comportamiento óptimo. En [Sut-98] se mencionan algunas aproximaciones a este problema, por ejemplo: dada cierta probabilidad escoger acciones de exploración aleatorias, usar heurísticas para escoger acciones de exploración, favorecer acciones que lleven al agente hacia estados no visitados recientemente, separar la fase de exploración de la fase de explotación, que la exploración sea guiada por un instructor, etc.

Los conceptos de RL comentados anteriormente se pueden ver gráficamente en la Figura 2-8, donde se muestra cómo un agente autónomo analiza el entorno mediante sus sensores, y actúa sobre él en un instante t, mediante una acción de control a_t. Cada vez que el agente actúa con una acción, el entorno cambia su estado inicial, s_t, a un nuevo estado s_{t+1}, provocando una transición de estados. Un

entrenador puede entonces asignar una recompensa al agente en función de la bondad del estado alcanzado en el entorno bajo la perspectiva de los objetivos que se pretenden conseguir. A continuación, el agente revisará su conjunto de reglas para determinar cuál se adapta a la nueva situación del entorno y en el caso de que haya varias posibilidades, elegirá aquella con una mayor recompensa acumulada. Con esta idea es fácil entender que la mayoría de estos algoritmos suficientemente entrenados acaban creando un homomorfismo, es decir, una relación uno a uno, entre todos los posibles pares estado-acción y un valor de aptitud.

Debido a la similitud entre esta técnica y los métodos de aprendizaje por *ensayo y error*, en los que el comportamiento de un animal ante una determinada situación se modifica en función del consiguiente refuerzo, Mendel y McLaren [Men-70] lo denominaron aprendizaje mediante refuerzo. En [Sut-84] [Bar-85a] [Bar-85b] [Bar-87] [And-89] se ha estudiado de forma extensa esta metodología y en [Wil-87] [Wil-88] [Gas-99a] se han desarrollado diferentes métodos de aprendizaje por refuerzo. Todos ellos incluyen una característica común, la obtención de la información para el aprendizaje a partir de la perturbación activa de las salidas del controlador.

El RL necesita en general de muchas iteraciones con el entorno para encontrar la convergencia a una política óptima. Esto se debe a que los ajustes que se hacen en la política de acciones durante la exploración son pequeños con respecto al número de estados del sistema. En teoría, cada estado del sistema debería ser visitado infinitamente a menudo para que el algoritmo encontrara convergencia a una política de acciones óptima. Existen algoritmos de rápida convergencia para problemas particulares pero todavía no existe una solución general a este problema. La rapidez de convergencia es un área activa de investigación.

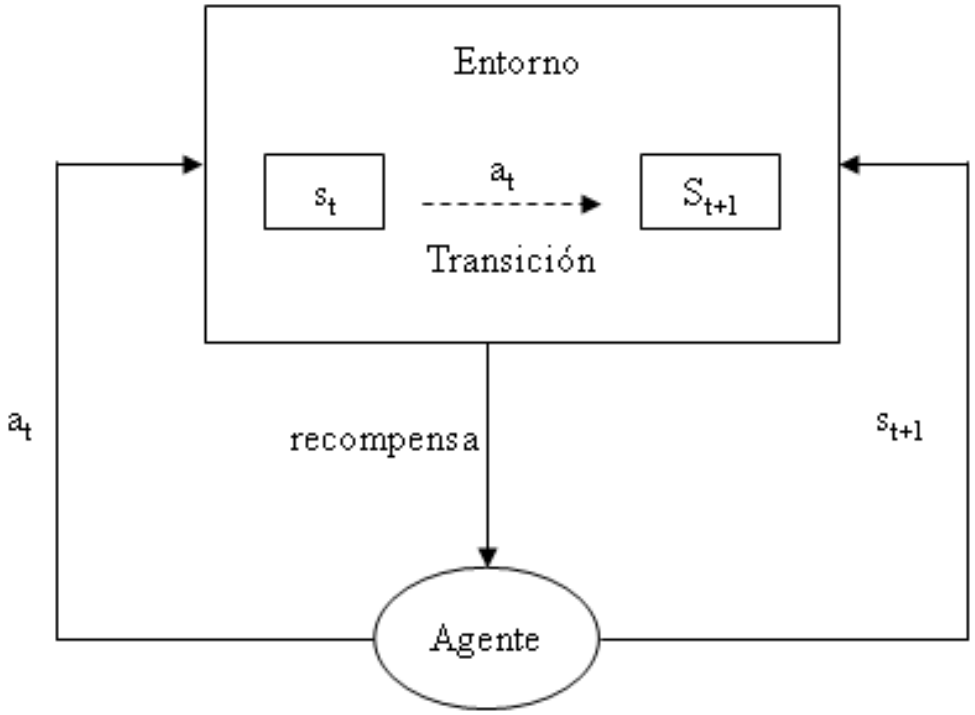


Figura 2-8: Esquema de aprendizaje por refuerzo

En esta metodología de aprendizaje por refuerzo, el agente aprende una política de acción recibiendo recompensas (refuerzos) del entorno cada vez que realiza una acción. Las recompensas pueden ser más o menos favorables en función de las transiciones producidas. Si un agente aprende la política de acciones óptima dada una estructura de recompensas y luego esa estructura cambia (el agente es puesto en otro entorno), el conocimiento del agente se torna obsoleto y tiene que aprender nuevamente una política desde cero, es decir, no existe transferencia del conocimiento aprendido.

2.3.2 Elementos utilizados en RL

En la técnica de RL se distinguen los siguientes elementos:

- *Política*: define la manera de comportarse el agente en un momento dado. Se podría considerar que la política consiste en asociar los estados percibidos del entorno a las acciones

asignadas a dichos estados. Generalmente, las políticas suelen ser procesos estocásticos.

- *Función de recompensa*: define el objetivo, es decir, asocia a cada estado percibido del entorno (o el par estado-acción) un valor numérico, indicando la proximidad de ese estado con el objetivo inicialmente planteado. Si a una acción seleccionada por la política le sigue una baja recompensa, entonces la política puede ser cambiada en el futuro para seleccionar otra acción en esa misma situación.
- *Función valor*: mientras que la función recompensa indica la mejor política en el momento más inmediato, la función valor expresa la mejor política a largo plazo. La función valor $V(i)$, de un estado i, es la cantidad total de recompensa que un agente puede acumular en cada estado empezando la política en ese estado. También se podría decir que la función valor acumula predicciones de recompensa. Las recompensas se obtienen directamente del entorno mientras que los valores que genera la función valor se estiman de las diversas observaciones que el agente realiza sobre el entorno. Es importante mencionar que estos valores están basados en las recompensas obtenidas del entorno mediante la *función recompensa*. Se podría considerar que esta función es la más importante de cualquier algoritmo basado en RL. Por ello, es fundamental definir un método para estimar estos valores de la forma más eficiente posible.

Por todo lo anterior, se puede decir también que los algoritmos de RL interactúan con el entorno y tratan de construir un conjunto de reglas que clasifican los datos percibidos en clases de problemas con sus respectivas soluciones.

2.3.3 Tipos de aprendizaje por refuerzo

En este apartado se introducirán los conceptos y avances actuales basados en RL. Se describirán los métodos clásicos de Diferencias Temporales (TD), así como dos variantes importantes: Aprendizaje por Refuerzo Relacional (RRL – Relational Reinforcement Learning) y Aprendizaje por Refuerzo Jerárquico (HRL – Hierarchical Reinforcement Learning). Finalmente se comentarán los métodos de Aprendizaje por Refuerzo Evolutivo (ERL – Evolutionary Reinforcement Learning), cuya particularidad respecto a los anteriores es que buscan políticas directamente en el espacio de políticas que provee el entorno.

Diferencias Temporales, TD(λ)

El aprendizaje por diferencias temporales es el más popular dentro de la teoría de RL descrita anteriormente, e inclusive puede ser considerado como un método estándar que es usado como punto de referencia para realizar comparaciones de resultados en investigaciones concernientes a RL. La idea general del método de diferencias temporales es descrita a continuación usando como ejemplo el algoritmo Q-learning [Wat-89] [Sut-90] [Sut-92] [Wat-92] [Kae-96] [Sut-98] [Krö-02].

Los algoritmos de diferencias temporales [Sut-98] aprenden el valor que poseen los estados del sistema Markoviano[3] y ajustan una función valor que es capaz de predecir el valor actual y futuro de los

[3] La propiedad de Markov consiste en que la evolución del entorno depende exclusivamente de su estado y de la acción realizada. Es decir, la evolución del entorno no depende de los estados anteriores ni de las acciones anteriores. Se dice que el entorno no tiene memoria.

Un proceso de Decisión de Markov (MDP) es un problema de aprendizaje por refuerzo en el que el entorno cumple la propiedad de Markov.

Un proceso de Decisión de Markov finito es un MDP en el que el espacio de estados y de acciones es finito.

estados. Más precisamente, la función valor predice el valor del entorno, dados el estado actual y una política de decisiones. Si la función valor es precisa, el agente puede basar todas sus decisiones en los valores predichos para futuros estados del entorno. En otras palabras, cuando selecciona la siguiente decisión, el agente considera el efecto futuro de esa decisión examinando el valor esperado que se obtendrá a causa de haber tomado dicha decisión.

El valor óptimo de la función valor se alcanza usando las diferencias de sucesivas observaciones de estados consecutivos para aprender las predicciones correctas de valor. Supongamos que en el instante t, observamos una transición del estado i al estado j, en donde actualmente $V(i) = -0.5$ y $V(j) = +0,5$. Esto sugiere que deberíamos aumentar $V(i)$ para ajustarlo mejor con su sucesor. Para ello, se puede utilizar la siguiente regla de actualización:

$$V_{t+1}(i) = V_t(i) + \alpha\left(r_{t+1}(i) + V_{t+1}(j) - V_t(i)\right) \qquad [1.23]$$

donde α representa la tasa de aprendizaje y r la recompensa que puede haber en el estado i. La tasa de aprendizaje es un valor comprendido entre 0 y 1. Debido a que esta regla de actualización usa las diferencias de la función valor de estados sucesivos, se le llama ecuación de Diferencias Temporales. Así, la diferencia en las predicciones $V(j) - V(i)$ de estados consecutivos es usada como medida para predecir el error. Es posible visualizar el proceso como una cadena de predicciones $V(0)...V(n)$ de estados consecutivos, de los cuales únicamente el estado $V(n)$ contiene un valor "verdadero" dado como recompensa por el entorno. Los valores de los otros estados son ajustados, de manera que cada uno se adecua con el siguiente y se logra la convergencia hacia el valor final $V(n)$. En otras palabras, el verdadero valor $V(n)$ es propagado hacia atrás a través de la cadena de predicciones.

Dependiendo de la importancia que se dé a la diferencia en las predicciones para actualizar las actuales, se puede formar una familia completa de algoritmos TD(λ), donde λ es el parámetro que pondera

esa importancia. Con TD(0) se mira únicamente un paso más adelante al hacer el ajuste de las predicciones de valores. Aunque éste eventualmente llegará al resultado correcto, puede tomarse mucho tiempo. TD(λ) es similar a TD(0), pero se aplica a cada estado de acuerdo a su elegibilidad, en vez de sólo al estado inmediatamente previo s. La elegibilidad de un estado s, mide el grado de visitas hechas al propio estado s en el pasado reciente. Es una variable adicional al propio estado utilizada para como mecanismo para acelerar la convergencia a una función valor aceptable. La regla general de la técnica TD(λ) se expresa a continuación:

$$V_t(i) = V_t(i) + \alpha\left(r_{t+1}(i) + V_{t+1}(j) - V_t(i)\right)e_t(i) \qquad [1.24]$$

donde $e(i)$ es la elegibilidad del estado i. Cuando la recompensa se recibe, ésta se usa para actualizar todos los estados que han sido recientemente visitados, de acuerdo a su elegibilidad. Es computacionalmente más costoso ejecutar el TD(λ) general, aunque frecuentemente converge considerablemente más rápido para grandes valores de λ [Kae-96]. La elegibilidad $e(i)$ se actualizaría del siguiente modo:

$$e_t(i) = \begin{cases} \lambda e_{t-1}(i); & \text{si} \quad i \neq i_t \\ \lambda e_{t-1}(i) + 1; & \text{si} \quad i = i_t \end{cases} \qquad [1.25]$$

El parámetro λ puede variar entre $0 \leq \lambda \leq 1$. Cuando λ=1 es equivalente a actualizar los estados de acuerdo al número de veces que han sido visitados sin realizar ninguna ponderación. Si λ=0 sería equivalente a TD(0).

El método Q-learning [Wat-89] [Wat-92] es un caso particular de diferencias temporales con λ=0, cuyo objetivo es desarrollar un aprendizaje para evaluar una política determinada. Se basa en encontrar una función $Q(s,a)$, que es un mapeo de estados y acciones de control hacia una estimación de la función valor, *Q-valor*. Es decir, la función $Q(s,a)$ representa la utilidad de tomar una acción a, en el estado s. Dada la función $Q(s,a)$, la política óptima

es la que selecciona para cada estado, la acción que tiene asociado el mayor valor esperado (*Q-valor*) acumulado. La función $Q(s,a)$ es aprendida usando la siguiente ecuación de ajuste de diferencias temporales:

$$Q(s_t,a_t) = Q(s_t,a_t) + \alpha\left(r_{t+1} + Q_{máx}(s_{t+1},a) - Q(s_t,a_t)\right) \qquad [1.26]$$

Esencialmente, esta ecuación ajusta $Q(s,a)$ basada en la recompensa actual y la recompensa predicha si todas las decisiones tomadas posteriormente fueran óptimas. Está probado que, de este modo, la función $Q(s,a)$ converge hacia los valores óptimos de la función valor. El sistema de RL puede entonces usar los valores *Q-valor* para evaluar cada decisión que es posible tomar desde cada estado. La decisión que retorne el mayor *Q-valor* es la óptima.

La forma de proceder del método Q-learning es la siguiente: el valor *Q-valor* que se deriva de la realización de una acción es la suma de la recompensa inmediata proporcionada por el entorno r, y el valor máximo de Q para el nuevo estado alcanzado. La transición al siguiente estado queda definida por la denominada función de transición T, afectada por el parámetro γ, denominado factor de descuento. Formalmente:

$$\begin{aligned} &s_{t+1} \leftarrow T(s_t,a_t) \\ &Q(s_t,a_t) = r_{t+1} + \gamma Q_{máx}(s_{t+1}); \quad 0 \le \gamma \le 1 \end{aligned} \qquad [1.27]$$

Basándonos en [1.26], los valores de Q se actualizarían mediante la siguiente regla [Sut-98]:

$$Q(s_t,a_t) = Q(s_t,a_t) + \alpha\left(r_{t+1} + \gamma Q_{máx}(s_{t+1},a) - Q(s_t,a_t)\right) \qquad [1.28]$$

Donde el parámetro α puede variar entre $0 \le \alpha \le 1$ y γ entre $0 \le \gamma \le 1$. Ambos juegan el siguiente papel: mediante el parámetro α se ajusta el mecanismo de aprendizaje. Por ejemplo, si $\alpha = 1$, el nuevo valor de Q(s,a) no tiene en cuenta la historia anterior del valor de Q, sino que será la recompensa directa sumada al valor máximo de Q para el nuevo estado s' corregido por el factor γ. El papel de γ es

ponderar las recompensas futuras, de manera que sean reducidas geométricamente. Así, si este valor es 1 se ponderaría por igual la recompensa inmediata y la futura. La aplicación de un método de aprendizaje por refuerzo usando Q-learning tiene la siguiente forma:

Tabla 2-3: Método de aprendizaje Q-learning

```
Inicializar Q(s,a)
Repetir para cada EPISODIO
     s ← estado actual
     Repetir hasta llegar al OBJETIVO
          a ← política(s)
          Ejecutar acción a y observar s' y r
          Q(s, a) ← Q(s, a) + α (r + γQ_máx(s', a') – Q(s, a))
          s ← s'
```

Los principales problemas del método Q-learning son la sensibilidad del algoritmo con la elección de los parámetros que lo gobiernan, así como la complejidad en el espacio y tiempo (necesidad de almacenamiento y gran número de iteraciones para garantizar la convergencia) debido al posible gran número de estados para un entorno real. La convergencia del algoritmo sólo se garantiza teóricamente para un número infinito de visitas a cada estado [Wat-89].

Aprendizaje por Refuerzo Relacional (RRL)

En RRL [Dri-04] [Tad-04] la representación de los estados del sistema se hace de manera estructurada o relacional. En esta representación se debe pensar el entorno como un conjunto de objetos, los cuales poseen propiedades y están relacionados unos con otros. Para lidiar con un ambiente que posee estas características es necesario tener una representación estructurada tanto de los estados como de las acciones, los cuales pueden representarse como objetos con propiedades y relaciones entre ellos. Dos representaciones son

usadas frecuentemente por [Dri-04] para establecer objetos y relaciones: un lenguaje formal basado en lógica de primer orden (que consta de predicados y hechos), y un grafo dirigido.

Usando una representación relacional es posible tratar con espacios de estados más grandes, ya que se abstraen las cualidades de los estados, acciones y objetos del entorno.

El RRL es muy similar a la versión no relacional, de hecho, el algoritmo Q-learning puede aplicarse casi sin modificaciones. La diferencia principal radica en que los estados del entorno no necesitan ser enumerados explícitamente, sino que pueden ser definidos a través de una representación relacional. Otra diferencia es que las recompensas recibidas deben ser sustituidas por un predicado llamado "objetivo", el cual mapea un estado hacia un valor *booleano*: `objetivo : S → {true, false}`. El algoritmo Q-Learning debe usar un proceso conocido como Regresión Relacional [Dri-04] para aprender una función Q(s,a) general que pueda predecir el valor real *Q-valor* para estados no vistos.

El proceso de Regresión Relacional es complejo e incluye la creación automática de "Árboles Relacionales", los cuales tienen un fin clasificador similar al de los Árboles de decisión. De hecho, los Árboles Relacionales de [Dri-04] son un tipo de Árbol de Decisión. Los Árboles Relacionales son árboles binarios donde cada nodo no-terminal contiene un test de decisión, que es una unión de literales de lógica de primer orden, mientras que cada nodo terminal contiene una predicción (valor real). Conforme el algoritmo aprende una política, el Árbol Relacional crece y actualiza sus estadísticas. El algoritmo de creación del Árbol Relacional se denomina "Algoritmo TG" [Dri-04]. Además, en [Dri-04] se propone otro método de regresión conocido como *Relational Instance Based Regression*, que se basa en una métrica para medir la distancia entre pares Q(estado, acción). El algoritmo guarda los pares (estado, acción) en memoria y luego realiza una búsqueda por rango *k-nearest neighbour* como método de regresión; computa el promedio de los *Q-valor* de los ejemplos

guardados en memoria y lo multiplica por un peso, que es inversamente proporcional a la distancia entre ejemplos.

Aprendizaje por Refuerzo Jerárquico (HRL)

El problema de dimensionalidad se reduce a través de la abstracción en el espacio de estados y acciones del entorno. Las abstracciones de "subproblema" y "subtarea", aplicadas al espacio de estados y acciones del entorno, dan lugar a crear arquitecturas jerárquicas de control y algoritmos de aprendizaje jerarquizados. Esto es en esencia HRL. El estudio de HRL está basado en una versión general de los MDP: los SMDP (Semi-Markov Decision Processes). A continuación se presenta la idea fundamental que está detrás de los SMDP así como un concepto más concreto de HRL.

Los SMDP son una generalización sobre los MDP convencionales. Los MDP son considerados tareas secuenciales de "un solo paso", mientras que los SMDP son considerados de "múltiples pasos". En los SMDP, el agente observa un estado s, consulta su política π, escoge una acción a y la ejecuta en un tiempo τ. El agente permanece en el estado s durante el tiempo τ hasta que la tarea termina y pasa a un estado s'. En los SMDP, la función de recompensa $r(s,a)$ denota la recompensa esperada (con descuento) en el estado s acumulada durante el tiempo de espera τ a que se realice la acción a. Si consideramos al tiempo τ como un número entero positivo, estamos ante un SMDP discreto. En los MDP de "un solo paso" el tiempo de espera τ es simplemente irrelevante, por lo que se considera aleatorio.

En HRL, se considera que el tiempo τ que puede demorar el agente en realizar una acción a, está en relación directa a la recompensa que podría obtenerse al pasar del estado s al estado s'. Es decir, suponiendo que la acción a es una subtarea, el tiempo τ es una medida de cuan óptima es la subtarea.

En [Sut-99] se plantea el formalismo *options* para describir HRL en forma genérica. Debido a que la mayoría de sistemas HRL se pueden

describir en términos de este formalismo, resulta conveniente usarlo como base para la introducción de los conceptos básicos de HRL.

En HRL se posee un "core MDP" que corresponde al MDP más alto de la jerarquía. El controlador (agente) completa su tarea actuando solamente sobre el "core MDP". De esta manera se abstrae del resto de la jerarquía. El agente en todo momento posee una serie de "opciones". Una opción es una terna (S, A, π), donde S es un conjunto de estados, A un conjunto de acciones y π una política. Al recorrer el "core MDP", el agente percibe un estado s, consulta la política π y escoge una opción o. La opción o, es en realidad un MDP secundario o subtarea, que tarda en ejecutarse un tiempo τ. Al terminar de ejecutar la opción o, se pasa al estado s'. El tiempo τ que le llevó al agente realizar la opción o, se usa para calcular la recompensa que corresponde al estado s. El estado s' puede usar también esta información del "pasado" si así lo requiere, sin embargo esto hace que el proceso viole el principio de Markov. Las opciones (S, A, π) en sí mismas pueden ejecutar otras opciones de forma jerárquica, hasta llegar a *acciones primitivas*.

El principio de optimalidad de Bellman (programación dinámica) se aplica a las jerarquías de opciones, ya que al hallar la política óptima para una opción, también se halla implícitamente para la solución completa ("core MDP"). En [Sut-99] se comprueba que la política óptima que se obtiene al hacer un "aplanamiento" (*flattening*) de la jerarquía de opciones hasta tener solo acciones primitivas, es equivalente a la política que se obtiene para el "core MDP" en la versión jerárquica. Gracias a esta propiedad, por ejemplo el algoritmo Q-learning tiene validez y es aplicable a HRL y por tanto a SMDP.

Otro formalismo se basa en Jerarquías de Máquinas Abstractas (HAM). Este formalismo es parecido al de *options* de [Sut-99], en el sentido de que cada opción es representada como una Máquina de Estado Finito no-Determinista. El SMDP está planteado como un programa que invoca subprogramas. El algoritmo MAXQ descrito en [Die-02] se puede plantear en términos del formalismo *options*, con la

salvedad de que las opciones son SMDP independientes cuyas soluciones pueden aprenderse simultáneamente.

Aprendizaje por Refuerzo Evolutivo (ERL)

Los algoritmos evolutivos buscan aprender la política óptima de comportamiento directamente, buscando un mapeo simple de los estados percibidos hacia acciones óptimas [Sut-98]. Estos algoritmos buscan resolver el problema de estimación de parámetros a través de un modelo estocástico que emula el proceso de selección natural que sucede durante la evolución de las especies en la naturaleza.

Las técnicas evolutivas, en especial los algoritmos genéticos, poseen características que los hacen adecuados para resolver problemas de control que tienen una recompensa infrecuente. Muchos sistemas proporcionan una recompensa sólo en los estados objetivo, mientras que el resto de estados intermedios permanece sin recompensa. Estos sistemas se denominan *Sparce Reinforcement Systems*. Además, al igual que en el algoritmo Q-Learning, los algoritmos genéticos no necesitan conocimiento del dominio en el cual realizan la búsqueda, es decir, no necesitan tener un modelo del entorno. Otra característica interesante de estos algoritmos es que, debido a la diversidad que poseen en su población (la cual deben mantener durante todo el proceso evolutivo), no quedan atrapados en mínimos locales. A continuación se revisa brevemente las diferencias y similitudes que existen entre los métodos evolutivos y métodos basados en TD.

Los métodos basados en TD aprenden una función valor ajustando sus valores hacia una política óptima cada vez que el agente recibe una recompensa. La recompensa es propagada hacia cada par estado-acción que llevó al agente al estado donde obtiene la máxima recompensa. Esto implica que, mientras el agente no alcance un estado objetivo, es decir no reciba una recompensa real, el ajuste de la función valor se hace propagando una estimación de la recompensa que se espera recibir. Esta estimación tiende a acercase al valor real conforme el agente se aproxima al objetivo. Los métodos

evolutivos, en cambio, comienzan planteando una "población" de políticas de acción, que pueden o no maximizar la recompensa del agente a largo plazo. En la aproximación genética, cada política está codificada como un "gen". La diversidad en la población inicial de políticas tiene que ser suficientemente grande. En este caso, la recompensa total que obtiene el agente se debe al conjunto de acciones de que se compone la política y no a una acción en particular. El proceso de exploración, necesario en RL, es inherente en los métodos evolutivos, ya que la exploración está implícita en los operadores genéticos que mantienen la diversidad de la población (p. e. mutación).

Si el espacio de estados del sistema es grande o bien es continuo, en cuyo caso el número de estados es considerado infinito, tanto los métodos basados en TD como en métodos evolutivos usan funciones aproximadoras para predecir el valor de un estado sin tener que usar una tabla que guarde esos valores. Las redes neuronales son las funciones aproximadoras más comunes ya que requieren un espacio de almacenamiento constante, se evalúan en tiempo lineal y permiten generalizar el sistema de estados ya que pueden interpolar o extrapolar su salida para estados nunca antes vistos.

Recientemente se han planteado exitosos métodos de RL basados en métodos evolutivos de poblaciones de redes neuronales [Kass-05] [Ken-02]. En este enfoque, conocido como *NeuroEvolution*, cada red neuronal mapea directamente estados en acciones, y evoluciona dentro de su entorno guiado por una recompensa poco frecuente.

2.3.4 Planificación en RL

En aplicaciones con sistemas reales, utilizando alguna de las técnicas de aprendizaje vistas en el apartado anterior (por ejemplo, Q-learning), se emplea demasiado tiempo haciendo miles de intentos para aproximar el comportamiento óptimo. Para acelerar el aprendizaje es necesario incorporar algún mecanismo de planificación que haga uso de la información que se va generando en cada transición y que por tanto, sea posible su almacenamiento

(estados, transiciones entre estados, función valor, y asignación de acción con función valor). Este mecanismo de planificación aprovechará el tiempo en el que el agente está aplicando una acción, para realizar un barrido entre estados, y de este modo estimar la función valor para cada par estado-acción, a partir de la historia reciente salvada en memoria (modelo).

La experiencia real tiene dos roles fundamentales: ser usada para mejorar el modelo haciéndolo más preciso al entorno y ser usada directamente para el cálculo de tanto la función valor como la política con cualquiera de los métodos de aprendizaje descritos en el apartado anterior. El primer rol se denomina aprendizaje mediante método indirecto y el segundo aprendizaje mediante método directo [Kae-96]. Sin embargo, el método indirecto va siempre ligado a uno directo con objeto de poder generar el modelo. Las posibles relaciones entre experiencia, modelo, función valor y política son representadas en la Figura 2-9. Cada flecha muestra la relación de influencia y la operación realizada. El concepto de planificación comentado anteriormente hace referencia a un aprendizaje mediante método indirecto.

Tanto el método indirecto como el directo tienen sus ventajas e inconvenientes: con el método indirecto normalmente se hace un uso limitado de la experiencia y de esta manera se consigue una mejor política con menos interacciones con el entorno. Por el contrario, los métodos directos son mucho más sencillos de implementar y no se ven influenciados por posibles "desviaciones" en el diseño del propio modelo.

Teniendo un modelo del entorno, se puede predecir el siguiente estado y la recompensa, dado un estado y una acción. La predicción puede ser un conjunto de posibles estados con su probabilidad asociada o puede ser un estado que es muestreado de acuerdo a la distribución de probabilidad de los estados resultantes. Dado un modelo, es posible hacer planificación. Lo interesante es que se pueden utilizar los estados y acciones utilizados en la planificación, también para aprender. De hecho al sistema de aprendizaje no le

importa si los pares estado-acción son dados de experiencias reales o simuladas.

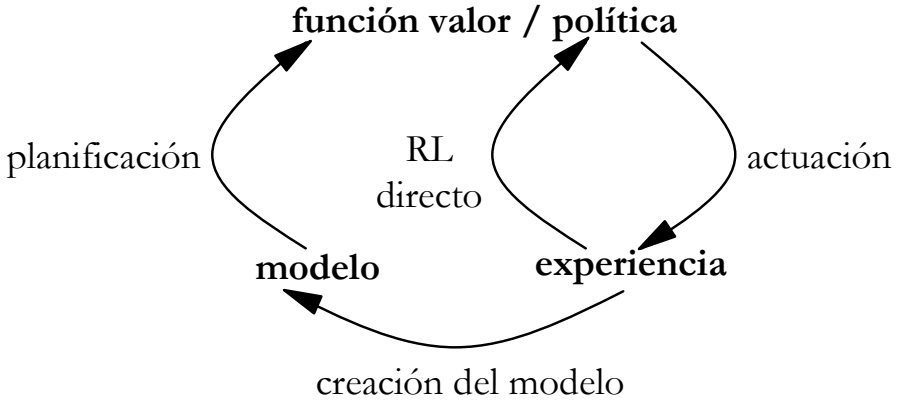


Figura 2-9: Relaciones entre aprendizaje, planificación y actuación.

Por tanto, dado un modelo del entorno, es posible seleccionar aleatoriamente un par estado–acción, usar el modelo para predecir el siguiente estado, obtener una recompensa y actualizar el valor de Q del estado correspondiente. Esto se puede repetir indefinidamente hasta converger al valor óptimo de Q para ese estado.

Dos de los mecanismos más ampliamente utilizados en planificación son *Dyna-Q* [Sut-90] o *Prioritized Sweeping* [Moo-93]. Ambos incluyen un mecanismo de búsqueda para asimilar las experiencias reales pasadas en un orden especificado. El algoritmo *Dyna-Q* (véase Tabla 2-4) combina experiencias con planificación para aprender más rápidamente una política óptima. La idea es aprender de experiencia, pero también usar un modelo para simular experiencia adicional y así aprender más rápidamente. El algoritmo de *Dyna-Q* selecciona pares estado-acción aleatoriamente.

Sin embargo, la planificación se puede usar mucho mejor si se enfoca a pares estado-acción específicos. Por ejemplo, enfocarse en el conjunto objetivo e irnos hacia atrás o más generalmente, irnos hacia atrás de cualquier estado que cambie su valor. Lo que se puede hacer es enfocar la planificación al estado que cambió su valor. Esto conduce a todos los estados que llegan a ese estado y que también cambiarían su valor. Este proceso se puede repetir sucesivamente. Sin

embargo, algunos estados cambian mucho más que otros. En este caso se pueden ordenar y cambiar sólo los que rebasen un cierto umbral, δ. Esto es precisamente lo que hace el algoritmo de *Prioritized Sweeping* (véase Tabla 2-5).

Tabla 2-4: Planificación *Dyna-Q*

```
Inicializar Q(s,a) y Modelo(s,a)
DO FOREVER
  s ← estado actual
  a ← política(s)
  Ejecutar acción a y observar s' y r
  Q(s, a) ← Q(s, a) + α (r + γQ_máx(s', a') – Q(s, a))
  Modelo(s,a) ← s', r
  Repetir N veces
       s_d ← estado seleccionado de forma aleatoria
       a_d ← acción aleatoria tomada en s_d
       s_d', r ← Modelo(s_d, a_d)
       Q(s_d, a_d) ← Q(s_d, a_d) + α (r + γQ_máx(s_d', a_d') – Q(s_d, a_d))
```

Tabla 2-5: Planificación *Prioritized Sweeping*

```
Inicializar Q(s,a), Modelo(s,a) y ColaP=0
DO FOREVER
  s ← estado actual
  a ← política(s)
  Ejecutar acción a y observar s' y r
  Modelo(s,a) ← s',r
  p ← |r + γQ_máx(s') − Q(s,a)|
  IF    p > δ
  THEN Insertar (s,a) en ColaP con prioridad p
       Repetir N veces, mientras ColaP≠0
         s_c, a_c ← Primero(ColaP)
         s_c', r ← Modelo(s_c, a_c)
         Q(s_c, a_c) ← Q(s_c, a_c) + α (r + γQ_máx(s_c', a_c') − Q(s_c, a_c))
         Repetir para cada par estado-acción predicho
         (s̄, ā), que llega a s_c
              Predecir recompensa, r̄
              p ← |r̄ + γQ_máx(s_c) − Q(s̄, ā)|
              IF    p > δ
              THEN Insertar (s̄, ā) en ColaP con prioridad
                   p
```

2.3.5 Ejemplos

Control del motor de tracción de un vehículo

Para probar la técnica de aprendizaje por refuerzo, ésta se ha implementado para el control del motor de tracción del vehículo utilizado en los trabajos llevados a cabo en esta tesis y que es presentado en el Capítulo 3.

Para excitar al motor de tracción se ha diseñado una función denominada `poner_tension(int valor)`, que dependiendo del valor que tenga su parámetro de entrada, se excita al motor con una

tensión continua proporcional en valor y signo. Con objeto de observar el cambio de estado del motor, se ha desarrollado otra función llamada `calcular_velocidad_posicion()`, que mediante las señales provenientes de un tacómetro y una rutina de retardo, que proporciona la señal de muestreo, se calcula la distancia y velocidad adquirida en ese periodo en función del número de pulsos emitidos por el tacómetro.

La política de actuación del algoritmo de control óptimo se corresponde con la excitación del motor de manera aleatoria. Partiendo de un estado inicial del motor donde éste se encuentra parado y posicionado en 0 radianes, el objetivo es que el motor alcance la posición de 70 radianes en tiempo mínimo, para una vez llegado a esta posición se pare de nuevo. La fase de aprendizaje requirió un tiempo de 20 horas para actualizar convenientemente la tabla Q. Se ha considerado una longitud máxima de recorrido del eje del motor igual a 80,5 radianes y una velocidad máxima del mismo de 31 rad/s.

Para definir los estados del motor se utiliza un espacio discreto de dos dimensiones en donde un eje representa la posición del motor y el otro la velocidad, de manera que el entorno queda dividido en celdas que representan los posibles estados por los que puede pasar el motor. El número de divisiones realizadas en el algoritmo es de 14x14, de manera que una celda tiene las siguientes dimensiones: 5,75 radianes y 2.21 rad/s. La razón de utilizar una rejilla como la indicada, es decir, no demasiado fina, radica en el hecho de disponer de poca memoria SRAM en el momento de la realización de la prueba.

La tabla Q está compuesta por 196 filas (número de estados posibles) y 10 columnas correspondientes a las posibles tensiones que se pueden aplicar al motor. En la fase de aprendizaje y dependiendo del estado del motor, las celdas de la tabla Q se actualizan de acuerdo a [1.28]. Para que el proceso de aprendizaje sea efectivo, cada una de las posiciones de la tabla Q (1960 en total) tiene que actualizarse varias veces y como la política de actuación es aleatoria, se debe

realizar un largo periodo de aprendizaje para que se actualicen correctamente.

Teniendo en cuenta que la memoria SRAM en la que se ha implementado el algoritmo es de 24Kbytes, esta característica como se ha comentado anteriormente, condicionó el número de celdas utilizadas, ya que a mayor número de celdas, mayor es el espacio en memoria SRAM que ocupa la tabla Q. Otra limitación la impuso el espacio máximo que el motor puede recorrer en un periodo de muestreo, ya que el motor no puede girar muy rápido o de lo contrario tendríamos problemas de desbordamiento del registro acumulador de pulsos del tacómetro.

El parámetro clave para poder ajustar el algoritmo es el periodo de muestreo, ya que está ligado directamente a los resultados que se obtienen en las lecturas periódicas de posición y velocidad del motor. En este experimento se ha ajustado el periodo de muestreo de manera que a velocidad máxima, el motor recorra aproximadamente un tercio de la longitud de una celda. Por lo tanto, una vez ajustado el periodo de muestreo en función del tamaño de celda, si se cambia el voltaje de la batería que alimenta el motor de tracción, se tiene que volver a reajustar el tiempo de muestreo, lo cual hace que se tenga que volver a calcular el tamaño de celda.

Finalizada la etapa de aprendizaje, comienza la etapa de ejecución en la que el motor tiene que situarse en un estado origen y realizar la trayectoria que dependerá del estado inicial en el que se encuentre el motor y de los datos almacenados en la tabla Q. Cada fila de la tabla está subdividida en 10 celdas de tipo `float`, en las que se encuentran los valores obtenidos por el método Q-learning. Al ejecutarse el algoritmo, se comprueba en qué estado se encuentra el motor, y se consulta la fila de la tabla Q que se corresponda con dicho estado. Se busca el mayor de los diez números contenidos en cada una de las subdivisiones, y se aplica la tensión al motor que corresponda con la posición del número mayor. En la Figura 2-10 se muestra la ejecución ideal del algoritmo y el resultado real obtenido después del aprendizaje.

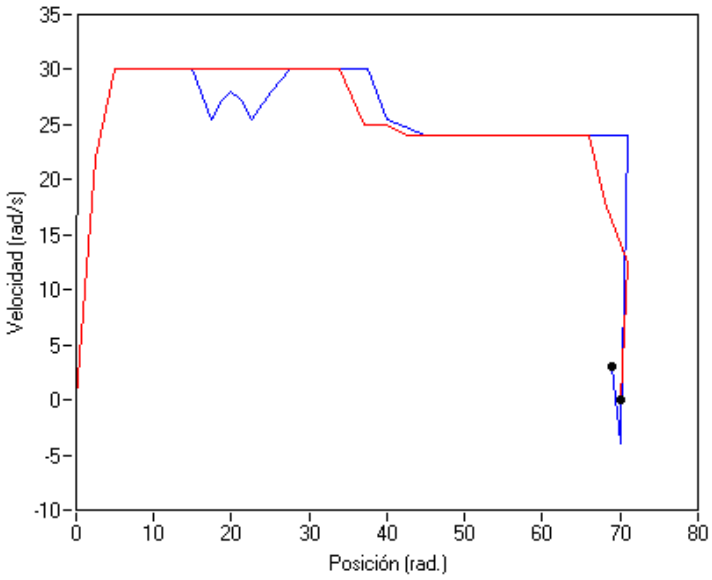


Figura 2-10: Solución del control del motor con Q-learning. En azul se representa la velocidad real del motor en rad/s y en rojo la velocidad ideal del mismo.

Control de un vehículo sobre la colina

Para aplicar aprendizaje por refuerzo y obtener una solución mediante este el método Q-learning, se considera una función recompensa caracterizada por estar acotada entre dos valores límite: –1 y +1. Será –1 si el coche se sale por el lado izquierdo del espacio de estados y varía linealmente entre los valores límite dependiendo de la velocidad del mismo cuando se salga por el lado derecho. La mejor recompensa será +1 y ésta se producirá cuando el coche alcance la cota derecha del espacio de estados (cima de la colina) con velocidad cero (Véase Figura 2-6). A continuación en la Figura 2-11 se muestra la controlabilidad de la solución óptima obtenida para el problema. Observando dicha Figura, se pueden distinguir tres zonas:

- *Frontera 1*: La trayectoria óptima que arranque desde cualquier punto situado por encima de esta frontera, permanecerá dentro del espacio de estados adquiriendo eventualmente un valor positivo de recompensa. Sin embargo, para uno por debajo de ella, cualquier control que se aplique conducirá al vehículo a la parte izquierda de la

colina (Figura 2-6), debido a que la velocidad inicial tiene un valor negativo alto.

- *Frontera 2*: Cualquier punto situado por encima de esta frontera puede alcanzar directamente la cima, mientras que por debajo de ella, el vehículo tendría que ir marcha atrás para conseguir la energía necesaria y alcanzar la cima.
- *Frontera 3*: Por debajo de esta frontera el coche acelera para alcanzar el objetivo lo más rápido posible, mientras que por encima de ella, el vehículo frena para alcanzar la cima con la velocidad más pequeña recibiendo la máxima recompensa.

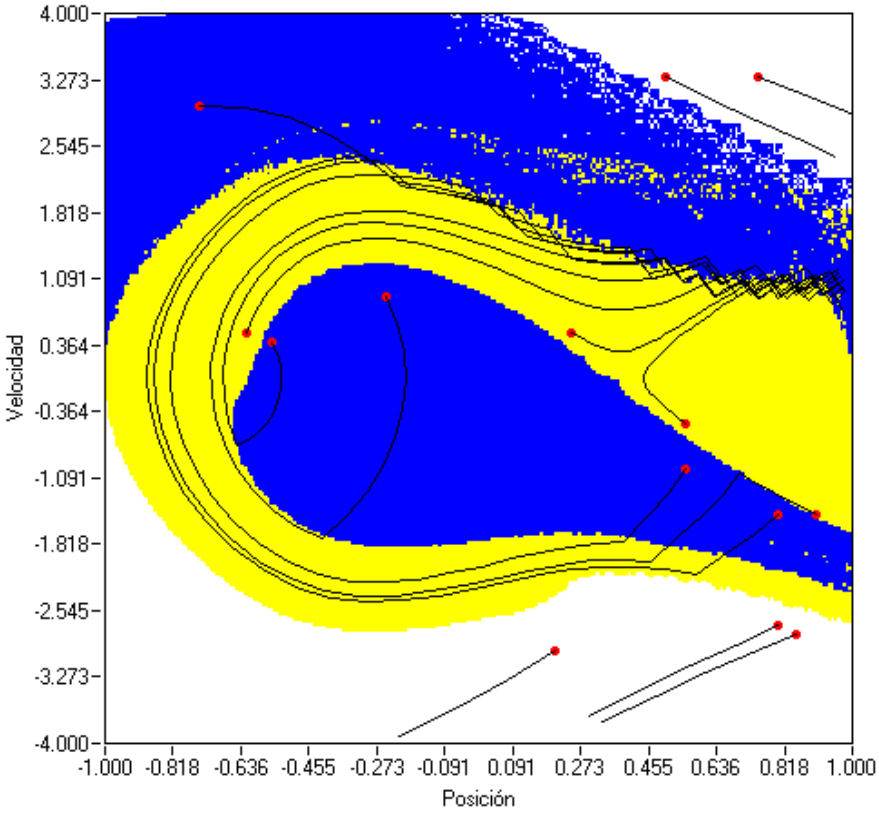


Figura 2-11: Controlabilidad del vehículo sobre la colina. Se ha obtenido para una rejilla de 257x257 celdas (usando un factor de descuento γ=0.6). El periodo de muestreo empleado ha sido 20 ms. En el área de color amarillo se aplica una acción positiva y en el área azul negativa.

2.3.6 Conclusiones

En el apartado 2.3 se ha estudiado la técnica RL y se han descrito algunos tipos de metodologías basadas en RL, siendo el aprendizaje por diferencias temporales considerado como punto de referencia para realizar comparaciones de resultados en investigaciones concernientes a RL. A través de la estimación de parámetros, el aprendizaje basado en diferencias temporales puede utilizarse cuando se aplica a procesos de decisión de Markov parcialmente observables, dinámicos y totalmente desconocidos a priori. En un proceso iterativo, el algoritmo TD reduce las diferencias temporales y eventualmente encuentra la convergencia hacia una política de acciones óptima. Además, en el apartado 2.3.4 se ha indicado la conveniencia de realizar una planificación para estimar las recompensas de cada pareja estado-acción, a partir de las experiencias reales pasadas, con objeto de disminuir el tiempo de aprendizaje.

Además, los mismos dos problemas resueltos en el apartado 2.2 mediante las técnicas de Cell Mapping se han resuelto aquí haciendo uso del aprendizaje por refuerzo. Para el problema del control del motor de tracción, en vez de utilizar un modelo matemático se ha ejecutado el método de aprendizaje por refuerzo en una plataforma real, directamente sobre el motor. De esta manera y al cabo de un tiempo, el aprendizaje finaliza almacenando en la tabla Q y de una forma implícita el modelo real del motor. Esta tabla hace las funciones del controlador óptimo siendo éste aplicado para generar una trayectoria determinada, obteniendo unas curvas típicas de un problema bang-bang.

Para el problema del vehículo sobre la colina, se ha procedido de la misma manera pero en este caso se ha utilizado el modelo matemático para generar los resultados de forma simulada. Si se compara la Figura 2-11 con la Figura 2-7 se puede apreciar que la fuerza de empuje va adquiriendo el mismo signo en el espacio de estados considerado.

Capítulo 3

La mayoría de las ideas fundamentales de la ciencia son esencialmente sencillas y, por regla general pueden ser expresadas en un lenguaje comprensible para todos.

ALBERT EINSTEIN (1879-1955. Científico y Nobel de Física)

3. Desarrollo de la investigación

3.1 Introducción

En este capítulo se presenta el desarrollo de la investigación llevada a cabo en esta tesis. Este desarrollo empieza con la descripción de un nuevo algoritmo de control óptimo que se propone como fruto de la investigación realizada y que es aplicable a la resolución del problema de planificación de movimiento en vehículos no holónomos, utilizando como base técnica otros métodos anteriores expuestos en el Capítulo 2. Con objeto de llevar a la práctica el algoritmo anterior, se evaluó y se dedujo una nueva transformación de las transiciones con la que el tiempo de aprendizaje de dicho algoritmo se fuera reduciendo hasta valores que permitieran su uso desde un punto de vista práctico.

3.2 Combinación de CACM con RL

El principal objetivo del algoritmo ideado en esta tesis [Gom-07a] [Gom-08] [Gom-09a] [Gom-09b] y descrito en este Capítulo 3, será realizar una planificación óptima del movimiento de cualquier vehículo. En particular, el vehículo considerado para la realización de todas las pruebas y verificar el algoritmo planteado, ha sido no holónomo, dotado de cuatro ruedas, de las cuales las dos delanteras marcan la dirección, pudiendo realizarse la tracción bien en el eje delantero o bien en el trasero. En general, las investigaciones recientes han estado centradas en diferentes métodos basados en algoritmos de generación de trayectorias. Todos estos métodos tienen un objetivo común y es obtener una secuencia real de movimientos evitando los posibles obstáculos presentes en el entorno [Lam-01] [Gom-01] por donde se desplaza el vehículo. La implementación de estos algoritmos se basa en un método *off-line*, por el cual se obtiene una trayectoria libre de obstáculos a partir de información almacenada previamente del entorno (posición de obstáculos, características y formas geométricas del espacio, etc.). Si además, se tiene la necesidad de que la trayectoria generada cumpla con ciertos criterios (tiempo mínimo, energía mínima, distancia mínima, etc.), significa que no vale cualquier trayectoria libre de obstáculos, por lo que el vehículo debería generar de forma inteligente una trayectoria determinada. Así, en [Ran-96] se describe la implementación de un algoritmo de búsqueda basado en la técnica A^* para planificación de trayectorias en vehículos móviles autónomos usando un mapa de todos los obstáculos conocidos para determinar la trayectoria más corta libre de colisión. Por lo tanto, el objetivo de estos algoritmos es realizar un control de manera que el vehículo siga el camino obtenido de modo *off-line*. Obviamente, este control depende de las características dinámicas y mecánicas del vehículo.

Normalmente los vehículos tienen ruedas motrices bien delanteras o traseras y ruedas de dirección delanteras. En este tipo de vehículos, se utilizan diferentes algoritmos de control óptimo para generar de forma inteligente las trayectorias. Basándose en el Modelo de Control

de la Velocidad del Ángulo de Dirección (SAVCM), [Qin-00a] trata con el problema de la planificación de trayectorias en vehículos caracterizados por tracción trasera y dirección delantera cuya dirección no puede variar instantáneamente. En este caso concreto, el control óptimo se hace para ofrecer una solución con el criterio de energía mínima. Sin embargo, en el trabajo presentado [Qin-00b] se describe un planificador de trayectorias compuesto por un generador de trayectorias y por un planificador de la estrategia de control. Como estrategia de control se utiliza bang-bang, basada en restricciones cinemáticas para obtener la solución de la distancia más corta como trayectoria óptima.

En otros trabajos [Lam-01] [Fra-01] [Gom-07b] se proponen diferentes algoritmos de planificación para generar trayectorias que producen movimientos suaves. Sin embargo, estas trayectorias se realizan en lazo abierto y en entornos reales donde cualquier vehículo puede estar sujeto a perturbaciones e incertidumbres y que, por lo tanto, parecería más apropiado realizar un control en lazo cerrado.

Existen soluciones en lazo cerrado que incluyen obstáculos haciendo uso de la programación dinámica descrita en el apartado 2.1.3 [Jba-89]. Aunque la programación dinámica es eficiente si la comparamos con la búsqueda directa, requiere un gran coste computacional. Por lo tanto, podríamos decir que la planificación óptima de trayectorias es una línea de investigación abierta, especialmente considerando la presencia de obstáculos. Existen otras soluciones en lazo cerrado basadas en técnicas de Cell Mapping, por ejemplo CSCM [Gom-02], pero tienen el inconveniente de que el controlador al haber sido generado a partir de un modelo matemático del comportamiento del sistema, es muy sensible a posibles perturbaciones u holguras mecánicas de la propia plataforma a la que se aplica el control óptimo.

En esta tesis, se propone una solución para la planificación óptima de trayectorias en presencia de obstáculos donde se reduce notablemente la complejidad computacional y se realiza el control globalmente. Esta solución está basada en la combinación de técnicas

de Cell Mapping con aprendizaje por refuerzo (CACM-RL), aprovechando las ventajas de ambas.

3.2.1 Variables de estado y acciones

La aplicación del nuevo algoritmo, fruto de la combinación de las técnicas CACM y RL, se ha llevado a cabo teniendo en cuenta la definición de cuatro variables de estado que se indican a continuación:

- Velocidad, v
- Coordenada X, x
- Coordenada Y, y
- Orientación, θ

Estas cuatro variables y según los apartados 4.3 y 4.4 se han utilizado para recabar información de la dinámica del propio vehículo y hacer un control óptimo más acorde a la propia realidad de sus movimientos. Sin embargo, en el apartado 4.2 donde el algoritmo CACM-RL se aplica bajo condiciones de simulación considerando exclusivamente la cinemática, el número de variables se reduce a tres (coordenadas X, Y y orientación), pasando la velocidad a desempeñar el papel de acción de control.

Las cuatro variables anteriores son continuas y dado que una de las técnicas en las que se basa el algoritmo propuesto es la técnica de *Cell Mapping* por la cual se discretiza el estado de cualquier sistema, se hace necesario establecer intervalos de aplicabilidad para las variables, así como una discretización de dichos intervalos (véase apartado 4 para los distintos casos que se han probado). Conviene destacar que el algoritmo CACM-RL no tiene ninguna limitación en cuanto a límites de alguna de las variables anteriores u otras que se pudieran considerar.

Referente al proceso de discretización de las variables, éste vendrá marcado por la distancia de transición que se considere (véase

apartado 2.2.7). Así, por ejemplo para una distancia $d_k = 1$, una posible discretización podría ser la indicada en la Tabla 3-1. En general, cuanto mayor sea la distancia de transición mayor número de celdas por variable será necesario tener y mejor aproximación se obtendrá a la solución óptima [Mar-99b].

Tabla 3-1: Ejemplo de discretización de variables de estado

VARIABLE	N_C
Velocidad	7
X	19
Y	19
Orientación	23

Se van a contemplar dos variables de control: una relacionada con la tracción del vehículo y otra con su dirección (véase el apartado 4 para más detalles sobre los valores considerados y tipos de acciones).

3.2.2 Descripción del algoritmo CACM-RL

En la Figura 3-1 se presenta el nuevo algoritmo o técnica que se ha implementado como consecuencia de los trabajos de investigación llevados a cabo. Este algoritmo se aplicará al vehículo indicado en la Figura 3-2.

El algoritmo indicado en la Figura 3-1 se ha denominado CACM-RL debido a que implementa los conceptos de CACM (apartado 2.2) y aprendizaje por refuerzo (apartado 2.3), ambos integrados, particularizados y constituyendo una única solución para la planificación óptima del movimiento del vehículo. Dentro de las técnicas de aprendizaje por refuerzo, el algoritmo propuesto se basa en Q-learning debido a que se trata de un método de aprendizaje por diferencias temporales y por tanto, tal y como se expuso en el Capítulo 2 es considerado como punto de referencia para realizar comparaciones de resultados en investigaciones concernientes a RL.

Las dos estructuras de datos más significativas del algoritmo de la Figura 3-1 son la *Tabla-Q* y *Tabla-Modelo*. La primera contiene los valores del factor Q para cada estado y para cada acción, Q(s,a), (véase apartado 2.3) y la segunda, el valor de los nuevos estados a los que se transita desde el origen de coordenadas para cada una de las acciones consideradas (véase apartado 3.3), es decir, el modelo a partir del cual se podrán deducir, aplicando las ecuaciones [3.11], el resto de transiciones para los demás estados. El estado en el que se encuentra en un momento dado el vehículo se representa en el algoritmo por el vector real, x, que se convierte a un estado discreto, s, mediante la llamada a la función `celda()`. La función `Dk-adyacencia()` se utiliza para determinar si se cumple la propiedad de adyacencia.

```
Inicializar Tabla-Q(s,a) y Tabla-Modelo
   x ← estado actual
   s ← celda(x)
   IF   s ∈ sumidero or s ∈ objetivo or s ∈ Zona_guarda
   THEN f_reactiva(x)
   ELSE IF    Dk-adyacencia(x,x')
        THEN  Tabla-Q(s,a) ← s',r
              Tabla-Modelo ← ETI(x,x')
        a ← política(s)
        Ejecutar acción a sobre el vehículo
        Observar el nuevo estado x' y recompensa r
HASTA terminar la fase de aprendizaje
FOR todos (s,a), repetir N veces
   x̄' ← Tabla-Modelo, ETD
   s̄' ← celda(x̄')
   Tabla-Q(s,a) ← s̄',r
```

Figura 3-1: Pseudocódigo del algoritmo CACM-RL

Figura 3-2: Vehículo usado en la implementación del algoritmo CACM-RL

Tabla-Q

La Tabla-Q se actualiza de acuerdo a la expresión general indicada en [1.28] y particularizada conforme a:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left(r + \gamma Q_{máx}\left(s'\right) - Q\left(s,a\right)\right) \qquad [3.1]$$

En la ecuación [3.1], la tasa de aprendizaje α se ha fijado igual a 0.4 y el factor de descuento γ a 1. Sin embargo, cabe destacar la capacidad implícita que tiene el algoritmo de ajustar de forma automática la tasa de aprendizaje, de manera que se podría promediar la recompensa en función del número de veces que se pasa por el mismo estado y así, evitar fijarlo a un valor durante todo el aprendizaje. En cuanto al factor de descuento, el hecho de particularizar el control óptimo en tiempo mínimo, hace que este factor se pueda poner constante a 1.

Los valores iniciales utilizados para la tabla de valores de Q tienen una influencia importante en el proceso de convergencia. Intuitivamente, si estos valores están cercanos a los valores óptimos, la convergencia será rápida. Por el contrario, determinadas configuraciones iniciales pueden hacer que la convergencia sea muy lenta en función de la situación relativa de los valores óptimos, o que

incluso esta convergencia al valor óptimo no se produzca. Su inicialización se hace teniendo en cuenta dos posibles condiciones iniciales:

$$\begin{aligned} Q(s,a) &= 0; \qquad \forall s \notin C_{OBJ}, \forall a \\ Q(s,a) &= K_{OBJ}; \quad s \in C_{OBJ}, \forall a \end{aligned} \qquad [3.2]$$

siendo C_{OBJ} el conjunto objetivo. Para todos los estados con excepción del conjunto objetivo el valor del factor Q se inicializará a 0 y para el objetivo este valor vendrá dado por una constante K_{OBJ}.

Dado que el criterio de optimización es tiempo mínimo, la idea es tener un valor suficientemente alto del factor Q en el estado objetivo con respecto al resto de estados, como para que durante el proceso de aprendizaje este valor se vaya propagando a otros estados que conduzcan a dicho conjunto objetivo.

Cuando se transita a un nuevo estado que no pertenece al conjunto objetivo y está dentro del espacio de estados considerado, la recompensa asignada es:

$$r = -T_t \qquad [3.3]$$

donde, T_t, es la duración de la transición. El motivo de hacer esta asignación negativa en la recompensa es debido a que el criterio de optimización es el tiempo, que ha de ser mínimo.

La idea es encontrar una acción de control que minimice en cada paso el tiempo acumulado desde cualquier estado al objetivo. El tiempo acumulado está directamente relacionado con los valores de Q(s,a). Estos valores se estiman a partir de la experiencia recibida del propio entorno, de acuerdo a la ecuación [3.1]. A medida que el vehículo se mueve por el entorno, el tiempo acumulado en el estado origen se suma al asociado en el nuevo estado. Mientras el coche se siga moviendo, la propagación se irá extendiendo al resto de estados. Al final del proceso de aprendizaje, los valores de Q(s,a) se habrán propagado a todos los estados controlables. Por lo tanto, la

propagación se lleva a cabo hacia atrás tanto en tiempo como en espacio. Se puede afirmar que el comportamiento del algoritmo sigue el principio de programación dinámica (véase apartado 2.1.3).

En caso de transitar al objetivo, la recompensa asignada es K_{OBJ}. La aplicación de la ecuación [3.1] sólo se va a realizar cuando el valor $Q_{máx}(s')$ sea positivo. Aunque teóricamente la actualización del valor de Q(s,a) se puede hacer siempre, independientemente del valor máximo de $Q_{máx}(s')$, en este algoritmo se ha optado por hacerlo de esta forma para no acumular demasiados valores negativos de Q(s,a) que puedan provocar una lenta y costosa recuperación hasta valores positivos y así hacer que el aprendizaje sea más rápido. Conviene aclarar que la acción, a, es un índice que hace referencia a los dos tipos de acciones manejadas por el sistema: par de tracción y el ángulo de dirección del vehículo.

Tabla-Modelo

Conceptualmente, la Tabla-Modelo es una estructura de datos constituida por un número de elementos igual a multiplicar el número de estados por las acciones de control utilizadas. Sin embargo, por las características particulares del problema tratado en esta tesis y relacionadas con la simetría en rotaciones y traslaciones en los movimientos del vehículo dentro del espacio cartesiano, la Tabla-Modelo se va a poder simplificar almacenando las transiciones que se generan desde un estado origen, cuando se "barren" todas las acciones de control para cada valor de velocidad. El algoritmo calcula estas transiciones aplicando las *ecuaciones de transformación inversas* (ETI), descritas en el apartado 3.3. Por simplicidad, el estado origen utilizado en esta tesis ha sido el origen de coordenadas. Es muy importante para el algoritmo que esta tabla se complete para cada acción de la manera más precisa (véase apartado 3.3.1 para más detalles).

Una vez se tiene la Tabla-Modelo, el siguiente paso es utilizar la información almacenada en ella para deducir las transiciones del resto del espacio de estados y así calcular también el factor Q para cada

estado y para cada acción, Q(s,a). Esto se realiza a través de un número N de iteraciones, tal y como se indica en el algoritmo de la Figura 3-1, aplicando las *ecuaciones de transformación directas* (ETD), descritas en el apartado 3.3.

Función `política(s)`

En control óptimo, la política se puede definir como una manera explícita de incorporar la exploración (descubrir o adaptar el valor de las acciones) durante el aprendizaje del controlador, es decir, la determinación de la siguiente acción de control que se aplicará cuando el vehículo se encuentre en un estado, s. Se trata de estimar a través de la experiencia adquirida la función valor de la acción cuando el agente se encuentra en un estado s y aplica una acción determinada.

Las políticas más empleadas son ε*–greedy* y *softmax* [Sut-98]. El primero tiene un comportamiento "avaro" la mayor parte del tiempo, es decir elige la acción con la que piensa obtener el máximo beneficio a largo plazo. Pero cada vez que tiene que tomar una decisión tiene una pequeña probabilidad ε con la que puede elegir una acción aleatoria independiente de la estimación del valor de la acción. Esto asegura una exploración a lo largo del tiempo.

La política *softmax* propone una alternativa más atractiva, y al igual que ε*–greedy* trata de balancear la relación entre exploración y explotación. El método más común utilizado para esta política es la distribución de Gibbs o Boltzan. Para permitir la exploración, se define la probabilidad de elegir una acción, a, en el estado, s, de manera proporcional al valor estimado Q(s,a). Para implementar este criterio, se calculan las probabilidades π de elegir cada una de las acciones posibles de la siguiente forma:

$$\pi(s,a) = \frac{e^{Q(s,a)/\tau}}{\sum_{b=1}^{n} e^{Q(s,b)/\tau}} \qquad [3.4]$$

donde $\tau \geq 0$ es un parámetro conocido en la literatura como "temperatura". Cuando las estimaciones de los valores de las acciones tienen asociadas una elevada incertidumbre, por ejemplo al comienzo del aprendizaje, conviene elegir un valor elevado de τ, de modo que todas las acciones sean igualmente probables. Conviene disminuir o "enfriar" la exploración, a medida que la estimación de los valores de las acciones es más confiable. En el límite cuando $\tau \to 0$, el criterio *softmax* elegirá la acción según el criterio *greedy* (elección de la acción asociada al factor Q más elevado en ese estado), sin margen alguno para permitir explorar.

En el algoritmo de la Figura 3-1, durante la fase de aprendizaje no se hace explotación sino solamente exploración. La explotación durante el aprendizaje normalmente se utiliza para que éste sea más rápido. En este caso y debido a que por el hecho de usar las transformaciones descritas en el apartado 3.3, el aprendizaje se acelera, la explotación no aporta grandes ventajas. El método llevado a cabo para la exploración ha sido una selección pseudoaleatoria de una acción cada vez que se produce una nueva transición. De esta manera, una vez finalizado el aprendizaje, todas las acciones han sido seleccionadas varias veces, asegurando la convergencia [Sut-98].

En el enfoque dado al nuevo algoritmo, el aprendizaje se ha separado de la planificación. Sin embargo, perfectamente estas dos fases se podrían hacer de forma simultánea tal y como normalmente se implementa mediante los métodos comentados anteriormente. El motivo de haber separado estas dos fases es debido a que el tiempo de aprendizaje se ha reducido por el hecho de haber realizado una transformación de transiciones (véase apartado 3.3). Cuando este algoritmo se quiera aplicar con otras variables de estado donde no sea posible realizar transformación de transiciones y por tanto, el tiempo de aprendizaje sea más elevado, es más conveniente que aprendizaje y planificación se ejecuten de forma simultánea.

Función *Dk-adyacencia(x,x')*

De acuerdo a lo expuesto en el apartado 2.2.4, relativo al método CACM, esta función será la que determine la distancia $d_k (d_k > 1)$, entre el estado actual del vehículo x', y el anterior x. La manera de establecer la distancia de transición será la siguiente:

- Para cada variable, se calculará la diferencia en valor absoluto entre el valor actual y el que tenía el vehículo anteriormente.
- Si cualquiera de estas diferencias en valor absoluto supera el tamaño de celda de su variable respectiva en d_k unidades, se considerará como distancia d_k. La distancia será la parte entera del tamaño superado.
- Si cualquiera de estas diferencias en valor absoluto, no supera el tamaño de celda de su variable respectiva, la distancia será 0.

Función reactiva *f_reactiva()*

Existen situaciones reales en los movimientos del vehículo que motivan el uso de una función reactiva. Concretamente, cuando el vehículo se sale del espacio de estados (celda sumidero), alcanza el conjunto objetivo C_{OBJ}, o alcanza la zona de guarda asignada a un obstáculo. En estos casos, se llama a la función `f_reactiva()` con objeto de seguir con el aprendizaje, bien haciendo que el vehículo entre de nuevo al espacio de estados desde el sumidero, o bien desplazando al vehículo a otro punto del espacio de estados.

Para el caso en el que el vehículo alcanza una velocidad que se encuentra fuera del rango previsto, la función `f_reactiva()` debe contemplar tres posibles alternativas:

a) *Sobrepasar la velocidad máxima*: si la velocidad que lleva el coche superara a la máxima permitida, ésta se va a

decrementar frenando al vehículo (aplicación de par negativo).

b) *Sobrepasar la velocidad mínima*: cuando la velocidad que lleva el coche superara a la mínima permitida, ésta se va a incrementar acelerando al vehículo (aplicación de par positivo).

c) *Coche parado*: Este caso se daría cuando el coche se saliera por alguna de las otras variables y además se encontrara parado. Para poder salir de esta situación y continuar con el aprendizaje, el vehículo debe arrancar y empezar a moverse. Lo que se hace en este caso es acelerarlo hasta que la velocidad alcanza el valor máximo. También se podría acelerar negativamente, ya que de lo que se trata es de moverlo para posteriormente, maniobrar y "meterlo" dentro del rango de las otras variables.

Cuando la posición del coche supera los límites máximos absolutos en la coordenada X o Y, la función `f_reactiva()` debe hacer que el vehículo entre de nuevo al espacio de estados aplicando unas acciones de control contrarias a las aplicadas en el momento de sobrepasarse.

En el caso de que el vehículo alcance el conjunto objetivo, se hace perceptivo seguir con la fase de aprendizaje moviéndolo hacia otro estado. A diferencia del caso anterior, dependiendo del conjunto objetivo que se considere, la manera de mover el coche hacia otro estado va a depender precisamente de dicho conjunto.

Por último, la función `f_reactiva()` debe sacar al coche de la zona de guarda asignada a un obstáculo con objeto de prevenir un choque con el mismo. Una vez que se ha conseguido sacar al coche de la zona prohibida, hay que comprobar y actuar si fuera necesario, si se ha salido del espacio considerado (sumidero) o si ha alcanzado incluso el objetivo.

3.2.3 Temporización

El algoritmo presentado en la Figura 3-1 no tiene una ejecución secuencial continua conforme se indica, sino que desde un punto de vista práctico dicho algoritmo está constituido por distintos flujos de ejecución que se ejecutan de forma concurrente y sincronizados.

En primer lugar, cabe destacar por su gran importancia, la existencia de una rutina de interrupción ejecutada periódicamente. El periodo de ejecución de esta rutina vendrá impuesto por el tiempo medio de transición o también denominado tiempo de procesamiento de celda [Pap-97]. Es el tiempo que el vehículo necesita para moverse a otra celda distinta de la actual con una distancia de transición determinada. La mayor parte del trabajo realizado en esta tesis se ha realizado para una distancia de transición $d_k = 1$.

Considerando el tiempo anterior, cada acción de control que actúe sobre el coche lo hará durante este tiempo. Una vez transcurrido, se generará la interrupción con objeto de que la subrutina asociada, observe el nuevo estado y actúe en consecuencia en función de si es sumidero, objetivo, zona de guarda o nueva transición. Si el nuevo estado fuera el sumidero, objetivo o zona de guarda, se ejecutaría la función `f_reactiva()`. Sin embargo, hay que tener en cuenta que en la ejecución de esta función una vez se actúe sobre el coche con una acción de control determinada se retorna de la subrutina hasta la llegada de la nueva interrupción.

En el caso de que la nueva transición sea distinta del sumidero, objetivo o zona de guarda, lo que se hace dentro de la subrutina es comprobar si el nuevo estado cumple con la condición de adyacencia en cuyo caso se actualiza el valor de Q(s,a) y la Tabla-Modelo (véase apartado 3.3.1). Posteriormente, se selecciona una nueva acción de control mediante la implantación de una política determinada para actuar sobre el coche y retornar de la subrutina. Como se indicó en el apartado 3.2.2, la política establecida en esta tesis ha sido una selección pseudoaleatoria.

Durante el tiempo medio de transición en el que el vehículo se encuentra moviéndose bajo la aplicación de una acción de control determinada, existe un programa principal (bucle final del algoritmo de la Figura 3-1) que apoyándose en la información almacenada en la Tabla-Modelo actualiza la Tabla-Q, que será finalmente el controlador óptimo.

3.2.4 Evaluación del controlador

Para evaluar la calidad del algoritmo propuesto y conocer hasta qué punto los resultados obtenidos como consecuencia de su aplicabilidad se aproximan al óptimo, nos vamos a basar por un lado en el concepto de controlabilidad aplicado a un espacio celular y por otro en la denominada *curva de switching* [Bry-75] [Pap-97] [Son-02].

Una gran ventaja de utilizar un espacio celular es la capacidad de poder evaluar de forma global en todo el espacio el comportamiento del controlador. Un espacio celular unido al propio controlador se considera como un control en lazo cerrado. El uso de técnicas de Cell Mapping van a permitir poder evaluar al controlador desde dos puntos de vista: *cualitativo* y *cuantitativo*. En cuanto al primero, las regiones de estabilidad o controlables pueden verse por inspección. Con respecto al segundo, la controlabilidad también puede calcularse analíticamente como se describirá más adelante.

El controlador propuesto en esta tesis, al estar basado en las técnicas de Cell Mapping y RL, examinando su controlabilidad para valores de parámetros diferentes a los considerados en el diseño, se puede evaluar la robustez del propio controlador [Pap-97]. Esta robustez se podría definir como la capacidad del controlador para reaccionar ante cambios significativos en sus parámetros. El controlador se considerará más robusto cuanto menos varíe su controlabilidad ante cambios de sus parámetros. Otra forma de poder evaluar la robustez de forma práctica [Pap-01], sería observar cómo varían cada una de las trayectorias cuando se modifican ciertos parámetros. Dado que el número de trayectorias es generalmente alto, resulta fácil evaluar por ejemplo la variación producida en el número de celdas controlables y

el tiempo óptimo promedio, ambos explicados más adelante en este apartado.

Problema de Zermelo

Para ilustrar el concepto de controlabilidad se va a resolver un problema clásico en la literatura de control óptimo como es el problema de Zermelo [Bry-75] [Lew-86]. Considérese un barco que navega por una zona con fuertes corrientes, las cuales dependen de la posición de dicho barco. La velocidad de navegación, V, es constante, siendo posible únicamente controlar la dirección del barco representada por $\theta(t)$. La corriente del agua se dirige en la dirección creciente del eje x, con una velocidad, u, dependiente de y:

$$u = Vy, \qquad [3.5]$$

Las ecuaciones de estado del sistema son:

$$\begin{aligned} \dot{x} &= V\cos(\theta) + Vy \\ \dot{y} &= V\mathrm{sen}(\theta) \end{aligned} \qquad [3.6]$$

Se desea determinar la dirección $\theta(t)$ necesaria para desplazar el barco desde cualquier posición inicial $\left(x(t_0), y(t_0)\right)$, hasta el origen (0,0) en tiempo mínimo. Por tanto, la función de coste es:

$$J = \int_0^T 1\,dt \qquad [3.7]$$

A partir de una formulación variacional se resuelve el problema planteado, obteniéndose el siguiente resultado[4]:

[4] En las referencias [Bry-75] [Lew-86] se resuelve analíticamente con detalle el problema de Zermelo.

$$y(t) = \sec(\theta(T)) - \sec(\theta(t))$$

$$x(t) = \frac{1}{2}\left[\begin{array}{l} \sec(\theta(T))\left(\tan(\theta(T)) - \tan(\theta(t))\right) - \\ -\tan(\theta(t))\left(\sec(\theta(T)) - \sec(\theta(t))\right) + \\ +\ln\dfrac{\sec(\theta(T)) + \tan(\theta(T))}{\sec(\theta(t)) + \tan(\theta(t))} \end{array}\right] \qquad [3.8]$$

La función de control $\theta(t)$ no se puede determinar de forma explícita a partir de las ecuaciones anteriores, sino que se debe calcular, junto con $\theta(T)$, mediante un método iterativo. Para resolver este problema de forma aproximada utilizando las técnicas de Cell Mapping, por ejemplo CACM, se han empleado los siguientes datos de entrada:

- Variables de estado: 2
  - x_1: x. Rango: $-0.125 \leq x_1 \leq 10.125$ rad. Nº de celdas : 41.
  - x_2: y. Rango: $-2.2 \leq x_2 \leq 2.2$ rad/s. Nº de celdas : 41.
- Variables de control: 1
  - u: θ. Rango: $0 \leq u_1 \leq 6.086$ rad. Nº de niveles: 32.
- Tiempo de muestreo: 1 ms.

La Figura 3-3 tiene dos zonas claramente diferenciadas: la región controlable en color gris y la no controlable en negro situada en las esquinas superior derecha e inferior izquierda. La región de color gris se corresponde con la zona de controlabilidad de la resolución del problema de Zermelo [Mar-99b] definido por las ecuaciones [3.6] y parámetros de discretización indicados anteriormente. En la Figura 3-4 se muestra la solución del problema de Zermelo representando gráficamente diferentes trayectorias que alcanzan el mismo estado objetivo desde diferentes orígenes.

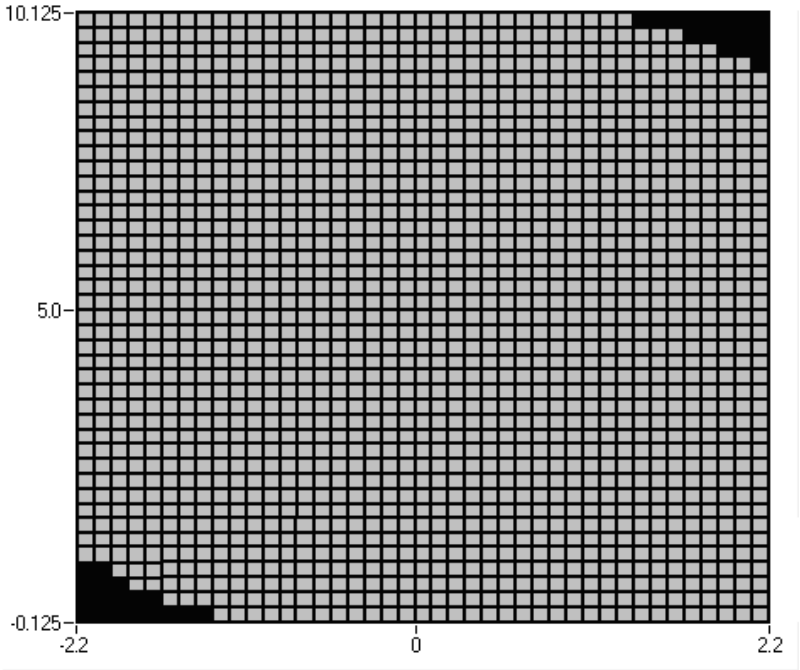


Figura 3-3: Problema de Zermelo. Controlabilidad en el espacio de estados celular

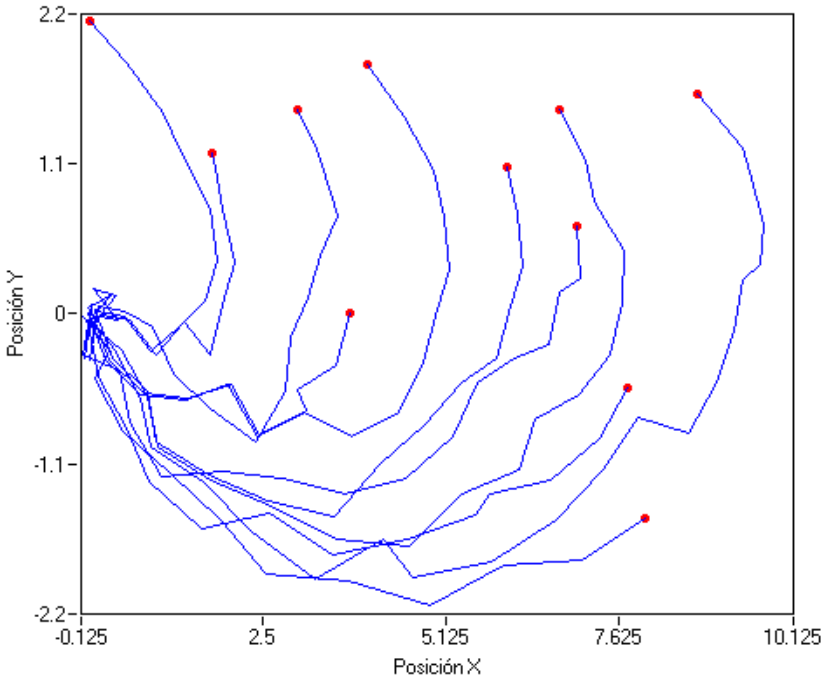


Figura 3-4: Problema de Zermelo. Diferentes trayectorias que alcanzan el mismo objetivo (0, 0) desde distintos estados orígenes

La descripción del controlador desde un punto de vista cuantitativo puede ser hecha desde varias perspectivas, detalladas seguidamente.

Controlabilidad

La controlabilidad se va a considerar como una característica intrínseca a cualquier controlador y en este caso, dado que se ha obtenido bajo un estado celular o discreto, se va a medir considerando unos criterios que difieren respecto a la teoría de control clásica. Por tanto, mediante la controlabilidad vamos a validar el comportamiento del controlador generado. También, a veces puede resultar interesante tener la capacidad de comparar este controlador con otros controladores diseñados y generados utilizando otras técnicas o metodologías diferentes siempre y cuando sea posible.

Una celda del espacio celular será controlable, si y solo si, partiendo de ella y aplicando las distintas acciones que el controlador tiene almacenadas para cada celda, se va transitando a otras celdas hasta alcanzar el objetivo [Son-02].

En el caso de implementar una discretización no uniforme para un espacio celular, la consideración de diferentes tamaños de celdas implica tener un número de celdas también diferente para un mismo espacio. Por lo tanto, con esta premisa se podría decir que el número total de celdas controlables va a depender de los tamaños de celda considerados en las distintas regiones del espacio. Así, una medida buena de la controlabilidad es el porcentaje de celdas controlables y si la discretización es uniforme, esta medida es independiente de los tamaños considerados. La controlabilidad se define como [Son-02]:

$$C_{\%} = \frac{N_{cont}}{N} \qquad [3.9]$$

donde N_{cont} es el número total de celdas controlables y N el número de celdas considerado en el espacio celular. Dado que en esta tesis los resultados se han centrado en una discretización uniforme del

espacio, la ecuación [3.9] puede usarse directamente como una medida de controlabilidad del espacio celular.

Al construir o generar la tabla de control óptimo o el controlador propiamente dicho, es necesario definir una función de coste [1.2] que nos caracterice cada una de las transiciones vinculadas a las celdas del espacio considerado. Esta caracterización denominada también coste, puede ser tiempo, energía, distancia, etc. Para el caso que nos ocupa y para la implementación del algoritmo propuesto, el coste considerado ha sido el tiempo, por lo que el control óptimo deberá consistir en generar trayectorias hacia el objetivo en el tiempo mínimo haciendo uso de las acciones disponibles.

Evolución de la controlabilidad en el aprendizaje del vehículo

A medida que el algoritmo propuesto en la Figura 3-1 se ejecuta, se puede hacer una representación gráfica de cómo evoluciona temporalmente su controlabilidad. Por ejemplo, en la Figura 3-5 se muestra dicha evolución considerando un espacio cartesiano totalmente libre sin obstáculos. En la Figura 3-6 se puede observar también dicha evolución junto con la colocación de unos obstáculos en el mismo espacio cartesiano anterior.

En la Figura 3-6 el porcentaje de celdas controlables es similar al de la Figura 3-5 donde no se consideraban obstáculos. En este caso, el número total de celdas introducidas en la ecuación [3.9] se ha reducido por el hecho de tener en cuenta las celdas obstáculo. Conviene puntualizar que la forma de la curva en un caso y otro no tiene por qué ser la misma, ya que ésta va a depender fundamentalmente de los estados por los que se vaya moviendo el vehículo y de la política implementada para la selección de las acciones que se van a aplicar en cada momento. Lo importante es que independientemente de este hecho, ambas curvas van a alcanzar un porcentaje de controlabilidad superior al 90% en un tiempo realizable desde un punto de vista práctico.

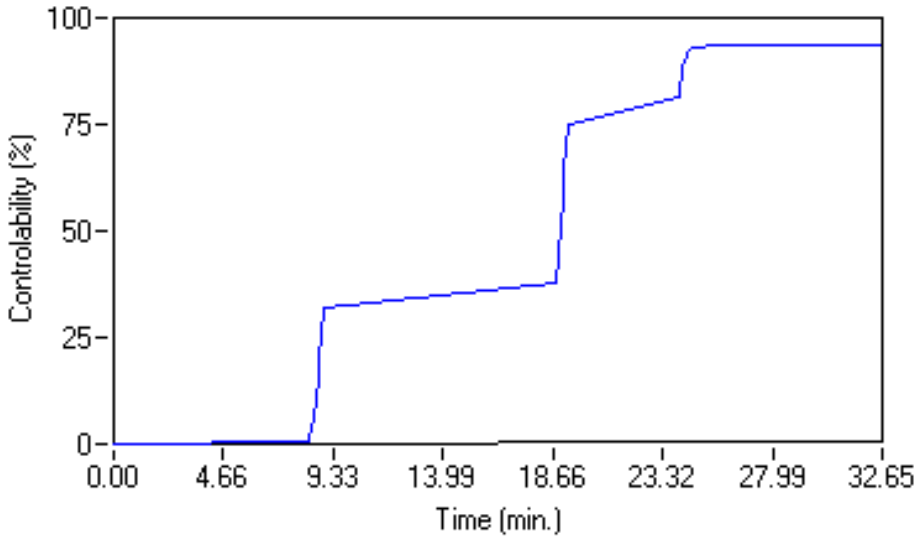


Figura 3-5: Aprendizaje del vehículo. Evolución de la controlabilidad con el espacio cartesiano libre de obstáculos

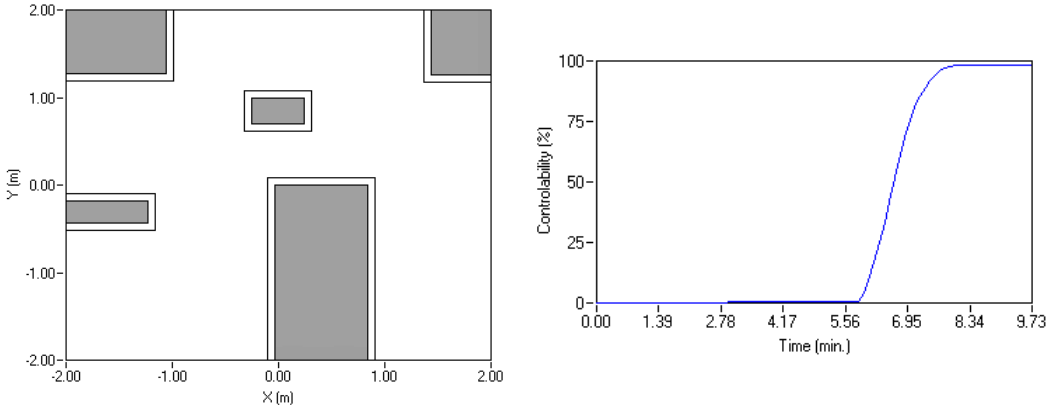


Figura 3-6: Aprendizaje del vehículo. Espacio cartesiano con obstáculos y evolución de la controlabilidad

Curva de conmutación en forma de 'S'

Una vez que el controlador se ha generado (por ejemplo, mediante el algoritmo propuesto en esta tesis) y se ha comprobado que su controlabilidad es alta y aceptable, es perceptivo realizar otra medida de calidad del mismo con objeto de conocer su aproximación al óptimo. Nuestro caso es un claro ejemplo de control bang-bang, ya que tenemos dos acciones de control acotadas y el objetivo es la

planificación óptima del movimiento del vehículo en tiempo mínimo. Por tanto, la medida de calidad del controlador generado se va a realizar haciendo una representación gráfica en dos dimensiones de dos variables de estado del espacio bajo estudio, visualizando el valor de la acción para cada celda. En definitiva, se trata de representar la *curva de conmutación en forma de 'S'* comentada en el apartado 2.1.2.

Debido a que el algoritmo de la Figura 3-1 se está aplicando a un sistema de 4 dimensiones, la representación de la curva de conmutación se va a hacer considerando una orientación igual a cero y una velocidad constante para todos los estados en el plano X-Y y un número de celdas por eje igual a 61. Dado que este número de celdas es alto para el rango de valores asignado a cada variable, el controlador se ha generado para una distancia de transición igual a 3. El objetivo se ha situado en el origen de coordenadas. Esta curva está representada en la Figura 3-7 e incluye dos trayectorias que el vehículo genera para alcanzar el objetivo en tiempo mínimo.

Observando la Figura 3-7 el vehículo empieza a moverse mientras va girando hasta que alcanza la curva, momento en el cual, gira en dirección contraria para alcanzar el objetivo. Independientemente de donde se posicione el robot, las trayectorias siempre alcanzarán la curva y una vez alcanzada, el robot la seguirá. De acuerdo a la Figura 3-7, cada color o símbolo representa una acción de control específica que actúa sobre el vehículo. En la zona roja (punteada), el vehículo avanza hacia delante girando hacia la derecha. Si se encontrara en la zona amarilla, iría marcha atrás girando a la izquierda. Cuando estuviera en la zona azul (líneas verticales), iría hacia delante girando a la izquierda también. Y finalmente, si el coche fuera marcha atrás girando hacia la derecha es porque se encontraría en la zona de color cian (líneas horizontales).

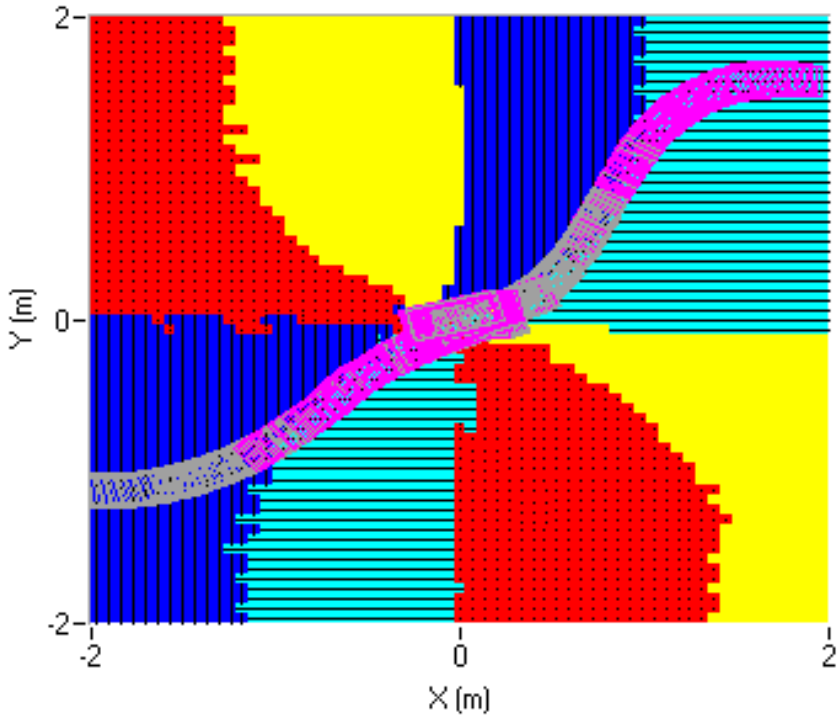


Figura 3-7: Curva de conmutación en forma de 'S'

Tiempo óptimo promedio

Las características de tiempo óptimo asociado a un controlador se podrían evaluar calculando para cada celda del espacio celular, el tiempo que necesita el sistema para alcanzar el objetivo o punto de equilibrio. De esta manera se define T(z) como el tiempo en alcanzar el objetivo o punto de equilibrio partiendo desde la celda z. Este tiempo es una medida local de la optimalidad del tiempo.

Con esta idea se puede hacer un histograma representando en el eje de abscisas los tiempos invertidos en alcanzar el objetivo o punto de equilibrio. De esta manera se obtiene una buena medida mediante histograma que permite evaluar globalmente el grado de optimalidad del tiempo. Para hacer esta representación, en el eje de abscisas se va a representar un número de intervalos de tiempo, N_t, de anchura a_i y cada T(z) se asignará a un intervalo. Así obtendremos un número de trayectorias completadas para cada uno de los intervalos.

Por tanto, para la evaluación del tiempo óptimo promedio que se consigue mediante la ejecución del algoritmo propuesto en esta tesis, se han realizado dos histogramas. En la Figura 3-8 se presentan dichos histogramas con objeto de dejar más clara su representación y de esta manera poder realizar una comparación entre un controlador generado con una distancia de transición igual a 1 (curva de color rojo) y otro con distancia igual a 3 (curva de color azul). Ambos se han creado para alcanzar el siguiente conjunto objetivo:

$$C_{OBJ}\begin{cases} x_{velocidad} = 0 \\ x_x = 0 \\ x_y = 0 \\ x_{orientación} = 0 \end{cases} \qquad [3.10]$$

Se puede observar que para el histograma representado mediante la curva roja, existe una mayor varianza en el tiempo óptimo promedio que para el de la curva azul. Esto significa que a medida que aumenta la distancia de transición, los tiempos óptimos de cada uno de los estados tienden a concentrarse en un intervalo de tiempo más pequeño. Se puede afirmar que el tiempo óptimo promedio tiende aproximarse al origen de coordenadas conforme aumenta la distancia de transición en el aprendizaje, es decir, su valor numérico disminuye con la distancia y por tanto, se acerca al óptimo.

De forma general, se podría decir que la mayor parte de las trayectorias que llegan al objetivo o punto de equilibrio tardan un tiempo similar y no se aprecian regiones periódicas por la distribución de tiempos obtenida. Este hecho se hace más notable a medida que aumenta la distancia de transición. Para concluir se puede afirmar que el tiempo óptimo promedio da una idea de cómo el controlador aproxima al sistema a su comportamiento óptimo.

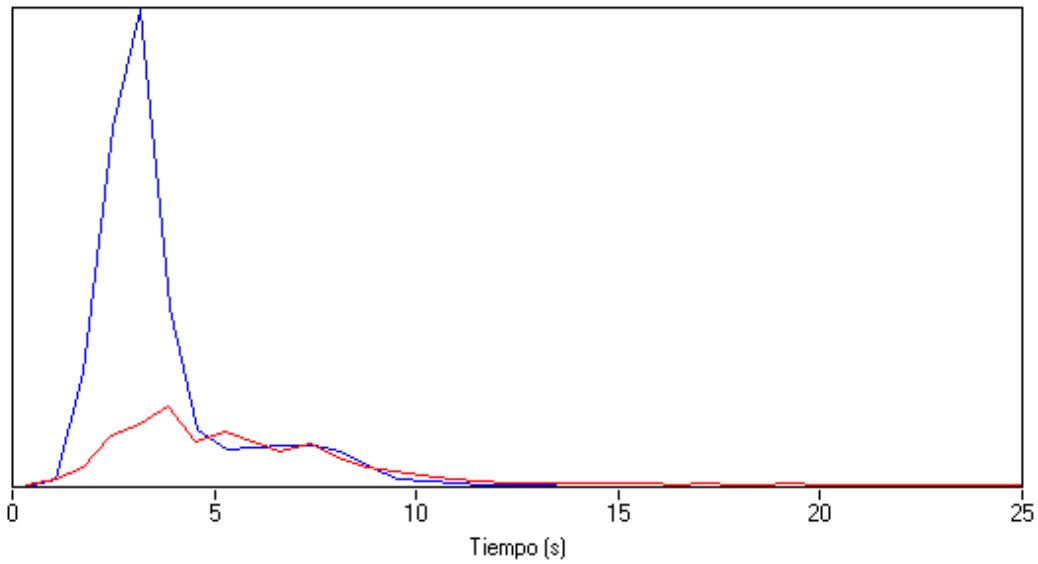


Figura 3-8: Histogramas que representan los valores del tiempo óptimo promedio para alcanzar el conjunto objetivo. La gráfica de color rojo se corresponde a un controlador generado con una distancia de transición igual a 1 y el de color azul con una distancia igual a 3.

3.3 Transformación de las transiciones

Mediante la aplicación de la nueva técnica propuesta en el apartado 3.2 se consigue un doble objetivo: por un lado generar el controlador que será el responsable del comportamiento óptimo del sistema y, por otro, dotar al sistema de la capacidad de aprender la propia dinámica del vehículo conforme éste se va moviendo por el plano X-Y. Esta capacidad de aprendizaje se sintetiza en un conjunto de Cell-to-Cell Mapping, es decir, un conjunto de transiciones para cada celda y para cada acción. Precisamente si somos capaces de generar y calcular esta información, podemos decir que tenemos todo el conocimiento necesario de la dinámica del vehículo y, por lo tanto, a partir de ella se puede generar el controlador.

Aplicando la técnica descrita en el apartado 3.2 es totalmente posible que el vehículo pueda aprender automáticamente su propia dinámica. El problema radica en que desde un punto de vista computacional, el tiempo que tardaría en aprenderla sería tan grande que resultaría

inviable su implementación práctica. Por este motivo, otra de las aportaciones incluidas en esta tesis ha sido la introducción de unas *ecuaciones de transformación directas* [3.11] mediante las cuales el vehículo sólo necesita aprender las transiciones generadas desde el origen de coordenadas para cada una de las acciones [Mar-04]. El resto de transiciones entre diferentes estados se obtienen haciendo uso de estas nuevas ecuaciones. De esta manera, se reduce notablemente el tiempo de aprendizaje haciendo viable desde un punto de vista práctico la implementación del algoritmo propuesto en el actual trabajo de investigación.

$$\begin{aligned} x_f &= x_0 + x_m \cos(\theta_0) - y_m \operatorname{sen}(\theta_0) \\ y_f &= y_0 + x_m \operatorname{sen}(\theta_0) + y_m \cos(\theta_0) \\ \theta_f &= \theta_0 + \theta_m \end{aligned} \qquad [3.11]$$

El significado de las variables usadas en las ecuaciones [3.11] es el siguiente:

- x_f, y_f y θ_f se refieren al nuevo estado alcanzado una vez producida la transición.
- x_0, y_0 y θ_0 se refieren al estado inicial.
- x_m, y_m y θ_m es el estado alcanzado en la transición local generada desde el origen.

Resulta conveniente describir la manera en que las ecuaciones indicadas en [3.11] han sido implementadas en el nuevo algoritmo propuesto en esta tesis. La primera consideración que se debe hacer es que el vehículo durante la fase de aprendizaje va a estar moviéndose y desplazándose por el plano X-Y de una manera continua, es decir, arrancará en una posición determinada y a partir de ahí sus movimientos serán continuos provocando transiciones entre las celdas que cubren el espacio de estados. Por esta razón y debido a que las ecuaciones anteriores [3.11] sólo pueden ser

aplicadas cuando el vehículo transita desde el origen de coordenadas, hay que establecer un mecanismo que basándose en ellas permita transformar cualquier transición, tt_i, generada desde cualquier punto genérico, x_i, a otra transición, tt_j, generada desde otro punto genérico distinto al anterior, x_j. La solución establecida para resolver esta nueva transformación es deducir a partir de las ecuaciones [3.11] unas nuevas ecuaciones que denominaremos *ecuaciones de transformación inversas*. Estas ecuaciones se indican a continuación:

$$\begin{aligned} x_m &= (x_f - x_0)\cos(\theta_0) + (y_f - y_0)\,\mathrm{sen}(\theta_0) \\ y_m &= -(x_f - x_0)\,\mathrm{sen}(\theta_0) + (y_f - y_0)\cos(\theta_0) \\ \theta_m &= \theta_f - \theta_0 \end{aligned} \qquad [3.12]$$

Mediante las ecuaciones [3.12] se obtiene la transición que se produciría desde el origen de coordenadas a partir de cualquier transición genérica generada desde cualquier punto también genérico. De esta manera mediante el algoritmo descrito en 3.2 y aplicando las ecuaciones [3.12], el tiempo de aprendizaje se dedicará exclusivamente a aprender las transiciones generadas desde el origen para cada valor de acción. Una vez que se tiene este conocimiento, con objeto de calcular el resto de transiciones, se trataría de aplicar directamente las ecuaciones [3.11] para cada uno de los estados pertenecientes al espacio celular global considerado. Conviene darse en cuenta que para aplicar las ecuaciones [3.11] vamos a utilizar la información almacenada para el origen, que se ha obtenido de forma experimental mediante la aplicación de las ecuaciones [3.12]. La idea es aplicar las ecuaciones [3.12] cada vez que el vehículo como consecuencia de sus movimientos por el plano X-Y produzca una transición.

3.3.1 Creación de la *Tabla-Modelo*

Cuando el vehículo inicia sus movimientos en un espacio cartesiano con o sin obstáculos, su único objetivo es ir aprendiendo mediante la

ejecución del algoritmo de la Figura 3-1 tanto su dinámica como el propio espacio por el que se mueve. En este aprendizaje, el coche al moverse realiza transiciones entre estados diferentes (celdas) y esta información, mediante la aplicabilidad de las ecuaciones [3.11] [3.12], es la base esencial para extender el conocimiento local de la transición actual al resto de estados.

Debido a que el coche puede empezar sus movimientos en cualquier punto del plano X-Y y éstos son continuos, cada transición localmente generada se utiliza en las ecuaciones de transformación inversa [3.12] para crear la Tabla-Modelo que almacene todas las transiciones desde el origen, por cada acción de control y velocidad:

$$\text{Origen}\begin{cases} x_x = 0 \\ x_y = 0 \\ x_{\text{orientación}} = 0 \end{cases} \qquad [3.13]$$

Un buen aprendizaje de la Tabla-Modelo será fundamental para conseguir un buen controlador, ya que de lo contrario cualquier error en la misma o una creación incompleta, se extenderá al resto de celdas mediante las ecuaciones [3.11] y como consecuencia, los errores se irán extendiendo.

Como se ha visto en el apartado 3.3, las ecuaciones [3.11] [3.12] sólo se aplican para tres variables de las cuatro manejadas por el algoritmo, es decir, la variable `velocidad` no puede ser calculada de forma precisa cuando se aplican las ecuaciones anteriores, sino que se hará una estimación de la misma, de forma que según transcurra el tiempo de aprendizaje se irá aproximando a su valor real.

Con objeto de realizar una aproximación más real y así ser lo más inmunes a posibles perturbaciones del entorno, se calculará el promedio para cada variable. De este modo, se consigue un efecto de filtrado en las variables que reduce el ruido de las medidas realizadas. De forma general, la estimación de cada variable, como transición desde el origen, se hace de acuerdo a:

$$\overline{x}_n' = \frac{n-1}{n}\overline{x}_{n-1}' + \frac{1}{n}\overline{x}' \qquad [3.14]$$

donde:

$\overline{x}_n'$, es la estimación de la variable para el instante n.

$\overline{x}_{n-1}'$, es la estimación de la variable en el instante n-1.

$\overline{x}'$, es el valor calculado para la variable.

El valor calculado para la variable $\overline{x}'$ se genera aplicando las ecuaciones de transformación inversa [3.12]. Para el caso de la variable `velocidad`, ésta se calculará a partir de la desviación con respecto al centro de celda, que se produce en la transición local. A continuación a partir de la Figura 3-9 se va a describir el método llevado a la práctica para la estimación del valor de velocidad en la transición ocurrida en el origen (Tabla-Modelo).

La Tabla-Modelo almacena el estado imagen al que llegaría el vehículo si éste estuviera en el estado origen [3.13] y se le aplicara la misma acción de control que la que se aplicó para provocar la última transición del coche en su movimiento real continuo. Conviene recordar que por cada transición real que se produce en el coche a lo largo de su movimiento, se actualiza la Tabla-Modelo. La velocidad inicial, V_1, con la que partía el vehículo cuando generó la transición pertenecerá a una celda determinada. Como se puede observar en la Figura 3-9, esta velocidad no tiene por qué coincidir con la velocidad central de la celda V_{C1} ($\Delta V = V_1 - V_{C1}$). También se puede ver en la misma figura que la velocidad alcanzada por el vehículo en la transición es V_2 que tampoco coincide en general con el centro de celda ($\Delta V' = V_2 - V_{C2}$). Por lo tanto, la velocidad que se almacenará en la Tabla-Modelo será igual a $V_2 - \Delta V$.

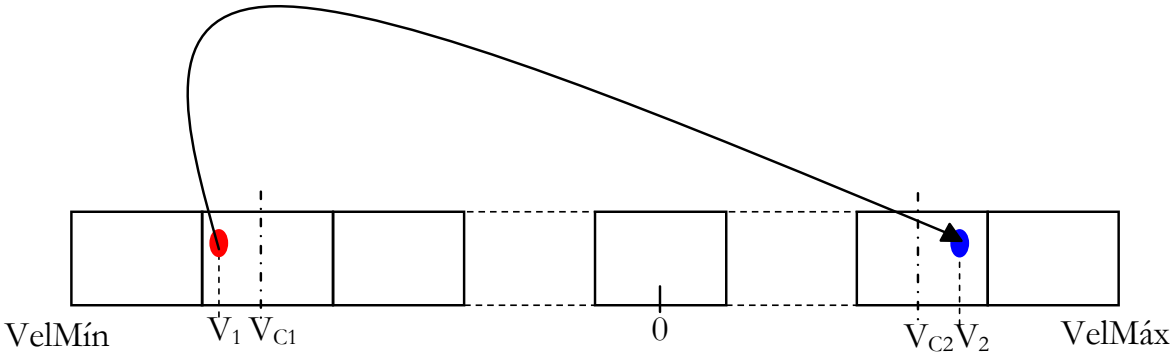


Figura 3-9: Transiciones en la variable `velocidad`

3.4 Conclusiones

En este Capítulo 3 se ha presentado, descrito, diseñado y evaluado un nuevo algoritmo de control para llevar a cabo una planificación óptima de movimiento en vehículos móviles autónomos no holónomos.

Debido a que el nuevo algoritmo (véase Figura 3-1) se puede considerar un híbrido de las técnicas CACM y RL, es importante indicar las ventajas derivadas de la integración realizada:

1. Ausencia de modelo matemático. La diferencia más significativa con respecto a CACM es la ausencia de un modelo matemático a partir del cual se lleve a cabo el aprendizaje. En su lugar, se interacciona directamente con el entorno creando de una manera implícita el modelo real del sistema/entorno.
2. Mayor flexibilidad en las búsquedas de control óptimo que en el caso de CACM. Con el nuevo algoritmo se puede optar a hacer un barrido por cada uno de los estados para determinar el valor de la acción de control óptima a ese estado y así, generar una tabla de control óptimo formada por tantas filas como estados se tengan y una columna asociando el valor de la acción de control óptima al estado.

También es posible hacer un tipo de búsqueda más selectiva a partir de un estado determinado y así crear un árbol de búsqueda jerárquico. Este árbol va a partir de un estado objetivo.

3. Reducción del número de parámetros de ajuste con respecto a Q-learning. En concreto la tasa de aprendizaje α y el factor de descuento γ. De acuerdo a la ecuación [3.1], aunque α se ha fijado a un valor constante, el algoritmo puede perfectamente ajustar automáticamente de forma implícita este valor promediando la recompensa en función del número de veces que se pasa por el mismo estado. En cuanto al parámetro γ, debido a que el criterio de optimización del controlador es tiempo mínimo, cualquier transición que se produzca siempre va a tener asociada una recompensa con un cierto valor (véase ecuación [3.3]). Por esta razón, se dice que la recompensa es "apropiada" [Ber-96], lo cual significa que la convergencia está asegurada [Bar-95] y por ello, el parámetro γ puede ser fijado a 1, que es como si no se utilizara.

   Sin embargo, en aquellos problemas donde sólo se obtiene recompensa cuando se alcanza el objetivo, es necesario establecer el parámetro γ a un cierto valor, para ponderar la distancia al objetivo desde cualquier estado.

4. El tiempo de muestreo no es crítico. En el algoritmo presentado, el tiempo de muestreo utilizado para ejecutar el controlador no es un parámetro crítico ya que independientemente del valor al que se fije, no se producen variaciones en su controlabilidad. Sin embargo, en los algoritmos basados exclusivamente en aprendizaje por refuerzo, la elección precisa del periodo de muestreo es muy importante, ya que dependiendo de su valor la controlabilidad final puede tener grandes variaciones.

Capítulo 4

Si buscas resultados distintos no hagas siempre lo mismo.

ALBERT EINSTEIN (1879-1955. Científico y Nobel de Física)

4. Resultados

4.1 Introducción

Con objeto de ver los resultados ofrecidos por el nuevo algoritmo propuesto en la Figura 3-1, éste se ha probado considerando por un lado, condiciones puramente cinemáticas y por otro, condiciones y efectos dinámicos en el movimiento del propio vehículo.

4.2 Pruebas de simulación con modelo cinemático

Aprovechándonos de las ventajas de poder aplicar el algoritmo propuesto en esta tesis, independientemente de la naturaleza del problema y del número de variables de estado considerado, primeramente se ha resuelto el problema cinemático en condiciones de simulación, por ser más sencillo que el problema final dinámico expuesto en los apartados 4.3 y 4.4. Como se verá a lo largo del

apartado, el algoritmo ofrece buenos resultados, condición necesaria para poder ser aplicado en un problema más complejo como el dinámico, donde el número de variables de estado es mayor y también influyen de forma implícita algunos efectos externos como el rozamiento, deslizamiento o los pares aplicados directamente a los motores (tracción y dirección), a partir de tensiones en los mismos.

El problema cinemático planteado en esta tesis está basado en un modelo matemático que estudia el sistema desde la perspectiva de las siguientes variables de estado:

- Coordenada X, x
- Coordenada Y, y
- Orientación, θ

El algoritmo además de procesar las variables anteriores, actuará sobre el vehículo mediante la aplicación de dos acciones de control:

- Velocidad angular, ω
- Ángulo de dirección, ψ

En las ecuaciones [3.15] se indica el modelo matemático por el que se describe la evolución de la cinemática del sistema (en este caso el coche o vehículo):

$$\begin{aligned} x' &= R \cdot \omega \cos(\theta)\cos(\psi) \\ y' &= R \cdot \omega \operatorname{sen}(\theta)\cos(\psi) \\ \theta' &= \frac{R \cdot \omega}{d}\operatorname{sen}(\psi) \end{aligned} \qquad [3.15]$$

donde R es el radio de la rueda del coche y d la distancia entre ejes del mismo (véase Apéndice A). El rango de valores que se ha considerado para cada variable en las pruebas de simulación realizadas son los indicados en la Tabla 4-1.

Tabla 4-1: Rango de valores para las variables utilizadas en el problema cinemático

VARIABLE	Rango
X	$-2 \leq x \leq 2$ (m)
Y	$-2 \leq y \leq 2$ (m)
Orientación	$-180^{o} \leq \theta \leq 180^{o}$

El algoritmo se ha probado para varios casos, considerando el mismo conjunto objetivo, C_{OBJ}, para todos:

$$C_{OBJ} \begin{cases} x_x = 0 \\ x_y = 0 \\ x_{orientación} = 0 \end{cases} \qquad [3.16]$$

Para presentar los resultados al problema cinemático se han establecido dos escenarios: uno con un entorno libre de obstáculos y el otro con obstáculos. Para cada escenario se han asignado diferentes casos de prueba cuyo objetivo será generar una trayectoria que alcance el conjunto objetivo [3.16] en tiempo mínimo, y por tanto se aproxime a la solución óptima. Para cada caso de prueba se han considerado las mismas posiciones iniciales del coche, siendo éstas indicadas en la Tabla 4-2.

Por otro lado, cada caso de prueba estará basado tanto en una resolución de rejilla determinada como en un número concreto de acciones de control (véase Tabla 4-3). La resolución de rejilla se refiere al número de celdas establecidas para cada una de las variables de estado indicadas al principio del presente apartado. El número de acciones de control se refiere a la combinación de valores de velocidad y dirección con los que se puede actuar sobre el vehículo (en este caso sobre el modelo indicado en [3.15]). En la Tabla 4-4 se muestran cada uno de los casos de prueba que se han ejecutado en el algoritmo propuesto.

Tabla 4-2: Identificadores de trayectoria para cada posición inicial

Identificador de trayectoria	Posición inicial
$\#T_1$	$x_x = 1.7$ $x_y = -1.7$ $x_{orientación} = \pi$
$\#T_2$	$x_x = -1.7$ $x_y = 1.7$ $x_{orientación} = 0$
$\#T_3$	$x_x = 1.7$ $x_y = 1.7$ $x_{orientación} = 0$

Un aspecto importante que hay que tener en cuenta durante la fase de aprendizaje, es que dependiendo de la rejilla seleccionada (#R1, #R2 o #R3), ha sido necesario establecer una distancia de transición particular (véase apartado 2.2.7). Así por ejemplo, para una rejilla como la $\#R_1$, con una distancia igual a 1 se pueden conseguir buenos resultados, como muestran las Figuras 4-1 y 4-4. Sin embargo, para las rejillas $\#R_2$ y $\#R_3$, por el número elevado de celdas y consecuentemente el pequeño tamaño de las mismas para los rangos utilizados, se ha utilizado una distancia igual a 3, produciendo también buenos resultados y presentados en las Figuras 4-2, 4-3, 4-5 y 4-6.

Tabla 4-3: Rejillas y acciones de control para los casos de prueba

Rejilla		Acciones de control	
$\#R_1$	21x21x21	$\#A_1$	$6\begin{cases}\varpi = -\frac{100}{3}, \frac{100}{3} \ [\text{rad/s}] \\ \psi = -0.4, 0, 0.4 \ [\text{rad}]\end{cases}$
$\#R_2$	45x45x45	$\#A_2$	$16\begin{cases}\varpi = -\frac{100}{3}, \frac{100}{3} \ [\text{rad/s}] \\ \psi = -0.4, -0.286, \ \ldots \ , 0.286, 0.4 \ [\text{rad}]\end{cases}$

Rejilla		Acciones de control	
$\#R_3$	61x61x61	$\#A_3$	$26\begin{cases}\varpi = -\frac{100}{3}, \frac{100}{3} \ [\text{rad/s}] \\ \psi = -0.4, -0.333, \ \ldots \ , 0.333, 0.4 \ [\text{rad}]\end{cases}$

Tabla 4-4: Casos de prueba para el problema cinemático

Caso de prueba	Rejilla	Acciones de control	Escenario
#1	$\#R_1$	$\#A_1$	1
#2		$\#A_2$	
#3		$\#A_3$	
#4	$\#R_2$	$\#A_1$	1
#5		$\#A_2$	
#6		$\#A_3$	
#7	$\#R_3$	$\#A_1$	1
#8		$\#A_2$	
#9		$\#A_3$	
#10	$\#R_1$	$\#A_1$	2
#11		$\#A_2$	
#12		$\#A_3$	
#13	$\#R_2$	$\#A_1$	2
#14		$\#A_2$	
#15		$\#A_3$	
#16	$\#R_3$	$\#A_1$	2
#17		$\#A_2$	
#18		$\#A_3$	

En las Figuras 4-1, 4-2 y 4-3 se muestran los resultados obtenidos una vez aplicado el algoritmo de la Figura 3-1 en un escenario libre de obstáculos, para los casos de prueba #1, #2, #3, #4, #5, #6, #7, #8 y #9.

En la Figura 4-1, con el tamaño de celda considerado para cada caso de prueba (relativamente grande en comparación con las dimensiones del coche –véase Apéndice A–), se observa una mejora de la trayectorias tanto en la forma de alcanzar el objetivo como en el trazado de las mismas, cuantas más acciones de control de dirección se tengan. Además, para verificar esta mejora se puede observar como a medida que aumentan las capacidades de giro del coche, las trayectorias se van aproximando a la curva de conmutación en forma de 'S' de la Figura 3-7.

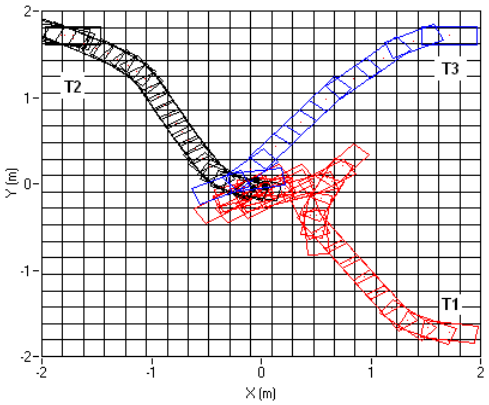
Para el caso concreto de las trayectorias T_1 y T_3 y en el caso de prueba #1, se necesitan hacer varias maniobras con grandes giros (sobre todo en la T_1), mientras que para los casos #2 y #3 el movimiento general de la trayectoria es más curvo y "suave" para alcanzar el objetivo sin giros tan "bruscos". Sin embargo, si nos fijamos en los tiempos invertidos en alcanzar el objetivo no se aprecia una correspondencia favorable (tiempo mínimo) cuando se usan más valores de acciones de control.

En general, es importante interpretar cada una de las trayectorias desde dos perspectivas:

- Cuantas menos acciones de dirección tengamos, las trayectorias generadas serán más rectilíneas y por tanto, los movimientos "bruscos" en la dirección, se producirán principalmente próximos al objetivo.
- Siempre hay que tener en cuenta el efecto indeseado pero siempre presente del proceso de discretización que dependiendo del tamaño de celda e incluso del periodo de muestreo, se manifiesta en movimientos del coche hacia atrás y hacia delante quedando "atrapado" en celdas adyacentes hasta que después de hacerlo varias veces

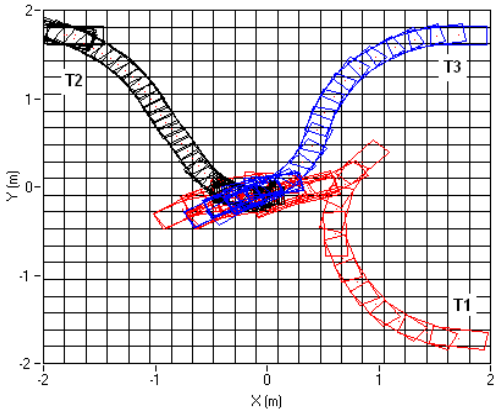
consigue "caer" en otra celda, aplicándose otra acción de control y permitiendo al vehículo salir de esa situación.

#1



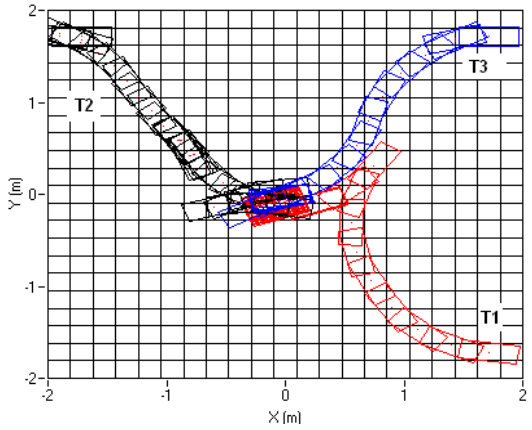
$T_1 \equiv 6.6$ s. $T_2 \equiv 3.2$ s $T_3 \equiv 3.24$ s.

#2



$T_1 \equiv 9.6$ s. $T_2 \equiv 4.84$ s. $T_3 \equiv 5.53$ s.

#3



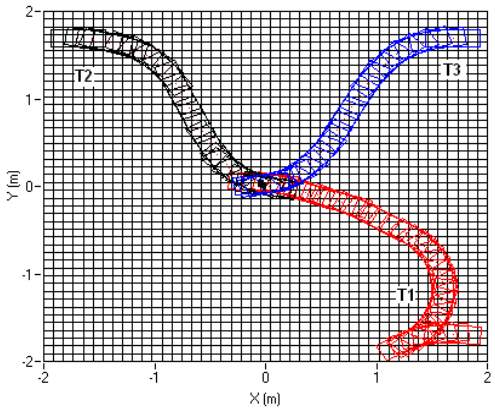
$T_1 \equiv 7.65$ s. $T_2 \equiv 5.8$ s. $T_3 \equiv 4.8$ s.

Figura 4-1: Trayectorias generadas en un escenario libre de obstáculos para los casos de prueba #1, #2 y #3. Se usa rejilla #R1 igual a 21x21x21 y se aplican las acciones #A1, #A2 y #A3 en #1, #2 y #3 respectivamente.

En la Figura 4-2, los casos de prueba considerados se han realizado con un tamaño de rejilla inferior a los de la Figura 4-1. Esto hace que las trayectorias que se generan presenten un trazado más suave, aproximándose mejor a la curva de la Figura 3-7. Además, por utilizar una rejilla más pequeña, el objetivo se alcanza en unos tiempos inferiores para las mismas acciones de control. No obstante, la

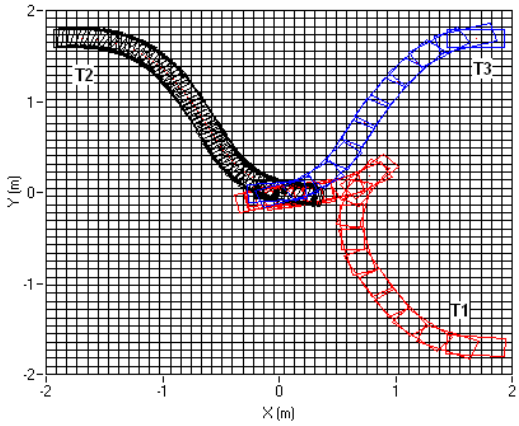
interpretación de cada una de las trayectorias hay que hacerla también teniendo en cuenta las mismas dos perspectivas anteriores.

#4



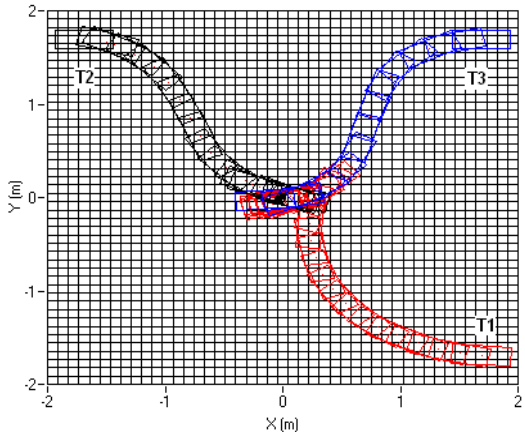
$T_1 \equiv 3.88$ s. $T_2 \equiv 3.25$ s. $T_3 \equiv 2.95$ s.

#5



$T_1 \equiv 5.58$ s. $T_2 \equiv 2.98$ s. $T_3 \equiv 3.15$ s.

#6

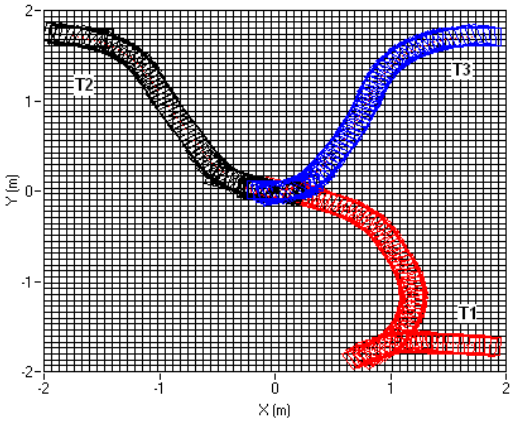


$T_1 \equiv 5.51$ s. $T_2 \equiv 3.6$ s. $T_3 \equiv 3.24$ s.

Figura 4-2: Trayectorias generadas en un escenario libre de obstáculos para los casos de prueba #4, #5 y #6. Se usa rejilla #R2 igual a 45x45x45 y se aplican las acciones #A1, #A2 y #A3 en #4, #5 y #6 respectivamente.

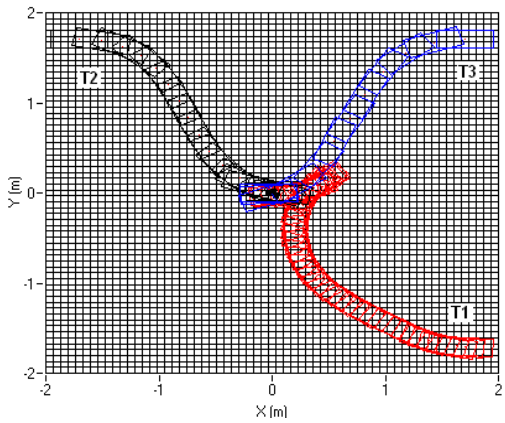
En la Figura 4-3 no se aprecian grandes cambios por el hecho de utilizar una rejilla de 61x61x61. El coste computacional derivado del aumento de celdas no aporta ventajas considerables con respecto a la rejilla #R2, ya que tanto los tiempos en alcanzar el objetivo como los movimientos realizados por el coche, son muy similares.

#7



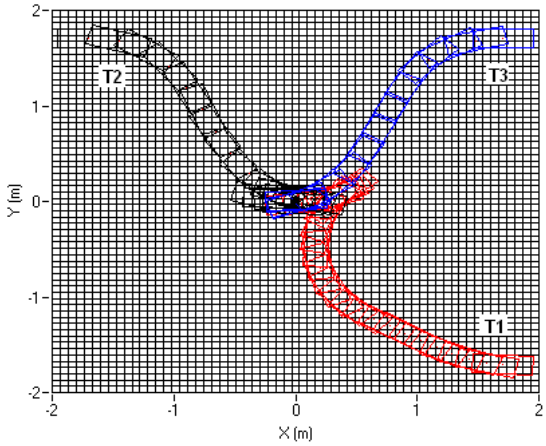
$T_1 \equiv 3.85$ s. $T_2 \equiv 3.5$ s. $T_3 \equiv 3.1$ s.

#8



$T_1 \equiv 3.72$ s. $T_2 \equiv 3.36$ s. $T_3 \equiv 3.3$ s.

#9

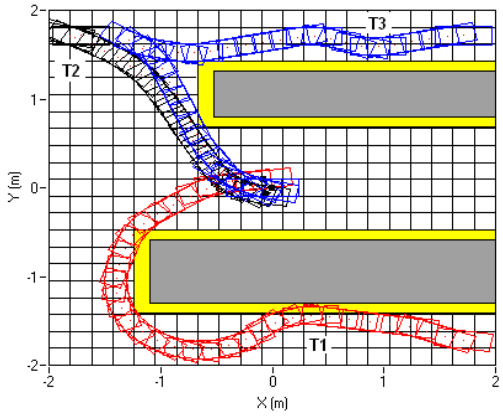


$T_1 \equiv 3.85$ s. $T_2 \equiv 4.59$ s. $T_3 \equiv 3.36$ s.

Figura 4-3: Trayectorias generadas en un escenario libre de obstáculos para los casos de prueba #7, #8 y #9. Se usa rejilla #R3 igual a 61x61x61 y se aplican las acciones #A1, #A2 y #A3 en #7, #8 y #9 respectivamente.

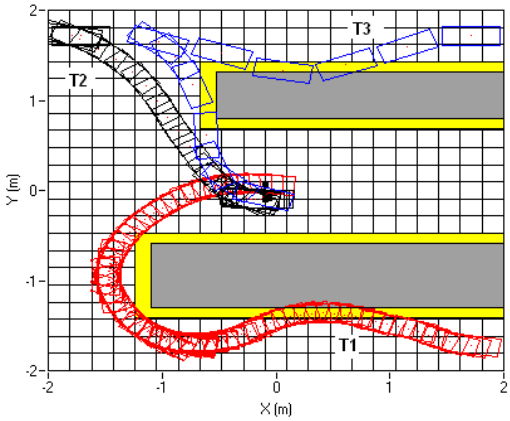
En las Figuras 4-4, 4-5 y 4-6 se muestran los resultados obtenidos aplicando el algoritmo en un escenario con obstáculos, para los casos de prueba #10, #11, #12, #13, #14, #15, #16, #17 y #18. Rodeando a cada uno de los obstáculos se han previsto unas zonas de guarda. Comparando las Figuras 4-1 y 4-4, se observa un cambio de trazado en las trayectorias T_1 y T_3 a las que les afecta directamente los obstáculos colocados.

#10



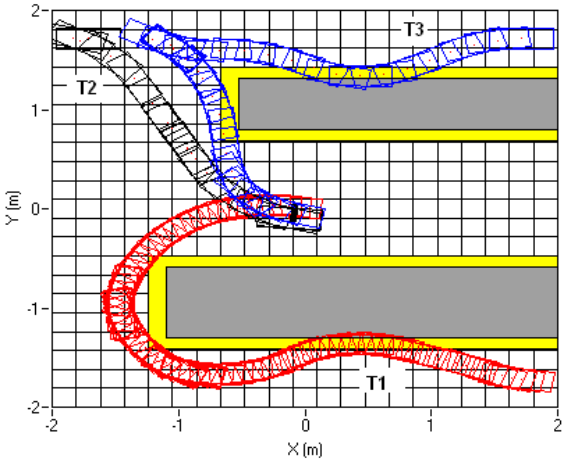
$T_1 \equiv 5.94$ s. $T_2 \equiv 3.2$ s. $T_3 \equiv 6.4$ s.

#11



$T_1 \equiv 8$ s. $T_2 \equiv 4.84$ s. $T_3 \equiv 7.26$ s.

#12



$T_1 \equiv 8.44$ s. $T_2 \equiv 5.25$ s. $T_3 \equiv 7.54$ s.

Figura 4-4: Trayectorias generadas en un escenario con obstáculos para los casos de prueba #10, #11 y #12. Se usa rejilla #R1 igual a 21x21x21 y se aplican las acciones #A1, #A2 y #A3 en #10, #11 y #12 respectivamente.

En general, con la presencia de obstáculos alcanzar el mismo objetivo desde los mismos orígenes implicará un mayor tiempo. Pero en la práctica y debido al número de acciones de control en dirección y al efecto indeseado del proceso de discretización, como se comentó para la Figura 4-1, a veces el tiempo invertido en alcanzar el objetivo es incluso inferior con la presencia de obstáculos (por ejemplo para la trayectoria T_1 en los casos de prueba #10 y #11). Para la trayectoria

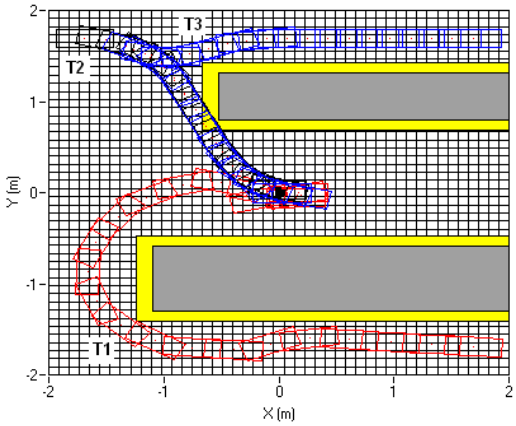
T_2 son similares, incluso exactamente iguales para los casos de prueba #10 y #11.

Debido a los valores de las acciones de control para la dirección y sobre todo al tamaño de rejilla utilizada en la Figura 4-4, a veces la zona de guarda prevista se queda demasiado pequeña sobrepasándola el vehículo y llegando incluso a chocar con el obstáculo (por ejemplo la trayectoria T_3 del caso de prueba #11).

En el caso de la Figura 4-5, la comparación con respecto a la Figura 4-2 en los tiempos invertidos por las trayectorias afectadas por los obstáculos (T_1 y T_3), sí son mayores en los tres casos de prueba #13, #14 y #15, como es lo más lógico desde un punto de vista teórico. Para la trayectoria T_2 los tiempos son similares. Si los tiempos son comparados con los obtenidos para las trayectorias de la Figura 4-4, en algunos casos son inferiores (por ejemplo, T_3 de #13, T_1 de #14, T_2 de #14, T_2 de #15 y T_3 de #15), y en otros superiores (por ejemplo, T_1 de #13, T_2 de #13 y T_3 de #14). La razón de estas diferencias está fundamentalmente basada en el efecto indeseado de la discretización comentado para la Figura 4-1.

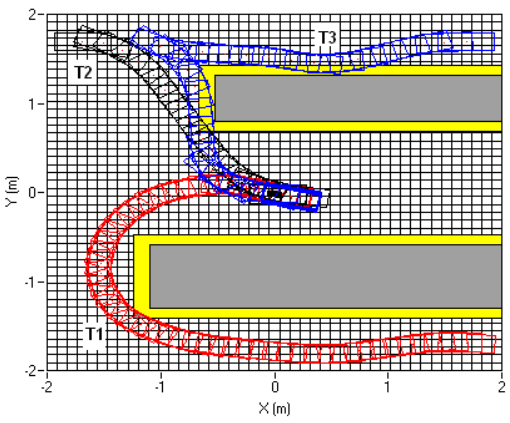
En esta figura y sólo con el hecho de aumentar el número de celdas en cada variable en comparación con la Figura 4-4, las zonas de guarda sí se respetan sin necesidad de haberlas ampliado. Además, los trazados de cada una de las trayectorias, de nuevo por el hecho de utilizar una mayor resolución en la rejilla, son más suaves que los mostrados en la Figura 4-4.

#13



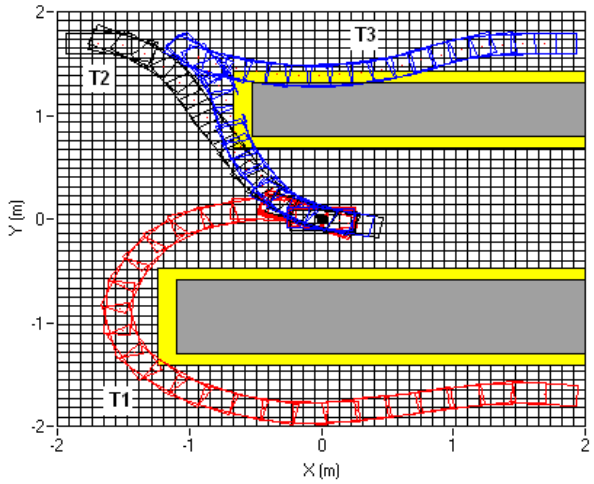
$T_1 \equiv 7.74$ s. $T_2 \equiv 3.4$ s. $T_3 \equiv 5.76$ s.

#14



$T_1 \equiv 6.3$ s. $T_2 \equiv 3.4$ s. $T_3 \equiv 8.4$ s.

#15

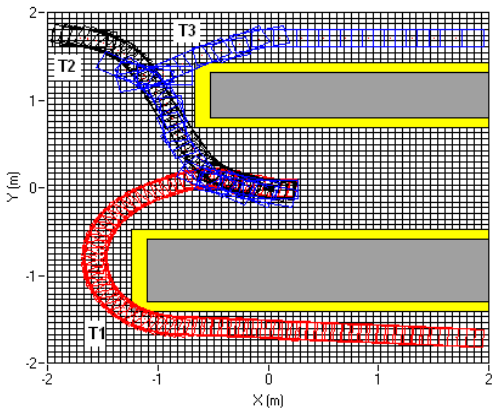


$T_1 \equiv 8.4$ s. $T_2 \equiv 3.4$ s. $T_3 \equiv 5.44$ s.

Figura 4-5: Trayectorias generadas en un escenario con obstáculos para los casos de prueba #13, #14 y #15. Se usa rejilla #R2 igual a 45x45x45 y se aplican las acciones #A1, #A2 y #A3 en #13, #14 y #15 respectivamente.

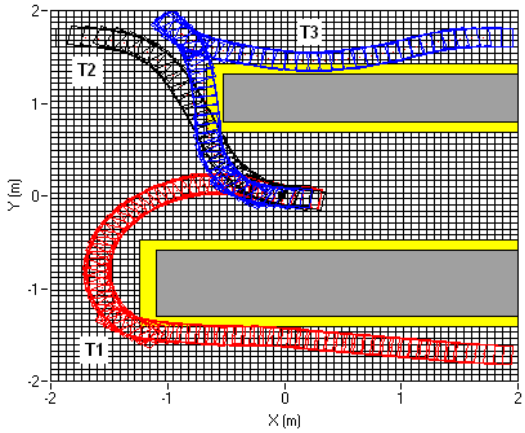
Por último, al igual que ocurría con la Figura 4-3, en la Figura 4-6 no se aprecian grandes cambios favorables por utilizar una rejilla de 61x61x61. Por tanto, lo mismo que se dijo para la Figura 4-3 sigue siendo válido para ésta.

#16



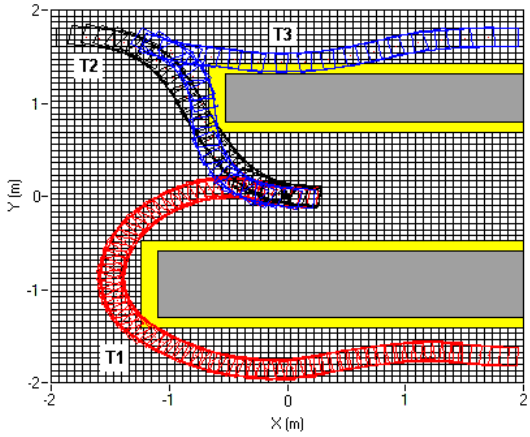
$T_1 \equiv 6.16$ s. $T_2 \equiv 3.28$ s. $T_3 \equiv 7$ s.

#17



$T_1 \equiv 6.56$ s. $T_2 \equiv 3.24$ s. $T_3 \equiv 5.84$ s.

#18



$T_1 \equiv 6$ s. $T_2 \equiv 3.18$ s. $T_3 \equiv 5.88$ s.

Figura 4-6: Trayectorias generadas en un escenario con obstáculos para los casos de prueba #16, #17 y #18. Se usa rejilla #R3 igual a 61x61x61 y se aplican las acciones #A1, #A2 y #A3 en #16, #17 y #18 respectivamente.

4.3 Efecto dinámico en el movimiento

En el movimiento real del vehículo representado en la Figura 3-2 se van a ver involucrados diversos efectos dinámicos (tiempo en alcanzar un ángulo de dirección, rozamiento, inercia), que van a aumentar la complejidad del controlador y a su vez, dependiendo de

la intensidad del efecto dinámico en cuestión, disminuir la controlabilidad. Por ejemplo, la Figura 4-7 muestra diversos movimientos del vehículo para distintas velocidades del ángulo de dirección. Se puede observar que cuanto más rápida es dicha velocidad antes se alcanza el ángulo deseado y por tanto el efecto dinámico va disminuyendo aproximándose al caso ideal, donde el ángulo se alcanzaría instantáneamente. Cuando la velocidad de dirección es lenta, a veces se podría transitar a otro estado sin haber llegado a la acción de ángulo marcada como consigna.

Con objeto de realizar un control óptimo en la planificación del movimiento teniendo en cuenta los efectos dinámicos que se puedan producir, se ha considerado el vehículo como un sistema con cuatro variables de estado, siendo una de ellas la velocidad que actúa como componente dinámica. Igual que en el movimiento cinemático, el vehículo conforme va ejecutando el nuevo algoritmo va aprendiendo, en este caso, tanto la cinemática como la propia dinámica. Durante la ejecución, el algoritmo hace uso de la información incluida en el apartado 3.3.1 para la creación de la Tabla-Modelo. Por tanto, las variables incorporadas para considerar el efecto dinámico son:

- Velocidad, v
- Coordenada X, x
- Coordenada Y, y
- Orientación, θ

El algoritmo aplica las siguientes acciones de control:

- Par de tracción, $\tau_{\text{tracción}}$
- Ángulo de dirección, ψ

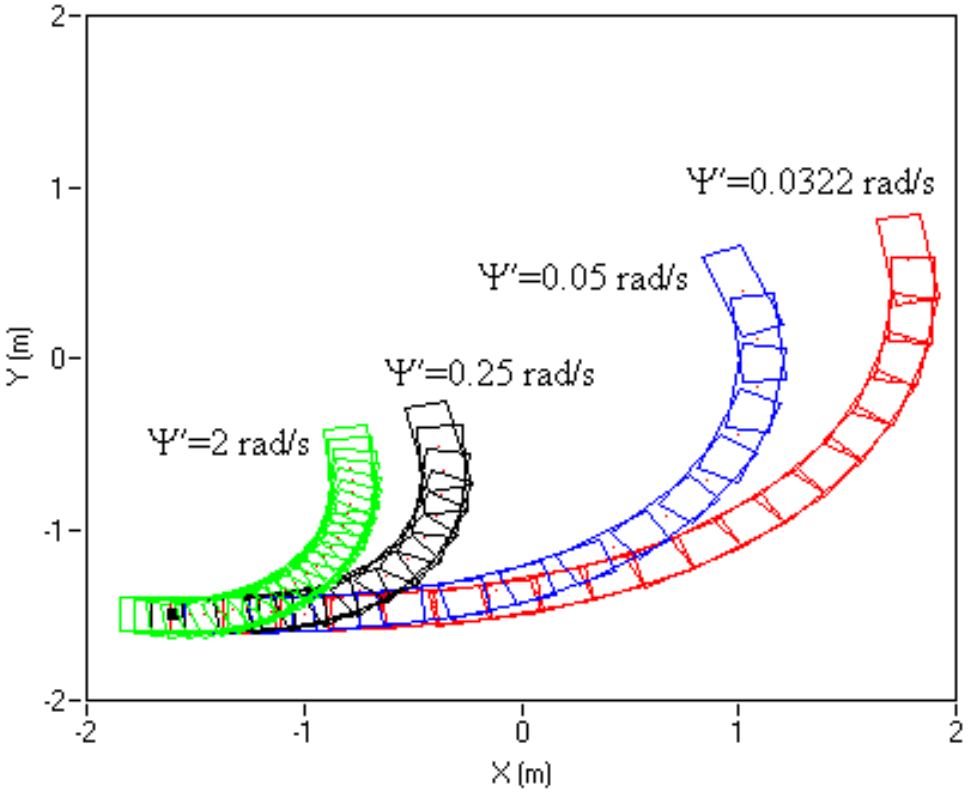


Figura 4-7: Efecto de la velocidad de dirección en el movimiento

Conviene puntualizar que el par de tracción se obtiene a partir de la aplicación de la tensión correspondiente al motor de tracción. Por lo tanto, no es una acción directa introducida por el algoritmo ya que éste aplica tensión. Por tanto, este par se ha tenido en cuenta solamente para simulaciones, haciendo uso del siguiente modelo matemático (véase Apéndice A) que describe la dinámica del movimiento del coche:

$$\begin{aligned} \omega' &= (\tau_{\text{tracción}} - (f_9 \cdot \omega \cdot V_\psi)) / f_7 \\ x' &= R \cdot \omega \cos(\theta) \cos(\psi) \\ y' &= R \cdot \omega \operatorname{sen}(\theta) \cos(\psi) \\ \theta' &= \frac{R \cdot \omega}{d} \operatorname{sen}(\psi) \end{aligned} \qquad [3.17]$$

donde ω es la velocidad angular de tracción del vehículo, R el radio de la rueda del coche y d la distancia entre ejes del mismo. La función f_9 depende del ángulo de dirección ψ, y se encuentra definida en el Apéndice A. La variable V_ψ se corresponde con la velocidad de dirección. Aunque para tener en cuenta los efectos dinámicos se ha establecido un espacio de estados de cuatro variables, para las simulaciones llevadas a cabo se han considerado dos variables más que no forman parte de dicho espacio: ángulo de dirección y velocidad de dirección. Tal y como se ha comentado anteriormente, una de las acciones de control que aplica el algoritmo es precisamente el ángulo de dirección. Desde el punto de vista de la simulación, este ángulo será la consigna que se deberá alcanzar cuando se intente aplicar a la dirección. Sin embargo, y tal como se representa en la Figura 4-7 alcanzar o no alcanzar este ángulo en un tiempo inferior al de transición dependerá de la velocidad de dirección (véanse ecuaciones [3.17]). Cuanto más elevada sea esta velocidad, la variación del ángulo mejor se aproximará al caso instantáneo.

En la Figura 4-8 se representan tres gráficas donde se han reflejado tres trayectorias en cada una de ellas. Cada gráfica se corresponde con una velocidad de dirección determinada. Se observa que a mayor velocidad se consigue una mayor facilidad de giro del coche. Con una velocidad de 1 rad/s y con el periodo de muestreo considerado, al vehículo apenas le da tiempo alcanzar la consigna de dirección establecida, y en general el resultado es un movimiento con tramos excesivamente rectos. Esto provoca mayor dificultad para alcanzar el objetivo como se puede observar. Sin embargo, el hecho de aumentar a 2 rad/s la velocidad, manteniendo el periodo de muestreo, implica poder alcanzar la consigna de dirección y por tanto, favorece el trazado de la trayectoria y la llegada al objetivo.

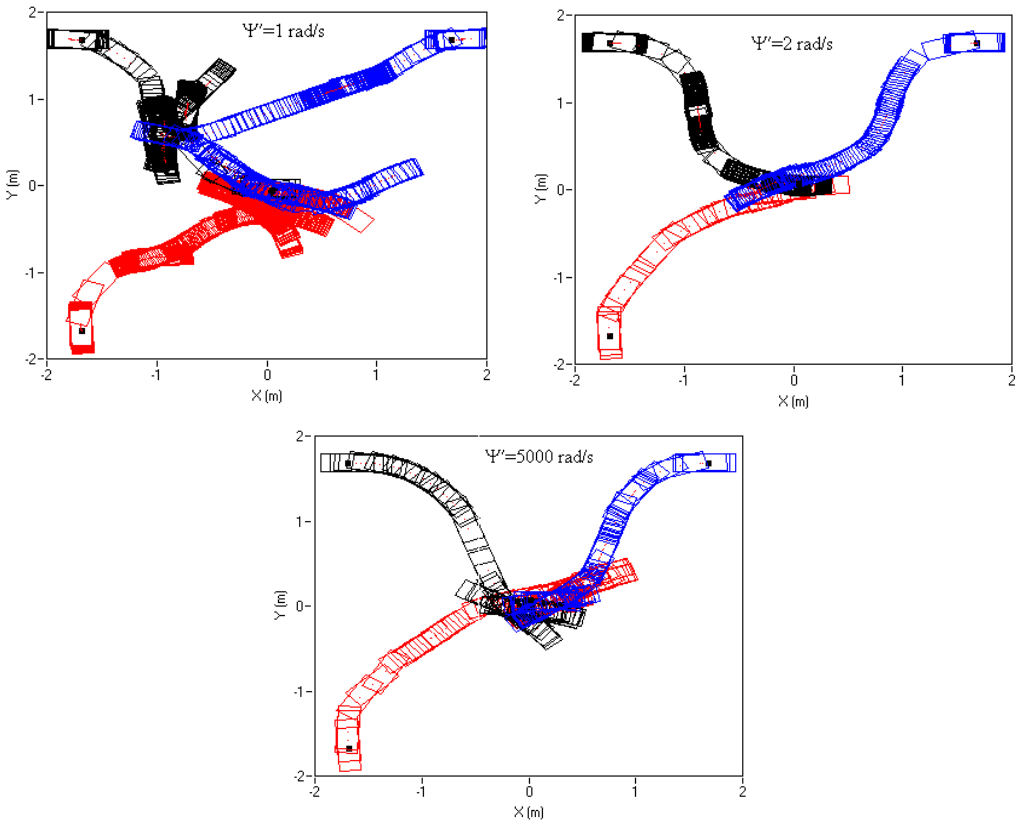


Figura 4-8: Efecto de la velocidad de dirección en la planificación del movimiento. Es significativa, la gráfica cuya velocidad de dirección es de 1 rad/s donde las trayectorias tienen poca curvatura. Un efecto contrario ocurre en los otros casos.

La gráfica correspondiente a la velocidad de dirección de 5000 rad/s se ha generado para simular el comportamiento de un alcance instantáneo de la consigna de dirección. Ahora, evidentemente esta consigna se alcanza inmediatamente y al mantenerse durante el periodo de muestreo hace que el coche haga un mayor recorrido de giro que en el caso anterior. Este efecto puede provocar que en estados próximos al objetivo, el vehículo tenga que maniobrar algo más como así ocurre.

4.4 Pruebas reales

Para comprobar la viabilidad y bondad del algoritmo propuesto en la Figura 3-1 se ha considerado un escenario real de pruebas, mostrado de forma esquemática en la Figura 4-9.

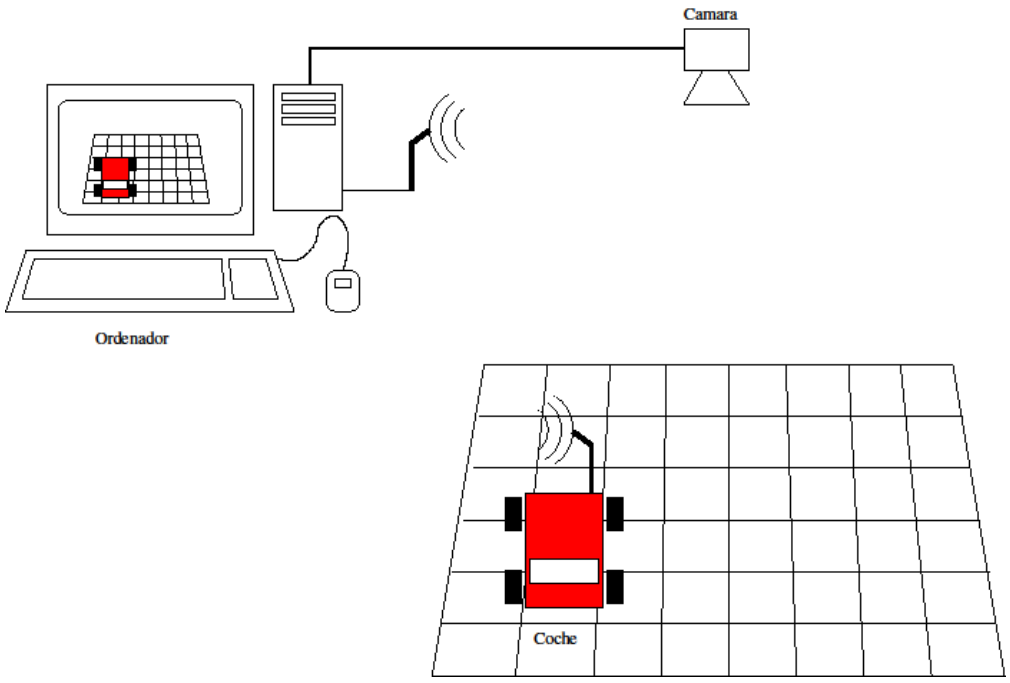


Figura 4-9: Arquitectura hardware del escenario de pruebas

La idea de montar esta arquitectura ha sido tener la información de localización del vehículo centralizada en el ordenador, de manera que la cámara esté continuamente enviándole la posición y orientación, para que cuando el algoritmo que se ejecuta en el coche necesite esa información, la solicite al ordenador vía radio. De esta manera, mediante la ejecución del nuevo algoritmo, el vehículo aprenderá su propia dinámica sin necesidad de modelos matemáticos.

Espacio de estados

El espacio de estados tenido en cuenta para la ejecución del algoritmo propuesto considera las cuatro variables descritas en el

apartado 3.2.1, con valores acotados, discretizados y con un número de celdas, N_C, indicados en la Tabla 4-5. En la Figura 4-11 se muestra el recinto (plano X-Y) por donde el vehículo va a desplazarse ejecutando el aprendizaje y la planificación de movimiento.

Tabla 4-5: Variables de estado para las pruebas reales

VARIABLE	N_C	Rango
Velocidad	3	$-1.5 \leq v \leq 1.5$ (m/s)
X	9	$-90 \leq x \leq 90$ (cm)
Y	13	$-130 \leq y \leq 130$ (cm)
Orientación	31	$-180^o \leq \theta \leq 180^o$

A continuación, se justifica el motivo de haber asignado un número de celdas igual al que se indica en la Tabla 4-5. En primer lugar, hay que tener en cuenta que debido a la aplicabilidad de las ecuaciones de transformación (véase apartado 3.3), con sólo generar la Tabla-Modelo a partir de las transiciones generadas por el vehículo en sus movimientos, podríamos considerar que el aprendizaje queda realizado. A modo de ejemplo, en la Figura 4-10 se muestra un modelo generado en el aprendizaje de un experimento, considerando el vehículo inicialmente parado. En la gráfica de dicha figura se representan hasta seis transiciones desde el estado origen (0, 0, 0, 0). Las transiciones son provocadas a partir de seis acciones de control indicadas en la Tabla 4-6.

Tabla 4-6: Acciones de control utilizadas en el aprendizaje de un experimento para la generación de un modelo

Par de tracción, τ	Ángulo de dirección, ψ
$\tau > 0$	$\psi < 0$
	$\psi = 0$
	$\psi > 0$
$\tau < 0$	$\psi < 0$
	$\psi = 0$
	$\psi > 0$

Un número elevado de celdas influye principalmente en el tiempo invertido en realizar diversas actualizaciones a la Tabla-Q (véase apartado 3.2.2). Este tiempo dependerá fundamentalmente de la velocidad de procesador que para nuestro caso ha sido el LEON3 (véase Apéndice B.3). Con las celdas de la Tabla 4-5, una actualización completa de la Tabla-Q se realiza en un tiempo de aproximadamente 3 s.

Las dimensiones del área reflejada en la Figura 4-11 son de 1.80 m en el eje X y 2.60 m en el Y. Para que el tamaño de celda en cada dimensión sea el mismo, y por tanto, tener la misma precisión, el número de celdas en X e Y ha de ser diferente, e igual a 9 y 13 respectivamente, obteniendo un tamaño de celda igual a 20 cm de lado. En cuanto a la orientación, se ha ajustado a 31, para además de conseguir precisión, ser capaces de que cuando se transite en X o Y, yendo el vehículo a velocidad mínima con las ruedas de dirección giradas al máximo, también se transite en orientación. Para la velocidad no se ha optado por tener precisión y 3 celdas han sido suficientes para probar el comportamiento del algoritmo. Un detalle común al número de celdas de cada variable, es que éste es impar para tener siempre el 0 como valor central de una de las celdas.

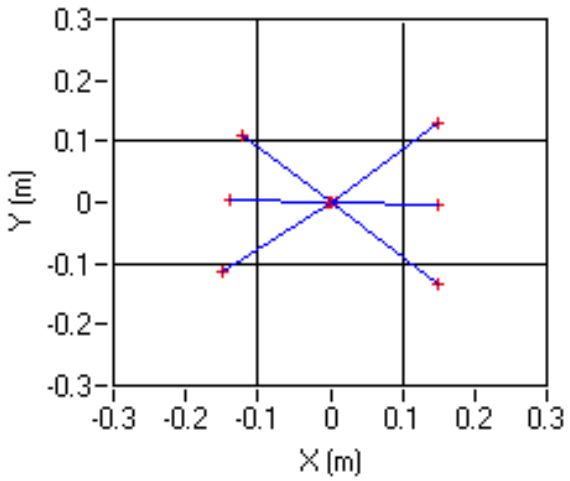


Figura 4-10: Ejemplo del modelo obtenido durante el aprendizaje de un experimento

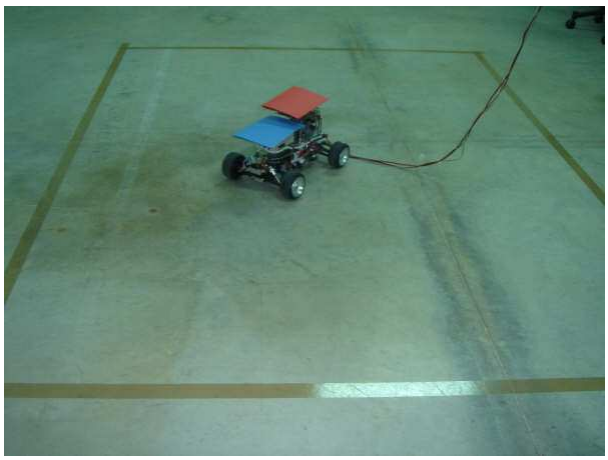


Figura 4-11: Plano X-Y por donde se mueve el vehículo

En la Tabla 4-7 se indican las dos acciones de control utilizadas por el algoritmo, la discretización utilizada para cada una de ellas, N_U, y los valores que pueden adquirir. Conviene prestar especial atención al valor 0 V. que podría adquirir la acción correspondiente al control de tracción. El algoritmo de la Figura 3-1 sólo aplicará esta acción cuando el coche esté moviéndose. Sin embargo, en el supuesto de que no lo esté, esta acción de 0 V. no se aplicaría ya que crearía un punto de equilibrio espurio y esta situación no es deseable. En su lugar se aplicaría cualquiera de los otros dos valores.

Las dos acciones son aplicadas durante un periodo de muestreo, transcurrido el cual se analiza el nuevo estado, de acuerdo a lo descrito en el apartado 3.2.2. Las velocidades mínima y máxima que el vehículo es capaz de adquirir aplicando las tensiones mínima y máxima en el motor de tracción son -1.3 m/s y 1.3 m/s respectivamente.

Tabla 4-7: Discretización de acciones para las pruebas reales

ACCIÓN	N_U	VALORES		
Tensión en motor de tracción	3	-18 V.	0 V.	18 V.
Ángulo dirección	3	-23º	0º	23º

Cámara

El estudio y análisis de técnicas odométricas no son objeto de esta tesis, y por tanto, para proporcionar la información de localización del vehículo al nuevo algoritmo, nos vamos a apoyar en una solución de laboratorio basada en un dispositivo externo como es el modelo de cámara CMUcam3. Con las imágenes captadas por esta cámara se van a realizar los cálculos de posicionamiento necesarios para el algoritmo.

La cámara se encuentra instalada en el techo, donde su campo de visión se ha ajustado al recinto de la Figura 4-11. El método seguido para obtener la localización del coche ha sido la implementación en el propio software de la cámara de un programa que permita hacer *tracking* de dos colores (azul y rojo) y calcule el centroide de cada uno de ellos, para de esta manera poder posteriormente calcular la posición del vehículo y su orientación. Existe una limitación inherente a la cámara de 300 ms. de retardo en captar una imagen. Esta restricción ha sido tenida en cuenta tanto para fijar el tamaño de las celdas como para establecer el periodo de muestreo.

Mediante el programa de *tracking* ejecutándose en la cámara se obtienen las coordenadas (X_{cen_azul}, Y_{cen_azul}) y (X_{cen_rojo}, Y_{cen_rojo}). De acuerdo a la Figura 3-2, la cartulina de color azul se ha situado en la parte delantera del coche y la roja en la trasera. Teniendo en cuenta esto, el criterio de signos seguido para el cálculo de la orientación, θ, es el representado en la Tabla 4-8.

La orientación puede variar entre $0^o \le \theta \le -180^o$ y $0 \le \theta \le 180^o$. De acuerdo a la Tabla 4-8, la orientación, $\theta = 0^o$, se consigue cuando el vehículo está mirando hacia valores positivos del eje X, mientras que en el caso inverso, es decir, cuando mira hacia valores negativos, su orientación sería, $\theta = 180^o$ o $\theta = -180^o$. Sin embargo, en el caso de que el coche mire hacia valores positivos del eje Y su orientación sería $\theta = 90^o$ y $\theta = -90^o$ en el caso inverso. Por tanto, habiendo

obtenido los centroides de cada uno de los colores considerados, los cálculos de la posición y orientación se pueden deducir fácilmente.

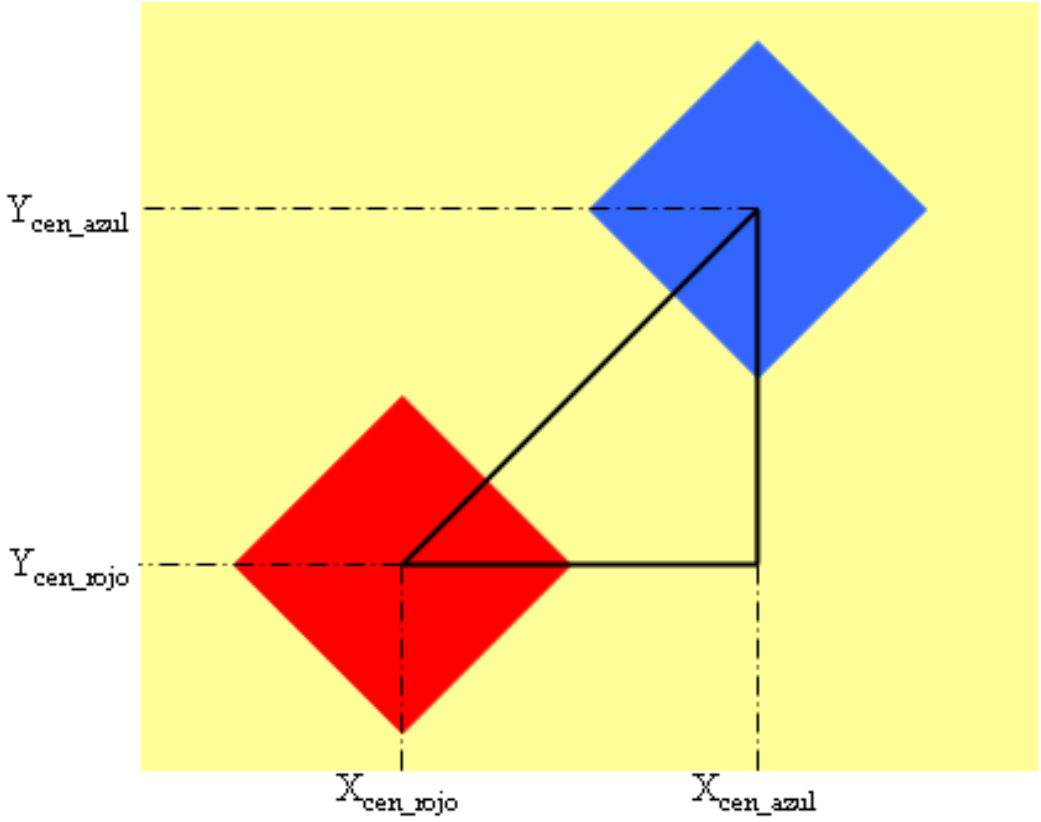
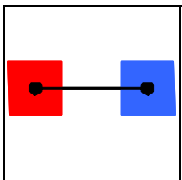
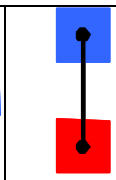
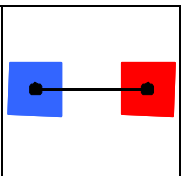
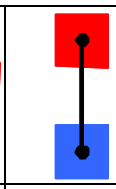


Figura 4-12: Cálculo de la orientación del vehículo

Tabla 4-8: Criterios de signo para la orientación

			
$\theta = 0^{o}$	$\theta = 90^{o}$	$\theta = 180^{o}$, o $\theta = -180^{o}$	$\theta = -90^{o}$

Ordenador

El ordenador va a servir como punto de enlace entre la cámara y el vehículo, de manera que siempre tendrá disponible la última posición de aquél. Además, se ha optado por esta solución debido a que la cámara no tiene comunicación inalámbrica y de esta manera se facilita la conexión con el coche que sí la tiene, y es cómo se comunica con el ordenador (véase Apéndice B.7). Además, este ordenador va a servir como Centro de Control del vehículo, inspirado en [Gom-03], con una interfaz hombre-máquina disponible para labores de depuración.

El algoritmo propuesto en la Figura 3-1 se ejecuta de forma autónoma sobre el propio vehículo y sólo cada periodo de muestreo se solicitará la información de localización al ordenador vía *bluetooth*. Éste recibe continuamente esta información por parte de la cámara, por lo que cada vez que se la solicite mandará la última que tuviera almacenada.

Periodo de muestreo

Debido a la limitación hardware existente en la cámara en cuanto a su retardo, tanto el periodo de muestreo como el tamaño de las celdas se han calculado en función del periodo. Además, tal y como se ha comentado anteriormente, la localización enviada al vehículo será la última que hubiera captado la cámara por lo que se evitarán retardos en el algoritmo a costa de introducir un pequeño error en la localización por no ser la real del coche, sino la que éste tenía unos milisegundos antes. Este error no es significativo de cara al algoritmo siempre que el periodo de muestreo sea superior a los 300 ms. de retardo de la cámara. Además, con esta solución de centralizar en el ordenador la información de posicionamiento, evitaremos poder asignar un periodo de muestreo que provoque una solicitud a la cámara mientras ésta se encuentra todavía capturando una imagen de una solicitud anterior.

Por todo lo anterior, y debido a los valores mínimo y máximo que se han considerado para cada variable, se puede deducir que el valor del periodo de muestreo será elevado por lo que no tendrá mucho sentido fijar tamaños de celda demasiado pequeños, ya que se estaría muestreando en instantes de tiempo donde el coche todavía permanecería en la misma celda. También, es importante asignar un tamaño de celda para las variables X e Y que sea inferior al del coche (véase Apéndice A), por lo que este tamaño queda fijado en 20 cm, siendo cubierto el vehículo con tres celdas en el plano X-Y.

Como conclusión final, el periodo de muestreo que se ha fijado para las diferentes pruebas realizadas ha sido de 0,8 s. consiguiendo resultados satisfactorios como muestran los vídeos introducidos en [Gom-09c]. En [Gom-09c] se recoge el comportamiento real del algoritmo propuesto en esta tesis. Se puede observar en cada uno de los vídeos, la fiabilidad y el comportamiento esperado de la planificación de movimiento realizada en el vehículo para alcanzar el objetivo.

A continuación y a partir de los vídeos incluidos en [Gom-09c], se muestran distintas figuras que reflejan mediante fotogramas el movimiento real del vehículo para cada uno de los casos de prueba especificados en la Tabla 4-9. A la vista de la Tabla 4-9, los diferentes casos de prueba se podrían clasificar en dos grupos: un primer grupo para alcanzar el objetivo (0 m/s, 0 m, 0 m, 0º), y un segundo grupo para alcanzar el objetivo (0 m/s, 0.86 m, -0.05 m, 24.65º). Para cada grupo, el coche arranca desde distintos orígenes para llegar al conjunto objetivo considerado.

Tabla 4-9: Casos de prueba reales considerando dinámica

Caso de prueba	Estado origen	Estado objetivo
#1	v = 0 m/s x = –0.36 m y = –0.95 m θ = 42.4º	v = 0 m/s x = 0 m y = 0 m θ = 0º

Caso de prueba	Estado origen	Estado objetivo
#2	v = 0 m/s x = 0.74 m y = 0.93 m θ = –103.7º	v = 0 m/s x = 0 m y = 0 m θ = 0º
#3	v = 0 m/s x = –0.46 m y = 1.25 m θ = 142.1º	
#4	v = 0 m/s x = 0.81 m y = –0.94 m θ = –68.7º	
#5	v = 0 m/s x = 0.46 m y = –1.2 m θ = 0º	v = 0 m/s x = 0.86 m y = –0.05 m θ = 24.65º
#6	v = 0 m/s x = 0.25 m y = 0.66 m θ = 28.66º	
#7	v = 0 m/s x = 0.42 m y = –1.15 m θ = 141.02º	
#8	v = 0 m/s x = –0.5 m y = 0.93 m θ = 66º	

En el caso de prueba #1 de la Figura 4-13, dado que el coche se encuentra en el tercer cuadrante (Figura 4-13a) y el objetivo está en el centro de coordenadas, su arranque ha de ser hacia delante con las ruedas giradas a la izquierda (Figura 4-13b), para seguir la hipotética curva teórica de la Figura 3-7. Según avanza va invirtiendo el sentido de giro de las ruedas para aproximarse al objetivo (Figura 4-13c). Debido a la granularidad de celda y valor del periodo de muestreo, en vez de entrar en el objetivo, se pasa de él (Figura 4-13d) y tiene que dar marcha atrás girando a la izquierda (Figura 4-13e), para posicionarse y entrar en el objetivo de frente (Figura 4-13f).

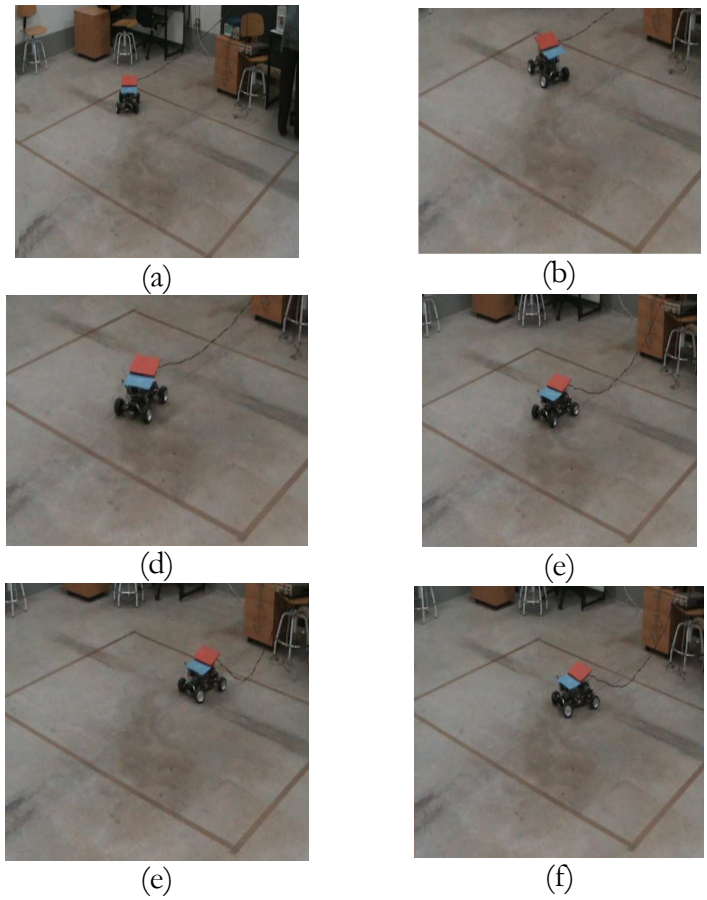
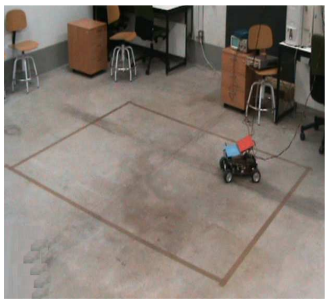


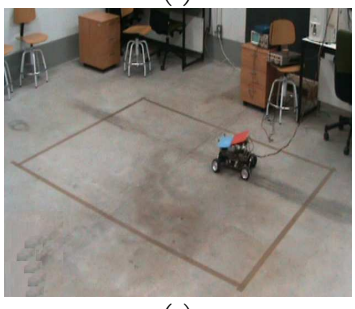
Figura 4-13: Caso de prueba #1

En el caso de prueba #2, el coche parte con un estado que hace que el objetivo se encuentre delante del mismo (Figura 4-14a). Inicialmente arranca marcha atrás girando hacia la izquierda (Figura 4-14b), y así encarar el objetivo yendo hacia delante con las ruedas giradas a la derecha (Figura 4-14c). A continuación siguiendo en el mismo sentido de marcha, gira las ruedas a la izquierda y derecha para aproximarse al objetivo (Figura 4-14d, Figura 4-14e). En este momento y como consecuencia del proceso de discretización y periodo de muestreo se produce un fenómeno indeseado pero real y es que el coche se queda "enganchado" entre dos estados yendo hacia delante y hacia atrás (Figura 4-14f, Figura 4-14g, Figura 4-14h, Figura 4-14i, Figura 4-14j). Una vez alcanzado el estado correspondiente a la Figura 4-14k, arranca hacia delante con las ruedas giradas a la izquierda para posicionarse con la misma orientación que el objetivo (Figura 4-14k). En este momento, con las ruedas rectas da marcha atrás alcanzándolo finalmente (Figura 4-14l).

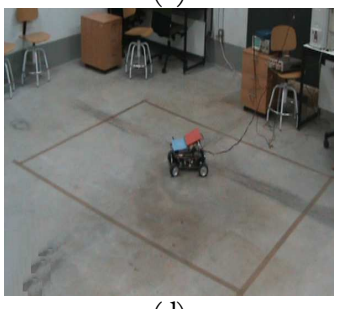
(a)



(b)



(c)



(d)

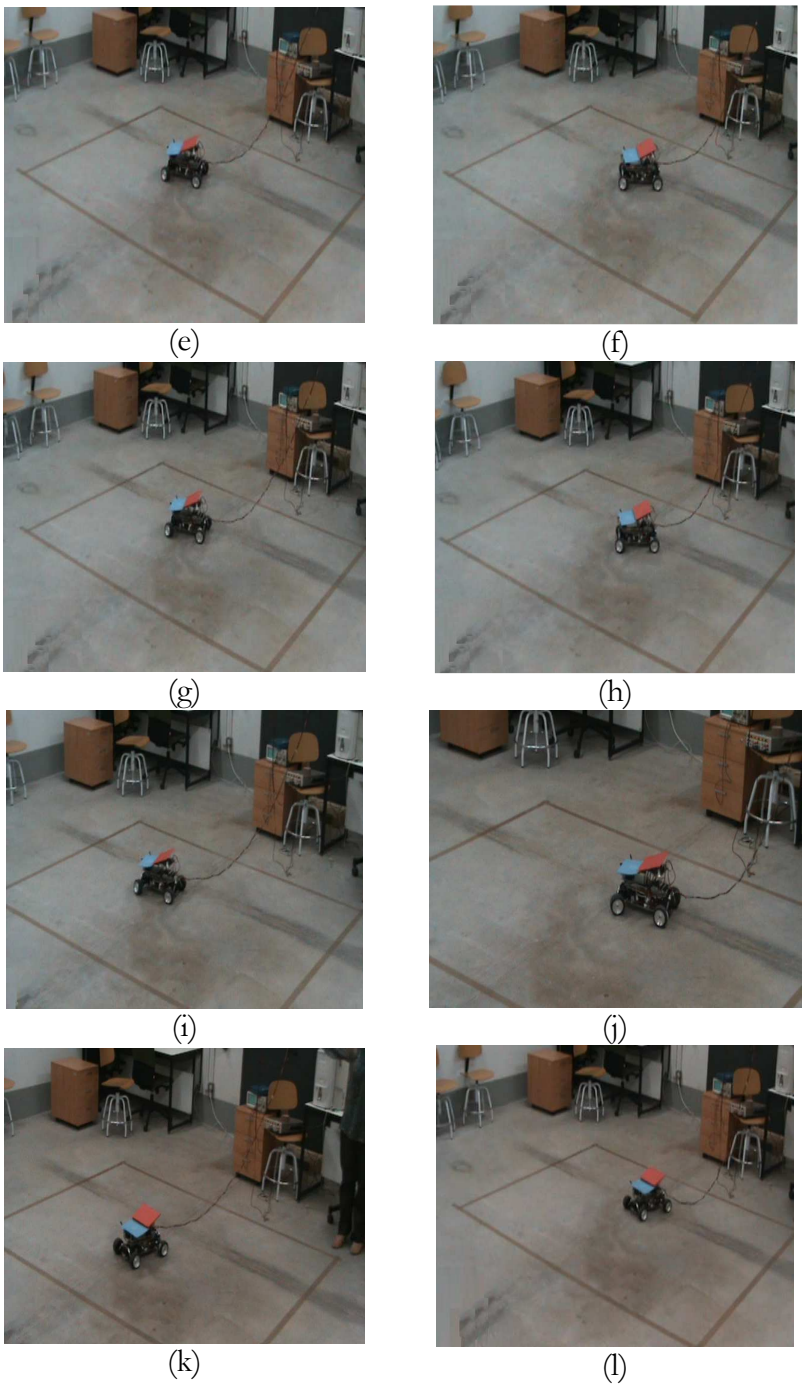


Figura 4-14: Caso de prueba #2

En el caso de prueba #3 representado en la Figura 4-15, el coche arranca marcha atrás (Figura 4-15a, Figura 4-15b), para después girar y avanzar hacia delante (Figura 4-15c). Ocurre sin embargo, que en la maniobra de giro, el vehículo se sale del espacio cartesiano (Figura 4-15c), y en definitiva del espacio de estados. Este hecho no debería ocurrir, pero debido a las limitaciones de la propia cámara en cuanto a su ángulo de visión, el espacio de estados es más reducido del que debería considerarse para un coche como el utilizado en estas pruebas (véase Figura 3-2). Por lo tanto y aplicando la función reactiva descrita en 3.2.2, debe de nuevo ir marcha atrás hasta que entra de nuevo al espacio de estados, momento en el cual empieza a moverse hacia su frente (Figura 4-15d, Figura 4-15e), con las ruedas rectas. Llegado a cierto punto donde si continuara con ese movimiento se pasaría del objetivo (Figura 4-15e), empieza a ir marcha atrás con las ruedas giradas a la derecha (Figura 4-15f), y de nuevo avanzar hacia delante girando las ruedas a la izquierda (Figura 4-15g) y así aproximarse al objetivo. Debido a la discretización realizada y periodo de muestreo utilizado, tiene que hacer otra maniobra de giro, marcha atrás con las ruedas hacia la derecha (Figura 4-15h), para arrancar a continuación e ir hacia el frente con las ruedas giradas a la izquierda llegando finalmente al objetivo (Figura 4-15i).

(a)

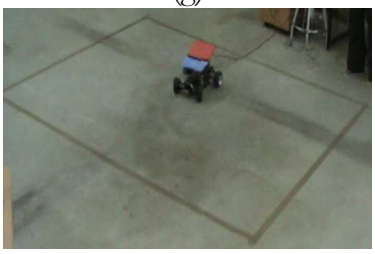
(b)

(e)

(d)

(f)

(g)



(h)

(i)

Figura 4-15: Caso de prueba #3

En la Figura 4-16 se representan los fotogramas de la trayectoria del caso de prueba #4, en la cual el coche se sitúa en el cuarto cuadrante (Figura 4-16a) con una orientación tal, que tiene que arrancar marcha atrás girando a la derecha (Figura 4-16b, Figura 4-16c), para llegar a cierto estado en el que cambia el sentido de marcha girando a la izquierda (Figura 4-16d), y así colocarse con la misma orientación que el objetivo (Figura 4-16e). A continuación dando marcha atrás con las ruedas rectas alcanza dicho objetivo (Figura 4-16f).

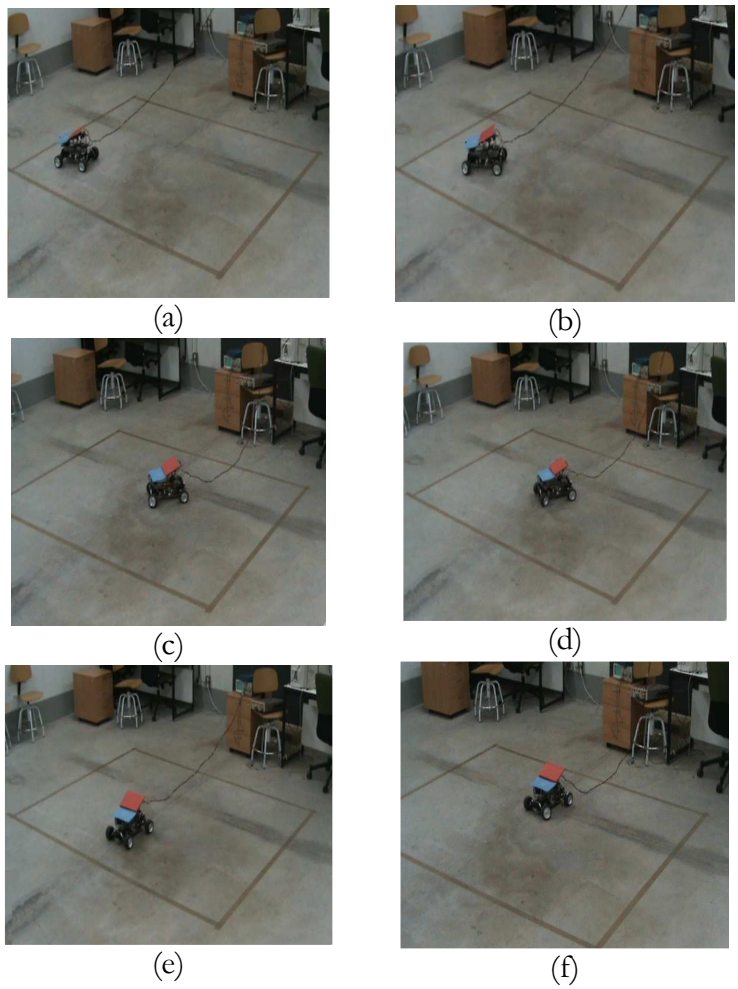
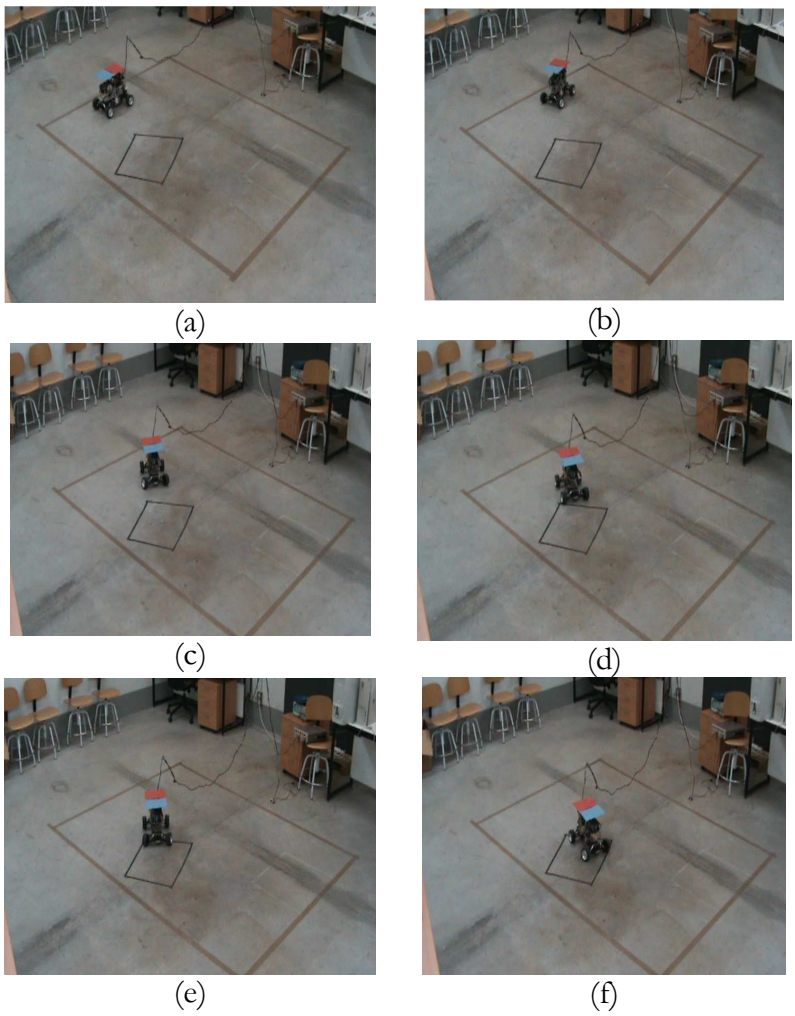


Figura 4-16: Caso de prueba #4

La idea mostrada en los siguientes cuatro casos de prueba es visualizar las trayectorias generadas por el vehículo para realizar un aparcamiento en la zona delimitada con una cinta negra.

Para la Figura 4-17, por la posición con la se inicia la trayectoria (Figura 4-17a), el coche va marcha atrás (Figura 4-17b), para luego avanzar hacia delante con las ruedas giradas a la izquierda y aproximarse al estado objetivo (Figura 4-17c, Figura 4-17d, Figura 4-17e). Seguidamente tiene que maniobrar en sentido de marcha y dirección (Figura 4-17f, Figura 4-17g), para finalmente entrar en el objetivo previsto (Figura 4-17h).



(a) (b)

(c) (d)

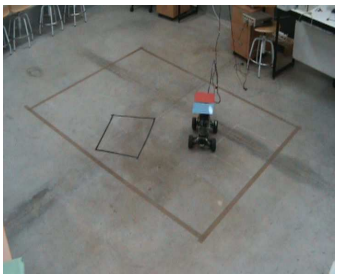
(e) (f)

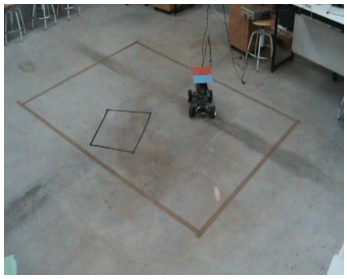

(g)


(h)

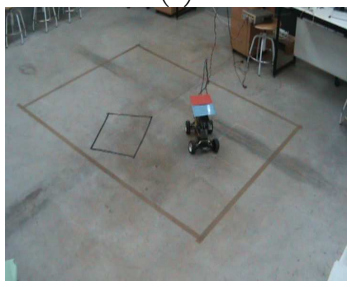
Figura 4-17: Caso de prueba #5

En la trayectoria de la Figura 4-18, partiendo de la posición indicada en la Figura 4-18a, el coche inicia una serie de movimientos cambiando el sentido de marcha y de dirección (Figura 4-18b, Figura 4-18c, Figura 4-18d, Figura 4-18e, Figura 4-18f, Figura 4-18g, Figura 4-18h), con objeto de posicionarse en un estado tal, que mediante un movimiento sin maniobras pueda alcanzar el objetivo (Figura 4-18i). Ocurre sin embargo, que arranca de frente con las ruedas giradas a la derecha y como consecuencia del valor del periodo de muestreo, se pasa ligeramente del estado objetivo (Figura 4-18j). Por tanto, ahora da marcha atrás para quedarse en la misma dirección que la plaza delimitada (Figura 4-18k), y así mediante un movimiento de frente alcanzar correctamente el estado objetivo.


(a)


(b)


(c)


(d)

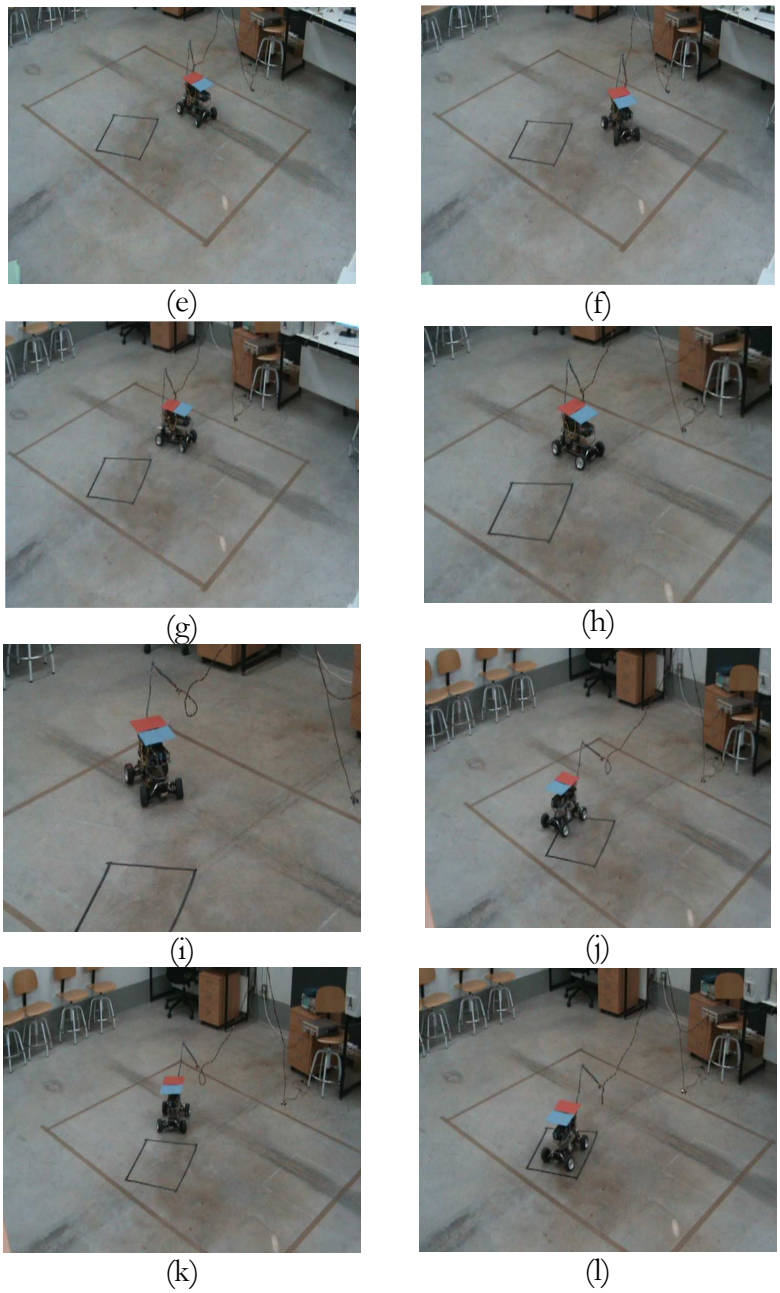
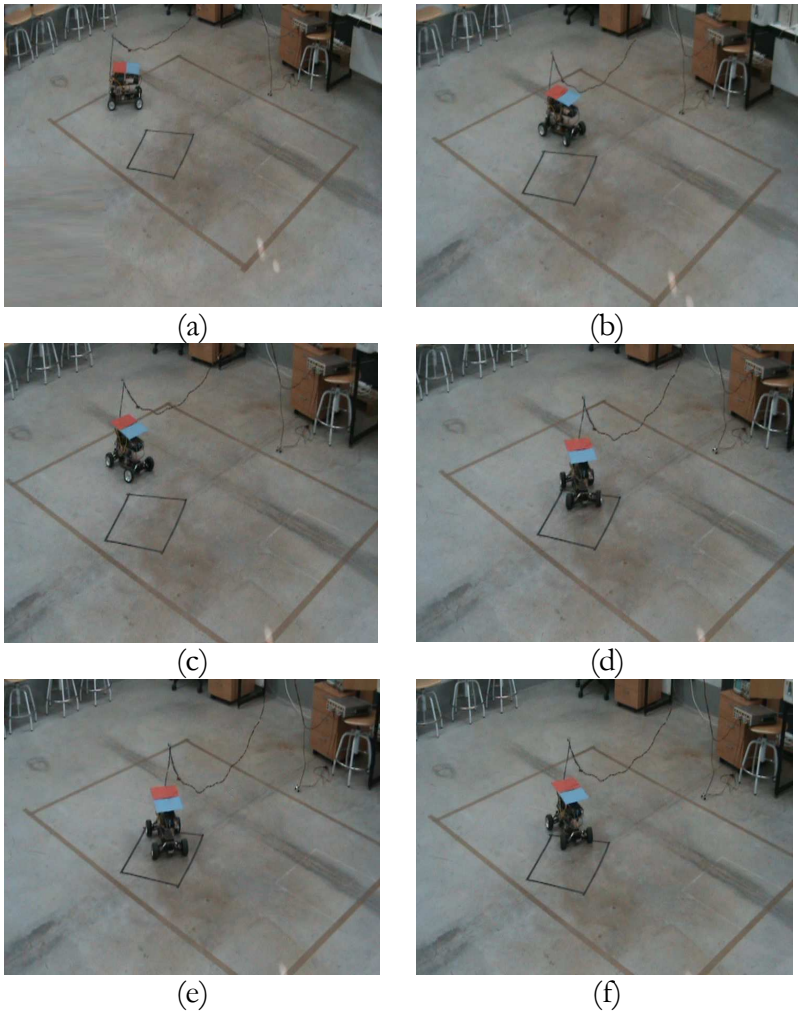


Figura 4-18: Caso de prueba #6

La Figura 4-19 muestra otra trayectoria partiendo el coche del cuarto cuadrante. Por su estado inicial (Figura 4-19a), el único movimiento que puede hacer para no salirse del espacio de estados es ir de frente, maniobrando también con las ruedas de dirección para llegar al objetivo (Figura 4-19b, Figura 4-19c, Figura 4-19d, Figura 4-19e). Observando la Figura 4-19e, vemos que el vehículo entra en la plaza pero con una orientación diferente a la del objetivo, por lo que de nuevo empieza a maniobrar tanto en sentido de marcha como en dirección (Figura 4-19f, Figura 4-19g, Figura 4-19h, Figura 4-19i, Figura 4-19j, Figura 4-19k), para poder llegar al objetivo entrando de frente y con las ruedas giradas hacia la derecha (Figura 4-19l).



(a) (b) (c) (d) (e) (f)

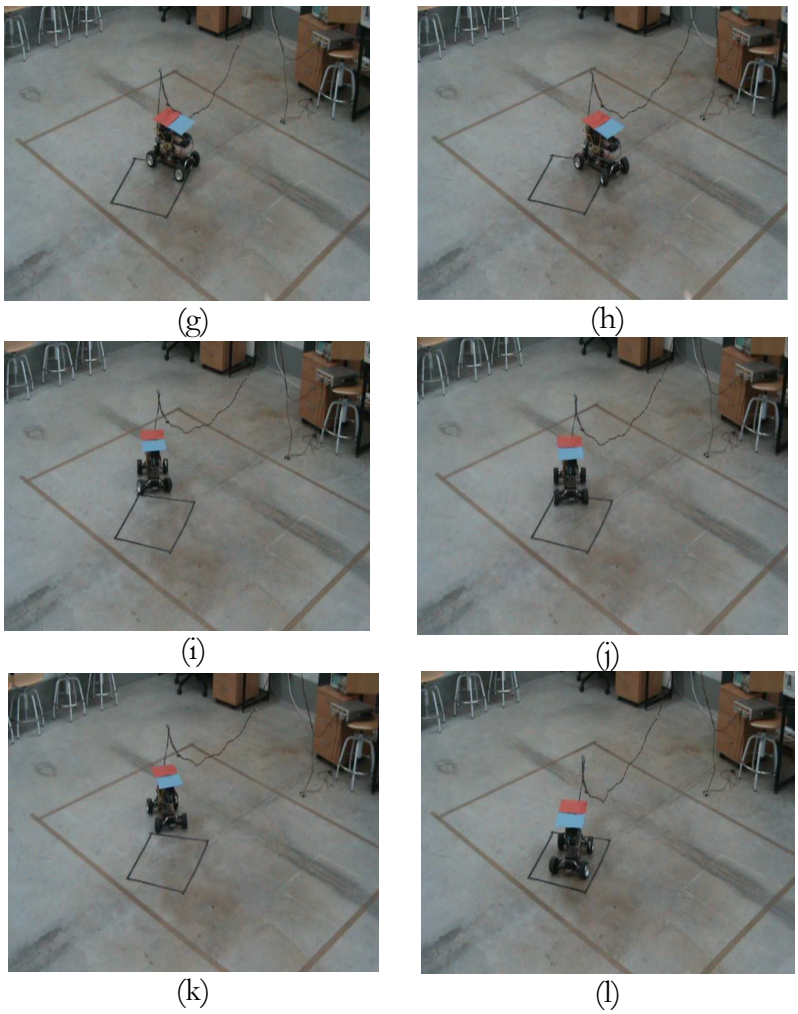
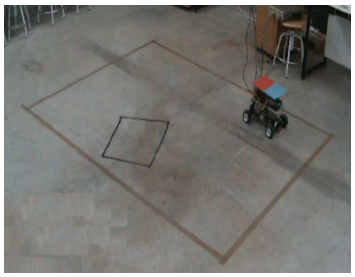


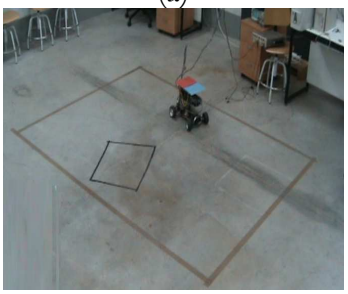
Figura 4-19: Caso de prueba #7

Partiendo de la posición de la Figura 4-20a, el coche inicialmente arranca marcha atrás para posicionarse a continuación en la dirección y sentido de la plaza de aparcamiento (Figura 4-20b, Figura 4-20c, Figura 4-20d, Figura 4-20e, Figura 4-20f). Una vez alcanzada la posición de la Figura 4-20f, el coche avanza hacia delante para entrar parcialmente en la plaza (Figura 4-20g, Figura 4-20h, Figura 4-20i). Sin embargo, dado que en orientación no cumple con el objetivo previsto, es necesario que haga ciertas maniobras para finalmente posicionarse correctamente en el estado objetivo (Figura 4-20j, Figura 4-20k, Figura 4-20l, Figura 4-20m, Figura 4-20n).

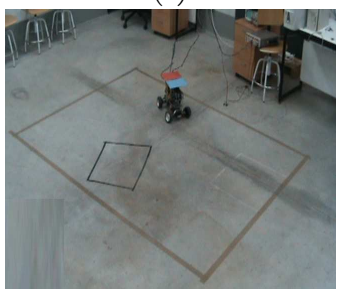


(a)

(b)



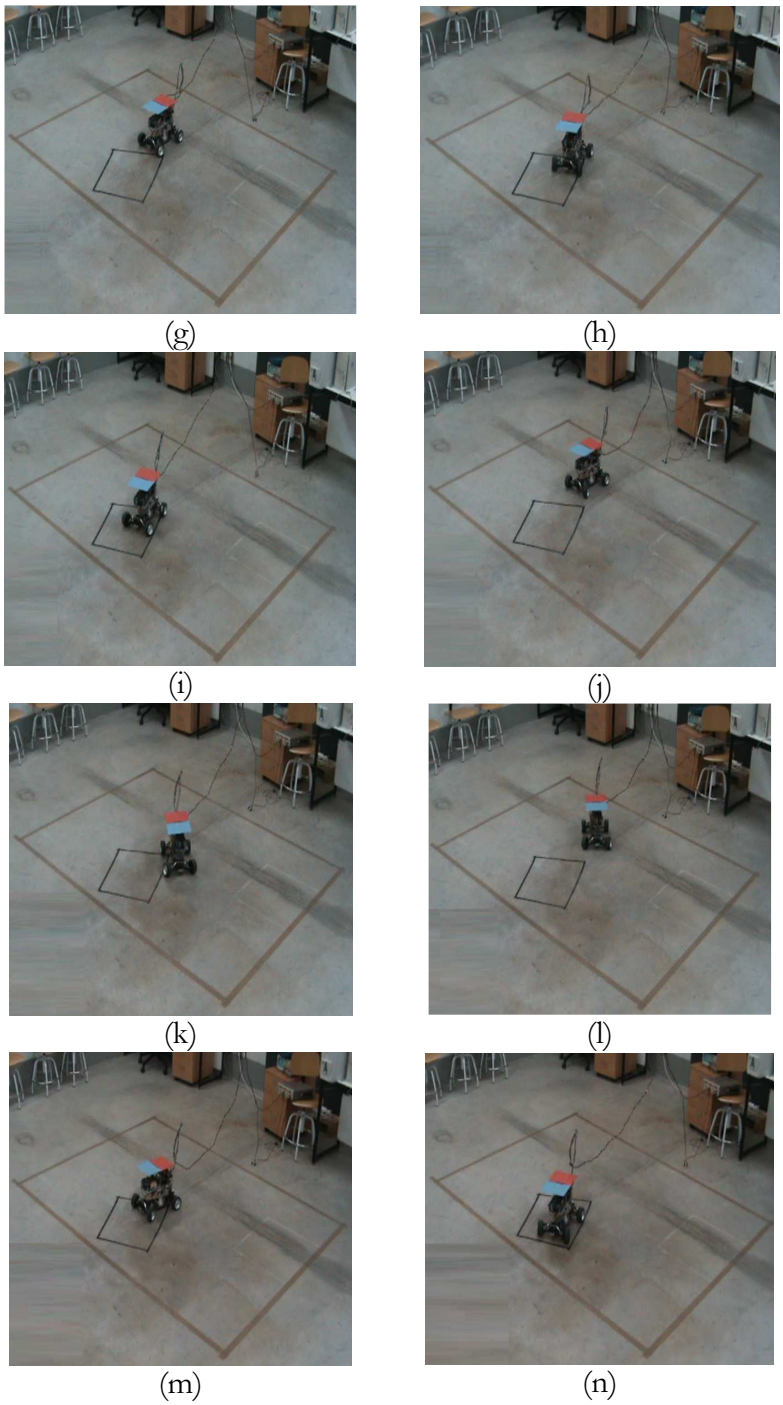
(c)



(d)

(e)

(f)



(g) (h) (i) (j) (k) (l) (m) (n)

Figura 4-20: Caso de prueba #8

Capítulo 5

Tengo una demostración maravillosa de esta proposición, pero este margen es demasiado estrecho para contenerla.

PIERRE DE FERMAT (1601-1665. Jurista y matemático)

5. Conclusiones

En este capítulo se resumen los principales resultados y conclusiones del trabajo realizado. Además, se indican las principales aportaciones, así como posibles líneas futuras de investigación.

5.1 Resumen y conclusiones del trabajo

En la presente tesis doctoral se ha investigado un nuevo algoritmo de control óptimo que puede ser aplicado en sistemas continuos no lineales, como es el caso particular de los vehículos móviles autónomos no holónomos. Previamente a la propuesta del algoritmo, se ha buscado que éste tuviera un carácter orientado a problemas de optimización, adaptación y aprendizaje sin modelo matemático subyacente.

La capacidad de aprendizaje sin modelo matemático, unida a la técnica de Cell Mapping como solución al problema de discretización e introduciendo el concepto de adyacencia entre celdas, como

característica intrínseca de transición, han producido los resultados demostrados en el Capítulo 4. Además, mediante la aplicación de las ecuaciones de transformación [3.11] [3.12] se consigue una mayor eficiencia en la ejecución del algoritmo debido a que el aprendizaje en tiempo real se reduce a conocer las transiciones en el origen del espacio de estados, para cada acción de control. A partir de este conocimiento parcial, se pueden calcular el resto de transiciones para el conjunto del espacio de estados. Con esta estrategia se consigue además reducir el tiempo total de aprendizaje para todo el conjunto del espacio de estados.

Un aspecto que merece ser destacado como consecuencia de las diferentes pruebas realizadas, es que un número elevado de acciones de control no mejora necesariamente el trazado del vehículo y, sin embargo, hace que el tiempo de aprendizaje se eleve considerablemente.

5.2 Aportaciones originales de la tesis doctoral

Las principales aportaciones extraídas como consecuencia de la investigación realizada son tres:

1. Implementación de un nuevo algoritmo de control óptimo estudiado teóricamente en simulación y verificado en vehículos reales, caracterizados por ser sistemas dinámicos continuos no lineales. Este algoritmo hace uso de cuatro variables de estado, combinando las virtudes de los métodos de aprendizaje por refuerzo y de las técnicas de Cell Mapping.
2. Desarrollo de un nuevo algoritmo que permite generar un modelo a partir del cual se extiende el conocimiento a todo el espacio de estados, reduciendo el tiempo global de aprendizaje del vehículo.

3. Generación de un controlador óptimo basado en un espacio de estados que considera la dinámica del vehículo sin tener en cuenta modelo matemático alguno.

5.3 Futuras líneas de investigación

A continuación se indican algunas posibles líneas de extensión de la investigación desarrollada en este trabajo. Algunas de estas líneas están asociadas a aspectos situados fuera de los objetivos de la tesis, pero directamente relacionados con los temas tratados:

- Adaptación del nuevo algoritmo a otras técnicas RL distintas de Q-learning de las expuestas en el apartado 2.3.3, con objeto de mejorar su capacidad de aprendizaje y reducir el tiempo del mismo, para hacerlo extensible a otros entornos intentando eliminar las ecuaciones de transformación.
- Introducción de nuevas variables de estado (por ejemplo, velocidad de dirección, ángulo de dirección) además de las cuatro existentes, para un mayor conocimiento dinámico del vehículo, aprovechando la potencialidad de las ecuaciones de transformación y de la técnica basada en la varianza con respecto al centro de celda de la velocidad del coche.
- Propuesta de una nueva solución para el modelado de diferentes tipos de terrenos por los que pueda moverse el vehículo. En estos casos, las ecuaciones de transformación no pueden ser directamente aplicables.
- Introducción de la variable de estado Z, además de las cuatro existentes, para contemplar una orografía de terreno con cierta altimetría. Con esta quinta variable no podrían utilizarse ni las ecuaciones de transformación ni la técnica basada en la varianza de la velocidad. Por lo tanto, el aprendizaje real habría que hacerlo en todo el espacio de estados, con el consiguiente tiempo invertido en ello.

- En este caso se abriría a su vez una nueva línea de investigación para la deducción de una estrategia óptima con el objetivo de reducir el tiempo de aprendizaje y que a partir de información exclusivamente dinámica (ángulo de tracción, ángulo de dirección, velocidad de tracción y velocidad de dirección), se obtuvieran las coordenadas X, Y, Z y θ del coche. De este modo, el algoritmo de control óptimo propuesto en esta tesis pasaría a tener al menos las variables de estado dinámicas.

- Diseño de un sistema empotrado de bajo coste capaz de ser integrado en un vehículo real consiguiendo una conducción autónoma bajo determinadas situaciones de congestión de tráfico, reduciendo el consumo de combustible, mejorando la confortabilidad del conductor y reduciendo las emisiones de CO_2 con la consiguiente reducción de la contaminación medio ambiental.

  La solución que se plantea para conseguir estos objetivos es mover el vehículo adecuadamente para seguir al vehículo precedente con un consumo mínimo de energía. Por lo tanto, en este caso, la función de coste que hay que minimizar es la energía. Mediante el algoritmo propuesto en esta tesis, utilizando ciertos sensores no explotados en este trabajo de investigación (por ejemplo, ultrasonidos, láser), y realizando los ajustes oportunos en cuanto a los tipos de variables de estado que se utilizarán, el vehículo identificará progresivamente su propia dinámica y la del entorno, al mismo tiempo que adaptará sus movimientos para conseguir los propósitos anteriores.

- Propuesta de una nueva técnica o método, que actuando sobre el vehículo durante la fase de ejecución en la que se hace uso del controlador óptimo generado, permita evitar independientemente del periodo de muestreo seleccionado, ciertos movimientos periódicos del coche que hagan que

éste se quede "enganchado" en determinadas celdas durante un tiempo excesivo.

- Desarrollo de un modelo matemático que demuestre que a medida que el número de celdas asignadas al espacio de estados considerado aumenta, la distancia de transición que se debe considerar en el algoritmo propuesto también debe aumentar.
- Adaptación de la técnica propuesta en esta tesis para su aplicabilidad en sistemas no lineales e inestables, haciendo que inicialmente el aprendizaje de estos sistemas sea guiado para terminar siendo totalmente autónomo.
- Estudio y análisis del algoritmo desarrollado en la tesis para su aplicabilidad en vehículos con motor de gasolina o diesel.
- Diseño de un sistema de autoguiado de vehículos controlando la velocidad, por los carriles de una autovía.
- Explotación de la discretización no uniforme de forma dinámica para conseguir una planificación de movimiento precisa en determinadas zonas del espacio de estados considerado. Para ello, será necesario el estudio e investigación de técnicas de discretización que conduzcan a los resultados de precisión requeridos.

Capítulo 6

Una palabra bien elegida puede economizar no solo cien palabras sino cien pensamientos.

HENRI POINCARÉ (1854-1912. Matemático)

6. Bibliografía

[And-89] C. W. Anderson, "Learning to control an inverted pendulum using neural networks," *in IEEE Control Systems Magazine*. Vol. 9, issue: 3, pp. 31-37, 1989.

[Bar-85a] A. G. Barto and P. Anandan, "Pattern recognizing stochastic learning automata," *in IEEE Transactions On Systems, Man and Cybernetics*. Vol. 15, pp. 360-375, 1985.

[Bar-85b] A. G. Barto, "Learning by statistical cooperation of self-interested neuron-like computing elements," *in Human Neurobiology*. Vol. 4, pp. 229-256, 1985.

[Bar-87] A. G. Barto and M. I. Jordan, "Gradient following without back-propagation in layered networks," *in Proceedings of the IEEE First Annual Conference on Neural Networks*. San Diego, CA. Vol. II, pp. 629-636, 1987.

[Bar-95] A. G. Barto, S. J. Bradtke, and S. P. Singh, "Learning to act using Real-Time Dynamic Programming," *Artificial Intelligence, Special Volume on Computational Research on Interaction and Agency*. Vol. 72, no. 1, pp. 81-138, 1995.

[Bar-03] A. G. Barto and S. Mahadevan, "Recent Advances in Hierarchical Reinforcement Learning," *in Discrete Event Dynamic Systems*. Vol. 13, no. 4, pp. 341-379, 2003.

[Ber-95] D. P. Bertsekas, "Dynamic programming and optimal control, Vol. I y II," *Athena Scientific*. 1995.

[Ber-96] D. P. Bertsekas and J. Tsitsiklis, "Neuro-Dynamic Programming," *Athenea Scientific*. 1996.

[Bel-57] R. E. Bellman, "Dynamic programming," *Princeton University Press*. 1957.

[Bor-91] J. Borenstein, Y. Koren, "The Vector Field Histogram – Fast obstacle avoidance for mobile robots," *in IEEE Journal of Robotics and Automation*. Vol. 7, no. 3, pp. 278-288, 1991.

[Bow-70] R. Bowen, "Markov partitions for Axiom A diffeomorphisms," *in Amer. J. Math.* Vol. 92, pp. 725-747, 1970.

[Bow-78] R. Bowen, "On Axiom A Diffeomorphisms," *CBMS Regional Conferences Series in Mathematics, A.M.S. Publications*. Vol. 35, 1978.

[Bry-75] A. E. Bryson and Y. C. Ho, "Applied optimal control," *New York: Hemisphere*. 1975.

[Cra-92] M. Crandall, H. Ishii and P. Lions, "User's guide to viscosity solutions of second order partial differential equations," *in Bulletin of the American Mathematical Society*. Vol. 27, no. 1, pp. 1-67, July 1992.

[Cra-83] M. Crandall and P. Lions, "Viscosity solutions of Hamilton-Jacobi Equations," *in Trans. of the American Mathematical Society*. 277 (1983), pp. 1-42.

[Dia-93] P. Diamond and P. E. Kloeden, "Spatial discretization of mappings," *in Comp. Math. Appl.* Vol. 25, pp. 85-94, 1993.

[Dia-98] P. Diamond, P. E. Kloeden, V. Kozyakin and A. Pokrovskii, "Monotonic dynamical systems under spatial discretization," *in Proceedings A.M.S.* Vol. 126, pp. 2169-2174, 1998.

[Die-02] T. Dietterich, "Overview of MAXQ Hierarchical Reinforcement Learning," *in Proceedings of the Symposium on Abstraction, Reformulation and Approximation SARA*. New York. Springer Verlag LNAI, pp. 26-44, 2002.

[Dri-04] K. Driessens, "Relational Reinforcement Learning," Ph.D. thesis, Katholieke Universiteit Leuven. 2004

[Dup-98] P. Dupuis and M. R. James, "Rates of convergence for approximation schemes in optimal control," *in SIAM Journal Control and Optimization*. Vol. 36, issue: 2, pp. 719-741, 1998.

[Fra-01] T. Fraichard and J. M. Ahuactzin, "Smooth Path Planning for Cars," *in IEEE Int. Conf. on Robot. Automat.* Seoul. Vol. 4, pp. 3722-3727, May 2001.

[Gas-99a] C. Gaskett, D. Wettergreen and A. Zelinsky, "Q-learning in Continuous State and Action Spaces," *in Proceedings of the 12th Australian Joint Conference on Artificial Intelligence.* Sydney, Australia. Springer LNCS 1747, pp. 417-428, 1999.

[Gas-99b] C. Gaskett, D. Wettergreen and A. Zelinsky, "Reinforcement Learning applied to the control of an Autonomous Underwater Vehicle," *in Proceedings of the Australian Conference on Robotics and Automation*. Brisbane, Australia. March 1999.

[Gas-00] C. Gaskett, L. Fletcher and A. Zelinsky, "Reinforcement Learning for a Vision Based Mobile Robot," *in Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2000)*. Takamatsu, Japón. October 2000.

[Gba-91] G. Barles and P. Souganidis, "Convergence of approximation schemes for fully nonlinear second order equations," *Asymptotic Analysis*. Vol. 4, pp. 271–283, 1991.

[Gom-01] M. Gómez, S. Sánchez, O. G. Dávila and D. Meziat, "An obstacle detector system based on laser technology," *in IEEE Circuits and Devices Magazine*. Vol. 17, issue: 5, pp. 9-15, 2001.

[Gom-02] M. Gómez, T. Martínez and S. Sánchez, "Optimal trajectory generation using the Simple Cell-Mapping method for wheeled mobile vehicles," *in Proceedings of the Seminario Anual de Automática, Electrónica Industrial e Instrumentación (SAAEI)*. Universidad de Alcalá, Spain. Conference Proceedings Book, pp. 463-467, 2002.

[Gom-03] M. Gómez, F. J. Bolívar, S. Sánchez, "A remote control system applied to wheeled mobile vehicles," editado por el Servicio de Publicaciones de la Universidad de Alcalá, con ISBN: 84-8138-560-3. Abril 2003.

[Gom-07a] M. Gómez, T. Martínez, S. Sánchez and D. Meziat, "Optimal control applied to Wheeled Mobile Vehicles," *in Proceedings of IEEE International Symposium on Intelligent Signal Processing*. Universidad de Alcalá, Spain. Conference Proceedings Book, pp. 83-88, 2007.

[Gom-07b] M. Gómez, S. López, S. Sánchez and D. Meziat, "Building an efficient maze robot that can get there fastest with the least effort. A single methodology applied to a robot for searching shortest path through a maze," *Embedded.com*. August 2007. [Online]: http://www.embedded.com/design/201801616.

[Gom-08] M. Gómez, T. Martínez-Marín, S. Sánchez and D. Meziat, "Optimal control applied for Wheeled Mobile Vehicles based on Cell Mapping techniques," *in Proceedings of IEEE Intelligent Vehicles Symposium*. Eindhoven University of Technology, The Netherlands. Conference Proceedings Book, pp. 1009-1014, 2008.

[Gom-09a] M. Gómez, T. Martínez-Marín, S. Sánchez and D. Meziat, "Integration of Cell-Mapping and Reinforcement Learning Techniques for Motion Planning of Car-Like Robots," *in IEEE Transactions on Instrumentation and Measurement (Special Issue)*. Vol. 58, issue: 9, pp. 3094-3103, September 2009.

[Gom-09b] M. Gómez, L. Gayarre, T. Martínez-Marín, S. Sánchez and D. Meziat, "Motion Planning of a Non-Holonomic Vehicle in a Real Environment by Reinforcement Learning," *in Proceedings of the International Work-Conference on Artificial Neural Networks (IWANN), Part I*. Salamanca, Spain. Springer LNCS 5517, pp. 813-819, June 2009.

[Gom-09c] M. Gómez, "Website of the research works related to motion optimal planning and reinforcement learning applied to autonomous mobile vehicles. It contains some videos of the tests performed to a non-holonomic vehicle," [Online]: http://atc1.aut.uah.es/~mariano/Research/OptimalControl_research.html.

[Gu-02] D. Gu and H. Hu, "Neural Predictive Control for a Car-like Mobile Robot," *in Journal of Intelligent and Robotic Systems*. Vol. 145, no. 2, pp. 367-393, 1990.

[Gut-89] R. S. Guttalu, "Steady state response of nonlinear systems by cell mapping method," *Int. Symp. On Advanced Computers for Dynamics and Design*. Tsuchiura, Japan. Pp. 303-308, 1989.

[Gut-93] R. S. Guttalu and P. J. Zufiria, "The adjoining cell mapping and its recursive unravelling, Part II: Application to selected problems," *Nonlinear Dynamics*. Vol. 4, no. 4, pp. 309-336, 1993.

[Hob-88] W. J. Hobbs, G. Hermon, S. Warner and G.B. Sheblé, "An Enhanced Dynamic Programming Approach for Unit Commitment," *in IEEE Transactions on Power Systems*. Vol. 3, no. 3, pp. 1201-1205, 1988.

[Hsu-80a] C. S. Hsu, "A theory of cell-to-cell mapping dynamical systems," *in J. Appl. Mech.* Vol. 47, pp. 931-939, 1980.

[Hsu-80b] C. S. Hsu and R. S. Guttalu, "An unravelling algorithm for global analysis of dynamical systems: an application of cell-to-cell mapping," *in J. Appl. Mech.* Vol. 47, pp. 940-948, 1980.

[Hsu-85] S. Hsu, "A discrete method of optimal control based upon the cell state space concept," *in J. Optim. Th. Appl.* Vol. 46, no. 4, 1985.

[Hsu-87] C. S. Hsu, "Cell-to-Cell Mapping: A Method of Global Analysis for Non-linear Systems," *Springer Verlag*. New York, 1987.

[Jba-89] J. Barraquand and J. C. Latombe, "On nonholonomic mobile robots and optimal maneuvering," *in Revue d'Inteligence Artificielle*. Vol. 3, no. 2, pp. 77-103, 1989.

[Kae-96] L. P. Kaebling, M. L. Littman and A.W. Moore, "Reinforcement learning: A survey," *in Journal of Artificial Intelligence Research*. Vol. 4, pp. 237-285, 1996.

[Kass-05] Y. Kassahun and G. Sommer, "Efficient reinforcement learning through evolutionary acquisition of neural topologies," *in Proceedings of the 13th European Symposium on Artificial Neural Networks (ESANN 2005)*. Bruges, Belgium. Conference Proceedings Book, pp. 259-266, April 2005.

[Ken-02] Kenneth O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *in Evolutionary Computation*. Vol. 10, no. 2, pp. 99-127, 2002.

[Kon-00] K. Konolige, "A Gradient Method for Realtime Robot Control," *in Proceedings of Intelligent Robots and Systems*. Vol. 1, pp. 639-646, 2000.

[Krö-02] Michael Krödel and Klaus-Dieter Kuhnert, "Reinforcement Learning to Drive a Car by Pattern Matching," *in Proceedings of the 24th DAGM Symposium on Pattern Recognition*. Springer LNCS 2449, pp. 322-329, September 2002.

[Kus-92] J. H. Kushner and P. Dupuis, "Numerical methods for stochastic control problems in continuous time," *in Applications of Mathematics*. Berlin. Springer-Verlag, 1992.

[Lam-01] F. Lamiraux and J. P. Laumond, "Smooth Motion Planning for Car-Like Vehicles," *in IEEE Trans. Robot. Automat.* Vol. 17, no. 4, pp. 498-502, August 2001.

[Lat-91] J. C. Latombe, "Robot Motion Planning," *Kluwer Academic*. 1991.

[Lew-86] F. L. Lewis, "Optimal Control," *John Wiley & Sons*. 1986.

[Mar-99a] T. Martínez-Marín and P. J. Zufiria, "Optimal Control of Nonlinear Systems through a Hybrid CACM/ANN Technique," *Int. J. Adaptive Control Signal Process*. Vol. 13, pp. 307-319. 1999.

[Mar-99b] T. Martínez-Marín, "Diseño de controladores óptimos combinando técnicas de Cell-Mapping y redes neuronales para el control de sistemas dinámicos no lineales," *Tesis Doctoral*. ETSIT, Universidad Politécnica de Madrid. 1999.

[Mar-03] T. Martínez-Marín, "Optimal Path Planning for Car-Like Vehicles in the Presence of Obstacles," *in Proceedings of Intelligent Transportation Systems*. Shangai, China. Vol. 2, pp. 1161-1164. October 2003.

[Mar-04] T. Martínez-Marín, "A reinforcement learning algorithm for optimal motion of car-like vehicles," *in Proceedings of Intelligent Transportation Systems*. Washington DC, USA. Pp. 47-51. October 2004.

[Mae-95] Maes Pattie, "Artificial Life Meets Entertainment: Life like Autonomous Agents," *Communications of the ACM*. Vol. 38, no. 11, pp. 108-114, 1995.

[Men-70] J. M. Mendel and R. W. McLaren, "Reinforcement learning control and pattern recognition systems," *In Adaptive, Learning and Pattern Recognition Systems*. K. S. Fu and J. M. Mendel, Eds. New York. Academic Press, pp. 287-318, 1970.

[Mil-94] Milos Zefran, "Review of the Literature on Time-Optimal Control of Robotic Manipulators," *GRASP Laboratory*. Department of Computer and Information Science, Technical Reports (CIS), University of Pennsylvania, 1994.

[Min-08] MinKi Choi, Woojin Chung, Yongkwan Kwon and Hyungchul Kim, "Safe and high navigation of a patrol robot in occluded dynamic obstacles," *in Proceedings of the 17th World Congress*. The International Federation of Automatic Control, Seoul, Korea. July 2008.

[Mon-08] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt and S. Thrun, "Junior: The Stanford Entry in the Urban Challenge," *in Journal of Field Robotics*. Vol. 25, issue: 9, pp. 569-597, September 2008.

[Moo-93] A. Moore and C. Atkeson, "Prioritized Sweeping: Reinforcement learning with less data and less time," *Machine Learning*. Vol. 13, pp. 103-130, 1993.

[Moo-95] A. Moore and C. Atkeson, "The parti-game algorithm for variable resolution reinforcement learning in multidimensional state space," *Machine Learning*. Vol. 21. 1995.

[Mun-92] R. Munos and A. Moore, "Variable Resolution Discretization in Optimal Control," *Machine Learning Journal*. Vol. 49, pp. 291-323, 2002.

[Oga-80] K. Ogata, "Ingeniería de control moderna," *Prentice Hall*. 1980.

[Oll-91] A. Ollero, "Control por computador. Descripción interna y diseño óptimo," *Marcombo*. 1991.

[Pap-97] M. Papa, H. M. Tai and S. Shenoi, "Cell Mapping for Controller Design and Evaluation," *in IEEE Control Systems Magazine*. Vol. 17, issue: 2, pp. 52-65. April 1997.

[Pap-01] M. Papa, J. Wood and S. Shenoi, "Evaluating controller robustness using cell mapping," *Fuzzy Sets and Systems*. Vol. 121, issue: 1, pp. 3-12. 2001.

[Pon-86] L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze and E. F. Mishchenko, "The Mathematical Theory of Optimal Processes," Classics of Soviet Mathematics, Gordon & Breach Science Publishers. New York, reprint of 1962 translation from the Russian by K. N. Trirogoff. 1986.

[Qin-00a] B. Qin, Y. C. Soh, M. Xie and D. Wang, "Optimal trajectory generation for wheeled mobile robot," *in the 5th International Conference on Computer Integrated Manufacturing*. Singapore. Vol. 1, pp. 434-444, March 2000.

[Qin-00b] B. Qin, Y. C. Soh, D. Wang and M. Xie, "Trajectory generation for velocity-varying wheeled mobile robot," *in the 6th International Conference on Applications of Advanced Technologies in Transportation Engineering*. Singapore. June 2000.

[Ran-96] A. L. Rankin and C. D. Crane III, "A multi-purpose off-line path planner based on an A* search algorithm," *in Proceedings of The ASME Design Engineering Technical Conferences and Computers in Engineering Conference*. Irvine, California, 1996.

[Ree-90] J. A. Reeds and R. A. Shepp, "Optimal paths for a Car that Goes both Forwards and Backwards," *Pacific J. Math.* Vol. 145, no. 2, pp. 367-393, 1990.

[Son-02] F. Song and S. M. Smith, "Cell-state-Space-Based Search," *in IEEE Control Systems magazine*. Vol. 22, issue: 4, pp. 42-56, August 2002.

[Sta-07] D. Stavens, G. Hoffmann and S. Thrun, "Online Speed Adaptation using Supervised Learning for High-Speed, Off-Road Autonomous Driving," *in Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. Hyderabad, India. 2007.

[Ste-86] R. F. Stengel, "Stochastic optimal control," *John Wiley & Sons*. 1986.

[Sut-84] R. S. Sutton, "Temporal Credit Assignment in Reinforcement Learning," *PhD thesis*. University of Massachusetts, Amherst, MA, 1984.

[Sut-90] R. S. Sutton, "First results with Dyna: An integrated architecture for learning, planning, and reacting," *in Neural Networks for Control*. Miller, T., Sutton, R.S., & Werbos, P., Eds. MIT Press. 1990.

[Sut-92] R. S. Sutton, "Reinforcement learning architectures," *in Proceedings of the International Symposium on Neural Information Processing*. Fukuoka, Japan. 1992.

[Sut-98] R. S. Sutton and A. Barto, "Reinforcement Learning: An introduction," *MIT Press*. 1998.

[Sut-99] R. S. Sutton, D. Precup and S. Singh, "Between MDPs and Semi-MDPs: A framework for temporal abstraction in Reinforcement Learning," *in Artificial Intelligence*. Vol. 112, pp. 181-211, 1999.

[Tad-04] P. Tadepalli, R. Givan and K. Driessens, "Relational Reinforcement Learning: An Overview," *in Proceedings of the ICML workshop on Relational Reinforcement Learning*. 2004.

[Thr-95] S. Thrun and A. Schwartz, "Finding structure in reinforcement learning," *in G. Tesauro, D. Touretzky, and T. Leen, editors, Advances in Neural Information Processing Systems (NIPS)* 7. Cambridge, MA, MIT Press. 1995.

[Thr-06] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley, the robot that won the DARPA Grand Challenge," *in Journal of Field Robotics*. Vol. 23, issue: 9, pp. 661-692, 2006.

[Ton-88] B. H. Tongue and K. Gu, "Interpolated cell mapping of dynamical systems," *in J. Appl. Mech*. Vol. 4, pp. 461-466, 1988.

[Van-94] J. van der Spek, "Cell Mapping Methods: Modifications and Extensions," *PhD thesis*. Eindhoven, University of Technology. 1994.

[Wat-89] Watkins Christopher J.C.H, "Learning from Delayed Rewards," *PhD thesis*. Cambridge University, Cambridge, England, 1989.

[Wat-92] Watkins Christopher J.C.H. and Dayan Peter, "Technical note: Q-learning," *Machine Learning*. Vol. 8, pp. 279-292, 1992.

[Wet-99] D. Wettergreen, C. Gaskett and Alex Zelinsky, "Autonomous Control and Guidance for an Underwater Robotic Vehicle," *in Proceedings of the International Conference on Field and Service Robotics*. Pittsbrugh, USA. September 1999.

[Wil-87] R. J. Williams, "Reinforcement learning connectionist systems," *Technical Report NU-CCS-87-3, College of Computer Science*. Northeastern University, 360 Huntington Avenue, Boston, MA, 1987.

[Wil-88] R. J. Williams, "On the use of backpropagation in associative reinforcement learning," *in Proceedings of the IEEE International Conference on Neural Networks*. San Diego. 1988.

[Zuf-90] P. J. Zufiria and R. S. Guttalu, "A computational method for finding all the roots of a vector function," *in Appl. Math. Comp.* Vol. 35, pp. 13-59, 1990.

[Zuf-93] P. J. Zufiria and R. S. Guttalu, "The adjoining cell mapping and its recursive unravelling, Part I: Description of adaptive and recursive algorithms," *Nonlinear Dynamics*. Vol. 4, no. 4, pp. 204-226, 1993.

[Zuf-03] P. J. Zufiria and T. Martínez-Marín, "Improved Optimal Control Methods based upon the Adjoining Cell Mapping Technique," *in J. Optimization Theory and Applications*. Vol. 118, no. 3, pp. 657-680, 2003.

APÉNDICES

Apéndice A Parámetros y modelos matemáticos de vehículos no holónomos

A. Parámetros y modelos matemáticos de vehículos no holónomos

A.1 Parámetros del vehículo

A continuación se indican los parámetros físicos del vehículo utilizado para los experimentos llevados a cabo en esta tesis:

$R = 0.03$ /* radio de la rueda en metros */

$m = 0.2$ /* masa de la rueda en kilogramos */

$M = 10$ /* masa del vehículo en kilogramos */

$$K_1 = \frac{mR^2}{2}$$

$$K_2 = \frac{MR^2}{2}$$

$$K_3 = \frac{1}{3}\left[1 + \left(\frac{2l}{l_1}\right)^2\right]$$

$l_1 = 0.1$ /* Mitad de la anchura del vehículo en metros */

$l = 0.25$ /* Mitad de la longitud del vehículo en metros */

$d = 0.32$ /* Distancia entre ejes en metros */

A.2 Variables de estado

Para definir tanto un modelo cinemático como el comportamiento dinámico de un vehículo no holónomo, se van a utilizar hasta seis variables de estado con sus correspondientes ecuaciones dinámicas. A continuación se indican dichas variables:

$X_1 = \omega$ /* Velocidad de tracción en rad/s */

$X_2 = x$ /* Coordenada x del vehículo en metros */

$X_3 = y$ /* Coordenada y del vehículo en metros */

$X_4 = \theta$ /* Orientación del vehículo en radianes */

$X_5 = \psi'$ /* Velocidad de dirección en rad/s */

$X_6 = \delta$ /* Ángulo de tracción en radianes */

$X_7 = \psi$ /* Ángulo de dirección en radianes */

A.3 Modelos cinemático y dinámico de un vehículo no holónomo

Para la definición de los modelos cinemático y dinámico de cualquier vehículo no holónomo, se definen las funciones indicadas en la Tabla A-1.

Tabla A-1: Funciones paramétricas del vehículo para la definición de su comportamiento cinemático y dinámico

$fd = \dfrac{1}{1+\dfrac{l_1}{2l}Tan(\psi)}$	$Dfd = -\dfrac{l_1}{2l} f_1 f_6$
$f_1 = fd^2$	$Df_1 = -\dfrac{l_1}{l} fdf_1 f_6$
$f_2 = 1 - fd$	$Df_2 = -Dfd$
$f_3 = f2^2$	$Df_3 = 2 f_2 Df_2$
$f_4 = \dfrac{2-fd}{fd}$	$Df_4 = -\dfrac{2Dfd}{f_1}$
$f_5 = f_4{}^2$	$Df_5 = 2 f_4 Df_4$
$f_{61} = Sec(\psi)$	$Df_6 = 2 f_6 Tan(\psi)$
$f_6 = f_{61}{}^2$	

$$F = 3K_1 f_1 (f_6 + f_5) + K_2 (1 + K_3 f_3)$$
$$DF = 3K_1 Df_1 (f_6 + f_5) + 3K_1 f_1 (Df_6 + Df_5) + K_2 K_3 Df_3$$

$$f_7 = 4F$$
$$f_8 = DF$$
$$f_9 = 4f_8 - \frac{Dfd(1 - fd) f_7}{(2 - fd) fd}$$

A continuación se indican dos modelos matemáticos que definen el comportamiento cinemático y dinámico del vehículo.

Tabla A-2: Ecuaciones diferenciales del comportamiento cinemático del vehículo

Modelo cinemático
$X_2' = R \cdot v \cdot \cos(X_4) \cos(\psi)$ $X_3' = R \cdot v \cdot \text{sen}(X_4) \cos(\psi)$ $X_4' = \frac{1}{d} R \cdot v \cdot \text{sen}(\psi)$
Acciones de control
$v \rightarrow$ Velocidad de tracción $\psi \rightarrow$ Ángulo de dirección

Tabla A-3: Ecuaciones diferenciales del comportamiento dinámico del vehículo desde dos perspectivas

Modelo dinámico_1*	**Modelo dinámico_2**
$X_1' = \frac{\tau_{\text{tracción}} - (X_1 X_5 f_9)}{f_7}$ $X_2' = R X_1 \cos(X_4)\cos(\psi)$ $X_3' = R X_1 \text{sen}(X_4)\cos(\psi)$ $X_4' = \frac{1}{d} R X_1 \text{sen}(\psi)$	$X_1' = \frac{\tau_{\text{tracción}} - (X_1 X_5 f_9)}{f_7}$ $X_5' = \frac{\tau_{\text{dirección}} + X_1^2 f_8}{K_1}$ $X_6' = X_1$ $X_7' = X_5$
Acciones de control	
$\tau_{\text{tracción}} \rightarrow$ Par de tracción $\psi \rightarrow$ Ángulo de dirección	$\tau_{\text{tracción}} \rightarrow$ Par de tracción $\tau_{\text{dirección}} \rightarrow$ Par de dirección

* En condiciones de simulación y con objeto de poder simplificar los cálculos, se podría considerar que la velocidad de dirección se aproxima a una señal *Delta Dirac* cuya duración sea más baja que el tiempo de respuesta del sistema.

Apéndice B Características generales del vehículo

B. Características generales del vehículo

La plataforma móvil (vehículo) utilizada para probar el nuevo algoritmo indicado en la Figura 3-1, y conseguir una planificación óptima de movimiento consta de:

- Chasis y placas separadoras.
- Soportes para motores de tracción y dirección.
- Placa GR-XC3S-1500.
- Motor DC de tracción (Maxon RE-36) y reductora acoplada (GP-32K)
- *Encoder* (HEDS 5540) acoplado al eje del motor de tracción.
- Servomotor de dirección (Hitec HS-985-MG).
- Comunicación *bluetooth*.
- Etapa de potencia.

En la Figura B-1 se muestra un diagrama esquemático de la plataforma.

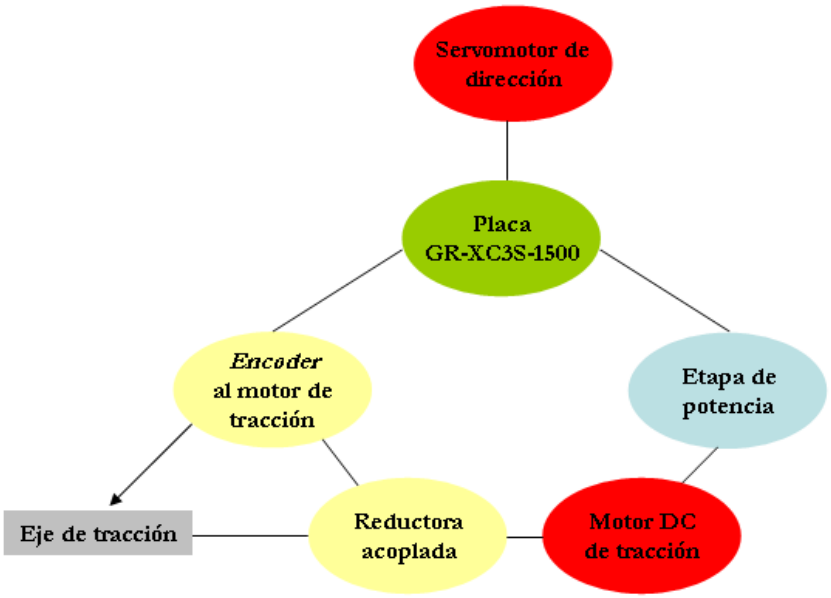


Figura B-1: Arquitectura de la plataforma móvil autónoma

B.1 Chasis y placas separadoras

El cuerpo de la plataforma está formado por el chasis modificado de un vehículo de radio-control escala 1:8. Se ha mantenido el sistema de suspensión, dirección y el diferencial trasero, sobre el que se ha adaptado el motor de tracción. Además, se han añadido unas plataformas en diferentes alturas con el mismo diseño que la base del propio vehículo, con objeto de ubicar el motor de tracción, servomotor, baterías y demás hardware de control.

En la Figura B-2 se puede apreciar el chasis del vehículo y las plataformas adicionales.

Figura B-2: Chasis de la plataforma

B.2 Soportes para motores

Para que tanto el servomotor de dirección como el motor de tracción queden lo mejor anclados al chasis del vehículo, se han construido las siguientes piezas mecánicas:

- para el servomotor se ha optado por realizar un soporte de aluminio de acuerdo a la Figura B-3.
- para el motor de tracción se ha diseñado un sistema basado en tres piezas (véase Figura B-4). La pieza radial (soporte II) se rosca abrazando a la reductora. Los materiales escogidos fueron de teflón para el soporte I, y aluminio para el II.

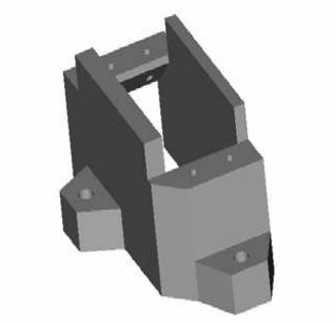


Figura B-3: Soporte del servomotor de dirección

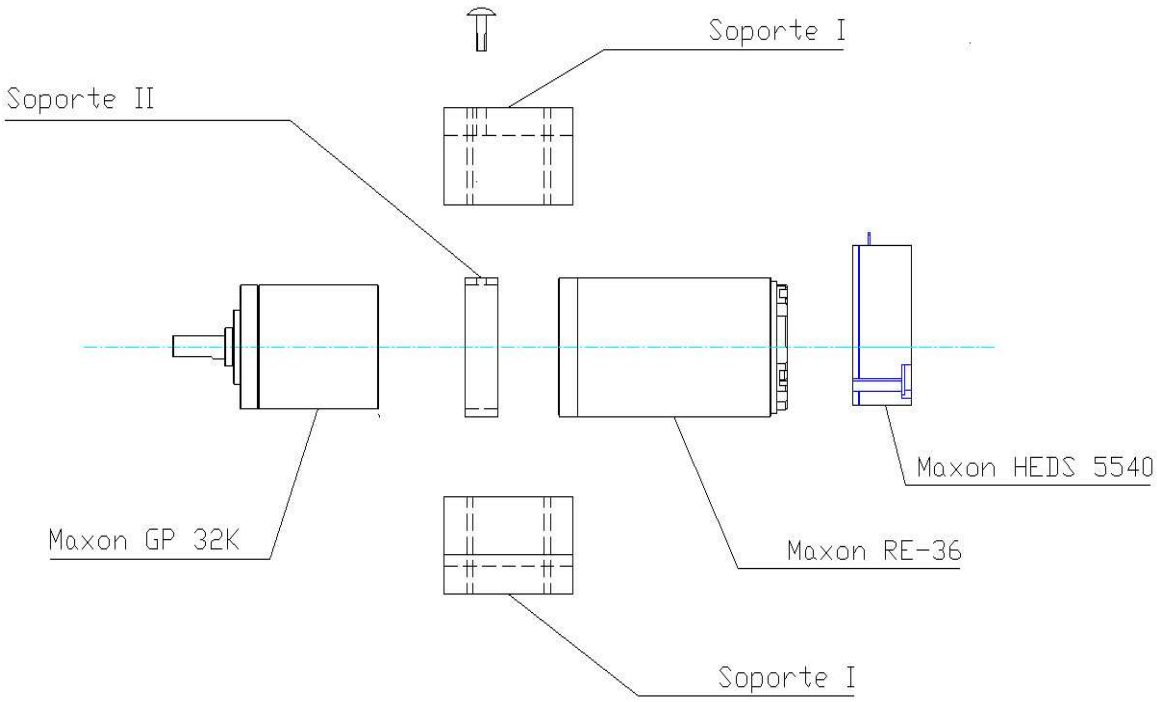


Figura B-4: Montaje del motor de tracción, reductora, *encoder* y soporte de teflón

B.3 Placa GR-XC3S-1500

La tarjeta utilizada para la realización de todas las pruebas consideradas en esta tesis ha sido la GR-XC3S-1500 mostrada en la Figura B-5 y basada en el procesador LEON3. Este procesador es un modelo sintetizable en VHDL sobre la FPGA de Xilinx Spartan3 acorde a una arquitectura de 32 bits SPARC V8. El nuevo algoritmo se ha desarrollado sobre el sistema operativo de tiempo real RTEMS 4.8 instalado y adaptado para dicha placa.

A continuación se describen brevemente las características principales de la placa.

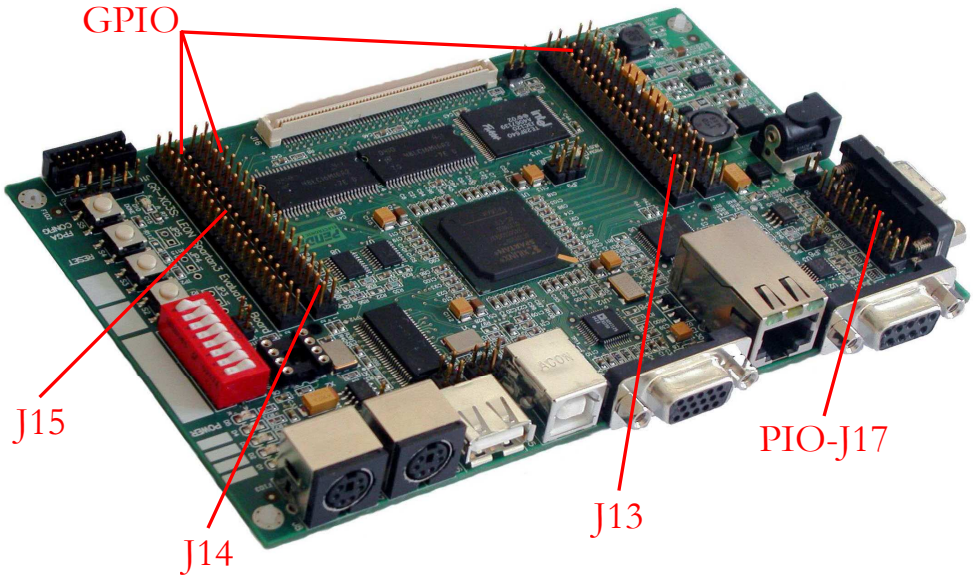


Figura B-5: Placa GR-XC3S-1500

Tamaño: Compact Eurocard (100x160 mm)

Xilinx XC3S1500-4FG456 FPGA

- Xilinx Spartan3 (456 BGA p*ackage*)
- 1 x 4 Mbit (XCF04S)
- 1 x 1 Mbit (XCF01S) Flash PROM

Alimentación

La placa se alimenta de una fuente de alimentación externa de 5V/1A e internamente se hace la regulación a otras tensiones. En la Figura B-6 se muestra el sistema interno de alimentación. La regulación realizada consiste en bajar a 3.3V/2A, 2.5V/300mA y 1.2V/2A.

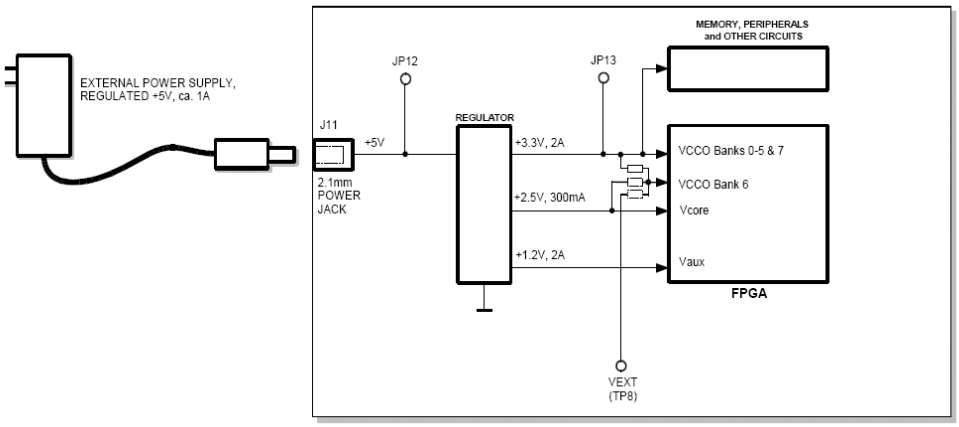


Figura B-6: Sistema de alimentación de la placa GR-XC3S-1500

Memoria

La placa dispone de 8Mbytes de memoria *flash* y 64Mbytes de memoria SDRAM, además se puede ampliar externamente a través de un conector que la propia placa proporciona. En la Figura B-7 se puede ver la disposición de la memoria.

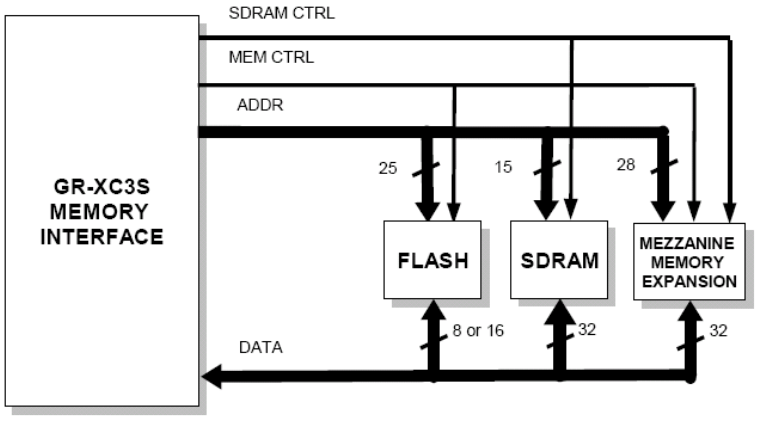


Figura B-7: Sistema de memoria de la placa GR-XC3S-1500

Puertos

La placa GR-XC3S-1500 dispone de 60 puertos GPIO (*General Purpose Input Output*), así como otros 16 puertos PIO (*Parallel Input Output*); todos ellos con niveles LVTTL (*Low Voltage Transistor - Transistor Level*). El primer paso para manejar los puertos es configurar los pines de la FPGA que se conectan directamente a los conectores J17, J13, J14, J15 mostrados en la Figura B-5.

Interfaces

- Ethernet 10/100 Mbit
- USB
- PS2
- 2 UART's (1Mbaudio RS232)
- Video DAC (ADV712550)
- JTAG

El procesador LEON3 posee como mecanismo de comunicación con el sistema de memoria y periféricos el bus AMBA BUS. En la Figura B-8 se muestra la arquitectura de la tarjeta.

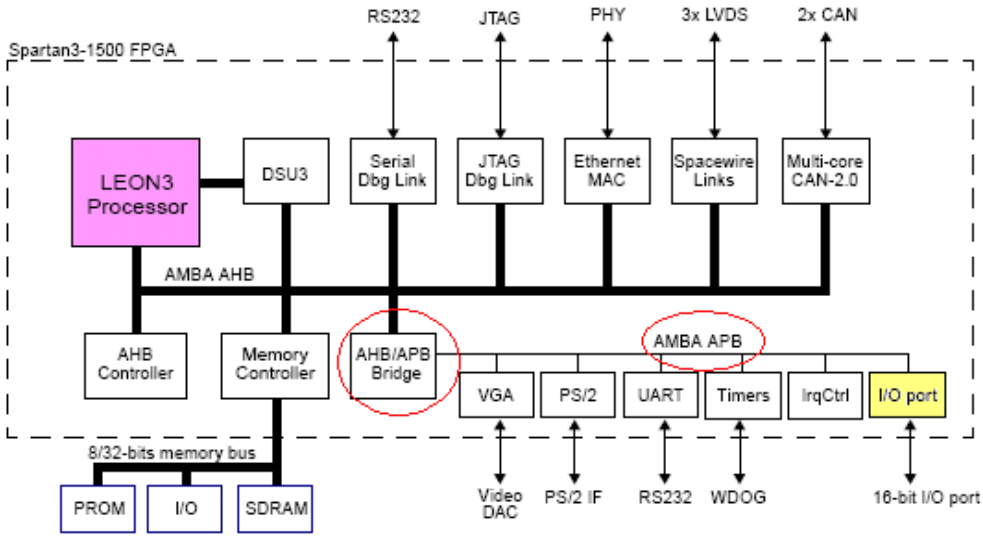


Figura B-8: Arquitectura de la tarjeta GR-XCS3-1500

B.4 Motor de tracción y reductora acoplada

El motor de tracción que está montado en la plataforma de la Figura B-2 es el *Maxon RE-36* mostrado en la Figura B-9. Es un motor de escobillas de grafito, con una potencia de 70W y un diámetro de 36mm. Tiene un voltaje nominal de 18V y se alimenta a través de una etapa de potencia descrita en el apartado B.8.

El funcionamiento del motor de tracción se basa en recibir una señal PWM, una señal de freno y otra de dirección. La velocidad es directamente proporcional al tiempo en alta de la señal PWM. La señal de dirección tiene que ser 0 para retroceder y 1 para avanzar hacia delante. La señal de freno bloquea el motor para que el coche pare con la mayor brevedad posible. Como se ha comentado anteriormente, la conexión del motor no es directa a la tarjeta, sino que estas tres señales pasan por la etapa de potencia (apartado B.8).

Acoplado al eje del motor de tracción se encuentra la reductora *Maxon GP-32K* (Figura B-10). Su factor de reducción es de 19:1 y utiliza dos etapas para poder alcanzar el factor de reducción deseado.

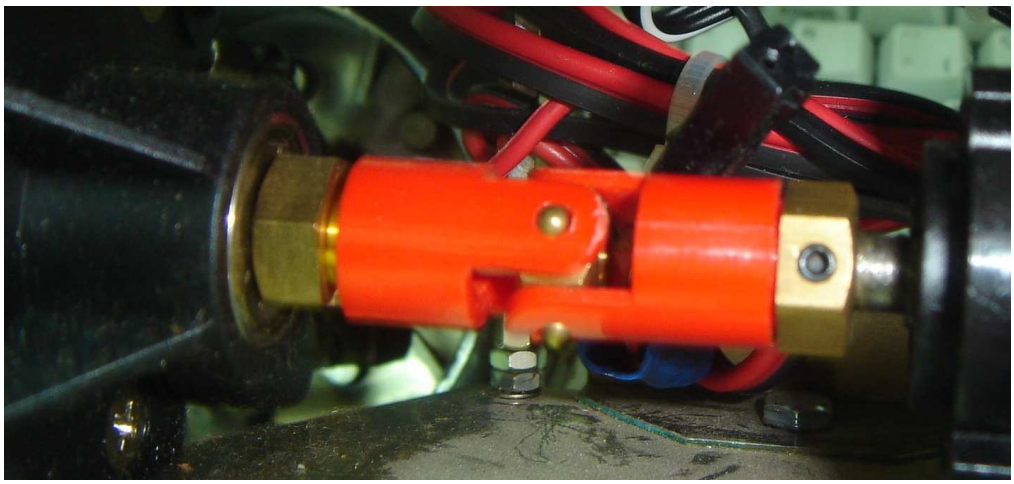


Figura B-9: Motor de tracción (ref. *Maxon* RE-36)

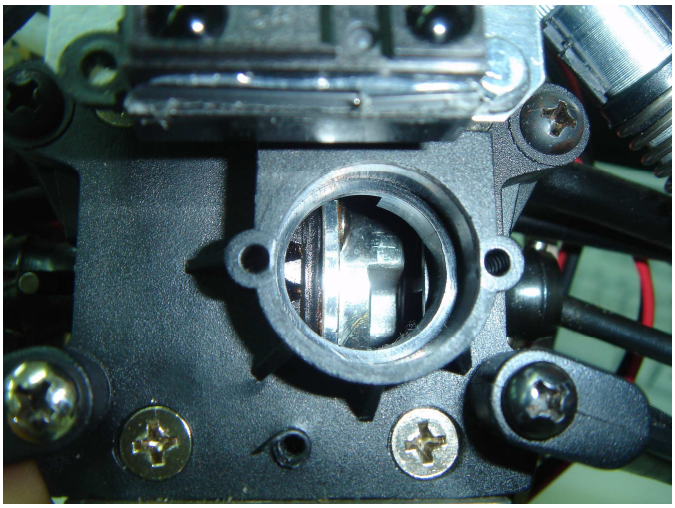


Figura B-10: Reductora 19:1 (ref. *Maxon* GP-32K)

Para permitir el movimiento del coche es necesario aplicar al motor una señal PWM de período 20 ms cuyo nivel en alta sea proporcional a la velocidad con la que se quiera desplazar el propio coche. La forma de onda sería la siguiente:

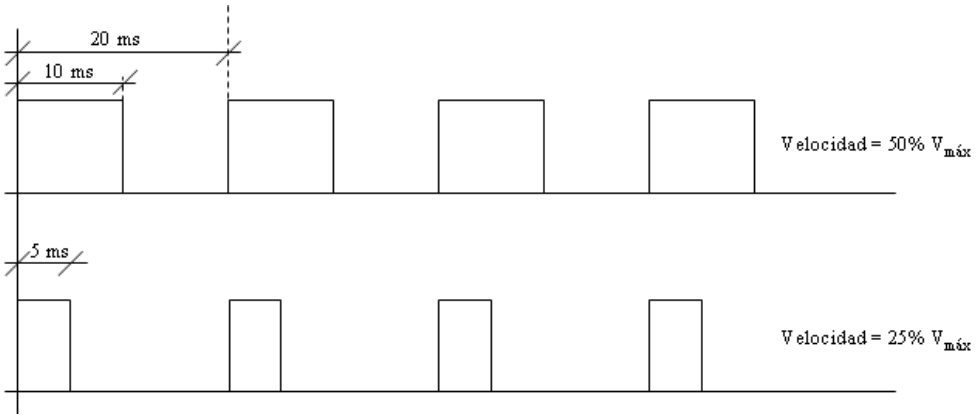


Figura B-11: Ejemplo de señal PWM para el motor de tracción

B.5 *Encoder* acoplado al eje del motor de tracción

El *encoder* elegido para la plataforma es el modelo *Maxon HEDS 5540*, de 500 pulsos por vuelta. Consiste básicamente en un diodo LED que haciendo pasar su luz por una rueda codificada excita unos

fotodetectores que son responsables de las tres señales de salida (véase Figura B-12).

Los canales A y B contienen idéntica información, pero no su fase. Cada 1/500 vueltas aparece un pulso en dichos canales, pero dependiendo del sentido de giro uno de los canales estará adelantado 90º sobre el otro. Analizando el desfase es posible calcular si la velocidad es positiva o negativa. Por último, conviene comentar que en el Canal A aparece un pulso cada vez que se realiza una vuelta completa, proporcionándonos información complementaria. Con estos datos es posible saber el ángulo girado por el eje del motor con una precisión de ± 1/9500 vueltas de eje de salida de la reductora (500 pulsos por *vuelta*factor_de_reducción*).

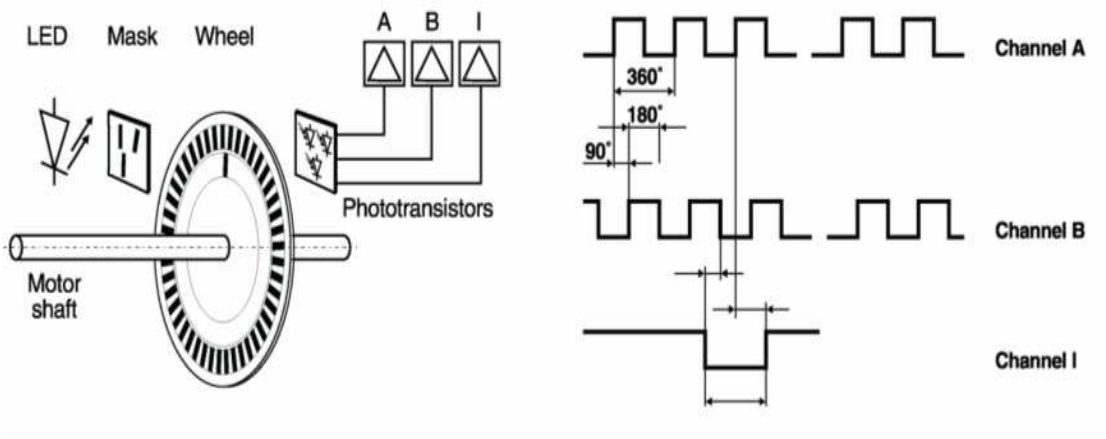


Figura B-12: Funcionamiento del *encoder*

B.6 Servomotor de dirección

Como motor de dirección se ha utilizado el servomotor *HS-985-MG*, representado en la Figura B-13. Es un dispositivo electromecánico con tres terminales: rojo y negro para alimentación y masa respectivamente, y un tercero para control (de color amarillo).

El eje del motor está unido al cursor de un potenciómetro. Gracias al valor del potenciómetro se puede calcular la posición exacta del eje, y comparando dicha posición con la demandada por el usuario, se puede determinar el nuevo sentido y ángulo de giro del eje.

La nueva posición de giro se genera mediante una señal PWM que se aplica al terminal de control. La señal de control posee las características que ilustra la Figura B-14:

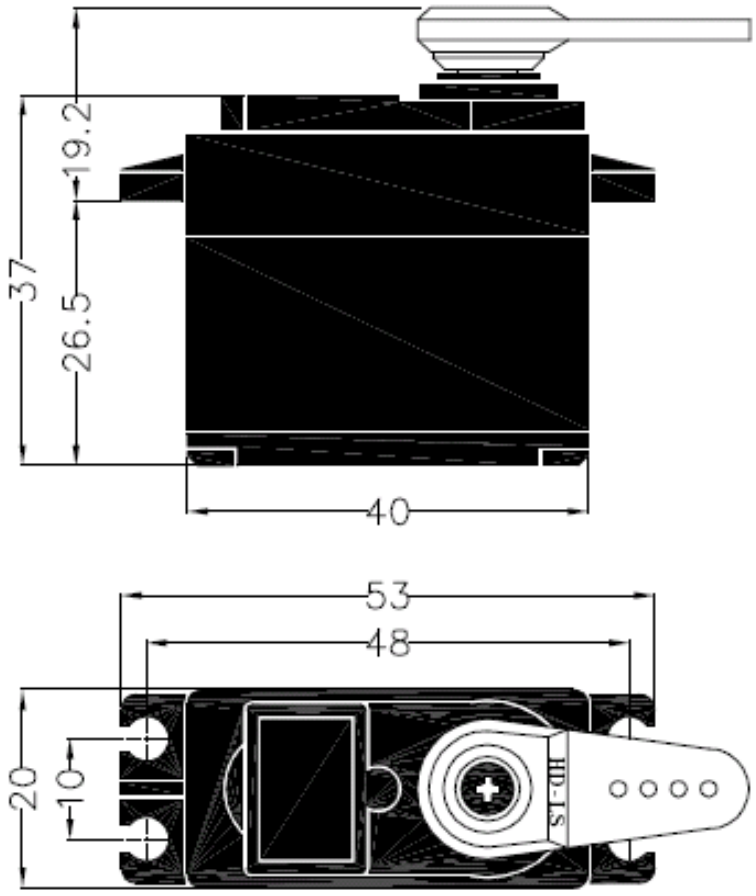


Figura B-13: Servo HITEC HS-985-MG

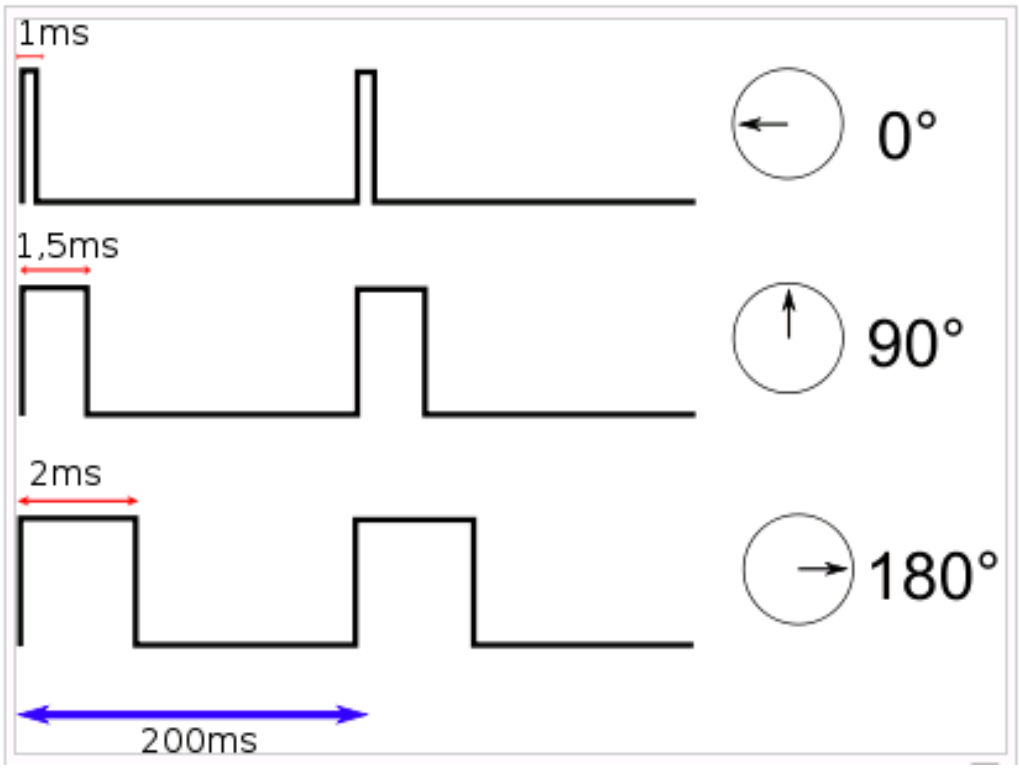


Figura B-14: Señal PWM de entrada al servo

En la Figura B-15 se muestra la localización del servo dentro de la plataforma:

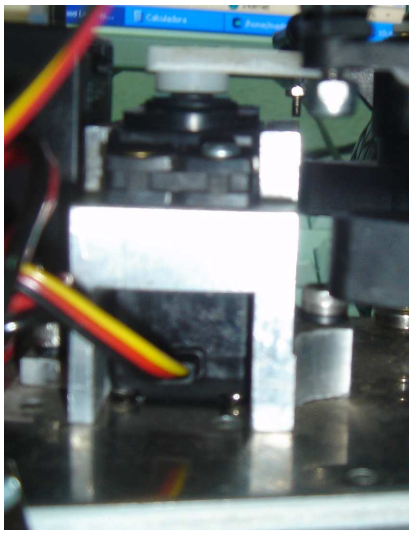


Figura B-15: Situación del servomotor dentro de la plataforma

Los valores de tiempo de la señal en alto están entre 1 y 2 ms, que posicionan al motor en ambos extremos de giro (0° y 180°, respectivamente). Los valores de tiempo en alto para ubicar al motor en otras posiciones se calcula mediante una relación completamente lineal: el valor 1,5 ms indica la posición central, y otros valores de duración del pulso dejarían al motor en la posición proporcional a dicha duración.

En la Figura B-16 se puede apreciar un diagrama con la correspondencia entre el nivel alto de la señal PWM y los grados recorridos por el servo.

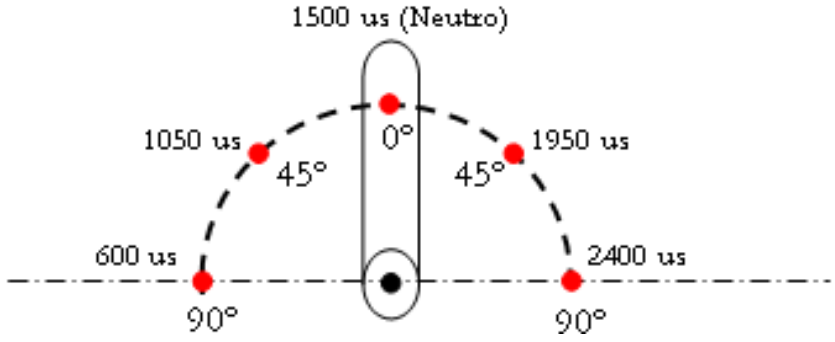


Figura B-16: Correspondencia entre la señal PWM y grados recorridos

Para bloquear el servomotor en una posición, es necesario enviarle continuamente la señal con la posición deseada. De esta forma, el sistema de control seguirá operando, y el servo conservará su posición y se resistirá a fuerzas externas que intenten cambiarlo de posición. Si los pulsos no se envían, el servomotor quedará liberado, y cualquier fuerza externa puede cambiarlo de posición fácilmente.

B.7 Comunicación *Bluetooth*

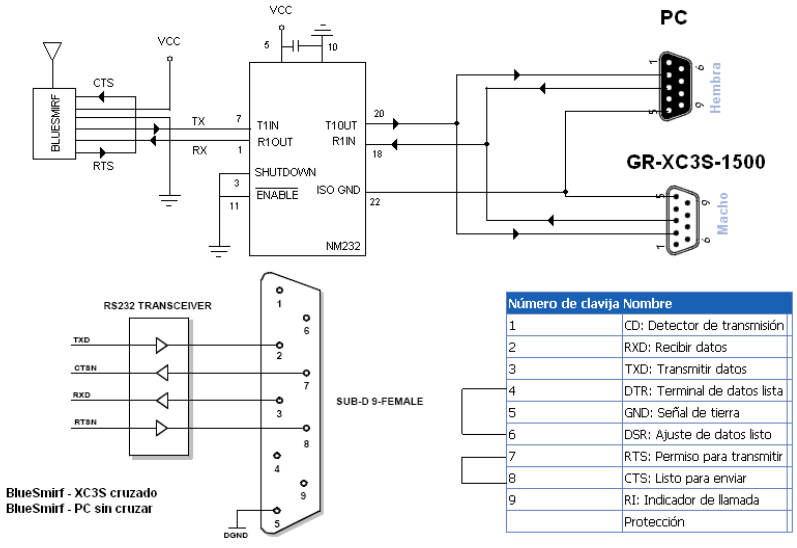
Se ha optado por este tipo de comunicación debido a la necesidad de disponer de un sistema inalámbrico que permitiera comunicar el vehículo con un ordenador. El uso de una tecnología de comunicación inalámbrica parece necesario en una plataforma

robótica que como fin principal tiene que poder desplazarse autónomamente.

Para la realización de la comunicación *bluetooth* se eligió el módulo *BlueSmirf* instalado en la plataforma móvil. Para su integración se ha diseñado un adaptador como el mostrado en la Figura B-17:

Figura B-17: Foto del transmisor y receptor

El diagrama de bloques del adaptador queda representado a continuación en la Figura B-18:



Número de clavija	Nombre
1	CD: Detector de transmisión
2	RXD: Recibir datos
3	TXD: Transmitir datos
4	DTR: Terminal de datos lista
5	GND: Señal de tierra
6	DSR: Ajuste de datos listo
7	RTS: Permiso para transmitir
8	CTS: Listo para enviar
9	RI: Indicador de llamada
	Protección

Figura B-18: Esquema adaptador *Bluetooth*

B.8 Etapa de potencia

Debido a que la corriente de salida de la tarjeta GR-XC3S-1500 no es lo suficientemente elevada como para excitar al motor de tracción, se hace imprescindible el uso de una etapa intermedia de potencia. Esto no ocurre con el servomotor de dirección, ya que el microcontrolador sólo excita la señal de control, la cual no proporciona energía eléctrica para mover el eje.

Dicha etapa de potencia consta de 2 puentes en H modelo *LMD 18200*, el optoacoplador *6N137* para proporcionar aislamiento eléctrico entre la etapa de potencia y digital, así como lógica adicional destinada a la excitación de motores. La Figura B-19 contiene un diagrama esquemático de la etapa de potencia.

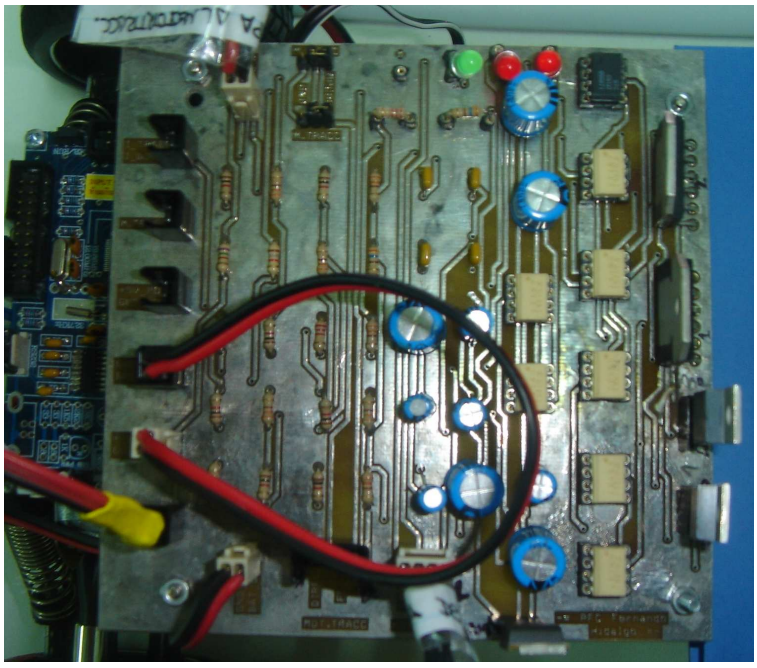


Figura B-19: Placa de potencia