UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE

INGENIEROS INDUSTRIALES

ARQUITECTURA DE CONTROL PARA LA CONDUCCIÓN AUTÓNOMA DE VEHÍCULOS EN ENTORNOS URBANOS Y AUTOVÍAS

TESIS DOCTORAL

JORGE GODOY

MÁSTER EN AUTOMÁTICA Y ROBÓTICA

2013

Departamento de Automática, Ingeniería Electrónica
e Informática Industrial
Escuela Técnica Superior de Ingenieros Industriales

Arquitectura de Control para la Conducción Autónoma de Vehículos en Entornos Urbanos y Autovías

---

Autor
Jorge Godoy
Máster en Automática y Robótica

Directores
Carlos González Fernández-Vallejo
Doctor en Ciencias Físicas

Vicente Milanés Montero
Doctor en Ingeniería Electrónica

2013

Este documento está preparado para ser impreso a doble cara.

Título:

Arquitectura de Control para la Conducción Autónoma de Vehículos en Entornos Urbanos y Autovías

Autor:

Jorge Luis Godoy Madrid

Máster en Automática y Robótica

Tribunal nombrado por el Mgfco. y Excmo. Sr. Rector de la Universidad Politécnica de Madrid, el día_____de_____de 2013

Presidente: _____

Secretario: _____

Vocal: _____

Vocal: _____

Vocal: _____

Suplente: _____

Suplente: _____

Realizado el acto de lectura y defensa de la tesis el día _____ de _____ de 2013, en _____, el tribunal acuerda otorgar la calificación de:

_____

EL PRESIDENTE EL SECRETARIO

LOS VOCALES

A mi madre,
mi apoyo incondicional.

A Mirvs, Eli y Tita,
porque más grande que el obstáculo
serán siempre sus ganas de superarlo.

AGRADECIMIENTOS

Cuatro años han pasado desde el momento en que iniciara la investigación cuyo principal fruto es la presente tesis doctoral. A lo largo de este tiempo me he beneficiado de la colaboración y apoyo de distintas personas y entidades, sin las cuales no habría sido posible alcanzar este punto.

En primer lugar, quiero agradecer al Consejo Superior de Investigaciones Científicas, que ha financiado mi formación por medio del programa de becas Junta para la Ampliación de Estudios.

A mis directores, Carlos González y Vicente Milanés, quienes han tenido la paciencia y constancia adecuada para guiarme en todo momento durante la elaboración de este trabajo, no sólo como directores sino también con una gran amistad plagada de anécdotas.

A Teresa de Pedro, quien como investigadora principal del Programa AUTOPIA me ha dado la oportunidad de formar parte de este grupo. Igualmente a su esposo Ricardo García (Q.E.P.D), cofundador del programa, quien me demostró que la jovialidad no se pierde con la edad y que lamentablemente por razones del destino no ha podido ver la finalización de este trabajo.

Un profundo agradecimiento a mis compañeros del Programa AUTOPIA, en especial a Joshué Pérez, Jorge Villagrá y Enrique Onieva; quienes me han brindado su apoyo tanto en el ámbito profesional como personal; además de haber confiado en mí para subirse a un coche sin conductor a más de 100 km/h con el único pretexto de querer cambiar el mundo.

Al profesor Manuel Ferre, quien ha asumido la importante responsabilidad de relevar al profesor Ramón Galán como tutor académico de este trabajo, presentando además una gran disposición a hacer que las cosas resulten más sencillas.

A Dominique Gruyer y Martin Lauer, responsables de los grupos de investigación del LIVIC y el MRT respectivamente, quienes me dieron la oportunidad de compartir la experiencia de investigación con sus grupos, aportándome conocimientos en nuevas áreas de los ITS.

A mis compañeros del CAR, en especial a Bego, Lola, Nacho, Jaime, Ángel, Jesús, Rodolfo, Agustín, Raúl, Fernando(s) y a tantos otros con los que he compartido el día a día de estos últimos años y que, desde mi llegada, han hecho más llevadera mi estancia en España.

Al personal del CAR, Maria Eugenia, Chicho, Rufina, Pepi(s), Alicia, África, Fernando, Tártalo, Miguel y Andrés, quienes aportan su grano de arena para que el centro funcione adecuadamente a pesar de las actuales adversidades.

A toda mi familia, que son tantos que necesitaría un capítulo entero para mencionarlos,

pero que en todo momento han estado pendientes de mí y de mis estudios, a pesar de toda la distancia y la diferencia horaria que nos separa. Gracias de verdad por todo su apoyo para seguir adelante.

A mis amigos, Mirvy, Erika, Giselle, Andrés, Natalia, Gaby, Luis, Manuel, Grey, Emmi, Xavier, Alejandro y todos aquellos que forman parte de esta comunidad venezolana en Madrid, haciéndome sentir siempre como en casa. También a los adoptados, Christian, Sylvia, Julie, Gaëlle y Pierre Antoine que como buenos embajadores me han recibido con cariño y afecto en Madrid.

A estas personas y a alguna más que se me haya podido escapar, pero que con su colaboración y apoyo han contribuido al desarrollo de esta tesis doctoral.

A todos, sinceramente mi más profundo y sincero agradecimiento.

RESUMEN

Hoy en día, el desarrollo tecnológico en el campo de los sistemas inteligentes de transporte (ITS por sus siglas en inglés) ha permitido dotar a los vehículos con diversos sistemas de ayuda a la conducción (ADAS, del inglés *advanced driver assistance system*), mejorando la experiencia y seguridad de los pasajeros, en especial del conductor. La mayor parte de estos sistemas están pensados para advertir al conductor sobre ciertas situaciones de riesgo, como la salida involuntaria del carril o la proximidad de obstáculos en el camino. No obstante, también podemos encontrar sistemas que van un paso más allá y son capaces de cooperar con el conductor en el control del vehículo o incluso relegarlos de algunas tareas tediosas. Es en este último grupo donde se encuentran los sistemas de control electrónico de estabilidad (ESP - *Electronic Stability Program*), el antibloqueo de frenos (ABS - *Anti-lock Braking System*), el control de crucero (CC - *Cruise Control*) y los más recientes sistemas de aparcamiento asistido. Continuando con esta línea de desarrollo, el paso siguiente consiste en la supresión del conductor humano, desarrollando sistemas que sean capaces de conducir un vehículo de forma autónoma y con un rendimiento superior al del conductor.

En este trabajo se presenta, en primer lugar, una arquitectura de control para la automatización de vehículos. Esta se compone de distintos componentes de hardware y software, agrupados de acuerdo a su función principal. El diseño de la arquitectura parte del trabajo previo desarrollado por el Programa AUTOPIA, aunque introduce notables aportaciones en cuanto a la eficiencia, robustez y escalabilidad del sistema.

Ahondando un poco más en detalle, debemos resaltar el desarrollo de un algoritmo de localización basado en enjambres de partículas. Este está planteado como un método de filtrado y fusión de la información obtenida a partir de los distintos sensores embarcados en el vehículo, entre los que encontramos un receptor GPS (*Global Positioning System*), unidades de medición inercial (IMU – *Inertial Measurement Unit*) e información tomada directamente de los sensores embarcados por el fabricante, como la velocidad de las ruedas y posición del volante. Gracias a este método se ha conseguido resolver el problema de la localización, indispensable para el desarrollo de sistemas de conducción autónoma.

Continuando con el trabajo de investigación, se ha estudiado la viabilidad de la aplicación de técnicas de aprendizaje y adaptación al diseño de controladores para el vehículo. Como punto de partida se emplea el método de *Q-learning* para la generación de un controlador borroso lateral sin ningún tipo de conocimiento previo. Posteriormente se presenta un método de ajuste *on-line* para la adaptación del control longitudinal ante perturbaciones impredeci-

bles del entorno, como lo son los cambios en la inclinación del camino, fricción de las ruedas o peso de los ocupantes.

Para finalizar, se presentan los resultados obtenidos durante un experimento de conducción autónoma en carreteras reales, el cual se llevó a cabo en el mes de Junio de 2012 desde la población de San Lorenzo de El Escorial hasta las instalaciones del Centro de Automática y Robótica (CAR) en Arganda del Rey. El principal objetivo tras esta demostración fue validar el funcionamiento, robustez y capacidad de la arquitectura propuesta para afrontar el problema de la conducción autónoma, bajo condiciones mucho más reales a las que se pueden alcanzar en las instalaciones de prueba.

ABSTRACT

Nowadays, the technological advances in the Intelligent Transportation Systems (ITS) field have led the development of several driving assistance systems (ADAS). These solutions are designed to improve the experience and security of all the passengers, especially the driver. For most of these systems, the main goal is to warn drivers about unexpected circumstances leading to risk situations such as involuntary lane departure or proximity to other vehicles. However, other ADAS go a step further, being able to cooperate with the driver in the control of the vehicle, or even overriding it on some tasks. Examples of this kind of systems are the anti-lock braking system (ABS), cruise control (CC) and the recently commercialised assisted parking systems. Within this research line, the next step is the development of systems able to replace the human drivers, improving the control and therefore, the safety and reliability of the vehicles.

First of all, this dissertation presents a control architecture design for autonomous driving. It is made up of several hardware and software components, grouped according to their main function. The design of this architecture is based on the previous works carried out by the AUTOPIA Program, although notable improvements have been made regarding the efficiency, robustness and scalability of the system.

It is also remarkable the work made on the development of a location algorithm for vehicles. The proposal is based on the emulation of the behaviour of biological swarms and its performance is similar to the well-known particle filters. The developed method combines information obtained from different sensors, including GPS, inertial measurement unit (IMU), and data from the original vehicle's sensors on-board. Through this filtering algorithm the localization problem is properly managed, which is critical for the development of autonomous driving systems.

The work deals also with the fuzzy control tuning system, a very time consuming task when done manually. An analysis of learning and adaptation techniques for the development of different controllers has been made. First, the Q-learning –a reinforcement learning method– has been applied to the generation of a lateral fuzzy controller from scratch. Subsequently, the development of an adaptation method for longitudinal control is presented. With this proposal, a final cruise control controller is able to deal with unpredictable environment disturbances, such as road slope, wheel's friction or even occupants'weight.

As a testbed for the system, an autonomous driving experiment on real roads is presented. This experiment was carried out on June 2012, driving from San Lorenzo de El Escorial up

to the Center for Automation and Robotics (CAR) facilities in Arganda del Rey. The main goal of the demonstration was validating the performance, robustness and viability of the proposed architecture to deal with the problem of autonomous driving under more demanding conditions than those achieved on closed test tracks.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

INTRODUCCIÓN

En las últimas décadas el transporte terrestre ha absorbido el crecimiento producido en la movilidad tanto de personas como de mercancías. Prueba de ello es que actualmente este sector representa cerca del 1,6 % del producto interno bruto de la Unión Europea (UE) y proporciona alrededor de 5 millones de puestos de trabajo (Comisión Europea, 2006). No obstante, este crecimiento acarrea diversos problemas en cuanto a la seguridad y a la fluidez del tráfico. Hoy en día la mayor parte de los accidentes de tráfico son debidos al conductor humano, por lo que la investigación en sistemas de ayuda a la conducción, bien sean de asistencia total o parcial, contribuye tanto a mejorar la calidad de vida de los conductores, reduciendo el tiempo en tránsito, como a incrementar la seguridad en las carreteras

Actualmente, los vehículos incorporan una diversidad de sistemas inteligentes, diseñados para incrementar su seguridad, confort y eficiencia. Estas soluciones se engloban dentro de los llamados sistemas de ayuda a la conducción, los cuales si bien iniciaron como sistemas totalmente pasivos, han alcanzado un nivel de desarrollo en el que son capaces de interactuar y cooperar con el conductor, a fin de llevar a cabo el control del vehículo de forma adecuada. La aparición de este tipo de soluciones nos acerca a la idea de la conducción 100 % autónoma de vehículos, aunque aún queda un largo camino por recorrer para que esta se haga una realidad en las carreteras.

En este sentido, se orienta el tema de la presente tesis doctoral hacia el desarrollo de una arquitectura de control para la conducción autónoma de vehículos. El trabajo realizado se engloba dentro del marco de desarrollo e investigación del Programa AUTOPIA, el cual es uno de los principales grupos de investigación españoles en el campo de la conducción autónoma. A continuación se explica detalladamente el propósito y los objetivos del presente trabajo, así como las principales aportaciones realizadas.

1.1. Propósito

De acuerdo a la información proporcionada por Eurostat[1] —oficina de estadísticas de la Unión Europea—, entre los años 2001 y 2009 se registró un aumento en torno al 12 % en el

[1] http://epp.eurostat.ec.europa.eu/

número de vehículos de pasajeros —i.e. vehículos para menos de 10 personas— en la UE. Este crecimiento se refleja directamente en el incremento de la congestión del tráfico (atascos) y emisiones de carbono a la atmósfera. Contrario a lo esperado, el número de victimas mortales por accidentes de tráfico ha disminuido en un 36 % en el mismo intervalo de tiempo, indicando que tanto las carreteras como los vehículos se están haciendo más seguros.

A pesar de esta notable reducción en el número de víctimas, tanto los fabricantes como los centros de investigación a nivel mundial continúan abocándose al desarrollo de sistemas y soluciones que permitan reducir aun más esta cifra. Así pues, el desarrollo de sistemas de conducción autónoma de vehículos es una de las lineas de investigación más prometedoras de los últimos años, a partir de la estimación de que la mayor parte de los accidentes de tráfico son causados por errores humanos. Por otro lado, la implementación de sistemas de control con mayor precisión a la de un conductor humano permitiría reducir la distancia de separación entre los vehículos, incrementando la capacidad de las carreteras (Shladover, 2009). De igual forma, sería posible emplear modos de conducción mucho más eficientes, desde el punto de vista del consumo de combustible y las emisiones de carbono a la atmósfera.

En España, el Programa AUTOPIA es uno de los principales grupos de investigación enfocado a esta línea de desarrollo, realizando la continua transferencia de técnicas basadas en inteligencia artificial al control autónomo de vehículos. El inicio de su respetada trayectoria se remonta hasta hace más de 15 años, cuando llevó a cabo la automatización de dos furgonetas eléctricas (Naranjo, 2005). Desde entonces los miembros del equipo han trabajado en pro del desarrollo de sistemas de conducción autónoma de vehículos, realizando continuas aportaciones a la arquitectura de control (Milanés, 2010). Así mismo se ha trabajado en el desarrollo de aplicaciones diseñadas para abordar algunos de los escenarios típicos de la conducción como son las intersecciones, adelantamientos e incorporaciones (Pérez et al., 2010; Milanés et al., 2011b, 2012b).

Aunque en principio AUTOPIA ha conseguido resolver parcialmente el problema de la conducción autónoma, el conjunto del sistema empleado hasta el momento de la realización de la presente tesis presenta algunas limitaciones importantes, la mayor parte de ellas relacionadas con la estructura empleada para la arquitectura de control. En primer lugar, se sufre de una alta dependencia de los receptores GPS para el posicionamiento del vehículo, lo cual impide abordar situaciones donde estos son afectados por interferencias del entorno. En segundo lugar, la constante evolución de la arquitectura de control ha alcanzado un punto donde los sistemas se han vuelto altamente dependientes entre si, dificultando la expansión del sistema y por lo tanto reduciendo su escalabilidad. De igual forma, el manejo de los recursos a nivel de software no es óptimo, acarreando diversos problemas en el procesamiento de datos y comunicación con los periféricos (Pérez, 2012). Por último, con excepción de la participación en el *Grand Cooperative Driving Challenge* (GCDC), la validación del sistema se ha realizado únicamente en pistas de pruebas, las cuales no someten el sistema a las mismas condiciones de las carreteras reales.

A partir de estos planteamientos, se establece como objetivo principal de este trabajo el desarrollo e implementación de una nueva arquitectura de control para la conducción

autónoma de vehículos, teniendo como prioridad la eficiencia y robustez del sistema. Para ello nos planteamos cumplir con las siguientes metas:

1. Realizar una reestructuración completa de la arquitectura de control, priorizando tanto en el manejo de los recursos como en la robustez y escalabilidad general de la misma.

2. Desarrollar un sistema de localización basado en la combinación de información de distintos sensores, reduciendo la dependencia del sistema a la calidad de la medición de los receptores GPS, al menos ante situaciones concretas.

3. Desarrollar controladores capaces de aprovechar su propia interacción con el entorno para mejorar su rendimiento ante el problema para el cual han sido diseñados.

4. Validar el funcionamiento de todo el sistema desarrollado a través de un experimento en carreteras reales, verificando principalmente la robustez de la arquitectura y el comportamiento de los distintos controladores implementados.

1.2. Estructura de la memoria

A continuación presentamos una breve introducción del contenido de los seis capítulos restantes:

En el **capítulo 2** se describe el estado actual del desarrollo y la investigación en el ámbito a los Sistemas Inteligentes de Transporte. Así, se inicia con la introducción a este concepto, incluyendo los principales motivos que incitan su desarrollo. Posteriormente, se describen las últimas soluciones disponibles comercialmente en lo referente a los sistemas de ayuda a la conducción, comentando además la situación actual de esta línea de investigación. Por último, el capitulo presenta las principales aportaciones de algunos grupos, centros y proyectos de investigación relacionados con este campo.

En el **capítulo 3** se presenta el diseño e implementación de la arquitectura de control propuesta en este trabajo, la cual se enfoca en la descentralización —a nivel de software— de los procesos de control. El capítulo inicia con la descripción de la arquitectura de control del Programa AUTOPIA, tal y como se encontraba en el punto previo al desarrollo de esta tesis doctoral. Seguidamente, se detalla la instrumentación de un vehículo prototipo, empleado como plataforma de desarrollo y validación del sistema propuesto. Finalmente, se introduce la arquitectura de control propuesta, resaltando las principales aportaciones respecto al esquema anterior.

El **capítulo 4** aborda uno de los retos fundamentales para la conducción autónoma: la localización fiable y robusta de los vehículos. En este capítulo se detalla el desarrollo de un algoritmo de localización basado en la fusión de información de distintos sensores. Para ello, se emplea un enfoque similar al de los ya conocidos filtros de partículas, considerando además un comportamiento de enjambre durante el proceso de estimación del estado del vehículo. Este algoritmo representa uno de los primeros pasos del Programa AUTOPIA en lo referente a la localización.

El **capítulo 5**, dividido en dos secciones, se centra en la aplicación de técnicas de inteligencia artificial, relacionadas con el aprendizaje y adaptación, al control de vehículos. En primer lugar, se describe el desarrollo de un sistema de control lateral basado en aprendizaje por refuerzo, comprobando la capacidad del sistema para obtener la mejor respuesta del vehículo sin ningún tipo de conocimiento previo. La segunda parte del capítulo introduce un método de aprendizaje y adaptación aplicado al control longitudinal del vehículo. El método se basa en la evaluación *on-line* del rendimiento de controladores borrosos, adaptando la estructura de los mismos para mejorar la respuesta del sistema ante posibles perturbaciones impredecibles del entorno.

En el **capítulo 6** se presentan los resultados de una demostración de conducción autónoma realizada en carreteras abiertas al tráfico. El objetivo de este experimento ha sido validar el funcionamiento, viabilidad y robustez de la arquitectura, algoritmos y métodos desarrollados a lo largo de este trabajo. La demostración ha consistido en un recorrido de 100 km realizados de forma autónoma a través de entornos urbanos y autovías.

Por último, el **capítulo 7** concluye el trabajo realizado, comentando las aportaciones de la presente tesis y las líneas futuras de trabajo que surgen de la misma.

1.3. Contribuciones

A juicio del autor, las principales contribuciones de la presente tesis son:

- El desarrollo de una nueva arquitectura de control para la conducción autónoma de vehículos, la cual hace hincapié en la optimización de recursos computacionales a fin de incrementar la eficiencia global del sistema. Por otro lado, gracias a la estructura modular de software que ha sido implementada, se facilita en gran medida la incorporación de nuevos elementos al sistema, así como la modificación de los ya existentes. De igual forma, es posible fragmentar la ejecución de software de control en varios ordenadores, extendiendo la capacidad de computo del sistema. Adicionalmente, esta estructura otorga cierto grado de independencia de ejecución entre los distintos componentes, minimizando el impacto que genera la falla de uno de los elementos sobre el rendimiento general del sistema.

- La mejora significativa del sistema de posicionamiento de los vehículos, gracias a un algoritmo de localización que combina la información obtenida desde distintos sensores embarcados. Con este algoritmo se ha conseguido sortear una de las principales debilidades del sistema previamente empleado por el Programa AUTOPIA, incapaz de operar en situaciones donde los receptores GPS presentan errores elevados en la medición.

- El estudio e implementación de técnicas de aprendizaje y adaptación para la mejora del control de los vehículos. La aplicación del aprendizaje por refuerzo al control lateral ha demostrado la capacidad del sistema para abordar el problema del guiado sin ningún tipo de conocimiento previo. Así mismo, la capacidad de adaptación alcanzada

con el método de control longitudinal permite sortear perturbaciones impredecibles del entorno, mejorando notablemente la respuesta del sistema.

- La realización de una demostración de conducción autónoma en carreteras reales, a través de la cual se ha demostrado la robustez y capacidad conjunta de los sistemas desarrollados para llevar a cabo la conducción autónoma de un vehículo de forma segura. Durante el experimento se ha circulado a través de entornos urbanos y autovías, alcanzando velocidades de hasta 100 km/h en un recorrido de poco menos de 100 km de longitud. Sin lugar a dudas, esta demostración constituye el logro más significativo del Programa AUTOPIA en los últimos años, ostentando la mayor marca de velocidad, distancia y tiempo de ejecución de todos los experimentos realizados hasta el momento.

1.4. Difusión de resultados

Durante el desarrollo del presente trabajo de tesis, el doctorando ha tenido la oportunidad de participar en distintos proyectos de investigación, tanto nacionales como europeos, los cuales le han valido para establecer vínculos de trabajo y colaboración con otros centros de investigación, universidades y empresas relacionadas. Así mismo, el doctorando ha realizado estancias internacionales en dos importantes centros de investigación:

- Durante el 2011, en el *Laboratoire sur les Interactions Véhicules-Infrastructure-Conducteurs* (LIVIC), del Instituto Francés de Ciencias y Tecnologías para el Transporte, Desarrollo y Redes, en Versalles (Francia), bajo la supervisión del Dr. Dominique Gruyer.
- Durante el 2012, en el equipo AnnieWAY del *Institut für Mess- und Regelungstechnik* (MRT) de la Universidad de Karlsruhe (KIT), en Karlsruhe (Alemania), bajo la supervisión del Dr. Martin Lauer.

Producto de todo el trabajo realizado a lo largo de los últimos años, tanto en el desarrollo de la tesis aquí presentada como en los proyectos, estancias y colaboraciones con otros centros; han surgido diversas publicaciones científicas en revistas y congresos relacionados con la robótica, el control, la inteligencia artificial y el transporte; las cuales se citan a continuación:

Publicaciones en revistas

Título: An Auxiliary V2I Network for Road Transport and Dynamic Environments
Autores: J. Godoy, V. Milanés, J. Pérez, J. Villagrá y E. Onieva
Revista: Transportation Research Part C
Estado: En revisión.

Título: A Modular Architecture for Fully-Controlled Vehicles
Autores: J. Godoy, J. Pérez, V. Milanés, J. Villagrá y E. Onieva
Revista: Mechanical Systems and Signal Processing

Estado: En revisión.

Título: A Driverless Vehicle Demonstration on Motorways and in Urban Environments
Autores: J. Godoy, J. Pérez, J. Villagrá, E. Onieva y V. Milanés
Revista: Engineering Applications of Artificial Intelligence
Estado: En revisión.

Título: Automated On-Ramp Merging System for Congested Traffic Situations
Autores: V. Milanés, J. Godoy, J. Villagrá y J. Pérez
Revista: IEEE Transactions on Intelligent Transportation Systems
Estado: Publicado. Vol. 12, no. 2, pp. 500–508, 2011

Título: Precise vehicle cruise control system based on on-line fuzzy control learning
Autores: E. Onieva, J. Godoy y J. Villagrá
Revista: Communications in Computer and Information Science
Estado: Publicado. Vol. 297, pp. 101–110, 2012

Título: On-line learning of a fuzzy controller for a precise vehicle cruise control system
Autores: E. Onieva, J. Godoy, J. Villagrá, V. Milanés y J. Pérez
Revista: Expert Systems with Applications
Estado: Publicado. Vol. 40, no. 4, pp. 1046–1053, 2013

Título: A fuzzy aid rear-end collision warning/avoidance system
Autores: V. Milanés, J. Pérez, J. Godoy y E. Onieva
Revista: Expert Systems with Applications
Estado: Publicado. Vol. 39, no. 10, pp. 9097–9107, 2012

Título: An evolutionary tuned driving system for virtual car racing games: The AUTOPIA driver
Autores: E. Onieva, D. A. Pelta, J. Godoy, V. Milanés y J. Pérez
Revista: International Journal of Intelligent Systems
Estado: Publicado. Vol. 27, no. 3, pp. 217–241, 2012

Título: An Intelligent V2I-Based Traffic Management System
Autores: V. Milanés, J. Villagrá, J. Godoy, J. Simó, J. Pérez y E. Onieva
Revista: IEEE Transactions on Intelligent Transportation Systems
Estado: Publicado. Vol. 13, no. 1, pp. 49–58, 2012

Título: Comparing Fuzzy and Intelligent PI Controllers in Stop-and-Go Manoeuvres
Autores: V. Milanés, J. Villagrá, J. Godoy y C. González
Revista: IEEE Transactions on Control Systems Technology
Estado: Publicado. Vol. 20, no. 3, pp. 770–778, 2012

Título: Cooperative controllers for highways based on human experience
Autores: J. Pérez, V. Milanés, J. Godoy, J. Villagrá y E. Onieva
Revista: Expert Systems with Applications
Estado: Publicado. Vol. 40, no. 4, pp. 1024–1033, 2013

Título: Ultrasonic sensors in urban traffic driving-aid systems.
Autores: L. Alonso, V. Milanés, C. Torre-Ferrero, J. Godoy, J. P. Oria y T. de Pedro
Revista: Sensors
Estado: Publicado. Vol. 11, no. 1, pp. 661–673, 2011

Título: Smooth path and speed planning for an automated public transport vehicle
Autores: J. Villagrá, V. Milanés, J. Pérez y J. Godoy
Revista: Robotics and Autonomous Systems
Estado: Publicado. Vol. 60, no. 2, pp. 252–265, 2012

Título: Traffic jam driving with NMV avoidance
Autores: V. Milanés, L. Alonso, J. Villagrá, J. Godoy, T. de Pedro y J. P. Oria
Revista: Mechanical Systems and Signal Processing
Estado: Publicado. Vol. 31, pp. 332–344, 2012

Título: Genetic optimization of a vehicle fuzzy decision system for intersections
Autores: E. Onieva, V. Milanés, J. Villagrá, J. Pérez y J. Godoy
Revista: Expert Systems with Applications
Estado: Publicado. Vol. 39, no. 18, pp. 13148–13157, 2012

Publicaciones en congresos

Título: Implementación de un sistema de localización para vehículos sin conductor
Autores: J. Godoy, V. Milanés, J. Pérez, J. Villagrá, T. de Pedro y C. González
Congreso: Seminario Anual de Automática, Electrónica Industrial e Intrumentación - SAAEI 2010

Título: Power electric aiding controller for automated bus stopping
Autores: J. Godoy, V. Milanés, J. Pérez, J. Villagrá y C. González
Congreso: Compatibility and Power Electronics (CPE), 2011

Título: Development of an particle swarm algorithm for vehicle localization
Autores: J. Godoy, D. Gruyer, A. Lambert y J. Villagrá
Congreso: IEEE Intelligent Vehicles Symposium - IV 2012

Título: Virtual Vehicle Approach for Longitudinal Control in Urban Environments
Autores: J. Godoy, J. Villagrá, T. de Pedro y R. Galán
Congreso: 14th Computer Aided Systems Theory - EUROCAST 2013

Título: V2I-based architecture for information exchange among vehicles
Autores: V. Milanés, J. Godoy, J. Pérez, B. Vinagre, C. González, E. Onieva y J. Alonso
Congreso:7th IFAC Symposium on Intelligent Autonomous Vehicles, 2010

Título: Traffic Light Intelligent Regulation Using Infrastructure Located Sensors
Autores: J. Alonso, J. Godoy, R. Sanz, E. Onieva, V. Milanés, J. Villagrá, C. González, T. de Pedro y R. García
Congreso: 13th Computer Aided Systems Theory - EUROCAST 2011

Título: Path following with backtracking based on fuzzy controllers for forward and reverse driving
Autores: J. Pérez, J. Godoy, V. Milanés, J. Villagrá y E. Onieva
Congreso: IEEE Intelligent Vehicles Symposium - IV 2012

Título: Nearly-time optimal smooth path planning using continuous curvature derivative primitives
Autores: J. Villagrá, J. Godoy, T. de Pedro y C. González
Congreso: 14th Computer Aided Systems Theory - EUROCAST 2013

Título: Implementación del control lateral sobre un vehículo de serie con servodirección asistida
Autores: E. de Torres, J. Villagrá, J. Godoy y T. de Pedro
Congreso: 8º Workshop Robocity2030-II – Robots de Exteriores, 2010

Título: Low speed control of an autonomous vehicle by using a fractional PI controller
Autores: I. Tejado, V. Milanés, J. Villagrá, J. Godoy, H. HosseinNia y B. M. Vinagre
Congreso: 18th IFAC World Congress, 2011

Título: A Reinforcement Learning Modular Control Architecture for Fully Automated Vehicles
Autores: J. Villagrá, V. Milanés, J. Pérez, J. Godoy, E. Onieva, J. Alonso, C. González, T. de Pedro y R. García
Congreso: 13th Computer Aided Systems Theory - EUROCAST 2011

Título: An approach to driverless vehicles in highways
Autores: V. Milanés, E. Onieva, J. Pérez, J. Godoy y J. Villagrá
Congreso: IEEE Conference on Intelligent Transportation Systems, 2011

Título: Longitudinal fuzzy control for autonomous overtaking

Autores: J. Pérez, V. Milanés, E. Onieva, J. Godoy y J. Alonso
Congreso: IEEE International Conference on Mechatronics - ICM 2011

Título: Path and speed planning for smooth autonomous navigation
Autores: J. Villagrá, V. Milanés, J. Pérez, J. Godoy y E. Onieva
Congreso: Workshop Navigation, Perception, Accurate Positioning and Mapping for Intelligent Vehicles, IEEE Intelligent Vehicles Symposium - IV 2012

Título: Modularity, adaptability and evolution in the AUTOPIA architecture for control of autonomous vehicles
Autores: J. Pérez, C. González, V. Milanés, E. Onieva, J. Godoy y T. de Pedro
Congreso: IEEE International Conference on Mechatronics - ICM 2009

Título: AUTOPIA Program Advances: How to Automate the Traffic?
Autores: V. Milanés, E. Onieva, J. Pérez, J. Villagrá, J. Godoy, J. Alonso, C. González, T. de Pedro y R. García
Congreso: 13th Computer Aided Systems Theory - EUROCAST 2011

Título: Study of Traffic Flow Controlled with Independent Agent-Based Traffic Signals
Autores: E. Onieva, V. Milanés, J. Pérez, J. Alonso, T. de Pedro, R. García, J. Godoy y J. Villagrá
Congreso: 13th Computer Aided Systems Theory - EUROCAST 2011

CAPÍTULO 2

ESTADO DEL ARTE

Hoy en día los vehículos totalmente automatizados siguen siendo una idea asociada al ámbito de la investigación, permaneciendo fuera del alcance del público general. No obstante, con el paso del tiempo se ha ampliado el abanico de sistemas de ayuda a la conducción (ADAS) disponibles para los vehículos comerciales. La principal motivación tras el desarrollo de este tipo de sistemas radica en el continuo incremento del número de vehículos en las carreteras a nivel mundial, lo cual demanda acciones para mejorar la seguridad y eficiencia en las mismas.

A lo largo de este capítulo revisaremos algunos de los avances más significativos que se han realizado en el ámbito de los ITS, los cuales engloban todas las tecnologías de información, comunicación y control aplicadas para el desarrollo de un transporte sostenible. Comenzaremos con una introducción al concepto de ITS para posteriormente comentar la situación actual desde el punto de vista de las soluciones comerciales. Además, presentamos algunos grupos y proyectos enfocados a esta línea de investigación, incluyendo al Programa AUTOPIA, donde se ha realizado la mayor parte del trabajo aquí presentado. Dada la gran diversidad de grupos y proyectos que existen en esta área, presentamos sólo aquellos cuyo trabajo guarda relación con las aportaciones de la presente tesis doctoral.

2.1. Los sistemas inteligentes de transporte

El Departamento de Transporte de EE.UU. (DOT por sus siglas en inglés), presenta los ITS como todos aquellos sistemas orientados a la *mejora de la seguridad y movilidad en el transporte a través de la integración de tecnologías avanzadas de comunicación en los vehículos y la infraestructura de transporte*[1]. La sociedad de ITS del IEEE (en inglés, Instituto de Ingenieros Eléctricos y Electrónicos) da una definición más amplia, presentándolos como *aquellos que emplean tecnologías sinérgicas y conceptos de ingeniería de sistemas para desarrollar y mejorar el transporte de todo tipo.*[2]. Por su parte, la organización europea de ITS ERTICO los introduce como *la integración de tecnologías de comunicación e información*

[1] http://www.its.dot.gov/
[2] http://sites.ieee.org/itss/

con la infraestructura de transporte, los vehículos y los usuarios; con el objetivo de sacar un mayor provecho a las redes de transporte, mejorando la seguridad y reduciendo el impacto en el medio ambiente.[3].

En concreto, los ITS se perfilan como la combinación de tecnología punta para el desarrollo de sistemas avanzados que, al ser integrados en los vehículos e infraestructura, mejoran la sostenibilidad del transporte. Si bien su aplicación final se extiende al transporte en general —i.e. aéreo, marítimo y terrestre—, nos centraremos sólo en su análisis desde el punto de vista del transporte por carretera, ya que es el ámbito dentro del cual se desarrolla este trabajo.

De acuerdo a (Shladover, 2009), la aplicación de los ITS al transporte por carretera responde a tres grandes necesidades actuales:

1. Mejorar la eficiencia en el uso de la infraestructura de transporte —i.e. carreteras—, reduciendo la congestión del trafico.

2. Mejorar la seguridad, a fin de reducir la frecuencia y severidad de los accidentes de trafico.

3. Reducir el consumo de energía y las emisiones de carbono asociadas al desplazamiento de los vehículos.

Aprovechando las ventajas de los ITS en la automatización de vehículos, sería posible por ejemplo emplear controladores automáticos que sean capaces de reducir la distancia de separación entre vehículos en las carreteras, aumentando la capacidad de estas últimas sin necesidad de modificar su configuración actual. Así mismo, se reduciría/eliminaría el factor humano en la conducción, responsable de cerca del 90 % de los accidentes ocurridos en España, de acuerdo a cifras de la Dirección General de Tráfico (DGT) (Soria, 2001). En cuanto a la eficiencia energética, se podría regular la velocidad de los vehículos automáticamente, buscando siempre la mayor eficiencia con bajas emisiones. Incluso, al tener una separación menor entre los vehículos se conseguiría reducir la carga aerodinámica de los mismos y, por lo tanto, su consumo (Shladover, 2009).

No obstante, no hace falta esperar la entera automatización del vehículo y la infraestructura para observar los beneficios que trae consigo la aplicación de los ITS al transporte por carretera. Hoy en día ya prácticamente todos los fabricantes de vehículos ponen a disposición del usuario distintos sistemas inteligentes (ADAS) que mejoran y facilitan significativamente la labor de conducción. En la siguiente sección presentamos algunas de las soluciones más recientes presentes en el mercado.

2.2. Soluciones comerciales y situación actual

El reciente y continuo desarrollo de sistemas inteligentes de ayuda a la conducción ha sido un gran avance en la industria automovilística, principalmente desde el punto de vista

[3]http://www.ertico.com/

de la seguridad aunque también desde la eficiencia en el consumo de los vehículos. Según sea su forma de funcionamiento, estos sistemas suelen clasificarse como pasivos o activos. Los sistemas pasivos se reservan a la emisión de mensajes y/o señales, visuales o acústicas, que notifican al conductor sobre ciertas circunstancias o situaciones especiales, como puede ser la presencia de otros vehículos o el consumo actual del coche. Por su parte, los sistemas activos van un paso mas allá y, además de advertir al conductor, son capaces de tomar el control del vehículo parcial o totalmente, evitando incluso algunas situaciones de riesgo.

Como ejemplos actuales de sistemas pasivos podemos encontrar:

- El control dinámico de la iluminación externa, pensado para regular automáticamente el alcance de las luces en función de ciertos factores como la velocidad, aceleración y carga del vehículo. En algunos fabricantes, el sistema se integra con la detección de vehículos, regulando el alcance y la intensidad de la iluminación en función de la situación del trafico.

- Las recomendaciones de cambio de marcha, donde se evalúa continuamente el rendimiento del motor y se notifica al conductor la marcha más adecuada para reducir el consumo de combustible.

- Los sistemas de reconocimiento de señales de tráfico, los cuales detectan las señales más próximas durante la circulación, presentando la información de las mismas a través de una pantalla ubicada en el tablero del vehículo. Estos sistemas son útiles para sortear las distracciones del conductor y recordarle, por ejemplo, el límite de velocidad de la zona por donde circula.

- Los sistemas de visión nocturna, pensados para mejorar el rango de visión del conductor durante la noche.

- Los sensores de proximidad traseros/delanteros y las cámaras de visión traseras, que incrementan el campo de percepción del conductor al dar marcha atrás o aparcar el vehículo.

- Los asistentes de cambio de carril, pensados para advertir al conductor de la presencia de otros vehículos en su punto muerto de visión.

La figura 2.1 muestra imágenes de los sistemas pasivos mencionados. En cuanto a las soluciones activas encontramos:

- El control de crucero, pensado para mantener la velocidad del vehículo en un valor fijado por el conductor.

- El control de crucero adaptativo (ACC - *adaptive cruise control*), que además de mantener la velocidad de referencia, respeta una distancia de seguridad con el vehículo que le precede.

(a) Control de iluminación externa (Audi).

(b) Recomendación de cambio de marcha (Audi).

(c) Reconocimiento de señales (Ford).

(d) Visión nocturna (Mercedes-Benz).

(e) Sensores de aparcamiento (Nissan).

(f) Asistentes de cambio de carril (Hella).

Figura 2.1: Sistemas comerciales pasivos de asistencia a la conducción.

- La advertencia de cambio de carril, que primero notifica al conductor de su salida involuntaria del carril, bien sea a través de un mensaje o una vibración en el volante y, ante una salida inminente, actúa directamente sobre el volante, ejerciendo un leve movimiento del mismo para mantener el vehículo en su carril.

- Los sistemas *start-stop*, diseñados para parar el motor del vehículo cuando este se detiene —e.g. en el semáforo de una intersección—, reiniciando la marcha cuando es necesario. El objetivo de estos sistemas es reducir el consumo de combustible y las emisiones del vehículo.

- El aparcamiento asistido, pensado para facilitar la labor de aparcar en paralelo. Dependiendo del fabricante el sistema puede ser semi-automático, donde sólo se controla el volante mientras que el conductor controla el movimiento longitudinal del vehículo; o automático, donde el sistema controla tanto el volante como el acelerador, quedando el conductor únicamente a cargo del cambio de marchas.

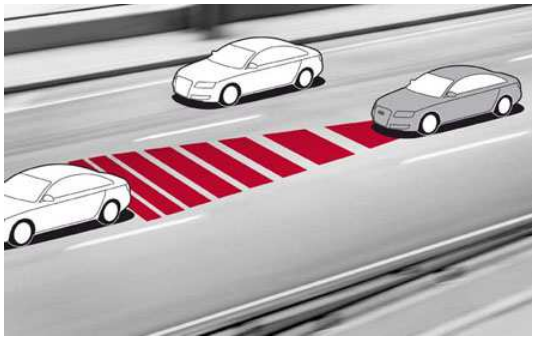
- Los sistemas anti-colisión, diseñados para minimizar los daños provocados por una colisión con el vehículo precedente o incluso evitarla totalmente. Estos sistemas suelen estar basados en sensores láser, radar o cámaras ubicadas en la parte frontal del vehículo, los cuales detectan los obstáculos en la vía del vehículo. Un ejemplo bastante conocido de este tipo de sistema es la solución *CitySafety* desarrollada por Volvo.

- El control de descenso en pendientes, encargado de mantener la velocidad del vehículo, mediante la acción sobre el freno, cuando este baja a través de una pendiente pronunciada o un terreno accidentado.

  La figura 2.2 muestra algunos de los sistemas mencionados.

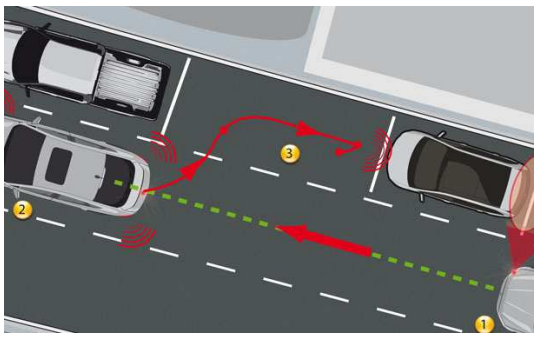
Si bien todas las soluciones mencionadas anteriormente se refieren a sistemas embarcados en el vehículo, la investigación también se ha inclinado hacia el desarrollo de sistemas cooperativos, basados en el intercambio de información entre los vehículos y la infraestructura. No obstante, aún es demasiado pronto para ver este tipo de soluciones en el mercado puesto que no existe un estándar definitivo para la comunicación entre los vehículos y la infraestructura. En este sentido, en los últimos años se ha observado un notable desarrollo de las tecnologías de comunicación para su implementación en soluciones ITS.

En el año 2003, el DOT lanzó la iniciativa de integración vehículo-infraestructura (VII - *Vehicle Infrastructure Integration*) con el objetivo de emplear canales dedicados de comunicación (DSRC - *Dedicated Short-Range Communications*) para incrementar la seguridad en las carreteras (DOT, 2010). Luego de seis años de investigación, esta fue renombrada como la iniciativa *IntelliDrive*, orientándose entonces a la integración de otras tecnologías de comunicación para estas aplicaciones. Como resultado de la iniciativa VII, la comisión federal de comunicaciones de EE.UU. reservó una banda de frecuencia en torno a los 5,9 GHz para aquellas aplicaciones relacionadas con el transporte. Así mismo, la IEEE definió los estándares IEEE 802.11p y IEEE 1609, pensados para arquitecturas de comunicación entre vehículos y la infraestructura (Uzcategui y Acosta-Marum, 2009).



(a) Sistema ACC (Audi).

(b) Advertencia de cambio de carril (Continental).



(c) Aparcamiento asistido (Ford).

(d) City-Safety (Volvo).

Figura 2.2: Sistemas comerciales activos de asistencia a la conducción.

En paralelo, la organización internacional de normalización (ISO – *International Organization for Standardization*) definió una arquitectura de comunicaciones vehículo-vehículo (V2V) y vehículo-infraestructura (V2I) conocida como CALM (*Communication Access for Land Mobile*) (ISO, 2010; Williams, 2004). Esta arquitectura está pensada para proporcionar enlaces de comunicación permanentes a través de la combinación de distintas tecnologías tales como las redes celulares 2G/3G, infrarojos y WiFi, entre otros. Una de las primeras implementaciones de esta arquitectura se realizó durante el proyecto CVIS –ver sección 2.4.1—, demostrando la viabilidad de su implementación (Toulminet et al., 2008; Ernst et al., 2009).

A fin de obtener una visión más clara de la situación actual de la investigación, presentamos en las secciones siguientes algunos de los principales grupos, centros y proyectos de investigación orientados al campo de los ITS.

2.3. Grupos y centros de investigación

A lo largo de esta sección se mencionan algunos de los grupos y centros de investigación enfocados en materia de los ITS. Comenzaremos la sección hablando de los grupos españoles, europeos e internacionales, para finalmente presentar una descripción del Programa AUTOPIA.

2.3.1. INSIA

El Instituto Universitario de Investigación del Automóvil[4] (INSIA) es un centro de investigación perteneciente a la Universidad Politécnica de Madrid (UPM), adscrito a la Escuela Técnica Superior de Ingenieros Industriales (ETSII) e integrado en el Parque Científico y Tecnológico de la UPM. Con más de 25 años de experiencia, el INSIA es un centro de referencia para el sector de la automoción, además de ser un laboratorio oficial autorizado para la homologación de diversos reglamentos y directivas. Sus actividades principales abarcan la investigación, formación y desarrollo en el ámbito automovilístico, así como estudios de seguridad e impacto en el medio ambiente.

El INSIA cuenta con una estrecha colaboración con diversos entes implicados en el sistema I+D+I —empresas, universidades y centros de investigación— entre los que destacan la DGT, la Empresa Municipal de Transportes de Madrid (EMT) y algunas empresas privadas como ALSA e IRIZAR. Durante los últimos años han participado en diversos proyectos orientados a vehículos eléctricos (OPERA4FEV[5], INNELBUS, INNVEXTRAN, EPISOL) y seguridad en accidentes (LITEBUS, FURGOSEG).

2.3.2. Robe-Safe

El grupo Robe-Safe[6], perteneciente a la Universidad de Alcalá de Henares (Madrid, España), se enfoca en los sistemas de percepción aplicados a la robótica de servicio y los sistemas *e-safety*. Está formado por más de 20 investigadores que trabajan en el desarrollo de sistemas de visión artificial para aplicaciones de percepción, localización y navegación, entre otras. Desde el año 2001 el grupo ha tenido una estrecha colaboración con el Programa AUTOPIA a través de distintos proyectos y trabajos conjuntos (Sotelo et al., 2004; Milanés et al., 2012a), de entre los que destaca el sistema de detección de peatones implementado en uno de los vehículos de AUTOPIA (Llorca et al., 2011). Actualmente ambos grupos trabajan en colaboración en el proyecto ONDA-F —*On Demand Autonomous Fleet in dedicated areas*—, cuyo objetivo principal es el desarrollo de lineas de transporte público dedicadas, basadas en la conducción autónoma.

2.3.3. LIVIC

El *Laboratoire sur les Interactions Véhicules-Infrastructure-Conducteurs*[7] es una unidad asociada al Instituto Francés de Ciencias y Tecnologías para el Transporte, Desarrollo y Redes (IFSTTAR, por sus siglas en francés). Su trabajo está orientado principalmente al estudio de la aplicación de nuevas tecnologías al desarrollo de sistemas de ayuda a la conducción. Así mismo, trabaja en colaboración con equipos de investigación de ciencias sociales a fin de analizar, desde el punto de vista del factor humano, el grado de aceptación de los sistemas

[4] http://www.insia-upm.es/
[5] http://www.opera4fev.eu/
[6] http://www.robesafe.es/
[7] http://www.inrets.fr/linstitut/unites-de-recherche-unites-de-service/livic/accueil.html

(a) (b)

Figura 2.3: Imágenes del simulador Pro-SIVIC desarrollado por el LIVIC.

desarrollados, su influencia en el comportamiento y su impacto directo en la seguridad de la carretera.

El LIVIC dispone actualmente de una amplia infraestructura de experimentación, compuesta por laboratorios de desarrollo y validación de prototipos, una flota de cuatro vehículos instrumentados y pistas de prueba privadas con una extensión cercana a los 7 km. Entre sus trabajos destacan la localización de vehículos mediante algoritmos multimodelo (Ndjeng et al., 2007), la percepción basada en visión artificial (Bak et al., 2012) y el desarrollo de un entorno de simulación para la evaluación de sistemas ADAS (Hiblot et al., 2010) —ver figura 2.3—.

Es en este grupo de investigación donde el autor realizó una estancia de tres meses de duración durante el año 2011, a partir de la cual se desarrolló el algoritmo de localización que se presenta en el capítulo 4.

2.3.4. IMARA

IMARA[8] (*Informatique, Mathématiques et Automatique pour la Route Automatisée*) es un grupo de investigación que forma parte del INRIA (*Institut National de Recherche en Informatique et Automatique*). Su objetivo es la coordinación y transferencia de toda aquella investigación desarrollada por el INRIA que pueda ser aplicada al concepto de carreteras automatizadas, introducido por el consorcio LaRA (*La Route Automatisée*). Con más de 15 años de experiencia en el área de los ITS, dirige su enfoque actual al desarrollo de aplicaciones relacionadas con el procesamiento de señales, control del vehículo, sistemas de comunicaciones y optimización de los sistemas de transporte.

Por su parte, LaRA es un consorcio coordinado creado en año 1997 por el INRIA, el INRETS (*Institut National de Recherche sur les Transports et leur Sécurité*) y el LCPC (*Laboratoire Central des Ponts et Chaussées*)[9]. Desde su creación, los miembros del consorcio han trabajado en colaboración para el desarrollo, integración y experimentación de tecnologías

[8] https://team.inria.fr/imara/

[9] Actualmente el INRETS y el LCPC se han fusionado en una única institución, llamada IFSTTAR.

de la comunicación, orientadas a mejorar la seguridad en las carreteras. Así mismo, trabajan para remover al conductor del bucle de control de los vehículos, aunque en paralelo también desarrollan sistemas de asistencia a la conducción.

Además de formar parte de este consorcio, el IMARA participa en una gran cantidad de proyectos, entre los que resaltan los ya finalizados CVIS, HaveIT y el CiberCars2.

2.3.5. AnnieWAY

AnnieWAY[10] es un equipo formado por investigadores y estudiantes de la Universidad de Karlsruhe (Alemania), el cual fue fundado en el año 2006. Su objetivo inicial fue el de participar en la competencia *DARPA Urban Challenge* —ver sección 2.4.5—, pero luego de esto ha continuado como un grupo de investigación enfocado a la conducción autónoma de vehículos. En el año 2011 participaron, al igual que AUTOPIA, en la competición GCDC —ver sección 2.4.6—, resultando ganadores de la misma (Geiger et al., 2011).

Su labor de investigación se centra en la percepción basada en visión artificial y la interpretación del entorno, aunque también han experimentado con técnicas de control, cooperación y procesado de información. Actualmente se encuentran desarrollando trabajos relacionados con la visión estereoscópica de alta definición, la planificación óptima de trayectorias, la percepción del entorno a partir de sensores lidar 3D y estrategias de control cooperativas para el control de vehículos en caravana.

Este grupo cuenta con dos vehículos experimentales: un Volskwagen Passat, equipado para una conducción totalmente autónoma; y un Audi Q7, que actúa como plataforma de captura de datos y como vehículo auxiliar en pruebas cooperativas. Durante el año 2012 el autor realizó una estancia de investigación en colaboración con este grupo.

2.3.6. TNO

La organización holandesa para la Investigación Científica Aplicada[11] (TNO por sus siglas en holandés) es una organización independiente que se dedica en manera general a la investigación con nuevas tecnologías. Dentro de su estructura cuenta con varios departamentos dedicados exclusivamente a la investigación de la seguridad y movilidad en la automoción, principalmente en el análisis de accidentes y el desarrollo de sistemas ADAS para la prevención de estos. Así mismo, ofrece sus servicios e instalaciones a un gran número de empresas asociadas para la experimentación y validación de los vehículos y sus sistemas incorporados.

TNO desempeñó un rol muy importante para la creación, en 2008, del consorcio que finalmente se encargaría de la organización de la competición GCDC en 2011.

2.3.7. VisLab - Universidad de Parma

El grupo VisLab se creó tras la participación de la Universidad de Parma en el proyecto EUREKA-PROMETHEUS, llevado a cabo en el período 1987-1995. A partir de esta expe-

[10] http://www.mrt.kit.edu/annieway/
[11] http://www.tno.nl

Figura 2.4: Fotografía de los 4 vehículos empleados en el experimento VIAC.

riencia, el profesor Alberto Broggi —uno de los responsables por parte de la universidad en el proyecto— lideró la construcción del prototipo ARGO, capaz de conducir de forma autónoma. Esto le valió al grupo para establecerse como otro referente de la conducción autónoma en Europa.

En el año 2010 el equipo llevó a cabo el experimento VIAC[12] (*VisLab Intercontinental Autonomous Challenge*), en el cual realizaron un recorrido de más de 15.000 km a lo largo de 9 países, desde Parma (Italia) hasta Shanghái (China). Para ello emplearon cuatro vehículos eléctricos totalmente automatizados —ver figura 2.4—, los cuales se iban alternando por pares a medida que se agotaba su batería. El experimento empleó un esquema líder-seguidor, en el que sólo uno de los vehículos funcionaba de forma autónoma, siguiendo la referencia marcada por un vehículo que le precedía, conducido manualmente.

2.3.8. Tartan Racing Team

Este grupo de investigación está formado por la universidad Carnegie Mellon en colaboración con General Motors Corporation, una de las mayores compañías de automoción del mundo. Además, cuentan con la colaboración de empresas como Cartepillar, Continental AG, Intel, Applanix, y Hewlett Packard entre otras. El equipo es reconocido por ser el ganador de la competencia *DARPA Urban Challenge* realizada en 2007 (Urmson et al., 2008). El vehículo utilizado en la competición es una Chevrolet Tahoe del año 2007 —ver figura 2.5— que fue modificada para permitir su conducción autónoma. Por razones de seguridad y normas de la competición, el vehículo mantiene los controles normales de conducción, de forma tal que un conductor humano pueda retomar el control del vehículo de forma rápida y sencilla en caso de fallas.

Actualmente el Dr. Cris Urmson —uno de los responsables del equipo durante la competencia— forma parte del equipo de investigación encargado del proyecto *Google Driverless Car*

[12]http://viac.vislab.it/

Figura 2.5: Prototipo *Boss* del Tartan Racing Team.

—ver sección 2.4.7—.

2.3.9. PATH

PATH[13] (*Partners for Advanced Transit and Highways*) es un consorcio creado en 1986 por el Instituto de Estudios de Transporte de la Universidad de California —en Berkeley— y el Departamento de Transporte de California (EE.UU.). Su misión principal es el desarrollo de tecnologías ITS que permitan incrementar la seguridad, flexibilidad, movilidad y administración de los sistemas de transporte del estado de California. Para ello, el consorcio cuenta con más de 40 investigadores y una estrecha relación con los fabricantes de vehículos y las agencias gubernamentales.

Este grupo es pionero en el desarrollo de aplicaciones ITS, lo que le ha valido para ser actualmente un referente de este tipo de tecnología a nivel mundial. Uno de sus experimentos más conocidos es la demostración de conducción autónoma realizada en agosto de 1997 (Tan et al., 1998) —ver figura 2.6—. Esta se llevó a cabo en San Diego (EE.UU.) y consistió en una maniobra conjunta de 8 vehículos viajando en formación de *platooning* —caravana de vehículos que viajan con una distancia de separación longitudinal muy corta—. Para ello incorporaron una serie de marcas magnéticas en una autopista, las cuales señalan la trayectoria a seguir por los vehículos. Para el control longitudinal emplearon radares que determinaban la distancia respecto al vehículo precedente, transmitiendo este valor junto con otros datos del vehículo a través de un sistema de radio comunicaciones. El principal objetivo de esta demostración fue validar técnicamente la capacidad de este tipo de sistemas para reducir la congestión del tráfico. Durante la demostración los vehículos viajaron con una distancia de separación de 6,5 metros. A esta distancia, la circulación de varias caravanas

[13] http://www.path.berkeley.edu

Figura 2.6: Vehículos del programa PATH durante el experimento de *platooning*.

de igual longitud separadas una distancia de 60 metros representaría una capacidad de 5700 vehículos por hora, más del doble del valor obtenido con conducción manual —2000 vehículos por hora—.

2.3.10. Programa AUTOPIA

El Programa AUTOPIA[14] es un grupo de investigación español, perteneciente al Centro de Automática y Robótica de la UPM y el Consejo Superior de Investigaciones Científicas (CSIC). Su principal objetivo es la conducción 100 % autónoma de vehículos, así como el desarrollo de aplicaciones que permitan mejorar la seguridad en la conducción, principalmente en entornos urbanos y situaciones de alto riesgo. El trabajo desarrollado durante la elaboración de la presente tesis doctoral se engloba totalmente dentro del marco de investigación de este grupo.

Desde su creación, en el año 1998, AUTOPIA ha enfocado sus estudios a la transferencia de técnicas de control de robots móviles a la conducción autónoma de vehículos, evitando en la mayor medida posible la introducción de modificaciones en la infraestructura. La mayor parte de las técnicas aplicadas al control están basadas en la inteligencia artificial, destacando el uso de la lógica borrosa por su facilidad para emular el razonamiento humano durante la conducción.

Actualmente el grupo cuenta con una flota de 5 vehículos automatizados: dos furgonetas eléctricas Citroën Berlingo, con las cuales inició el trabajo del Programa; dos Citroën C3, con motor a gasolina y un minibús eléctrico con capacidad para 14 pasajeros. Además, recientemente ha incorporado a su flota un coche Citroën DS3 Cabrio, el cual se encuentra en fase de instrumentación.

[14] http://www.car.upm-csic.es/autopia

A lo largo de su trayectoria, el grupo ha contado con financiación de distintos proyectos de investigación, entre los que destacan:

- **ORBEX**, financiado por la Comisión Interministerial de Ciencia y Tecnología (CICYT) para la creación de un núcleo de inferencia difusa denominado Ordenador Borroso Experimental —de allí el nombre del proyecto—, el cual se empleó como soporte para la implementación de los diferentes sistemas de navegación de los vehículos.
- **ZOCO**, financiado también por la CICYT y enfocado a la construcción de las pistas de prueba —conocidas como Zona de Conducción—. Estas están ubicadas en las instalaciones del CAR en Arganda del Rey, y es allí donde se llevan a cabo la mayor parte de los experimentos realizados por el grupo. Más adelante se presenta una breve descripción de la infraestructura de las mismas.
- **COVAN** y **GLOBO**, financiados por la Comunidad de Madrid y la CICYT respectivamente. Estos proyectos sirvieron para la adquisición e instrumentación de las dos furgonetas eléctricas con las que cuenta el Programa.
- **CYBERCARS-2**, financiado por la Comunidad Europea para la realización de maniobras cooperativas entre vehículos de diferente naturaleza. En el desarrollo de este proyecto participaron además otras 10 instituciones relacionadas con el sector del transporte.
- **CITYELEC**, proyecto enfocado a la implementación de un flujo de tráfico verde en ciudades españolas, en una línea de investigación solidaria con el medio ambiente.
- **GUIADE**, financiado por el Ministerio de Fomento y enfocado a la construcción de un sistema de guiado automático para vehículos de transporte público. El objetivo final de este proyecto fue la optimización de la eficiencia del sistema en términos de consumo energético, impacto ambiental, seguridad y calidad de los servicios de transporte público. La financiación otorgada con este proyecto permitió la adquisición el minibús eléctrico.
- **TRANSITO**, proyecto del plan nacional orientado a la coordinación local de un conjunto de vehículos. En esta misma línea de investigación se realiza actualmente el proyecto **ONDA-F**, en colaboración con la Universidad de Alcalá de Henares a través del grupo Robe-Safe. El objetivo de este último proyecto es extender la coordinación de los vehículos a todo el recinto de aplicación, organizando los trayectos de los vehículos y garantizando la seguridad de los desplazamientos.

Para la experimentación y validación de las aplicaciones desarrolladas por el Programa, el CAR dispone de una pista de prueba que simula un entorno urbano —ver figura 2.7—. Esta fue construida durante la ejecución del proyecto ZOCO, mencionado anteriormente. En principio la pista se diseñó con una forma reticular, simulando las manzanas de una ciudad, pero durante los años 2009 y 2010 se realizó la incorporación de elementos adicionales: una

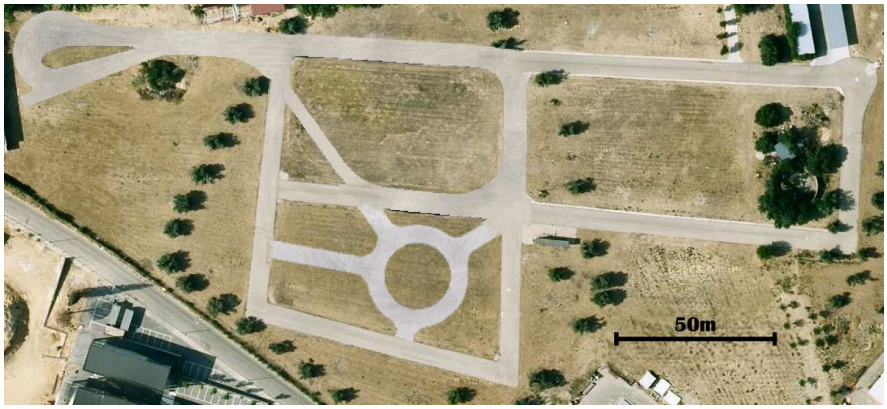


Figura 2.7: Pista de prueba del CAR.

rotonda en uno de los cuadrantes, semáforos en el cruce central y curvas adicionales en la recta principal. Con ello se consiguió un entorno más urbano, con un conjunto de curvas más amplio. Con la salvedad de un pequeño segmento diagonal, todas las calles tienen una anchura mínima de 6 metros, permitiendo la circulación en ambos sentidos.

2.4. Proyectos de investigación

Siguiendo la línea de descripción adoptada para los grupos y centros de investigación, continuamos con la introducción de algunos de los principales proyectos relacionados con los ITS.

2.4.1. CIVIS

CVIS[15] (*Cooperative Vehicle Infrastructure Systems*) fue un proyecto del sexto programa marco (FP6), ejecutado entre Febrero de 2006 y Enero de 2010. El objetivo principal de este proyecto consistía en el desarrollo de una solución unificada que permitiese la comunicación bidireccional, continua y transparente entre los vehículos y elementos de la infraestructura. Para ello planteó la creación de módulos de comunicaciones estándar, capaces de enlazar continuamente los vehículos con las unidades y equipos de la carretera a través de una combinación de distintas tecnologías, dentro de las que se incluían las redes de comunicación móvil, WiFi y DSRC entre otras. La idea tras este desarrollo era conseguir una sistema *middleware* que permitiese la comunicación V2V y V2I de forma transparente, de cara a los distintos elementos que interactúan entre sí. Por otro lado, se buscaba desarrollar un estándar abierto que garantizara la posibilidad de conexión independientemente de la localización o el proveedor del servicio.

Además de demostrar la capacidad de los sistemas basados en cooperación para mejorar la seguridad y eficiencia en las carreteras, el desarrollo de este proyecto contribuyó a la validación de la arquitectura de comunicaciones CALM, planteada por la ISO, y el protocolo

[15]http://www.cvisproject.org/ y https://team.inria.fr/imara/projet/cvis/

IEEE 802.11p. Parte de los avances alcanzados durante este proyecto sirvieron como base al consorcio encargado del GCDC para el desarrollo del protocolo de comunicaciones empleado durante esta competición (de Jongh, 2011).

2.4.2. SAFESPOT

SAFESPOT fue un proyecto FP6 orientado al estudio de la cooperación entre vehículos e infraestructura como medida para incrementar la seguridad en las carreteras. Para ello se enfocó en el desarrollo de un sistema denominado Asistente de Seguridad Marginal, el cual detecta de forma anticipada posibles situaciones de riesgo, incrementando el margen de tiempo disponible para tratar de evitarlo. La base de este sistema es un conjunto de mapas dinámicos digitales del entorno local, generados a partir de la información obtenida a través de enlaces de comunicaciones V2V y V2I. Algunas de las pruebas de validación realizadas durante este proyecto consistieron en cambios de carril, advertencia de colisiones frontales, reducción anticipada de la velocidad e información dinámica del estado de la carretera (Toulminet et al., 2008).

2.4.3. COOPERS

Al igual que CVIS y SAFESPOT, COOPERS[16] (*CO-OPerative SystEms for Intelligent Road Safety*) fue un proyecto englobado en el sexto programa marco, ejecutado entre 2006 y 2010 y orientado al desarrollo de aplicaciones telemáticas para la infraestructura del transporte por carretera (Toulminet et al., 2008). Su objetivo principal era salvar la brecha entre el desarrollo de sistemas en la industria automovilística y los operadores de infraestructura. Para ello se planteó una arquitectura de gestión de tráfico basada en enlaces continuos de comunicaciones V2I, similar a la propuesta por el proyecto CVIS, la cual permitiría el intercambio de información relevante para segmentos específicos de la carreteras.

Durante el desarrollo del proyecto se estudiaron las prestaciones de diversas tecnologías de la comunicación, entre las que destacan: DAB, DVB-H, GSM / GPRS, CALM-IR, CALM-M5 y WiMAX. Su arquitectura para los vehículos consiste en unidades de comunicación embarcadas, las cuales actúan como puertas de enlace entre las distintas plataformas de comunicaciones y el ordenador a bordo. De forma similar, la infraestructura emplea unidades de comunicación de corto alcance, combinadas con distintos centros de control y una plataforma de gestión de sensores.

2.4.4. Car2Car

El consorcio Car2Car[17] es una organización sin fines de lucro creada por diversos fabricantes de vehículos europeos y apoyado por varios proveedores de equipos, centros de investigación y terceros relacionados con la industria automovilística. El principal objetivo de este consorcio es la creación de estándares que definan las interfaces y protocolos empleados

[16] http://www.coopers-ip.eu/
[17] http://www.car-to-car.org

para la comunicación entre los vehículos y su entorno, a fin de garantizar la interoperabilidad entre los equipos de distintos fabricantes (C2C-CC, 2007). La idea es que toda esta tecnología se pueda implementar a corto plazo para el desarrollo de distintas aplicaciones cooperativas orientadas a mejorar la seguridad y eficiencia en la carretera. El trabajo de este consorcio se realiza bajo una estrecha cooperación con varias organizaciones de normalización, europeas e internacionales.

Desde mediados del año 2011 el Programa AUTOPIA es miembro de desarrollo de este consorcio, asociado a los grupos de trabajo de simulación y demostración.

2.4.5. DARPA Challenges

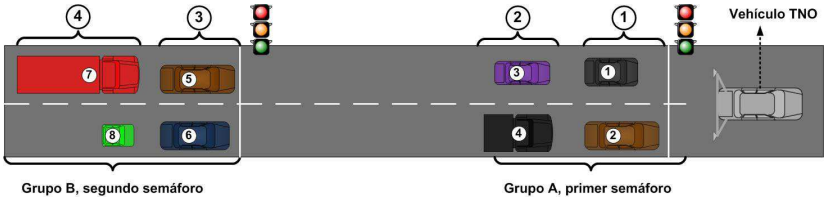
DARPA Challenges no es propiamente un proyecto de investigación sino una serie de competiciones de vehículos autónomos que han sido organizados por la agencia DARPA (*Defense Advanced Research Projects Agency*) del Departamento de Defensa de EE.UU. En cada competición intervienen vehículos terrestres automatizados, los cuales deben maniobrar en un determinado entorno sin ningún tipo de intervención humana. La primera competición, denominada *DARPA Grand Challenge*, se llevó a cabo en el año 2004 a través del desierto de Mojave (EE.UU.). En esta edición participaron más de 100 equipos internacionales, aunque ninguno de los vehículos logró completar el recorrido de 240 km. En el 2005, se realizó la segunda edición de la competición bajo el mismo nombre, resultando ganador el equipo de la Universidad de Stanford con su vehículo *Stanley* (Thrun, S. et al, 2006), el cual completó el recorrido de 212 km en 6 horas y 54 minutos.

Tras el éxito de esta competición, se realizó en 2007 la primera y única edición del *DARPA Urban Challenge*, la cual consistió demostrar las capacidad de conducción autónoma de los participantes en entornos urbanos. Al igual que en las ediciones del *DARPA Grand Challenge*, los vehículos debían ser capaces de completar el recorrido sin ningún tipo de intervención humana. En esta oportunidad resultó ganador el equipo Tartan Racing Team de la Universidad Carnegie Mellon —sección 2.3.8—, cuyo vehículo *Boss* completó el recorrido de 96 kilómetros en 4 horas y 10 minutos.

2.4.6. GCDC

En Mayo de 2011 se llevó a cabo en Helmond (Holanda) la primera competición europea de vehículos autónomos, el *Grand Cooperative Driving Challenge*, conocida como GCDC. La idea tras esta competición fue validar el funcionamiento de los sistemas desarrollados por distintos grupos de investigación en un entorno cooperativo. En concreto, los equipos debían demostrar su capacidad de control en una maniobra de control de crucero adaptativo con cooperación (CACC). El intercambio de información entre los vehículos se realizó a través de una red inalámbrica, desarrollada a partir de la especificaciones de la arquitectura CALM y el estándar IEEE 802.11p. Durante esta edición sólo se contempló el control longitudinal de los vehículos, dejando el control lateral a cargo del conductor del mismo.

El GCDC se realizó en la autovía holandesa A270, entre las poblaciones de Helmond y



(a) Formación de salida para las pruebas del GCDC.

(b) Vehículos preparándose para la formación de salida.

Figura 2.8: Imágenes del GCDC.

Eindhoven, la cual se cerró totalmente al tráfico durante el evento. A pesar de que inicialmente se postularon 11 equipos, sólo 9 de ellos superaron los requisitos mínimos establecidos por la organización, estando el Programa AUTOPIA entre estos participantes. Durante la competición se realizaron varias pruebas con la formación de pelotón mostrada en la figura 2.8. En cada prueba, los vehículos del pelotón B debían aproximarse al pelotón A tan pronto como su semáforo les permitiera el paso. Acto seguido, la totalidad de los vehículos debía esperar la activación del siguiente semáforo para iniciar el seguimiento del vehículo líder, perteneciente a la organización. Durante el recorrido —de aproximadamente 6 km— la velocidad del vehículo líder variaba de forma aleatoria, a fin de verificar el efecto de tal modificación sobre el comportamiento del resto de los vehículos.

Como criterio de evaluación la organización tuvo en cuenta el comportamiento individual y cooperativo de cada equipo. Finalmente, luego de dos días de pruebas, el equipo AnnieWAY se hizo con el primer lugar de la competición, con un total de 17 puntos. A pesar de las dificultades enfrentadas por el equipo (Pérez, 2012), AUTOPIA logró alcanzar la quinta posición en la tabla de clasificación, con un total de 11 puntos.

2.4.7. Google Driverless Car

Google Driverless Car es un proyecto del gigante informático Google, enfocado al desarrollo de vehículos sin conductor. En sus inicios el proyecto fue liderado por el ingeniero Sebastian Thrun, quién fuera responsable del equipo de la Universidad de Stanford durante el la edición del *DARPA Grand Challenge* en 2005. Actualmente está bajo la dirección técnica del Dr. Chris Urmson, quien formó parte del equipo ganador del *DARPA Urban Challenge*.

Su arquitectura de control se basa en la combinación de cámaras de vídeo y sensores radar

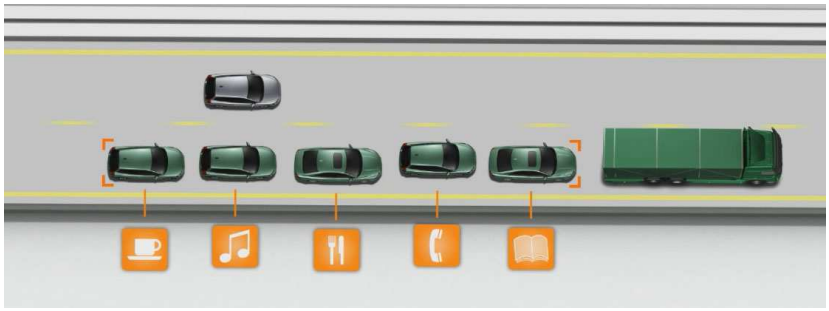


Figura 2.9: Esquema del proyecto SARTRE.

y lidar para la percepción de otros vehículos, peatones y elementos de la infraestructura. Además, combina la información de mapas digitales para la navegación de los vehículos, capturados con la tecnología similar a la empleada por la herramienta *Google Maps*. Toda la información capturada por los vehículos es procesada con ayuda de los servidores centrales de la empresa.

Actualmente el equipo cuenta con una flota de 10 vehículos, formada por seis Toyota Prius, un Audi TT y tres Lexus RX45h. Cada coche incorpora diversos sistemas de seguridad que permiten que un conductor —siempre presente durante las pruebas realizadas— retome el control del vehículo fácilmente en caso de fallos. Hasta la fecha Google indica que ha completado más de medio millon de kilómetros de conducción autónoma, circulando incluso por entornos con una alta densisdad de tráfico; como es el caso del puente *Golden Gate* en San Francisco (EE.UU).

2.4.8. SARTRE

El proyecto SARTRE[18] (*Safe Road Trains for the Environment*) es un proyecto englobado en el séptimo programa marco de la Comisión Europea (FP7) cuya ejecución ha finalizado en el año 2012. El objetivo de este proyecto fue el desarrollo y evaluación de tecnologías y estrategias para la operación de *platoonings* en autopistas convencionales, enfocado a los beneficios respecto a la seguridad, eficiencia y confort (Bergenhem et al., 2010). El proyecto fue liderado por Ricardo UK Ltd, y contaba con la participación de Robotiker-Tecnalia de España, el fabricante Volvo e institutos de investigación de Suecia y Alemania.

Una de las ideas tras este proyecto fue promover el cambio en el uso del transporte personal, desarrollando un esquema de control de *platoonings* que no requiere la modificación de la infraestructura. Su esquema de funcionamiento —ver figura 2.9— se basa en vehículos controlados por conductores profesionales, cuyos conductores son responsables (líderes) del recorrido realizado por los vehículos en el pelotón. Así, los vehículos seguidores entran en un modo de conducción semi-autónomo, lo que le permite a los conductores de los mismos concentrarse en otras tareas con seguridad —e.g. hablar por el móvil o leer un libro—.

[18] http://www.sartre-project.eu

CAPÍTULO 3

ARQUITECTURA DE CONTROL

A simple vista, dotar a un vehículo con la capacidad de conducirse de forma autónoma puede parecer un proceso bastante complejo. Sin embargo, dicho proceso es equiparable a otros problemas de automatización que hoy en día ya han sido resueltos gracias a la tecnología disponible. En este sentido, el programa AUTOPIA ha avanzado en el campo de la conducción autónoma a través de la transferencia de técnicas de control de robots móviles al control de vehículos comerciales.

El punto inicial de esta investigación lo constituye el trabajo desarrollado en (Naranjo, 2005), donde se presentó el primer enfoque de una arquitectura embarcada para el control de vehículos. Este sistema fue implantado en dos furgonetas eléctricas Citroën Berlingo con el objetivo de validar el funcionamiento y la aplicabilidad del mismo a través de una serie de experimentos de control lateral y longitudinal. Posteriormente, (Milanés, 2010) presentó una evolución del sistema, introduciendo una serie de mejoras entre las que se destacan (i) la incorporación de sensores inerciales e información proveniente de los sensores propios del vehículo; (ii) la inclusión de un sistema de frenado electro-hidráulico y (iii) la automatización de un vehículo Citroën C3 Pluriel.

Es objeto de la presente tesis continuar con esta línea de evolución, haciendo especial hincapié en la optimización de recursos computacionales a fin de incrementar la eficiencia del sistema. Para ello, presentamos un nuevo enfoque de la arquitectura a nivel de software, evitando la centralización en un programa único de control y abocando a la implementación de distintos módulos de software que trabajan en paralelo y de forma conjunta.

El desarrollo de este capítulo se va a estructurar de la siguiente manera. En primer lugar, se presenta la arquitectura de control de AUTOPIA tal y como se encontraba estructurada al comienzo del desarrollo de esta tesis doctoral. Seguidamente se detalla la instrumentación de hardware realizada a un vehículo comercial a fin de utilizarlo como plataforma de prueba para el nuevo esquema desarrollado. Finalmente, se explica la nueva estructura de software del sistema, resaltando las principales aportaciones realizadas por este trabajo.

3.1. Esquema de control del grupo AUTOPIA

El patrón de circulación que siguen los conductores, en general, se puede definir de la siguiente forma: guiar al vehículo a un destino determinado a través de la red de carreteras disponible, respetando todas las normas que rigen el entorno y evitando colisiones con el resto de entes presentes en el mismo —i.e. peatones y otros vehículos—. A tal fin, el conductor recibe continuamente información de primera mano acerca de su entorno de conducción —e.g. posición relativa de otros entes, señales de tránsito y condiciones meteorológicas— y, en base a esta información, planifica y ejecuta las acciones que considere pertinentes. Estas acciones están orientadas en su mayoría al control de los tres actuadores principales del vehículo: acelerador, freno y volante.

Tomando como base este esquema jerárquico de percepción - decisión - actuación, el Programa AUTOPÍA ha desarrollado una arquitectura de control que emula el comportamiento humano. A nivel general, este sistema se descompone en tres etapas principales que se describen a continuación:

La etapa de **Percepción** engloba todos los sensores embarcados en el vehículo, los cuales proporcionan información tanto del estado del mismo como del entorno que lo rodea. La entrada principal de esta etapa proviene de un sistema de posicionamiento global diferencial (DGPS por sus siglas en inglés) que, combinado con una unidad de medición inercial, otorga información precisa sobre la posición y orientación del vehículo.

La etapa de **Planificación** con tres tareas principales en el siguiente orden: (i) selección de la ruta óptima para alcanzar el destino indicado desde la posición actual; (ii) selección del modo de conducción más adecuado para la ejecución del recorrido dada las condiciones actuales del entorno —e.g. atascos, intersecciones, etc.— y (iii) determinación de las acciones correspondientes que se deben aplicar sobre cada uno de los actuadores del vehículo.

Finalmente, la etapa de **Actuación** se encarga de la ejecución de las órdenes enviadas desde la planificación a cada uno de los actuadores del vehículo. De cara a la planificación, el funcionamiento de esta última etapa es 100 % transparente, encargándose esta de la transducción de los valores recibidos dentro de un rango normalizado a las señales analógicas y/o digitales correspondientes a cada uno de los actuadores. Esto es necesario ya que, dependiendo de la tecnología utilizada, la automatización de los actuadores puede variar de un vehículo a otro.

Como demuestran los resultados de distintas pruebas y experimentos realizados previamente (Milanés et al., 2008; Pérez et al., 2010), la arquitectura de control presentada ha resultado adecuada para su aplicación en el guiado automático de vehículos. No obstante, esta presenta limitaciones en cuanto a la incorporación o modificación de sus elementos, dificultando la evolución de la misma. Por este motivo, hemos planteado como objetivo inicial de

Figura 3.1: Platero y su instrumentación.

la presente tesis el realizar la reestructuración de la arquitectura de control empleada por AUTOPIA, haciendo especial hincapié en la generalización del acceso a los datos proporcionados por cada uno de sus elementos. En este sentido, se emplearán técnicas análogas a aquellas que permiten a los Sistemas Operativos conceder acceso a sus periféricos de forma uniforme. En las secciones siguientes presentamos la descripción detallada de la nueva arquitectura propuesta, así como las aportaciones realizadas por esta tesis doctoral a dicha arquitectura.

3.2. Platero, el coche de prueba

Antes de introducir el nuevo enfoque de la arquitectura de control, presentamos en esta sección la descripción de la instrumentación realizada a un vehículo comercial con el fin de habilitarlo como plataforma de desarrollo y prueba para el sistema propuesto. El coche utilizado es un Citroën C3 convencional con motor a gasolina (ver Figura 3.1) bautizado como *Platero* y al cual se le han incorporado una serie de elementos de hardware que permiten su automatización.

Al igual que en las instrumentaciones realizadas previamente por AUTOPIA (Naranjo, 2005; Milanés, 2010), la base del sistema de automatización de este vehículo es un ordenador industrial que se ha instalado en el maletero del mismo. En este ordenador es donde se aloja la mayor parte del software desarrollado y desde donde se controlan cada uno de los actuadores instrumentados. Concretamente, se utiliza un PC modelo AEC6915 de la compañía AAENON. Este ordenador tiene la característica de que ha sido diseñado para aplicaciones en entornos con vibraciones, por lo que carece casi totalmente de partes móviles en su interior más allá del disco duro, el cual se encuentra instalado sobre una plataforma especial que

absorbe parte de las vibraciones. Para comodidad del usuario, el PC se controla a través de una pantalla táctil instalada en el parabrisas frente al asiento del copiloto y un teclado/ratón inalámbrico.

Las sub-secciones siguientes describen el resto de los elementos de hardware incorporados de acuerdo a su función principal: percepción o actuación.

3.2.1. Hardware embarcado para percepción

Desde el punto de vista de la percepción, se han embarcado tres dispositivos claves para el sistema: un receptor DGPS, una unidad de medición inercial y un adaptador CAN-USB; este último para capturar los datos procedentes del bus CAN (*Controller Area Network*) del vehículo. Como receptor DGPS se ha seleccionado una unidad Trimble BD960. Este equipo, comparado con los utilizados anteriormente, presenta dos características que resultaron fundamentales al momento de su elección. En primer lugar, tiene la capacidad de trabajar a una frecuencia de hasta 10 Hz; mientras que los equipos anteriores sólo alcanzaban los 5 Hz. En segundo lugar, utiliza una interfaz Ethernet de 100 Mbps para la transmisión de datos, lo cual reduce considerablemente el tiempo de transmisión de los mensajes comparado con el puerto RS-232, utilizado en las implementaciones previas.

En cuanto a la IMU instalada, seleccionamos el modelo VG440 fabricado por Crossbow. La elección de este equipo se realizó en función de los resultados obtenidos por (Milanés, 2010) utilizando un producto de la misma familia del fabricante. Esta unidad tiene una frecuencia de muestreo de hasta 100 Hz y posee un puerto serie para la transmisión de los distintos mensajes. No obstante, comparada con la unidad utilizada previamente este modelo posee un rango de lectura dos veces mayor (4 g en aceleración y 200 ^{o}/s en velocidad de rotación) y una velocidad de transferencia de datos hasta tres veces mayor (115200 baudios). Además, el modelo VG440 tiene la capacidad de emplear una señal de reloj externa, lo cual sumado a un enlace de conexión directa RS-232 entre el receptor DGPS y la IMU, ha permitido sincronizar el tiempo de muestreo utilizando el reloj del DGPS. Para ello se ha utilizado la señal de 1PPS (un pulso por segundo) proporcionada por este último.

Por último, el adaptador CAN-USB actúa como interfaz entre el ordenador de control y la red CAN del vehículo. A través de este enlace es posible acceder a las lecturas de distintos sensores embarcados de serie por el fabricante. Por ejemplo, se puede conocer directamente la velocidad de cada una de las ruedas, la posición del volante o el estado de activación del sistema ABS. El adaptador utilizado se corresponde con el modelo Lawicel CANUSB.

3.2.2. Hardware embarcado para actuación

A fin de permitir el control de cada uno de los actuadores del vehículo, ha sido necesario incorporar una serie de elementos de hardware específicos para cada uno de ellos. Se ha utilizado una tarjeta analógica para el control del acelerador, un sistema electro-hidráulico y un módulo E/S analógico/dígital para el freno y un controlador de motores CC para el control de la dirección. Adicionalmente, se emplea un sistema de relés e interruptores para

conmutar la operación de los actuadores entre los modos manual y automático.

El control del acelerador se realiza emulando el comportamiento del pedal electrónico del vehículo a través de la tarjeta analógica instalada (Advantech PCI-1723). De acuerdo a las especificaciones del fabricante, bajo condiciones normales de operación el pedal emite dos señales analógicas de entre 0 y 5 voltios, proporcionales a la presión ejercida por el conductor. Estas señales son interpretadas por el calculador central del coche, que se encarga finalmente de gestionar el control del motor de acuerdo a los valores recibidos.

Para el control del freno, se ha incorporado un sistema hidráulico en paralelo al sistema original del vehículo. El sistema añadido cuenta con una bomba electro-hidráulica cuya presión de salida es regulada a través de dos válvulas electrónicas. El control de cada una de estas válvulas se realiza mediante un módulo E/S analógico/digital que es controlado a su vez por el ordenador embarcado a través de una interfaz CAN, la cual no debe confundirse con el bus CAN del vehículo.

Por último, para el control del volante se modificó el sistema de dirección eléctrica asistida del vehículo. La base de este sistema es un motor eléctrico acoplado directamente a la barra de dirección. Bajo condiciones normales de operación, un calculador mide el par aplicado sobre el volante y utiliza el motor eléctrico para contribuir con el movimiento deseado, reduciendo así el esfuerzo del conductor. Como medida de seguridad, el calculador incorpora diversas reglas de verificación que desconectan automáticamente el sistema de ayuda en caso de que se detecte una anomalía, lo que dificultó la intervención directa del sistema. Por ello, esta se realizó colocando en paralelo al calculador original un controlador modelo MDL-BDC24 fabricado por Texas Instruments. De esta forma, al activar el modo de conducción autónoma las conexiones del motor son desconectadas del calculador de la dirección y conectadas al controlador añadido, permitiendo entonces al ordenador controlar la posición del volante a voluntad. El sistema de control implementado sigue el esquema de controladores en cascada presentado en (Pérez et al., 2011a).

Un esquema general de los elementos de hardware añadidos al vehículo se muestra en la figura 3.2. Los elementos de percepción y actuación son resaltados en color amarillo y azul respectivamente.

3.3. Diseño de la arquitectura

Uno de los principales aspectos que se deben tener en cuenta a la hora de diseñar, desarrollar e implementar un sistema es la modularidad del mismo (Parnas, 1972). Dentro del ámbito de la robótica es común encontrar una gran variedad de sistemas complejos que son implementados como una combinación de distintos componentes más sencillos (Montemerlo et al., 2003; Pérez et al., 2009). Esta descomposición trae consigo diversos beneficios que resultan claves para el desarrollo del sistema:

- Permite que distintos desarrolladores puedan trabajar de forma simultánea en diferentes secciones del mismo, sin que esto conlleve una interferencia entre ellos.

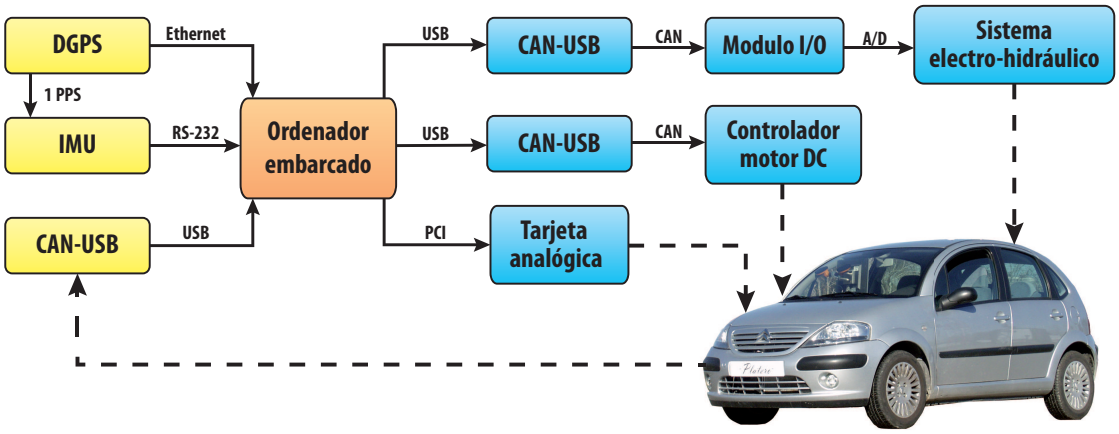


Figura 3.2: Esquema general del hardware instalado en Platero.

- En caso de que ocurriese algún fallo del sistema, la identificación del problema resultaría más sencilla, descartando rápidamente aquellos módulos cuyo funcionamiento es adecuado.

- Gracias al nivel de independencia que existe entre los distintos componentes es posible evitar, en algunos casos concretos, que la falla de uno de ellos provoque el colapso general.

- Se facilita la incorporación e intercambio de componentes, aumentando la escalabilidad del sistema.

Hasta el momento de la realización de esta tesis doctoral, la arquitectura de control desarrollada por AUTOPIA presentaba un nivel bajo de modularidad desde el punto de vista de su implementación en software. Esto se debe a que, a pesar de que se definen claramente tres etapas con sus respectivos componentes y funciones específicas, toda la gestión del sistema se realiza a través de un software único de control. En el caso de un sistema estable, estructurado adecuadamente, esta condición no supondría un problema mayor. Sin embargo, dada la naturaleza de prototipo de la arquitectura, nos encontramos ante un sistema cuya estructura sufre de continuas modificaciones y donde los fallos, presentes en todo proceso de desarrollo, son más comunes. Por otro lado, debido a la interrelación que existe entre los distintos módulos a nivel de programación, es posible que modificaciones realizadas sobre uno de ellos afecten de forma directa el rendimiento de otros componentes.

Partiendo de esta idea, planteamos como primera aportación del presente trabajo realizar la reestructuración general del sistema de AUTOPIA, fragmentando el software único de control en varios módulos. Con ello se busca independizar, en cierta medida, el funcionamiento de los elementos; facilitando finalmente el proceso de desarrollo del sistema, es decir la incorporación, modificación o supresión de funcionalidades. Los módulos de software si bien se ejecutarán de forma independiente, trabajarán en conjunto.

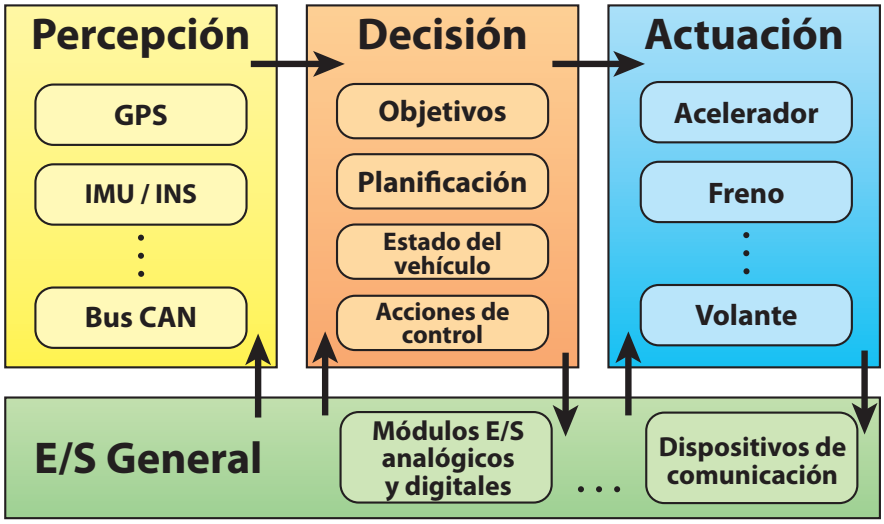


Figura 3.3: Esquema general la arquitectura de control.

3.3.1. Requisitos

Como primer paso para el desarrollo de esta arquitectura, debemos definir algunos requisitos mínimos que esta debe cumplir. Así pues tendremos en cuenta las siguientes consideraciones a lo largo de todo el proceso:

1. A cada componente de hardware se le asociará un único elemento de software, encargado de gestionar el control del mismo y de actuar como enlace entre este y los demás elementos de la arquitectura.

2. Debe existir un mecanismo de comunicación transparente entre los distintos procesos de software, permitiendo que la información generada por un componente esté disponible por igual para el resto de elementos que de ella requieran.

3.3.2. Esquema de la arquitectura de control

Una vez presentadas las limitaciones de la arquitectura previa, y considerando los requisitos mencionados en la sección anterior, se presenta a continuación un esquema general de la arquitectura propuesta. Como se puede apreciar en el diagrama de la figura 3.3, hemos dividido la arquitectura en 4 etapas:

1. **Percepción**, donde se agrupan todos los elementos encargados de la percepción del estado del vehículo y del entorno que lo rodea. A modo de ejemplo se muestran en la figura tres posibles componentes de esta etapa como lo son los procesos de lectura y procesado de la información proveniente del GPS, la unidad inercial y el bus CAN del coche.

2. **Decisión**, que por un lado, planifica la trayectoria del vehículo para alcanzar todos los objetivos establecidos y, por otro lado, selecciona las acciones de control en función de la información recibida de la etapa de percepción.

3. **Actuación**, que se encarga de ejecutar todas las acciones de control sobre cada uno de los distintos actuadores del vehículo.

4. **E/S General**, donde se agrupan todos los componentes de bajo nivel que pueden ser utilizados por los módulos de otras etapas indistintamente. Por ejemplo, los diferentes dispositivos de entradas y/o salidas analógicas y digitales que se pueden utilizar para percibir el estado de una variable, o bien para activar las válvulas del sistema de frenado.

Limitándonos a la estructura de la arquitectura, la principal diferencia que se puede observar respecto a la versión anterior es la presencia de la etapa E/S General, la cual juega un papel fundamental para satisfacer el requisito de independizar el control de los elementos de hardware. Por otro lado, es necesario resaltar que la subdivisión realizada a modo de ejemplo en cada una de las etapas se refiere a las funciones que en ella se agrupan, y no a los módulos de software que la componen.

Con el esquema de la arquitectura delimitado, continuamos en la siguiente sección con el próximo paso: definir el mecanismo de comunicación que permitirá el intercambio de información entre los distintos módulos de software del sistema.

3.3.3. Intercambio de información

Cuando se ejecuta un programa o proceso en un ordenador, se reserva de forma estática o dinámica un espacio de memoria donde se almacenan los datos del mismo. Esto permite que los diferentes elementos del proceso puedan compartir variables y funciones de forma sencilla, bajo ciertas condiciones definidas en la programación. Sin embargo, cuando un software se descompone en varios procesos independientes el intercambio de la información no resulta tan trivial. Esto es lo que se conoce como el problema de la comunicación entre procesos (IPC por sus siglas en inglés)

Dependiendo de la plataforma de hardware, el sistema operativo y el lenguaje de programación, existen diversas soluciones al problema de IPC. Algunas de las que se encuentran disponibles en sistemas basados en UNIX son:

- Memoria compartida y mapas de memoria: permiten la comunicación entre los procesos utilizando un bloque específico de memoria al cual todos ellos pueden acceder para escribir y/o leer información.

- Tuberías (*pipes*) y colas de mensajes: permiten el intercambio de información de forma secuencial.

- Sockets: permiten la comunicación entre procesos aún y cuando estos se ejecuten en ordenadores distintos. Hoy en día esta es una de las técnicas más utilizadas dada la alta propagación que ha tenido Internet a nivel mundial.

La selección de la técnica más adecuada para una aplicación depende de distintos factores como el número de procesos que deben comunicarse, la relación que mantienen entre ellos, su

sincronización, las limitaciones de permisos para la lectura/escritura y el número de ordenadores que hospedan el software (Mitchell et al., 2001).

Además de estas técnicas proporcionadas por los sistemas operativos, existen algunos paquetes o interfaces de programación de aplicaciones (API por sus siglas en inglés) orientados a resolver este problema en distintas plataformas. Normalmente estas APIs son una solución más amigable para los desarrolladores y, en algunos casos, ayudan a mantener un estándar de desarrollo, independientemente del lenguaje de programación o de la plataforma utilizada. Algunas de las APIs más conocidas en el ámbito de las aplicaciones de robótica son:

***Robot Operating System* (ROS)** [1], es la plataforma intermedia de software (*middleware*) más utilizada en robótica para el desarrollo de aplicaciones modulares. La comunicación entre los distintos elementos se basa en un esquema de interconexión *peer-to-peer* (P2P); es decir, utilizando un enlace directo entre dos procesos a comunicar, aún y cuando estos se encuentren hospedados en distintos ordenadores (Quigley et al., 2009). No obstante, esta requiere de la implementación de un proceso maestro centralizado, donde se mantiene una tabla con los servicios ofrecidos por cada proceso a fin de que puedan encontrarse mutuamente.

***Carnegie Mellon Robot Navigation Toolkit* (CARMEN)** [2], es un conjunto de herramientas de código abierto pensado para el control de robots móviles. Utiliza un esquema de conexiones TCP (Protocolo de Control de Transmisión, en inglés) para resolver el problema de IPC. Un proceso central se encarga de la coordinación de los enlaces entre los módulos y del enrutamiento de los mensajes (Montemerlo et al., 2003). Cada mensaje se envía utilizando un modelo de *publish/subscribe*; es decir, los módulos que requieren cierta información no envían un mensaje de petición al módulo proveedor, sino que la reciben directamente cuando esta se encuentra disponible.

***Mission Oriented Operating Suite* (MOOS)** [3], es un plataforma de comunicaciones entre procesos basada en una topología de conexión en estrella, en la que cada uno de los módulos tiene un canal de comunicación único con un servidor central. A través de su respectivo canal, cada proceso publica información en el nodo central, el cual la almacena en una base de datos global. Al mismo tiempo, puede acceder a la información publicada por otros módulos (Newman, 2008). En este caso la implementación de la base de datos permite que se almacene no sólo el último mensaje de cada proceso, sino también un histórico de los mismos.

***Lightweight Communications and Marshalling* (LCM)** [4], es un conjunto de librerías y herramientas para la comunicación entre procesos en sistemas de tiempo real desarrollado por el equipo *MIT DARPA Grand Challenge Team*. Utiliza un modelo *publish/subscribe* basado en la difusión *multicast* de mensajes UDP (Protocolo de Data-

[1] http://www.ros.org
[2] http://carmen.sourceforge.net/
[3] http://www.robots.ox.ac.uk/ mobile/MOOS/wiki/pmwiki.php/Main/HomePage
[4] http://lcm.googlecode.com

API	Modelo de comunicación	Transporte	Centralizado	Codificación	Lenguajes soportados
ROS	Pub./Sub. +servicio	TCP, UDP	Si	Binaria	C++, Java, Python
CARMEN	Pub./Sub.	TCP	Si	Binaria + Cadena de texto	C, Java
MOOS	Pub./Sub. (con petición)	TCP	Si	Cadenas de texto/*doubles*	C++, Java
LCM	Pub./Sub.	UDP	No	Binaria	C, C++, Java, Phyton, MATLAB

Tabla 3.1: Comparación de distintas APIs para IPC.

gramas de Usuario, en inglés) a través de canales de comunicación (Huang et al., 2010), identificados con un nombre único. Con esta herramienta cada proceso publica directamente su información en determinados canales y se suscribe a aquellos que cuenten con la información que necesite para su funcionamiento, todo ello sin utilizar un nodo central o un proceso maestro.

En nuestro caso, hemos seleccionado la opción del LCM como plataforma de intercambio de datos. La elección se realizó entre las APIs mencionadas anteriormente, teniendo en cuenta las características resumidas en la tabla 3.1. Se puede apreciar que las 4 técnicas comparten el mismo modelo de comunicación *publish/subscribe*; sin embargo, el LCM se diferencia de las otras tres en cuanto a que no requiere un proceso/nodo central, lo cual supondría un punto débil en caso de fallo. En lo que respecta a la codificación de los mensajes, hemos dado preferencia a los mensajes binarios sobre las cadenas de texto; ya que estas últimas si bien facilitan el proceso de depuración a los desarrolladores, requieren de un mayor ancho de banda para su transmisión (Huang et al., 2009). Finalmente, en favor de la escalabilidad del sistema optamos por que la plataforma soportase un amplio abanico de lenguajes de programación, además de tener la capacidad de trabajar con procesos hospedados en varios ordenadores.

Además de las características listadas, el paquete LCM proporciona tres herramientas que son de gran ayuda para el desarrollo e inspección del sistema:

1. Una herramienta de registro de datos capaz de capturar todo el tráfico LCM que circula por la red.

2. Un reproductor con capacidad de retransmitir a la red, en tiempo real, los datos almacenados en un fichero de registro. Una ventaja de esta herramienta es que permite al usuario seleccionar los canales a retransmitir, siendo posible acoplar la ejecución real del sistema con la reproducción de datos almacenados.

3. Por último, una herramienta de inspección que decodifica automáticamente todo el tráfico de la red, mostrando la información de cada uno de los mensajes a través de una interfaz gráfica. Además de ello, el programa proporciona estadísticas para cada canal

respecto a la frecuencia de envío de los mensajes, el número de ellos transmitidos y el consumo de ancho de banda de los mismos.

Con la plataforma de intercambio de información definida, continuamos en las secciones siguientes con la descripción del software implementado en cada una de las etapas de la arquitectura. La descripción de las etapas se presenta en orden de dependencia; esto es percepción, e/s general, actuación y finalmente decisión.

3.4. Percepción

En la sección 3.2.1 presentamos el listado de componentes de hardware embarcados en el vehículo relacionados con el sistema de percepción del mismo. A continuación se describen los módulos de software implementados para la lectura y/o control de cada uno de estos componentes.

3.4.1. GPS

El receptor DGPS embarcado (Trimble BD960) está configurado para proporcionar información acerca del estado del vehículo con una frecuencia de trabajo de 10 Hz. La comunicación entre esta unidad y el ordenador a bordo se realiza a través de una interfaz Ethernet, la cual se ha seleccionado frente a la interfaz RS-232 debido a que esta última presenta una velocidad de transferencia de datos mucho menor. En cada ciclo de lectura, los datos de la medida son transmitidos utilizando distintos mensajes definidos dentro del protocolo NMEA-0183 (NMEA, 2002), en los que la información no está codificada sino que se transmite como cadenas de caracteres legibles. En concreto, se procesan tres tipos de mensajes: (i) GPGGA, que contiene información del posicionamiento, incluyendo la calidad de la medida y el numero de satélites utilizados; (ii) GPVTG, con información de la velocidad y orientación del vehículo; y (iii) GPGST, con las estadísticas de error para la medida recibida.

Del lado del ordenador, el módulo de lectura recibe y decodifica las cadenas de caracteres, extrayendo la información relevante de cada una de ellas. En el caso de la trama GPGGA, el programa realiza además la conversión de la medida de posicionamiento de coordenadas geográficas —latitud y longitud— a coordenadas UTM (Universal Transversal de Mercator) —en metros—. Una vez que se completa la lectura de todas las tramas de un ciclo, el módulo empaqueta la información en una estructura de mensaje LCM llamada `gps_all` y la envía a través de un canal LCM homónimo. La figura 3.4 muestra la estructura definida para este mensaje utilizando pseudocódigo LCM.

3.4.2. IMU

Al igual que para el receptor DGPS, hemos desarrollado un módulo de lectura dedicado exclusivamente a la decodificación de los mensajes transmitidos por la IMU. En este caso la comunicación entre la unidad y el ordenador a bordo se realiza a través de una interfaz RS-232 con una velocidad de transferencia de 115200 baudios. Esta unidad dispone de varias

```
struct gps_all
{
    int32_t    Timestamp_sec;   // Tiempo de recepción
    int32_t    Timestamp_nsec;  // Tiempo de recepción
    double     Hora;            // Trama GPGGA
    double     UTMEste;         // Trama GPGGA
    double     UTMNorte;        // Trama GPGGA
    double     Altura;          // Trama GPGGA
    double     HDOP;            // Trama GPGGA
    double     Velocidad;       // Trama GPVTG
    double     HeadingVel;      // Trama GPVTG
    double     SemiMajorDesv;   // Trama GPGST
    double     SemiMinorDesv;   // Trama GPGST
    double     SemiMajorOrien;  // Trama GPGST
    int8_t     Calidad;         // Trama GPGGA
    int8_t     NroSat;          // Trama GPGGA
}
```

Figura 3.4: Definición del mensaje `gps_all` en pseudocódigo LCM.

estructuras de mensaje para la transmisión de las mediciones, todas ellas codificadas como estructuras de datos binarios. Cada mensaje se puede configurar para ser enviado de forma periódica o bien, bajo petición del lector (Crossbow, 2009). Para nuestra implementación, hemos seleccionado el mensaje *Nav Data Package 0* con un muestreo periódico a 100 Hz. Este mensaje engloba en una sola estructura tanto la información de la IMU como la información más reciente del DGPS, la cual se obtiene a través del enlace directo con el receptor GPS; tal y como se mencionó en la sección 3.2.1.

Luego de decodificar la información contenida en los mensajes, el programa genera una estructura del tipo `imu_all` y la transmite al resto de componentes utilizando un canal identificado con el mismo nombre. El contenido de esta estructura se muestra en la figura 3.5. El campo identificado como `IMUTime` contiene la hora del día —obtenida del GPS— en milisegundos. Este valor permite que un tercero pueda realizar la correlación de la información generada por el módulo del GPS y el módulo de la IMU.

3.4.3. Bus CAN del vehículo

Hoy en día los vehículos cuentan con un gran número de sensores a bordo, necesarios para el funcionamiento de los diferentes subsistemas integrados, como son el sistema ABS, el control electrónico de estabilidad y el panel de información al conductor, entre otros. La información generada por todos estos sensores y subsistemas circula por todo el vehículo a través de redes de comunicación embarcadas, normalmente definidas bajo el protocolo CAN. Una parte de estos datos, como la velocidad de las ruedas, es de gran utilidad para el control del vehículo; por lo que al leerlos directamente de la red es posible prescindir de la incorporación de otros sensores para estas variables.

Gracias al adaptador CAN-USB descrito en la sección 3.2.1, el módulo de lectura desarrollado es capaz de recibir todos los mensajes que circulan a través de la red CAN, tal y como haría cualquier otro subsistema embarcado por el fabricante. Esto ha permitido acce-

```
struct imu_all
{
    int32_t    Timestamp_sec;   // Tiempo de recepción
    int32_t    Timestamp_nsec;  // Tiempo de recepción
    double     Roll;
    double     Pitch;
    double     Yaw;
    double     Wx;              // vel. giro en eje X
    double     Wy;              // vel. giro en eje Y
    double     Wz;              // vel. giro en eje Z
    double     AcelX;           // acel. en eje X
    double     AcelY;           // acel. en eje Y
    double     AcelZ;           // acel. en eje Z
    double     VelN;            // vel. norte
    double     VelE;            // vel. al este
    double     VelZ;            // vel. al eje Z
    double     GPSLong;         // Longitud actual
    double     GPSLat;          // Latitud actual
    double     GPSAlt;          // Altura actual
    double     Temp;            // Temperatura interna
    int32_t    IMUTime;         // Tiempo (sincronizado con GPS)
    int16_t    BITStatus;       // Errores
}
```

Figura 3.5: Definición del mensaje `imu_all` en pseudocódigo LCM.

der a cuatro variables indispensables para el control del vehículo: la posición del volante, la velocidad de giro del volante, la velocidad del vehículo y su aceleración. En el caso de este módulo de lectura, no es posible presentar los detalles respecto a la recepción de las tramas o su proceso de decodificación, ya que existe un contrato de confidencialidad con el fabricante del vehículo.

Una vez extraída la información del bus, el módulo genera dos estructuras diferentes que se envían a través de sus respectivos canales homónimos: `can_coche_vel_acel`, para la velocidad y aceleración del vehículo; y `can_coche_volante` para la posición del volante y su correspondiente velocidad de rotación. La razón por la que se ha separado la información en dos estructuras es que las tramas utilizadas por el módulo de lectura circulan por la red CAN a distintas frecuencias: 25 Hz para los datos de velocidad y aceleración y 10 Hz para los datos del volante. La figura 3.6 muestra la definición de ambas estructuras.

3.5. E/S General

Como se menciona en la descripción de la arquitectura propuesta (sección 3.3), la etapa *E/S general* abarca los distintos componentes de bajo nivel que son utilizados por los elementos de otras etapas, independientemente de su función. En el caso específico de la implementación en Platero, la etapa incluye la tarjeta de salidas analógicas, el módulo de entradas y salidas analógicas/digitales y los elementos utilizados para la comunicación entre los vehículos y la infraestructura. A continuación se presentan los programas de control de los módulos de entradas y salidas, dejando la descripción del sistema de comunicaciones a la aplicación final de este trabajo —capítulo 6—.

```
struct can_coche_vel_acel       // Info longitudinal
{
    int32_t    Timestamp_sec;   // Tiempo de recepción
    int32_t    Timestamp_nsec;  // Tiempo de recepción
    double     Velocidad;       // Velocidad del coche
    double     Aceleracion;     // Aceleración del coche
}

struct can_coche_volante        // Info del volante
{
    int32_t    Timestamp_sec;   // Tiempo de recepción
    int32_t    Timestamp_nsec;  // Tiempo de recepción
    double     Angulo;          // Posición del volante
    double     Velocidad;       // Velocidad de rotación
}
```

Figura 3.6: Definición de los mensajes `can_coche_vel_acel` y `can_coche_volante` en pseudocódigo LCM.

```
struct control_analogico
{
    double     Valor;          // Valor de salida
    int8_t     Canal;          // Canal de salida
}
```

Figura 3.7: Definición del mensaje `control_analogico` en pseudocódigo LCM.

3.5.1. Salidas analógicas

En la arquitectura previa de AUTOPIA, se asociaba directamente el control de la tarjeta analógica al control del acelerador, ya que la función principal de la misma consistía en generar las señales para el control de este actuador. Esto imposibilitaba el uso de la tarjeta analógica para otras funciones si no se inicializaba correctamente todo el sistema de control del acelerador. Para resolver esto, la arquitectura propuesta implementa un módulo de control dedicado exclusivamente a la gestión de este componente.

El programa desarrollado se encarga de inicializar correctamente todos los canales de la tarjeta analógica y, posteriormente, se queda a la espera de las posibles peticiones provenientes de otros módulos para establecer el valor de salida en un canal específico. Las peticiones son recibidas como estructuras de datos `control_analogico` a través de su respectivo canal LCM —ver figura 3.7—. Por último, el programa se encarga de limitar los valores recibidos dentro del rango apropiado de la tarjeta, pero sin realizar ninguna correspondencia entre los canales y las aplicaciones específicas a las que se dedican —e.g. control del acelerador—.

3.5.2. Módulo analógico/digital

Continuando con la línea de diseño elegida para la arquitectura, hemos desarrollado un programa específico para el control del módulo analógico/digital. La incorporación de este componente de hardware responde a las especificaciones técnicas de los distintos elementos del vehículo; ya que mientras que el control del acelerador requiere de una precisión elevada

Orden	Descripción
0x01 — 0x02	Establecer salida analógica. IdSalida = Orden - 0x01
0x11 — 0x18	Establecer salida digital. IdSalida = Orden - 0x11
0x21	Establecer salidas digitales.

Tabla 3.2: Tipo de petición de acuerdo al valor del campo `Orden`.

y de una baja potencia, el control del freno demanda una mayor potencia del dispositivo con una precisión moderada.

En este caso, el programa de control además de inicializar las salidas del módulo, monitoriza continuamente los valores de todos los canales analógicos y digitales, tanto de entrada como de salida. Este proceso se realiza a una frecuencia de 20 Hz, la máxima soportada por el módulo. En cada ciclo, la información obtenida se empaqueta dentro de una estructura del tipo `modulo_can_estado` —ver figura 3.8— y se envía a través del canal del mismo nombre. Los motivos que han llevado a implementar la lectura continua (*polling*) sobre una lectura por petición (*on-request*) son: (i) garantizar que la frecuencia de lectura no es mayor a la soportada por el módulo, ya que empleando la lectura por petición la frecuencia depende directamente de los programas que envían las peticiones; (ii) proveer la información del sistema tan pronto como esta esté disponible, reduciendo el tiempo de petición; y (iii) utilizar un número reducido de órdenes para la lectura, ya que se emplean comandos que realizan la petición conjunta de todos los valores digitales y/o analógicos.

En paralelo a este proceso de lectura, el software atiende a las peticiones de escritura provenientes de otros elementos de la arquitectura. Las peticiones se reciben a través del canal `modulo_can_orden` dentro de mensajes con una estructura de datos del mismo nombre. La estructura, mostrada en la figura 3.8, es válida tanto para órdenes analógicas como digitales. El tipo de petición, analógica o digital, es indicado por el campo `Orden` de acuerdo a la tabla 3.2. El campo `IDModulo` señala el módulo especifico sobre el cual se llevará a cabo la acción ya que es posible utilizar el mismo bus para conectar módulos adicionales. Para esta implementación sólo se ha utilizado un módulo con id 0x01.

3.6. Actuación

A lo largo de esta sección se describen los diferentes módulos que conforman la etapa de actuación del sistema; es decir, el software utilizado para el control del acelerador, freno y volante. A pesar de que cada programa está orientado a un actuador específico, el funcionamiento de todos ellos es similar entre sí: recibir y procesar las órdenes normalizadas enviadas desde el programa de control, accionando adecuadamente el actuador correspondiente. En este sentido, se ha definido una estructura de mensajes común para todos los programas, llamada `orden_actuador`. Como se puede apreciar en la figura 3.9, la estructura cuenta únicamente

```
struct modulo_can_estado
{
    int32_t    Timestamp_sec;   // Tiempo de recepción
    int32_t    Timestamp_nsec;  // Tiempo de recepción
    byte       IdModulo;        // ID del módulo
    double     InputVolA;       // Entrada analógica 1
    double     InputVolB;       // Entrada analógica 2
    double     InputCorrA;      // Entrada analógica 1 (C)
    double     InputCorrB;      // Entrada analógica 2 (C)
    double     OutputAnlg1;     // Salida analógica 1
    double     OutputAnlg2;     // Salida analógica 2
    boolean    InputDig1;       // Entrada digital 1 ...
    boolean    InputDig2;       // ... 2 ...
    boolean    InputDig3;       // ... 3 ...
    boolean    InputDig4;       // ... 4
    boolean    OutputDig1;      // Salida digital 1 ...
    boolean    OutputDig2;      // ... 2 ...
    boolean    OutputDig3;      // ... 3 ...
    boolean    OutputDig4;      // ... 4 ...
    boolean    OutputDig5;      // ... 5 ...
    boolean    OutputDig6;      // ... 6 ...
    boolean    OutputDig7;      // ... 7 ...
    boolean    OutputDig8;      // ... 8
}

struct modulo_can_orden
{
    double     ValorAnalog;     // Valor analógico
    byte       IdModulo;        // ID del módulo
    byte       Orden;           // Tipo de orden
    byte       ValorDigital;    // Valor digital
}
```

Figura 3.8: Definición de los mensajes `modulo_can_estado` y `modulo_can_orden`.

```
struct orden_actuador
{
    double     Valor;           // Valor de la orden
    int8_t     Prioridad;       // Prioridad de la orden
}
```

Figura 3.9: Definición del mensaje `orden_actuador` en pseudocódigo LCM.

con dos campos: `Valor`, que contiene el valor normalizado de la acción sobre el actuador; y `Prioridad`. Este último permite al módulo clasificar las órdenes recibidas, priorizando aquellas con un valor más cercano a cero. A continuación se agrupan los módulos en dos categorías de acuerdo a su función: control longitudinal (acelerador y freno) y control lateral (volante).

3.6.1. Control longitudinal

Tan pronto como se inicia su ejecución, el módulo de control del acelerador se suscribe al canal `orden_acelerador`, a través del cual recibe las órdenes provenientes de la etapa de decisión. Cada orden recibida es procesada, determinando el valor de las salidas analógicas que emulan el comportamiento del pedal electrónico del acelerador. Una vez determinados los valores, el programa transmite la información al módulo de control de la tarjeta analógica —ver sección 3.5— que finalmente establece los valores calculados en las salidas correspondientes.

De forma análoga, el programa de control del freno recibe las órdenes a través del canal `orden_freno`. En este caso, el valor recibido se emplea para determinar el estado de cada una

de las válvulas del sistema electro-hidráulico, las cuales son accionadas a través del módulo analógico/digital. Para ello, el programa de control del freno envía las órdenes respectivas a través del canal `modulo_can_orden`.

El campo `Prioridad` presente en las órdenes de cada actuador permite que los programas prioricen las órdenes recibidas, mejorando así la seguridad del sistema. Por ejemplo, si implementamos un software de seguridad con una prioridad más alta —valor más pequeño— que el software general de control, garantizamos que las órdenes de seguridad se ejecuten sobre las órdenes del programa general de control, las cuales serían descartadas automáticamente.

Como medida de seguridad adicional, ambos programas cuentan con un temporizador que hace las veces de *watchdog*, estableciendo automáticamente los valores mínimos de actuación en caso de que no se reciban órdenes desde el programa de control. El tiempo de espera del temporizador es de 250 ms, equivalente a 2,5 ciclos de control a una frecuencia de trabajo de 10 Hz. Gracias a esta medida se evita que, en caso de un fallo del programa general de control, los actuadores mantengan permanentemente la última acción recibida, lo cual podría provocar un accidente.

3.6.2. Control lateral

El control lateral del vehículo cuenta con un único elemento de software, encargado de gestionar las acciones sobre el volante. Al igual que los programas homólogos para control longitudinal, el software de control del volante recibe las órdenes desde la etapa de decisión a través de un canal LCM dedicado, en este caso el canal `orden_volante`. Cada orden recibida se decodifica como una posición de referencia para el volante dentro del rango [-540, 540], estableciendo como referencia la última orden recibida, respetando la prioridad de las mismas.

En el caso de este actuador, hemos optado por implantar el control de bajo nivel directamente dentro del módulo de control del volante, utilizando el controlador físico únicamente como *driver* del motor. Esta decisión se ha tomado por tres razones: (i) la posición y velocidad del volante se obtienen a través del bus CAN del vehículo con un protocolo propio del fabricante, el cual no se adecúa a la interfaz del hardware instalado; (ii) la frecuencia de muestreo de los datos implica el uso de controladores discretos, mas que continuos; y (iii) al emplear el control por software, se deja abierta la posibilidad de utilizar otras técnicas de control distintas al PID clásico, que es la única soportada directamente por el dispositivo instalado.

Para la ejecución del control, el módulo primero se suscribe al canal `can_coche_volante`, a través del cual recibe toda la información sobre el volante del coche. Posteriormente, en base a los datos obtenidos en cada mensaje el software calcula el error de posición respecto a la última referencia almacenada y, utilizando un controlador basado en lógica borrosa, determina la consigna de voltaje a aplicar sobre el motor a fin de realizar la corrección del error. Finalmente, se transmite la consigna de control al dispositivo, que actúa directamente sobre el motor. La figura 3.10 muestra el esquema del controlador, detallando las funciones de pertenencia utilizadas para la codificación de las entradas y salidas del mismo, así como

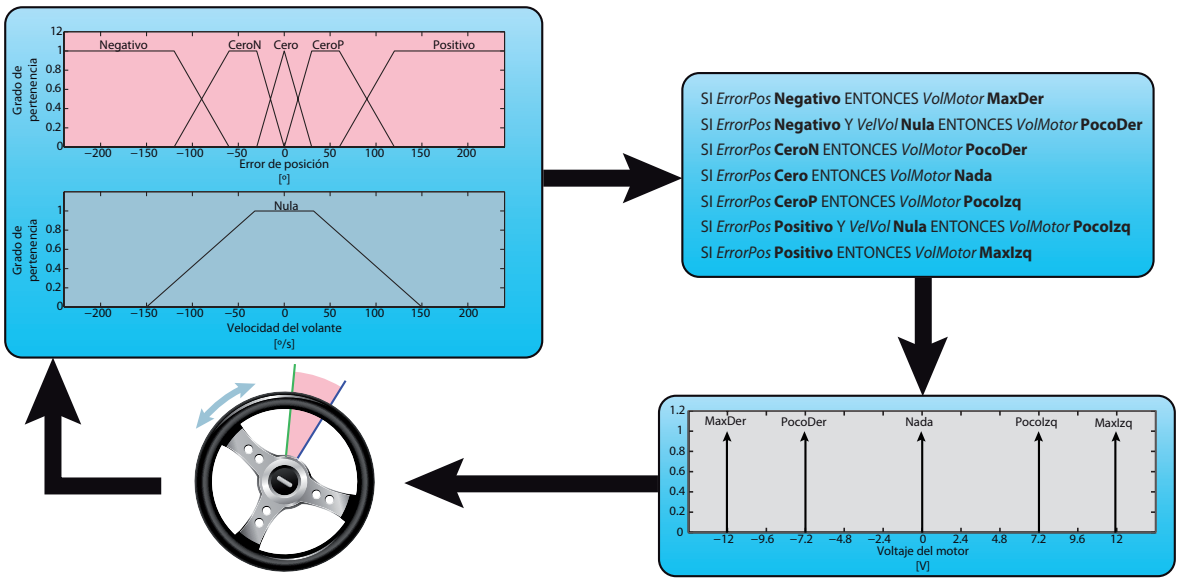


Figura 3.10: Esquema del controlador de bajo nivel para el volante.

las reglas de control utilizadas.

De forma análoga al control longitudinal, se emplea un temporizador como medida de seguridad del programa. En este caso, si se supera el tiempo máximo programado —2,5 ciclos de control— el módulo inhabilita automáticamente el dispositivo de control. Esto libera el motor de la dirección, facilitando que el conductor retome el control del vehículo sin percibir ningún par asociado a la última consigna de control recibida.

3.7. Decisión

Teniendo ya definidas las etapas externas, encargadas de controlar la percepción y actuación del vehículo, presentamos la columna vertebral del control autónomo del vehículo: la etapa de decisión. A diferencia de las etapas descritas previamente, la etapa decisión está unificada dentro de un único elemento de software; aunque internamente sus componentes se encuentran divididos y clasificados en distintas clases de acuerdo a su función. Todos estos elementos comparten información a través de distintas funciones, las cuales les permiten publicar y leer información utilizando un componente común llamado `pizarra`. A continuación se presentan los distintos elementos de control que conforman esta etapa.

3.7.1. Misión

Desde el punto de vista jerárquico de esta etapa, el módulo de misión representa el primer nivel de la planificación del vehículo. Al igual que ocurre con otros sistemas robotizados, el vehículo sin conductor requiere que se le indique la tarea específica que llevará a cabo. Esta tarea se puede definir de forma simple, como *ir desde el punto A hasta el punto B por el camino más rápido*; o bien podría ser más compleja, como sería en el caso de un autobús autónomo, el cual debe seguir una trayectoria predefinida, manteniendo una frecuencia de

paso determinada mientras realiza paradas en puntos específicos y espera que los pasajeros embarquen y desembarquen. En cualquiera de los dos casos, la tarea se puede presentar como un conjunto de sub-objetivos que se deben alcanzar hasta completar la misma. Esta lista de objetivos es lo que se considera la misión del vehículo, dentro de la arquitectura propuesta.

Para el desarrollo de este componente, se han definido dos elementos de software fundamentales: `puntoMision` y `moduloMision`. La clase `puntoMision` contiene los campos necesarios para la descripción de los sub-objetivos de acuerdo a la tarea a cumplir. Por su parte, la clase `moduloMision` contiene el grueso de la definición del módulo, siendo la encargada, por una parte, de gestionar adecuadamente la lista de sub-objetivos que definen la tarea y, por otra parte, de proveer una vía segura para que otros módulos puedan acceder a la información de la tarea. La lista de objetivos ha sido programada como una lista dinámica doblemente enlazada. Esto permite que la tarea se pueda describir antes de la ejecución del programa de control. de forma estática, o bien durante la ejecución del mismo, de forma dinámica.

A fin de garantizar que el acceso a la lista por parte de módulos externos no afecte la integridad de la información almacenada en la misión, se ha definido una herramienta denominada *marcador* para la lectura de la misma. Un marcador es un apuntador a una posición de la lista que sólo puede ser manipulado por el gestor de la lista, es decir el módulo misión. Cuando un componente solicita acceso a la información de la misión por primera vez, se le asocia un marcador único que emplea como identificador cada vez que accede a la misma. Además, los desplazamientos de avance o retroceso a lo largo de la lista son gestionados por el módulo misión a petición del componente que realiza la lectura. Esto garantiza que no se realicen accesos a direcciones de memoria no inicializados, o bien que un componente externo pueda modificar directamente la información de la misión.

Este módulo ha sido diseñado para facilitar la adaptación del sistema a las condiciones de la tarea que el vehículo debe llevar a cabo. Por ejemplo, si la tarea asignada es ir desde un punto a otro de la ciudad haciendo una parada intermedia en el supermercado, la misión incluiría sólo el punto inicial, la posición del supermercado y el punto final. Sin embargo, si se le encomienda el seguimiento de un vehículo líder, la misión estará conformada de una secuencia de puntos que rastrean la evolución del líder. En ambos casos, el proceso de adaptación de la misión a una trayectoria de referencia es realizada por el módulo planificador, presentado en la siguiente sección.

3.7.2. Planificador

Una vez definida la misión del vehículo, el siguiente paso en la planificación consiste en generar una trayectoria de referencia adecuada para completar la tarea. Con este fin se ha desarrollado el módulo planificador, segundo en la jerarquía dentro de la etapa de decisión.

Dada la similitud en funcionamiento entre los módulos misión y el planificador, para la implementación de este último también se han desarrollado dos elementos de software `puntoPlanificador` y `moduloPlanificador`. Como ocurre en el caso del módulo misión, la clase `moduloPlanificador` contiene la definición total del módulo, encargándose prin-

cipalmente de la gestión de la serie de puntos consecutivos que definen la trayectoria de referencia del vehículo. Cada uno de estos puntos se define utilizando una instancia de la clase `puntoPlanificador`. Esta clase se compone de diversos campos con información relacionada al punto de referencia tales como las coordenadas UTM del mismo, la consigna de velocidad señalada, la distancia al punto de inicio de la trayectoria y la curvatura de esta en ese punto especifico, entre otros. De ser necesario, la clase permite además la incorporación y personalización de nuevos campos en función de los requisitos específicos para una nueva tarea.

Al iniciar su ejecución, la primera acción del planificador es solicitar acceso al módulo misión actual, señalado por la `pizarra`. Una vez concedido el acceso a los datos del módulo, el planificador inicia una lectura continua de la información almacenada en la misión, uniendo cada uno de los puntos definidos como sub-objetivos a través de una sucesión de puntos equidistantes entre sí. Hemos decidido emplear una distancia de separación constante entre los puntos de referencia con el objetivo de suavizar la evolución de las variables de control que son estimadas a partir de la trayectoria de referencia. El valor de la distancia de separación es personalizable, teniendo un valor por defecto igual a la mitad de la longitud del coche en cuestión.

Mientras que no se indique lo contrario, y siempre y cuando los datos almacenados en la misión sean suficientes, el planificador genera una trayectoria de hasta 100 metros por delante de la posición actual del vehículo. Esto permite que, más allá de esta distancia, la misión pueda ser modificada de forma dinámica, sin necesidad de reiniciar totalmente la planificación. Si ocurriese el caso en el que la modificación afecte parte de la trayectoria ya generada, sería necesario reiniciar el módulo planificador desde el último sub-objetivo alcanzado.

Al igual que el módulo misión, el planificador proporciona acceso a su información a través de marcadores. En este caso, éstos son de gran utilidad ya que permiten que varios componentes, o incluso el mismo, puedan acceder a la vez a distintas zonas de la trayectoria generada.

3.7.3. Control

El tercer y último componente de la etapa de decisión es el módulo de control; encargado, como su nombre indica, de generar las acciones de control pertinentes en función del estado del vehículo y de la tarea asignada. Como evidencian la mayoría de los trabajos desarrollados por AUTOPIA, el enfoque general no ha estado orientado en ningún momento al desarrollo de un controlador global para todos los escenarios a los que debe enfrentarse un vehículo. Por el contrario, se ha optado por el análisis individual de aquellos que son más comunes como los adelantamientos (Pérez et al., 2011b), las incorporaciones (Milanés et al., 2011b), el ACC (Naranjo et al., 2006) y las frenadas de emergencia (Milanés et al., 2011c), entre otros; desarrollando controladores específicos para cada uno de ellos y permitiendo que el sistema seleccione el más adecuado para la tarea de acuerdo a las condiciones del entorno (Milanés, 2010).

En este sentido, la primera tarea de este módulo en cada ciclo consiste en la estimación de cada una de las variables necesarias para control del vehículo. A fin de mantener el tiempo de ejecución tan bajo como sea posible, esta tarea ha sido dividida en dos fases. Durante la primera fase el sistema calcula sólo las variables que son requeridas para realizar la selección del modo de control más adecuado, como por ejemplo la velocidad del vehículo y los errores laterales y angulares respecto a la trayectoria. En una segunda fase, posterior a la selección del modo de control, el programa calcula aquellas variables que son específicas a este último; como son los intervalos de tiempo y/o distancia que separan los vehículos, la velocidad relativa entre ellos o bien la prioridad de un vehículo ante una intersección.

Por último, el módulo procede con la ejecución de los controladores definidos para el modo de control seleccionado. La mayor parte de los trabajos desarrollados por AUTOPIA demuestran una clara preferencia por el uso de controladores basados en lógica borrosa, pero la arquitectura propuesta está abierta a la implementación de otras técnicas de control, siempre que los controladores utilizados sean coherentes con las restricciones de la arquitectura en cuanto a los controladores de bajo nivel (i.e. rango normalizado de las salidas).

Los controladores borrosos son definidos y ejecutados dentro de un sub-componente basado en el ORBEX (García y De Pedro, 2000). No obstante, este componente incluye diversas modificaciones pensadas para incrementar la funcionalidad del sistema, mejorando sensiblemente las prestaciones del mismo hacia una versión 2.0 de ORBEX. A continuación se describe brevemente su funcionamiento y las mejoras realizadas.

3.7.3.1. Orbex 2.0

La lógica borrosa se basa en la teoría de conjuntos borrosos introducidos por (Zadeh, 1965). A diferencia de la lógica clásica, donde la pertenencia de un elemento a un determinado rango se expresa de forma binaria —pertenece o no pertenece al conjunto—; la teoría de los conjuntos borrosos define esta relación a través de valores continuos dentro del rango [0, 1], donde 1 representa el máximo grado de pertenencia a un conjunto determinado. Por otra parte, la asociación de etiquetas lingüísticas a los distintos conjuntos facilita la aplicación de un razonamiento más humano a través de la computación con palabras (Zadeh, 1999).

Convencionalmente, la arquitectura de AUTOPIA se apoyaba en la herramienta ORBEX para la ejecución en tiempo real de los controladores borrosos que regían el comportamiento del vehículo. En ORBEX, los controladores son definidos a través de un fichero de configuración utilizando un lenguaje de entrada muy sencillo basado en el lenguaje natural (García Rosa y De Pedro, 1998). Las entradas se describen utilizando funciones de pertenencia trapezoidales, definidas por 4 valores que representaban la posición de los 4 vértices del trapecio. Por su parte, las salidas son codificadas utilizando valores discretos definidos por funciones *singletons* (Sugeno, 1999). Esto se hace con el objetivo de reducir el tiempo de cómputo empleado en el cálculo de la salida del controlador. Finalmente, cada una de las reglas de inferencia está definida a través de la siguiente sintaxis de ejemplo:

```
SI Entrada1 Etiqueta1A Y Entrada2 Etiqueta2A ENTONCES Salida1 SalidaA
```

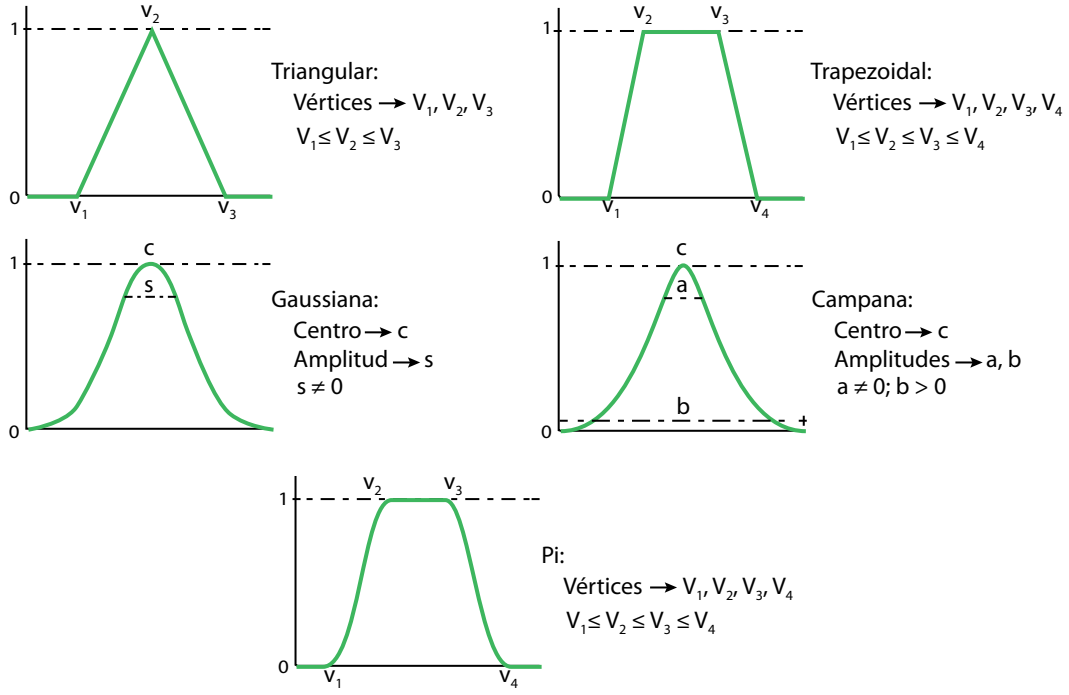


Figura 3.11: Funciones de pertenencia disponibles por defecto en la versión 2.0 del ORBEX.

donde `Etiqueta1A` y `Etiqueta2A` representan etiquetas lingüísticas asociadas a la `Entrada1` y `Entrada2` respectivamente (García Rosa y De Pedro, 1998).

A pesar de la amplia funcionalidad demostrada por esta herramienta, su aplicación en algunos escenarios está limitada por tres de sus características principales: (i) sólo permite el uso de funciones de pertenencia trapezoidales para definir las distintas entradas del sistema; (ii) la definición del controlador sólo puede realizarse a través de un fichero de configuración y siempre antes de su ejecución, por lo que resulta imposible realizar cambios en el controlador una vez que este se encuentra en ejecución; y (iii) sólo se permite la ejecución de un controlador a la vez.

Para sortear estas limitaciones en la nueva arquitectura, se ha desarrollado una versión actualizada del sistema ORBEX. Como primer cambio, se ha encapsulado el ORBEX dentro de una clase del sistema, garantizando así la coexistencia de varios controladores de forma simultánea sin que estos interfieran entre sí. Por otro lado, se han incorporado cuatro tipos de funciones de pertenencia: triangulares, gausianas, campana y trapecios suavizados (función pi); aumentando así el abanico de opciones para definir las entradas del sistema —ver figura 3.11—. Además, se pueden emplear funciones adicionales a través de la definición de una sencilla clase. Por último, se han incorporado herramientas que permiten la gestión dinámica del controlador, de modo que se pueden agregar, eliminar y/o modificar las entradas, salidas y reglas de forma individual.

CAPÍTULO 4

LOCALIZACIÓN POR ENJAMBRE DE PARTÍCULAS

Uno de los retos fundamentales en el campo de los vehículos inteligentes es conseguir un sistema de localización con la suficiente precisión, fiabilidad y robustez. Entre las posibles soluciones, el uso de los sistemas de localización basados en los sistemas globales de navegación por satélite (GNSS por sus siglas en inglés); como las redes GPS y GLONASS, operadas por EE.UU. y Rusia respectivamente. A pesar de su amplia aceptación, estos sistemas están limitados por los efectos de la interferencia, bloqueo y propagación multitrayecto de las señales provenientes de los satélites, lo cual merma considerablemente su precisión (Kaplan, 1996). En la mayoría de los casos se pueden sortear estos inconvenientes utilizando sistemas de corrección diferencial, también conocidos como DGPS. No obstante, esta solución incrementa los costos de los equipos utilizados y reduce el nivel de autonomía del sistema puesto que depende de la conexión con una estación base (Godoy et al., 2010). Por todo ello, las soluciones propuestas en la actualidad pasan por la integración de varios sensores —e.g. GPS, IMU, RADAR, LIDAR— para aumentar la fiabilidad y robustez del mismo (Toledo-Moreo et al., 2009; Martí et al., 2012)

Este capítulo está centrado en el diseño, desarrollo e implementación de un algoritmo de localización para vehículos. Parte de este trabajo se realizó en colaboración con el *Laboratoire sur les Interactions Véhicules-Infrastructure-Conducteurs* (LIVIC) en Versailles, Francia; durante la realización de una estancia de investigación por parte del autor en dicho centro. El algoritmo desarrollado utiliza un enfoque similar al de los filtros de partículas, que consideran de forma simultánea varios estados posibles del sistema con un grado de confianza asociado a cada uno de ellos. Sin embargo, el algoritmo propuesto integra también la emulación del comportamiento de los enjambres naturales —bandadas y cardúmenes— en el proceso de estimación del estado a partir de un estado previo. El algoritmo ha sido probado en dos fases: la primera durante el desarrollo del sistema en Francia, donde se utilizaron datos sincronizados, capturados con un vehículo prototipo instrumentado por el LIVIC; y una segunda fase en AUTOPIA, donde se utilizaron datos asíncronos capturados con el vehículo Platero.

4.1. Trabajos Previos

Al día de hoy se han presentado diversas soluciones plausibles para la fusión sensorial en el proceso de localización de vehículos. Uno de los enfoques más conocidos es el filtro de Kalman (KF) (Kalman, 1960), presentado en 1960 por el investigador del mismo nombre como una solución al problema del filtrado discreto (Welch y Bishop, 1995). Este funciona en dos fases alternadas de forma recurrente: predicción y corrección. Durante la fase de predicción se estima el estado del sistema a partir del estado anterior, empleando para ello un modelo del sistema. Posteriormente, en la etapa de corrección se actualizan los valores estimados en función de la observación realizada de acuerdo al modelo de medición.

Por definición teórica, el KF está pensado para problemas cuyo modelo de evolución es lineal, lo cual limita su aplicación a sistemas descritos a través de modelos no lineales (Kelly, 1994). Ante esta cuestión se han planteado variaciones del algoritmo, como el filtro de Kalman extendido (EKF) (Welch y Bishop, 1995) y el filtro de Kalman *unscented* (UKF) (Julier y Uhlmann, 2004), que abordan el problema de la no linealidad del sistema y mantienen la estructura de funcionamiento del KF original. Como demuestra la literatura, ambos enfoques —EKF y UKF— han sido aplicados notablemente a la localización de vehículos (Li y Leung, 2003; Zhang et al., 2005; Rezaei y Sengupta, 2007); llegando incluso a emplearse configuraciones de filtros en cascada para reducir la propagación de los errores del modelo (Bevly y Parkinson, 2007), u optimizado a través de su combinación con algoritmos genéticos (Lu et al., 2009). En cuanto al error de los modelos, el algoritmo supone que estos están bajo la influencia de ruidos con distribuciones Gaussianas, lo cual puede llegar a ser un factor limitante para algunas aplicaciones. Esto ha motivado el desarrollo y aplicación de un nuevo tipo de filtros conocidos como filtros de partículas (PF - *particle filters*).

El concepto de PF fue propuesto a principios de los 90 por (Gordon et al., 1993), en respuesta al vacío existente para la estimación de estados no observables en sistemas no-lineales, no-gaussianos. A diferencia del filtro de Kalman, que utiliza una única muestra del estado del sistema; este algoritmo emplea un conjunto de muestras que representan diversas soluciones posibles al estado del sistema de forma simultánea. Cada una de estas muestras es lo que se denomina partícula, lo que da origen al nombre del algoritmo.

A pesar de que existen diversas implementaciones del PF, todas ellas presentan un funcionamiento base similar. Durante la etapa de inicialización del filtro las partículas se esparcen de forma aleatoria sobre todo el espacio de estados. Posteriormente, las partículas son remuestreadas en cada ciclo en función de su probabilidad de ser una solución plausible al estado del sistema. Por su parte, el valor de la probabilidad se actualiza en función de las observaciones realizadas al estado del sistema en los ciclos que incluyen información de la medida (Arulampalam et al., 2002).

Uno de los inconvenientes a los que se enfrenta este tipo de filtros es la degeneración de las partículas después de varios ciclos de estimación; lo que significa que un gran número de las partículas presentes poseen una probabilidad despreciable de representar el estado real del sistema. La versión *Sampling Importance Resampling* (SIR) de este algoritmo (Gordon et al.,

1993; Arulampalam et al., 2002) aborda este problema implementando métodos de remuestreo, los cuales se encargan de eliminar las partículas con baja probabilidad y generan nuevas partículas en torno a aquellas con mayor probabilidad. Algunos ejemplos de implementaciones que consiguieron buenos resultados usando PF para la localización son el robot Minerva (Thrun et al., 1999) y el vehículo Boss, ganador del *DARPA Urban Driving Challenge* en 2007 (Urmson et al., 2008).

Una alternativa utilizada para mejorar el rendimiento de los filtros de partículas es la optimización por enjambre de partículas o PSO (*Particle Swarm Optimization*), la cual se basa en el comportamiento de los enjambres biológicos, como por ejemplo las bandadas de aves o bancos de peces (Nedjah y de Macedo, 2006). En este método, las partículas son igualmente inicializadas a lo largo de todo el espacio de estados, con la salvedad de que la evolución de las mismas se realiza tomando en cuenta tanto su evolución particular como la evolución del enjambre. Un ejemplo de la utilidad de este método es su aplicación para el desarrollo de un filtro de partículas optimizado por enjambre (Tong et al., 2006); donde el algoritmo de PSO se emplea como un método de remuestreo de las partículas cuando estas se degeneran más allá del umbral establecido.

Dentro del programa AUTOPIA y previo a la realización de esta tesis doctoral, la línea de investigación relacionada con la localización de los vehículos era un camino poco andado. Esto se debe a que habitualmente todo el desarrollo se lleva a cabo en zonas privadas de pruebas, donde la cobertura del GPS es prácticamente perfecta y se dispone de un sistema propio de corrección RTK (Godoy et al., 2010), siendo innecesario emplear métodos de mejora en la localización de los vehículos. No obstante, la intención de validar nuestros desarrollos en vías públicas nos ha obligado a enfrentarnos a los problemas de interferencia presentes en entornos reales. En este sentido, el presente capítulo describe el desarrollo de un algoritmo de posicionamiento basado en el método de optimización por enjambre de partículas.

4.2. Objetivos e idea general

Como se presentó anteriormente, el objetivo principal es el desarrollo de un nuevo algoritmo de localización basado en la emulación del comportamiento de los enjambres biológicos. Al igual que otros algoritmos de localización, esta propuesta —mencionada de ahora en adelante como LEP (Localización por Enjambre de Partículas)— fusiona la información obtenida a través de distintos sensores embarcados con la finalidad de estimar el estado del vehículo —posición y orientación— en tiempo real. A tal fin se define un grupo de partículas denominado enjambre, que representa parcialmente el conjunto de estados posibles del vehículo; evolucionando en cada ciclo de acuerdo a la información recibida de los sensores.

Como objetivo adicional, se desea disminuir la necesidad de receptores GPS de alta precisión como medida para reducir los costos finales de la aplicación. Por otro lado, se espera que esto incremente la robustez del algoritmo ante los errores en las medidas del GPS, o incluso ante la indisponibilidad de las mismas por ciertos períodos de tiempo. La reducción de la precisión en el GPS se contrarrestará con la inclusión de información de mapas digitales.

Desde el punto de vista del posicionamiento, el funcionamiento del algoritmo será el siguiente: en un primer paso, se inicializa el enjambre dentro del área de incertidumbre en la que se supone que se encuentra el vehículo. A medida de que se procesan las distintas mediciones de los sensores, se calcula la evolución de cada uno de los individuos de acuerdo a sus estimaciones individuales y a la evolución general del enjambre. Cada vez que una partícula evoluciona también lo hace su probabilidad de ser una solución plausible al estado del sistema. Dicha probabilidad se considera como la masa o el peso de las partículas, lo cual hace que las partículas con un mayor peso tengan una mayor independencia del enjambre, mientras que aquellas con un peso menor sufrirán una atracción mayor al movimiento del enjambre. Esto provocará que el área de incertidumbre se reduzca al cabo de algunos ciclos de evolución, convergiendo a una estimación más precisa del estado del vehículo. En cada ciclo el estado estimado estará determinado por el centro de masas —probabilidades— del enjambre.

4.3. Desarrollo del algoritmo

A lo largo de esta sección de describe el desarrollo del algoritmo de LEP de forma detallada. Este proceso se realizó totalmente *off-line*, es decir, utilizando datos capturados durante un recorrido realizado previamente con uno de los vehículos prototipos del LIVIC. El proceso de desarrollo se llevó a cabo con la herramienta MatLab, debido a las facilidades aportadas en cuanto al manejo de matrices y representación gráfica.

En el momento de la captura de los datos, el vehículo —mostrado en la figura 4.1— contaba con un GPS modelo Trimble AG132 y una unidad inercial Crossbow VG400, además de acceso al odómetro del vehículo y los datos de ángulo del volante. La información de todos los sensores se capturó de forma sincronizada, es decir, a la frecuencia de muestreo del GPS que es el sensor cuyo reloj interno posee una mayor precisión. Adicionalmente, el vehículo contaba con un segundo sistema de localización compuesto por un GPS RTK y una unidad inercial de alta precisión, el cual se utilizó para generar la trayectoria de referencia del vehículo de forma simultánea a la captura. El desarrollo y funcionamiento de este último sistema es un desarrollo totalmente independiente del LIVIC y no forma parte del trabajo realizado en la presente tesis.

El recorrido del vehículo se realizó dentro de las instalaciones de prueba ubicadas en Satory, Francia; específicamente en las pistas *Routière* y *Val d'Or* (ver figura 4.2). La primera pista posee características similares a la de una carretera, además de una vista despejada del cielo. En el caso de la pista *Val d'Or*, las características del camino se asemejan más a una ruta rural o a través de un puerto, teniendo una vista del cielo altamente afectada por la presencia de árboles en el entorno.

4.3.1. Modelado del sistema

El primer paso en el desarrollo del algoritmo ha sido definir tanto el vector de estado del sistema como el modelo de evolución del mismo a partir de un estado anterior. Dados los re-

Figura 4.1: Vehículo prototipo del LIVIC.

Figura 4.2: Pistas de prueba utilizadas por el LIVIC.

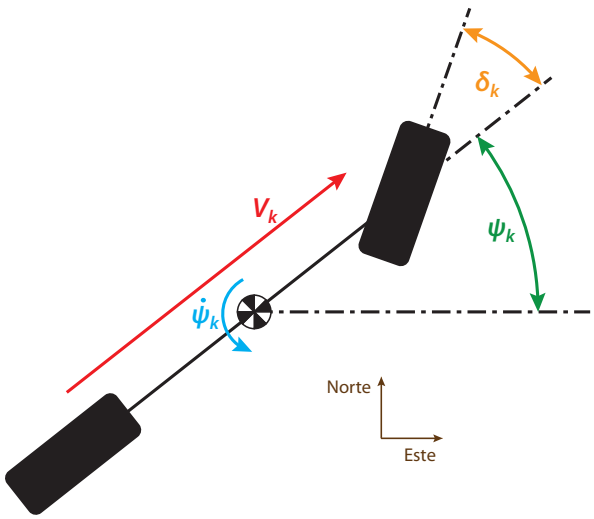


Figura 4.3: Modelo de la bicicleta empleado en la localización.

quisitos presentados en cuanto a las variables de localización, definimos el estado del vehículo —y por ende el de las partículas— empleando el vector de estados $[x_k, y_k, \psi_k]^T$, donde x_k e y_k representan la posición del vehículo dentro del marco de referencia $[X, Y, Z]$; y ψ_k representa el ángulo de giro del vehículo en torno al eje Z (guiñada). Los ejes X e Y representan los ejes Este y Norte de la tierra en el sistema de coordenadas UTM, respectivamente. El ángulo ψ se mide a partir de X con giro positivo en la orientación de Z. El sub-índice k hace referencia al instante —discreto— en el que se determina la variable.

Como modelo de evolución del sistema empleamos el modelo cinemático de la bicicleta:

$$\begin{cases} x_k = x_{k-1} + V_k \cdot \Delta t \cdot \cos\left(\psi_{k-1} + \frac{\Delta t \cdot \dot{\psi}_k}{2}\right) \cdot \cos(\delta_k) \\ y_k = y_{k-1} + V_k \cdot \Delta t \cdot \sin\left(\psi_{k-1} + \frac{\Delta t \cdot \dot{\psi}_k}{2}\right) \cdot \cos(\delta_k) \\ \psi_k = \psi_{k-1} + \Delta t \cdot \dot{\psi}_k \end{cases} \tag{4.1}$$

donde Δt es la diferencia de tiempo que existe entre el instante presente y el anterior, δ_k es el ángulo de la rueda delantera, V_k es la velocidad del vehículo y $\dot{\psi}_k$ es la velocidad angular en torno al eje Z de este. La figura 4.3 muestra una gráfica de las variables sobre un esquema del modelo. La inclusión del ángulo de la rueda delantera permite tomar en cuenta las restricciones cinemáticas del vehículo (Ndjeng Ndjeng et al., 2009).

4.3.2. Atracción del enjambre

Una vez definida la representación del estado del vehículo y el modelo de evolución del sistema, el siguiente paso consiste en definir la regla de atracción que rige el movimiento conjunto del enjambre. Como se menciona en la sección 4.2, el objetivo del algoritmo es que la evolución de cada partícula esté marcada, por una parte, por su evolución individual *a priori* y, por otra parte, por la tendencia general del enjambre. El peso que se asignará a cada una de estas evoluciones para la determinación del estado final de la partícula —*a posteriori*— deberá ser proporcional al peso de la partícula, que expresa su probabilidad de

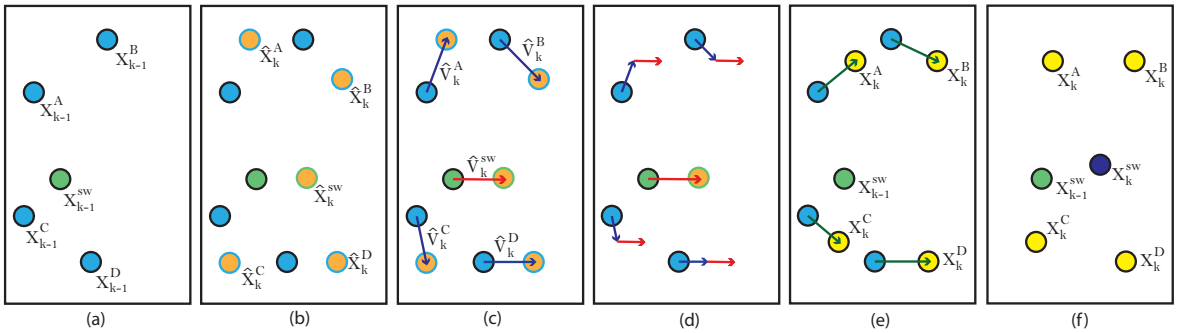


Figura 4.4: Evolución de las partículas en el algoritmo LEP.

ser una solución plausible al estado del sistema.

En una primera aproximación optamos por emplear una regla de atracción que modificara directamente el estado *a priori* de las partículas. Sin embargo, las primeras pruebas realizadas demostraron que, luego de varios ciclos, las partículas tendían a converger a un estado único al que eran atraídas constantemente. Esto provocaba que la varianza del enjambre se perdiera rápidamente sin llegar a generar una solución correcta. Otros métodos sometidos a consideración fueron la implementación de radios de atracción y la variación de la atracción en función de la distancia al centro de masas del enjambre. Ambas fueron descartadas finalmente debido a que provocarían una reducción de la funcionalidad del enjambre y sólo solucionarían el problema parcialmente. Finalmente, luego de analizar estas opciones optamos por emplear un método de atracción que afecta directamente la "velocidad" temporal de las partículas.

Para explicar mejor el funcionamiento de este método de atracción, consideremos un caso de ejemplo. Supongamos que tenemos un enjambre formado por sólo 4 partículas, identificadas como A, B, C y D. Al final del instante $k-1$, los estados de las partículas serán X^A_{k-1}, X^B_{k-1}, X^C_{k-1}, X^D_{k-1}; con los pesos W^A_{k-1}, W^B_{k-1}, W^C_{k-1}, W^D_{k-1}, respectivamente —figura 4.4a—. De igual forma tendremos el estado estimado por el enjambre X^{sw}_{k-1}. En el instante k, todas las partículas, incluyendo la estimación del enjambre, evolucionan de acuerdo con el modelo definido para el vehículo, generando un conjunto de partículas estimadas *a priori*: $\hat{X}^A_k$, $\hat{X}^B_k$, $\hat{X}^C_k$, $\hat{X}^D_k$ y $\hat{X}^{sw}_k$ —imagen (b)—. Para cada individuo, el cambio de estado entre los instantes $k-1$ y k se expresa a través de un vector de transición, que es lo que denominamos la velocidad *a priori* de la partícula —$\hat{V}^A$, $\hat{V}^B$, $\hat{V}^C$, $\hat{V}^D$, y $\hat{V}^{sw}$ en la figura 4.4c—.

Ahora, para incluir el movimiento del enjambre dentro de la estimación *a posteriori* de cada partícula, modificamos los vectores velocidad de acuerdo a la relación de probabilidad del individuo con el estado estimado del enjambre. De esta forma se garantiza que las partículas con menor probabilidad sufran una atracción mayor hacia el conjunto que aquellas que tienen un valor de probabilidad más alto —imagen (d)—. Finalmente, obtenemos los estados *a posteriori* X^A_k, X^B_k, X^C_k y X^D_k añadiendo los vectores de velocidad modificados a los respectivos estados anteriores de las partículas —figura 4.4e—. Por su parte, el nuevo valor estimado del enjambre X^{sw}_k se obtiene a partir de los nuevos estados generados y sus correspondientes valores de probabilidad, actualizados por la información más reciente de los sensores —imagen (f)—.

Empleando este tipo de atracción vectorial se garantiza que las partículas sigan el desplazamiento del enjambre sin tender a la convergencia en un estado único al cabo de unas pocas iteraciones. Adicionalmente, gracias a la definición de un peso mínimo de inercia $W^{inercia}$ para cada partícula, evitamos que aquellas que tengan una probabilidad cercana a cero sufran una atracción total al vector desplazamiento del enjambre en un único ciclo. A continuación presentamos las ecuaciones que describen todo este comportamiento de evolución:

$$\hat{V}_k^i = \hat{X}_k^i - X_{k-1}^i \tag{4.2}$$

$$\hat{V}_k^{sw} = \hat{X}_k^{sw} - X_{k-1}^{sw} \tag{4.3}$$

$$V_k^i = \frac{(W_k^i + W^{inercia}) \cdot \hat{V}_k^i + W_k^{sw} \cdot \hat{V}_k^{sw}}{W_k^i + W^{inercia} + W_k^{sw}} \tag{4.4}$$

$$X_k^i = X_{k-1}^i + V_k^i \tag{4.5}$$

4.3.3. Definición de los pesos

En secciones anteriores se ha mencionado que el objetivo de este algoritmo es expresar la probabilidad de cada partícula empleando el concepto de peso de la misma; a continuación, se define el método de cálculo de dicho peso a partir de la información recibida de los sensores. De entre todos los datos que se reciben, sólo resta el uso de la información del GPS; puesto que ya hemos incorporado los datos de odometría, sensor del volante y velocidad angular dentro del modelo de evolución del sistema. Por otro lado, a fin de satisfacer el requisito de reducir la dependencia de medidas GPS con alta precisión, hemos añadido información de mapas digitales en el cálculo del peso de las partículas, el cual se obtiene finalmente como una combinación de pesos auxiliares.

4.3.3.1. Peso por GPS

Cada medida que realiza un receptor GPS tiene asociada un área de incertidumbre a su alrededor, determinada fundamentalmente por la calidad de las señales recibidas de los satélites. Mientras mejor sea la calidad de la señal, mayor será la precisión de la medida y, por lo tanto, menor el área de incertidumbre. De acuerdo al modo de funcionamiento del receptor —autónomo, DGPS o RTK— es posible estimar el área de incertidumbre empleando valores típicos para cada uno de ellos; sin embargo, este es tan sólo un valor aproximado que no refleja las condiciones reales del entorno en el momento en que se realiza la medición.

Dependiendo del equipo receptor utilizado y de su protocolo de comunicación, es posible obtener directamente el área de incertidumbre asociada a cada medida. Por ejemplo, en el caso de los receptores utilizados por AUTOPIA y el LIVIC, esta información se obtiene a través de la trama `GPGST` del protocolo NMEA, la cual contiene las estadísticas del error de posicionamiento. En este mensaje, la desviación en torno a la medida se expresa como un

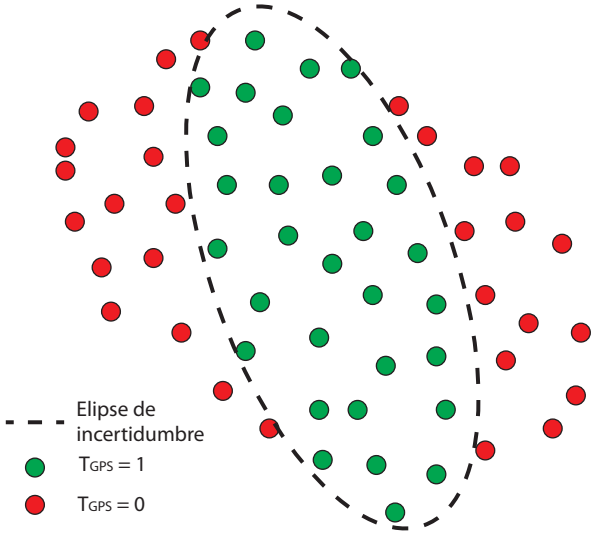


Figura 4.5: Ejemplo de estimación del peso por GPS.

área de incertidumbre con forma de elipse, descrita por los campos *semi-major deviation*, *semi-minor deviation* y *semi-major orientation*; que indican, respectivamente, el tamaño de los semiejes —mayor y menor— y la orientación de la misma (Trimble, 2008). El tamaño de cada semieje es igual al valor de la desviación típica en esa dirección, por lo que multiplicamos por 3 el tamaño de la elipse, a fin de aumentar al 99 % la probabilidad de que el área de incertidumbre contenga la posición real del vehículo.

Con cada mensaje del GPS recibido, el algoritmo primero calcula un peso binario temporal T^i_{gps} para cada partícula i, indicando si la misma se encuentra dentro —1— o fuera —0— del área de incertidumbre. Acto seguido, se actualiza el peso GPS de la partícula (W^i_{gps}) para incluir el valor de T^i_{gps}. Esto se realiza empleando una relación R_{gps} con el último valor del peso, lo cual impide que este cambie de forma drástica de un instante al siguiente, minimizando la influencia de errores temporales de medida. La figura 4.5 y la ecuación 4.6 ilustran, respectivamente, la asignación de T^i_{gps} y el cálculo del peso final por GPS para un enjambre de muestra en el instante k.

$$W^i_{gps_k} = R_{gps} \cdot W^i_{gps_{k-1}} + (1 - R_{gps}) \cdot T^i_{gps} \tag{4.6}$$

4.3.3.2. Peso por mapa

Dado que las medidas del GPS no son empleadas como muestras puntuales sino como áreas de incertidumbre, es necesario añadir información adicional al algoritmo a fin de que este pueda converger a una solución con una incertidumbre menor. En caso contrario sólo estaríamos desplazando el conjunto de partículas sin que estas sufran una modificación mayor a la generada por las variaciones en la incertidumbre del GPS. Por este motivo, hemos incorporado la información de los mapas digitales de las dos pistas de prueba del LIVIC. Como se aprecia en la figura 4.6, cada mapa está formado por dos series de datos, uno para cada borde de la carretera. En cada serie, los puntos se encuentran separados en torno a 1,5 metros

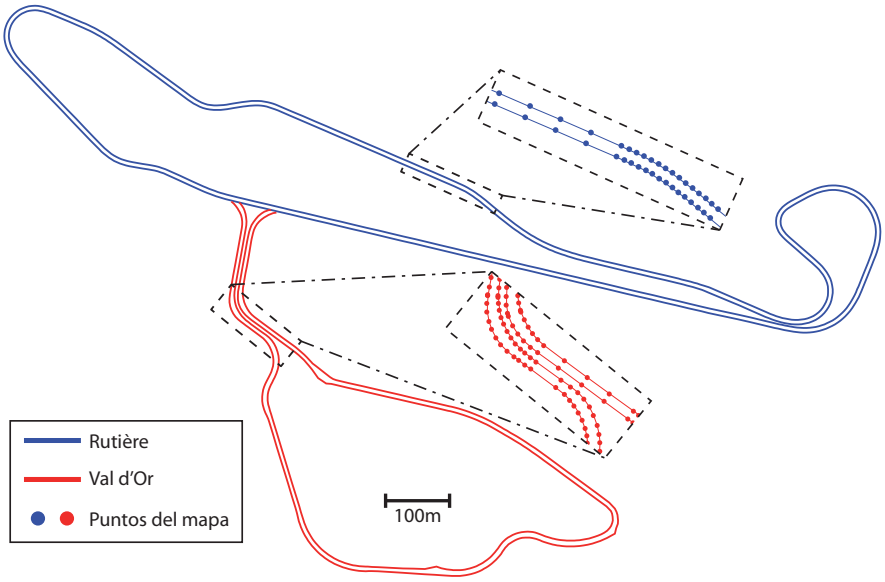


Figura 4.6: Mapas digitales de la pista del LIVIC.

en las secciones curvas y hasta 45 metros en los segmentos rectos. La figura 4.6 muestra una imagen con la secuencia de puntos de cada mapa.

Para facilitar la inclusión de esta información en el algoritmo, añadimos una restricción similar a la de los navegadores comerciales: se asume que el vehículo está ubicado siempre sobre el mapa, siendo imposible que este se salga del camino en ningún momento. Con esta restricción como base, el peso por mapa de la partícula se calcula de forma similar al peso por GPS. Primero se define un valor binario temporal T^i_{map} que indica si la partícula se encuentra dentro o fuera del área delimitada por los mapas. Finalmente, se actualiza el valor del peso W^i_{map} considerando la relación R_{map} con el último valor del peso —ver ecuación 4.7—.

$$W^i_{map_k} = R_{map} \cdot W^i_{map_{k-1}} + (1 - R_{map}) \cdot T^i_{map} \tag{4.7}$$

4.3.3.3. Peso final

Teniendo ya definido el método para calcular los pesos auxiliares —por GPS y por Mapa— el siguiente paso es definir el peso final de la partícula. Partiendo de la suposición de que el vehículo se encuentra siempre sobre el camino, resulta obvio que las partículas que satisfagan tanto esta condición como la del GPS deben tener un valor de probabilidad mayor que aquellas que sólo satisfacen una de ellas. En este sentido, definimos el peso final de las partículas como el resultado de la multiplicación de ambos pesos auxiliares (W^i_{gym}). No obstante, si sólo se considera este producto, las partículas que satisfacen al menos una condición serían despreciadas al igual que aquellas que no satisfacen ninguna de ellas. Por este motivo decidimos emplear una combinación que incluya tanto el valor individual de los pesos auxiliares como el resultado de su producto. La ecuación 4.8 muestra la relación propuesta como primera aproximación, siendo G_{gps}, G_{map} y G_{gym} las ganancias asignadas para cada condición: dentro de

la elipse del GPS, sobre el mapa y ambas, respectivamente.

$$W_k^i = G_{gps} \cdot W_{gps_k}^i + G_{map} \cdot W_{map_k}^i + G_{gym} \cdot W_{gym_k}^i \tag{4.8}$$

Una vez calculado el peso de todas las partículas, este se normaliza en relación con el máximo valor obtenido en el enjambre —ecuación 4.9—. En este caso no se considera ninguna relación con el valor anterior, ya que este comportamiento se introdujo previamente en el cálculo de los pesos auxiliares.

$$W_k^i = \frac{W_k^i}{\text{máx}\left(W_k^1, W_k^2, \ldots, W_k^N\right)} \tag{4.9}$$

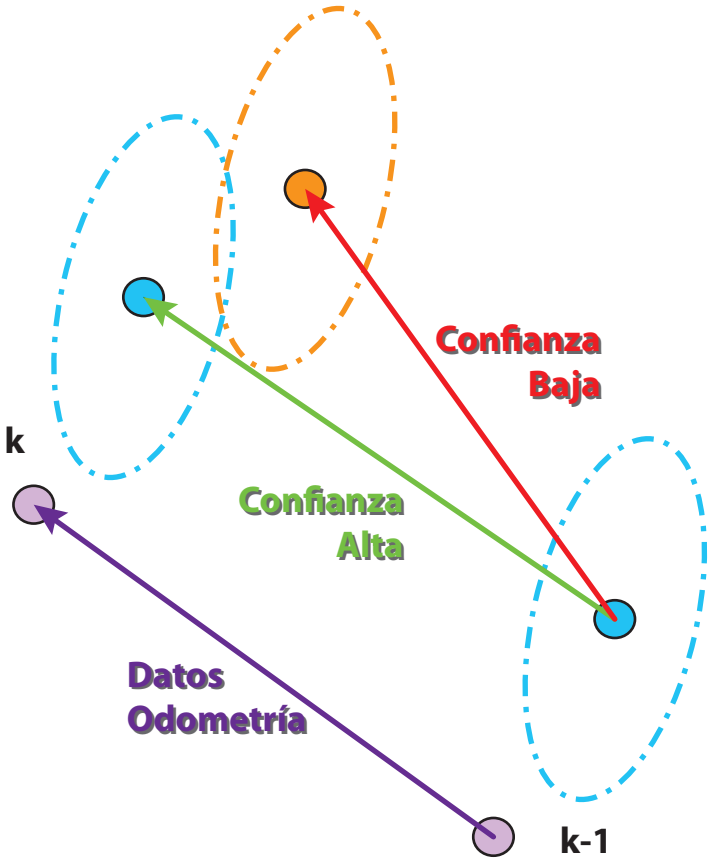
4.3.4. Confianza en el GPS

En apartados anteriores señalamos la interferencia en la señal de los satélites como principal causa de errores en la medida de los receptores GPS. El efecto de estas interferencias suele corregirse con el uso de sistemas de corrección diferencial; sin embargo, existen algunos escenarios donde las interferencias son muy cortas o están localizadas sobre un área muy pequeña. En estos casos ni siquiera la implementación de sistemas diferenciales es capaz de compensar los errores temporales, lo cual puede conducir a situaciones donde la medida del GPS venga acompañada de un valor bajo de incertidumbre, a pesar de encontrarse bastante más alejada de su verdadero valor.

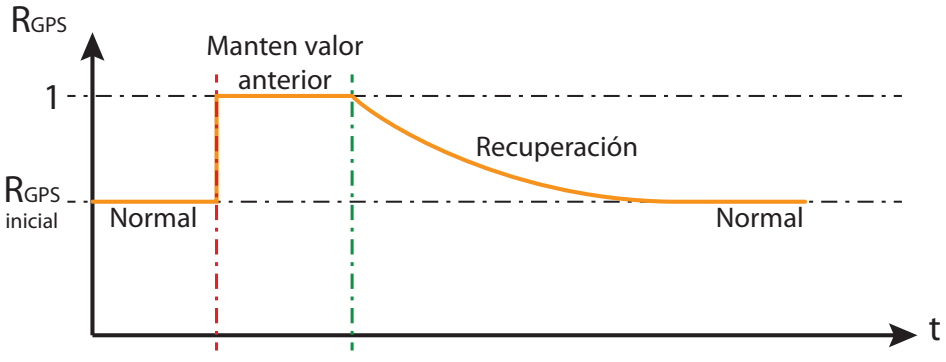
Con el objetivo de evitar que este tipo de eventos induzcan errores mayores en la estimación del algoritmo, hemos introducido el concepto de "confianza del GPS" como nueva variable de este. La idea es que ante valores muy bajos de confianza, la influencia de las medidas del GPS sea menor en la actualización de su peso auxiliar correspondiente y, por lo tanto, en la estimación final del algoritmo. Con este fin, comparamos en cada ciclo el vector de desplazamiento entre dos medidas consecutivas del GPS contra el vector desplazamiento calculado a partir de los datos de la odometría. En caso de que la diferencia entre ellos sea mayor al umbral definido para la aplicación, el valor de R_{gps} (ecuación 4.6) se establece en el máximo —1— haciendo que se mantenga el valor anterior del peso. Esta condición se mantiene hasta que se reestablece la confianza en el GPS, retornando progresivamente R_{gps} a su valor por defecto. La figura 4.7 ilustra un ejemplo de esta situación.

4.3.5. Remuestreo de partículas

De acuerdo al planteamiento del algoritmo, la idea es que las partículas evolucionen siguiendo el desplazamiento del enjambre además del suyo propio. Sin embargo, no todos los individuos se adaptarán de forma adecuada al movimiento del enjambre ni a las mediciones de los sensores; lo que significa que al cabo de varias iteraciones algunas partículas habrán perdido gran parte del valor de uno o varios de los pesos auxiliares. Esto provoca poco a poco la degeneración del enjambre, quedando tan sólo unas pocas partículas como candidatos



(a) Comparación de los vectores



(b) Evolución de R_{gps}

Figura 4.7: Cálculo de la confianza en el GPS.

posibles al estado del sistema; situación similar a la que ocurre en los filtros de partículas sin métodos de remuestreo (Doucet et al., 2000).

Para abordar este inconveniente, se ha definido un umbral mínimo para cada uno de los pesos auxiliares: U_{gps} y U_{map}, para el peso por GPS y mapa respectivamente. El valor asignado a estos umbrales nos permite identificar las partículas que deben considerarse "perdidas" en cada caso y, por ende, deben ser remuestreadas.

Dentro del algoritmo, hemos definido los métodos de remuestreo como reglas adicionales de atracción, con la diferencia de que en este caso la atracción sí modifica directamente el estado del individuo y no su vector de desplazamiento. Como se muestra en la ecuación 4.10, aquellas partículas cuyo peso por GPS sea menor a U_{gps} serán atraídas hacia la última medida del GPS $[x_{gps_k}\ y_{gps_k}]$, manteniendo su orientación actual. Esto último se hace para garantizar la varianza del enjambre en cuanto a orientación, además de evitar el uso de la orientación del GPS, la cual a baja velocidad presenta errores elevados debido a la proximidad de las muestras. De forma similar, en el caso de que el peso por mapa de una partícula sea menor a U_{map}, esta será atraída hacia el último valor estimado por el enjambre (ver ecuación 4.11).

$$\begin{bmatrix} x_k^i \\ y_k^i \\ \psi_k^i \end{bmatrix} = \frac{(U_{gps} + W^i_{gps_k}) \begin{bmatrix} x_k^i \\ y_k^i \\ \psi_k^i \end{bmatrix} + (U_{gps} - W^i_{gps_k}) \begin{bmatrix} x_k^{gps} \\ y_k^{gps} \\ \psi_k^i \end{bmatrix}}{2 \cdot U_{gps}} + \begin{bmatrix} D_x \cdot x_{rand} \\ D_y \cdot y_{rand} \\ D_\psi \cdot \psi_{rand} \end{bmatrix} \tag{4.10}$$

$$\begin{bmatrix} x_k^i \\ y_k^i \\ \psi_k^i \end{bmatrix} = \frac{(U_{map} + W^i_{map_k}) \begin{bmatrix} x_k^i \\ y_k^i \\ \psi_k^i \end{bmatrix} + (U_{map} - W^i_{map_k}) \begin{bmatrix} x_k^{sw} \\ y_k^{sw} \\ \psi_k^{sw} \end{bmatrix}}{2 \cdot U_{map}} + \begin{bmatrix} D_x \cdot x_{rand} \\ D_y \cdot y_{rand} \\ D_\psi \cdot \psi_{rand} \end{bmatrix} \tag{4.11}$$

En ambos casos, la ecuación de atracción incorpora al final un término aleatorio que dispersa las partículas. Esto se hace con el objetivo de que la atracción no las lleve a converger forzadamente en un único estado. Las constantes D_x, D_y and D_ψ especifican el rango máximo de dispersión que se aplica en cada componente del vector. $[x_{rand}, y_{rand}, z_{rand}]$ es un vector de valores aleatorios dentro del rango $[-1, 1]$, calculado para cada partícula.

4.3.6. Estimación

En la sección 4.3.2 se menciona que, al igual que el resto de partículas, el valor de la última estimación también evoluciona de acuerdo a la información proporcionada por los sensores. Sin embargo, esta estimación *a priori* se emplea únicamente como una representación del movimiento del enjambre, mas no corresponde con el valor estimado por el algoritmo en ese ciclo. Este valor se calcula al final de cada ciclo mediante la combinación de los estados *a posteriori* de todas las partículas del enjambre. Esto se realiza a través del cálculo del

promedio ponderado o centro de masas del enjambre (ecuación 4.12).

$$X_k^{sw} = \frac{\sum_{i=1}^{N} W_k^i \cdot X_k^i}{\sum_{i=1}^{N} W_k^i} \tag{4.12}$$

4.4. Validación del algoritmo

Una vez presentado tanto el desarrollo como la estructura del algoritmo, pasamos a la comprobación de su funcionamiento. Para ello se han realizado dos experimentos. El primero de ellos lo enfocamos al análisis de la influencia del número de partículas en el rendimiento del algoritmo, con el objeto de determinar cual de las configuraciones era más eficiente, considerando tanto la fiabilidad de la estimación como el tiempo de cómputo de la solución. Por su parte, en el segundo ensayo comparamos los resultados obtenidos por el algoritmo propuesto contra los resultados de un filtro de Kalman extendido, desarrollado previamente por el LIVIC (Ndjeng Ndjeng et al., 2009).

Ambos ensayos han sido realizados con datos registrados previamente, los cuales contienen información del GPS, IMU, odómetro y ángulo del volante. Los datos han sido capturados de forma sincronizada, utilizando una frecuencia de muestreo regulada por el reloj interno del GPS. Durante la captura de los datos, el GPS se configuró para trabajar en modo autonomo; es decir, sin utilizar ningún medio que permitiese aplicar la corrección diferencial a las medidas. Esto implica que durante todo el recorrido, la incertidumbre mínima de las mediciones es superior a la decena de metros.

En la figura 4.8 se muestra una imagen completa del recorrido realizado. Este inicia en la pista *Routière* y, tras recorrer aproximadamente la mitad de la misma, cambia a la pista *Val d'Or*. Una vez finalizada esta última, vuelve a la primera; recorriendo todo el tramo restante hasta alcanzar el punto final.

La principal diferencia que existe entre ambas pistas es que *Routière* se encuentra en una zona despejada, mientras que *Val d'Or* está dentro de una zona boscosa —ver figura 4.2—; por lo que en cualquier punto de ella, la visibilidad del cielo es mucho más limitada. Esta situación provoca que las medidas del GPS en esta zona tengan mayor incertidumbre asociada, como demuestran los datos del área de la elipse 1-σ asociada a cada medida del GPS —ver figura 4.9—. De igual forma, se ha visto afectado el sistema de localización utilizado para estimar la trayectoria de referencia; siendo prácticamente imposible que este funcionase en alguno de los modos RTK —*fixed* o *float*— dentro de la pista *Val d'Or* (figura 4.10).

4.4.1. Tamaño del enjambre

Considerando que, en cada ciclo, el estado estimado del vehículo se calcula como el centro de masas del enjambre; salta a la vista la idea de que, dependiendo del número de partículas que conformen el enjambre, el resultado del algoritmo será diferente. Para analizar esta situación en detalle, realizamos un experimento para observar el rendimiento del algoritmo bajo diferentes configuraciones. Durante el ensayo, se ejecutaron varias sesiones de localización,

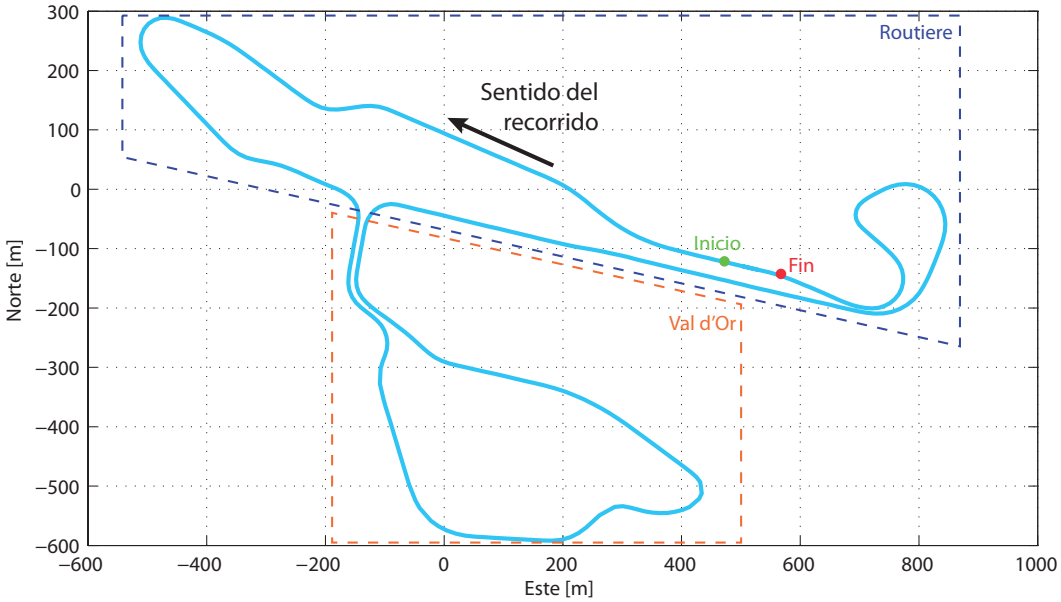


Figura 4.8: Trayectoria del recorrido realizado durante la captura de los datos.

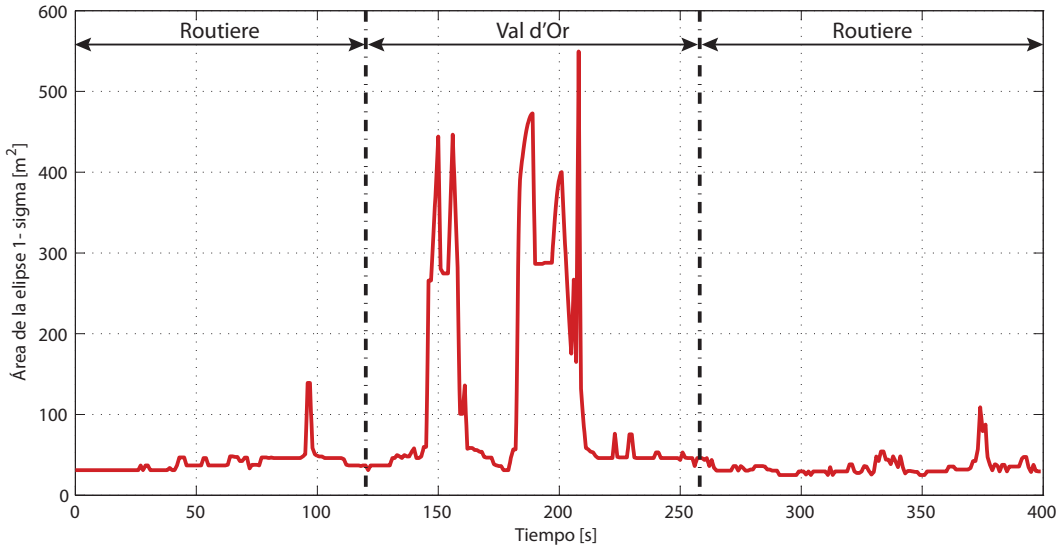


Figura 4.9: Área de la elipse de incertidumbre 1-σ asociada a cada medida del GPS.

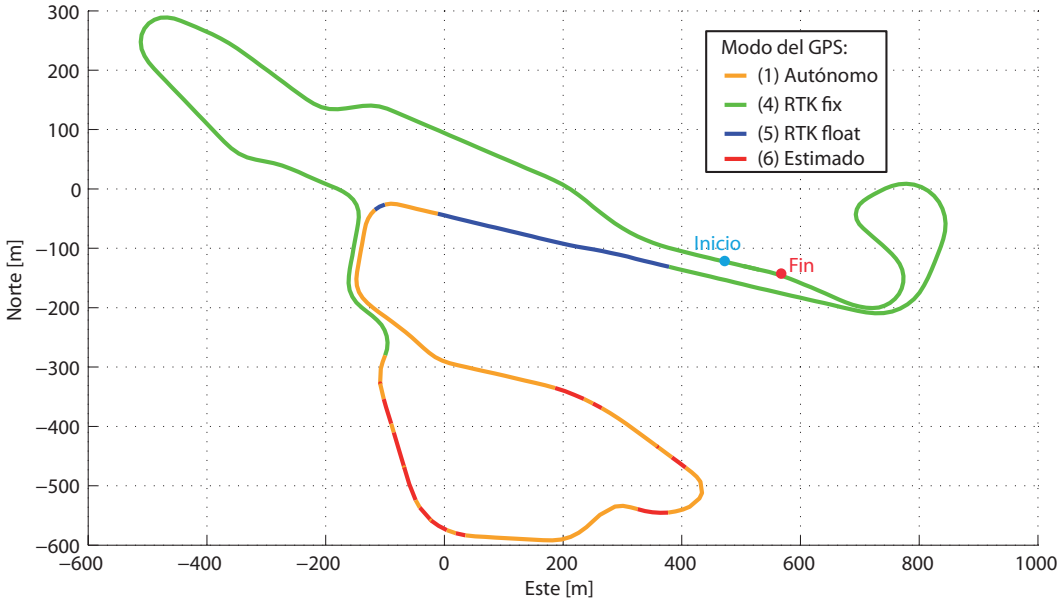


Figura 4.10: Evolución del modo de operación del GPS de referencia a lo largo del recorrido.

R_{gps}	R_{map}	U_{gps}	U_{map}	G_{gps}	G_{map}	G_{gym}
0,7	0,5	0,3	0,15	1	1	5

Tabla 4.1: Valores establecidos para las ganancias y umbrales de la localización.

utilizando tamaños de enjambre en el rango de [50, 1.000] partículas. La variación de tamaño se realizó en pasos de 50 partículas, por lo que finalmente se analizaron 20 configuraciones diferentes.

Dado a que el algoritmo presenta diversos factores aleatorios —como la inicialización y remuestreo de las partículas—, se ejecutaron 20 sesiones para cada tamaño del enjambre, considerando como resultado final el promedio de todas ellas. Como medida de comparación entre las distintas configuraciones, hemos considerado el error cuadrático medio (MSE) entre la trayectoria estimada y la trayectoria de referencia. La tabla 4.1 muestra los valores establecidos durante el experimento para las distintas ganancias y umbrales definidos en el algoritmo.

Los resultados obtenidos se muestran en la tabla 4.2 para cada configuración. Se puede apreciar que el promedio del MSE es similar para todas las configuraciones, Sin embargo, como se muestra en la figura 4.11, la desviación típica de este error disminuye a medida que se incrementa el tamaño del enjambre. Este resultado es lógico, puesto que la estimación del algoritmo se torna más estable al utilizar enjambres más grandes, donde la influencia de una partícula como individuo es menor para el conjunto y, por lo tanto, su correcta inicialización no es un factor tan crítico. En la gráfica, los valores en rojo señalan la reducción de la desviación típica con respecto al resultado obtenido para la configuración anterior —inmediatamente más pequeña—; mientras que los valores en verde indican la reducción de este valor respecto a la configuración con un enjambre de 50 partículas.

Nº partículas	50	100	150	200	250	300	350	400	450	500
MSE	5,531	5,619	5,140	5,123	5,491	5,345	5,452	4,897	5,439	5,422
Tiempo de ejecución [ms]	14,01	20,82	27,81	35,36	41,97	49,60	57,16	64,01	71,74	78,94
Nº partículas	550	600	650	700	750	800	850	900	950	1000
MSE	5,576	5,377	5,159	5,416	5,509	5,627	5,477	5,230	5,091	5,424
Tiempo de ejecución [ms]	86,53	93,70	100,09	107,26	114,22	122,12	128,85	136,81	142,87	151,01

Tabla 4.2: MSE promedio y tiempo de ejecución de acuerdo al tamaño del enjambre.

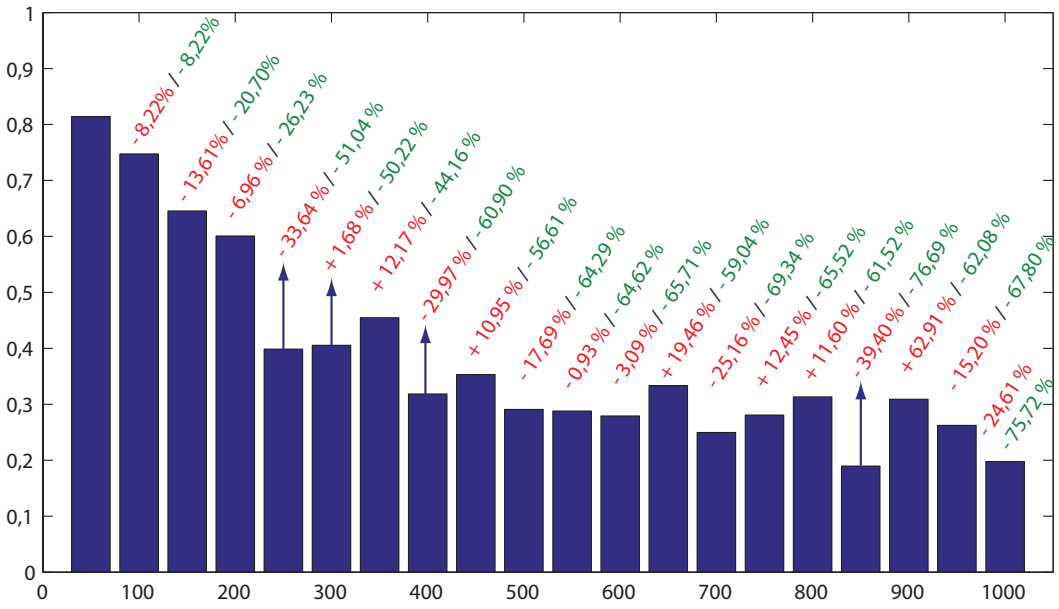


Figura 4.11: Evolución de la desviación típica para el MSE en función del número de partículas.

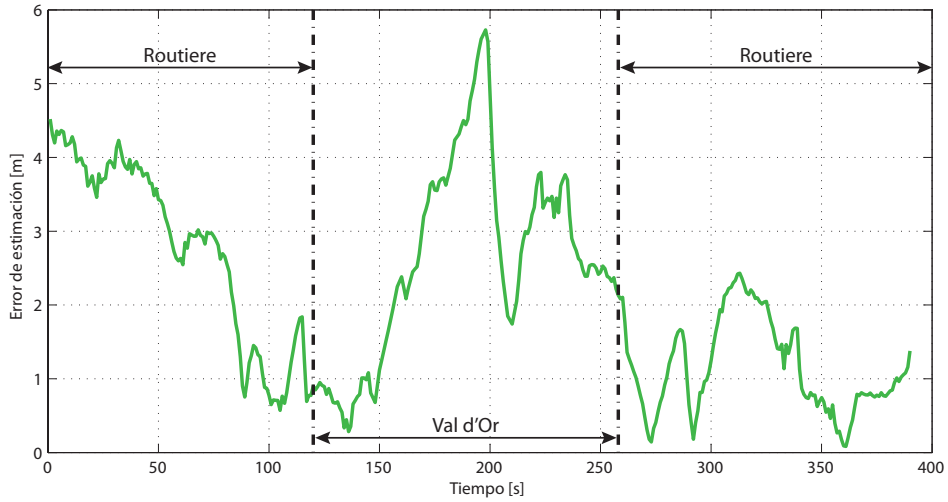


Figura 4.12: Evolución del error de estimación con un enjambre de 250 partículas.

A pesar de la estabilidad en la estimación que llegan a proporcionar los enjambres más grandes, los resultados también demuestran un incremento en el tiempo de ejecución, proporcional al tamaño del enjambre. Esto reduce la aplicabilidad de los enjambres de gran tamaño en la localización de vehículos en tiempo real. Por lo tanto, la selección del tamaño del enjambre es un compromiso entre el tiempo de ejecución y la estabilidad de la estimación. A partir de esta idea y de los resultados obtenidos, se puede apreciar que para un tamaño de enjambre entre 250 y 300 partículas, el equilibrio es adecuado. En este rango, la desviación típica ya se ha reducido al menos un 50 % respecto al enjambre de 50 partículas. De igual forma, el tiempo de ejecución no supera los 50 milisegundos —la cuarta parte del tiempo de muestreo empleado en el LIVIC para el GPS—.

En la figura 4.12 se muestra el peor resultado obtenido para una configuración con 250 individuos en el enjambre. Se puede ver en la gráfica que el error de la estimación disminuye de forma constante desde el inicio del recorrido hasta el segundo 100, permaneciendo por debajo de los 2 metros hasta el segundo 150; donde finalmente se observa un incremento considerable. Este aumento se corresponde con la zona mas boscosa de la pista *val d'Or*, donde la incertidumbre de las mediciones del GPS tuvo un aumento considerable (figura 4.9). No obstante, en esta zona la trayectoria de referencia también fue generada utilizando un estimador, lo que pudo contribuir a aumentar el error calculado. Tan pronto como el vehículo sale de esta zona y se acerca de nuevo a la pista *Routière*, los valores de incertidumbre disminuyen, lo que provoca que el error de estimación esté nuevamente por debajo de los 2 metros.

4.4.2. Comparación con EKF

Habiendo analizado la influencia del tamaño del enjambre en el rendimiento del algoritmo, se procedió a comparar el resultado de la estimación del LEP de 250 partículas contra un filtro de Kalman extendido, desarrollado por el LIVIC para el mismo escenario (Ndjeng Ndjeng et

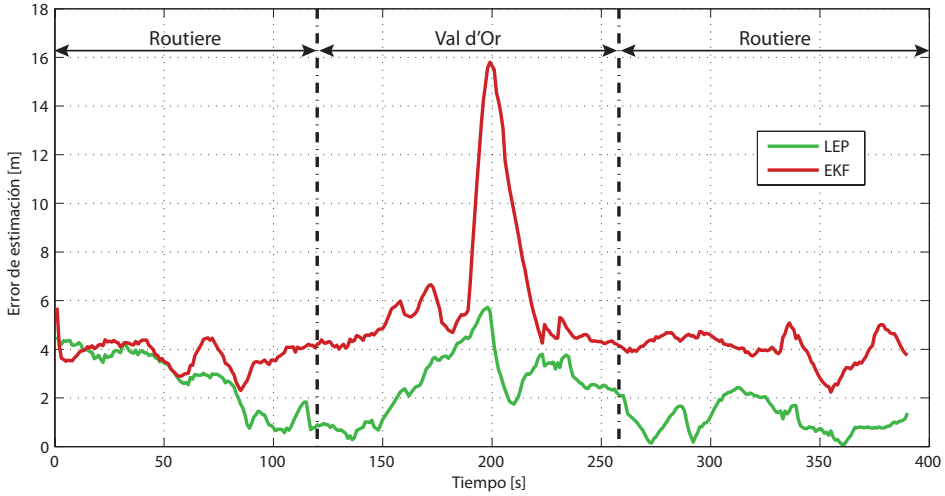


Figura 4.13: Comparación del error de estimación entre el LEP-250 y EKF.

al., 2009). Este último emplea como modelo de evolución el mismo modelo cinemático de la bicicleta utilizado por el LEP (ecuación 4.1). Sin embargo, la estimación del EKF no emplea ningún tipo de información relacionada con los mapas de las pistas, como si hace el LEP. Dadas estas diferencias, se debe resaltar que el objeto de la comparación no es determinar cual de los métodos de localización es mejor, sino demostrar que el algoritmo propuesto se encuentra al nivel de otras opciones ampliamente aceptadas.

Los resultados obtenidos por ambos enfoques se muestran en la figura 4.13. Se puede apreciar en la gráfica que la respuesta del LEP es más rápida, disminuyendo el error en los primeros 100 segundos de 4 a 2 metros aproximadamente, mientras que el error del EKF permanece más o menos estable en torno a los 4 metros. Posteriormente, al entrar en la zona de la pista *Val d'Or* ambos algoritmos presentan un notable incremento en el error de localización. Sin embargo, mientras que el LEP sólo alcanza los 6 metros de error, el EKF llega hasta los 16 metros. En este caso, el LEP es capaz de obtener una mejor respuesta debido a la inclusión de la información de los mapas, lo cual permite al algoritmo acotar el error, manteniendo siempre la estimación final dentro de la pista. Una vez que se regresa a la zona de la pista *Routière* la respuesta de ambos es similar al inicio de la estimación.

4.5. Aplicación en AUTOPIA

Los resultados presentados en la sección anterior validan el funcionamiento del algoritmo de localización propuesto para la plataforma de prueba desarrollada por el LIVIC. A continuación, para demostrar su independencia de la plataforma y aplicabilidad en otros sistemas, presentamos su implementación con la arquitectura de AUTOPIA. En este sentido, es necesario mencionar que el objetivo de esta aplicación no es mejorar la precisión en la localización de los vehículos, que cuentan con equipos DGPS con precisión centimétrica, sino disponer de un sistema que permita sortear los posibles fallos en la recepción de la señal, como puede ser su obstrucción temporal.

Antes de continuar, presentamos algunos cambios que se han realizado en el algoritmo durante el proceso de adaptación a nuestra arquitectura. En primer lugar, a diferencia de los datos utilizados en las fases de desarrollo y validación, nuestra etapa de percepción publica la información de cada uno de los sensores embarcados de forma independiente y a su máxima frecuencia de operación. Esto nos ha permitido aumentar la frecuencia de trabajo del algoritmo, ya que se emplea como base el tiempo de muestreo de la IMU, que es 10 veces mayor que el del GPS. El proceso de correlación de los datos ha sido posible gracias al enlace físico de sincronización que existe entre estos dos sensores (ver sección 3.2.1).

El segundo cambio está relacionado con el cálculo del peso de las partículas ya que, con el objetivo de no limitar la aplicabilidad del algoritmo, se tomó la decisión de no emplear —en una primera aproximación— ningún tipo de información de los mapas. Esto último ha sido posible dada la alta precisión con la que cuentan los receptores GPS, lo cual garantiza que desde un inicio todas las partículas se ubiquen dentro de una pequeña área de incertidumbre. No obstante, para compensar el vacío dejado por la supresión de ese tipo de información, hemos introducido un nuevo peso auxiliar denominado confianza. Este es proporcional al tiempo de vida de la partícula; es decir, a la cantidad de ciclos que han pasado desde su último re-muestreo.

Para poder utilizar el algoritmo en conjunto con nuestra arquitectura, se ha desarrollado un módulo de software independiente, dedicado únicamente a su ejecución. A diferencia de la versión de desarrollo, este ha sido programado en C++, optimizando al máximo su funcionamiento a fin de reducir el tiempo de ejecución al mínimo posible. Tras su inicio, la primera tarea del módulo consiste en reservar dinámicamente el espacio de memoria necesario para almacenar todo el enjambre. Acto seguido, el software se suscribe a los canales LCM que trasmiten la información del GPS, IMU y CAN del coche. Finalmente, la inicialización del algoritmo se pospone hasta el momento en que la velocidad del vehículo ha superado los 5 km/h. Esto se hace con el objetivo de reducir la incertidumbre inicial en la estimación de la orientación del coche, calculada por el receptor GPS a partir de mediciones consecutivas.

Una vez que se ha inicializado, el funcionamiento del software se rige por el diagrama de bloques presentado en la figura 4.14. Dependiendo del tipo de mensaje que se recibe —GPS, IMU o CAN— el algoritmo toma uno de los dos caminos principales: almacenar los datos a la espera del siguiente mensaje o ejecutar un ciclo de estimación. Idealmente, este último caso sólo se consigue cuando se recibe un mensaje de la IMU. Sin embargo, debido a la latencia en la lectura de los datos, existe un caso especial en el que se provoca el desencadenamiento de este bucle a partir de la recepción de un mensaje GPS. Para facilitar la comprensión del diagrama, se explicará este caso único más adelante.

En cada ciclo de estimación e inmediatamente después de predecir el nuevo estado de las partículas, el programa verifica si el tiempo asociado al mensaje de la IMU coincide con un ciclo de GPS. Si este es el caso, se realiza la actualización de los pesos por GPS y remuestreo de las partículas a partir de la información recibida para el GPS en este ciclo. Finalmente, se realiza la estimación del nuevo estado del vehículo y la actualización de la confianza de la partícula. El resultado final se publica a través del canal `filtro_lep` utilizando la estructura

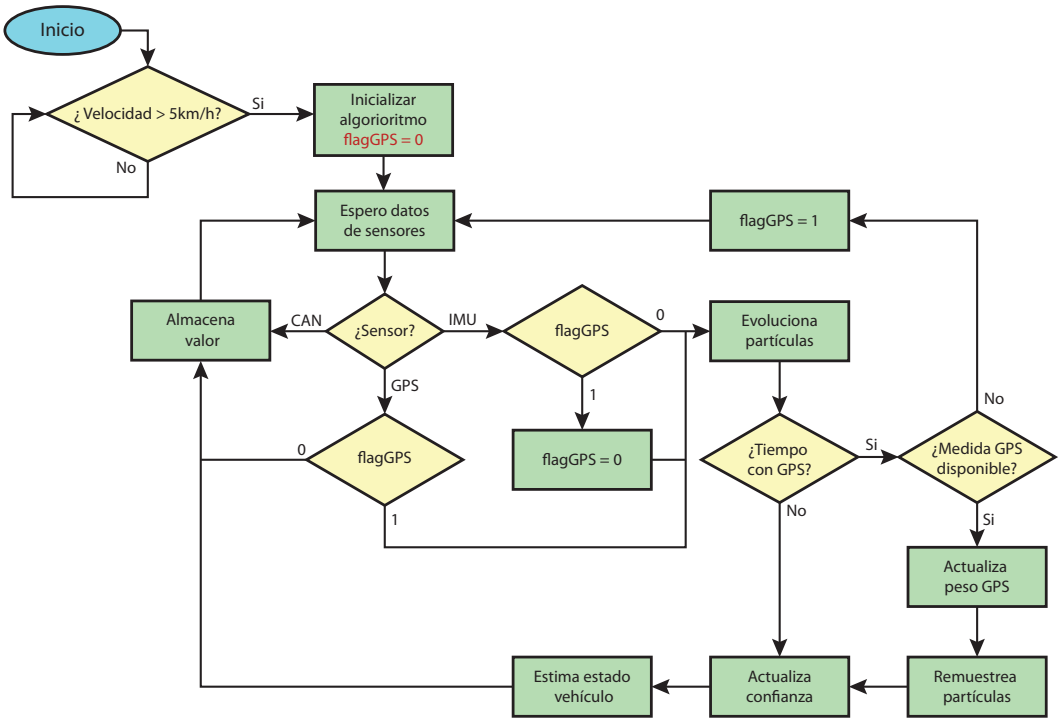


Figura 4.14: Diagrama del programa para la implementación del LEP en la arquitectura de AUTOPIA.

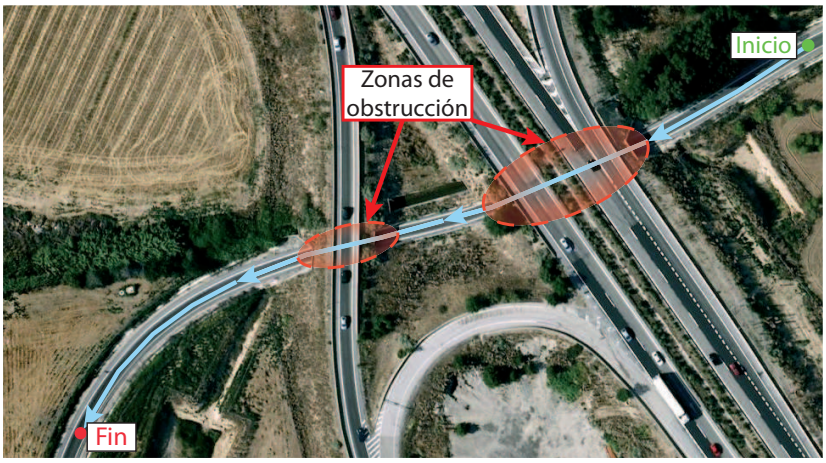
de datos presentada en la figura 4.15.

El caso especial que se mencionó anteriormente ocurre cuando el tiempo asociado a la IMU coincide con un ciclo de GPS —que como sabemos debe ocurrir por la sincronización existente entre ambos— pero la información de este último no ha sido recibida aún. Ante esta situación, el programa establece la bandera `flagGPS` al valor 1 indicando que la estimación está a la espera del mensaje del GPS. A partir de este momento pueden ocurrir dos cosas: si el siguiente mensaje recibido proviene del GPS —omitiendo los del CAN—, se reanuda la estimación en el punto en el que se había pausado anteriormente. Sin embargo, si se recibe antes otro mensaje de la IMU, el programa da por hecho que se ha perdido un mensaje del GPS, reanudando la estimación como si de un ciclo sin GPS se tratase. Ha sido necesario tomar

```
struct filtro_lep
{
    int32_t    Timestamp_sec;   // Tiempo de recepción
    int32_t    Timestamp_nsec;  // Tiempo de recepción
    double     IMUTime;         // Tiempo IMU
    double     UTMEste;         // Valor filtrado
    double     UTMNorte;        // Valor filtrado
    double     Orientacion;     // Valor filtrado
    double     Velocidad;       // Valor del CAN
    int8_t     Calidad;         // Trama GPGGA (GPS)
    int8_t     NroSat;          // Trama GPGGA (GPS)
}
```

Figura 4.15: Definición de la estructura de datos `filtro_lep` en pseudocódigo LCM.

(a) Recorrido urbano.



(b) Recorrido con túneles.

Figura 4.16: Recorridos utilizados para validar el funcionamiento del LEP en la arquitectura de AUTOPIA.

en cuenta esta situación debido a que el receptor GPS embarcado reduce automáticamente su frecuencia de operación, de 10 a 5 Hz, cuando pierde totalmente la señal de los satélites (modo 0 de operación).

Para validar el funcionamiento de este componente en escenarios reales, realizamos tres pruebas en situaciones que afectaban el rendimiento del GPS. La primera de ellas corresponde a un escenario urbano, donde los árboles y edificios obstruían parcialmente las señales de los satélites —figura 4.16a—. Por su parte, las otras dos pruebas se realizaron en un camino interurbano, en el que dos pasos elevados obstruían temporalmente la vista del cielo del receptor —figura 4.16b—.

En la figura 4.17 se muestra el resultado obtenido para el primer escenario, enfocándonos únicamente sobre la zona de obstrucción. En la imagen, la línea negra punteada representa la estimación del algoritmo, mientras que las líneas continuas representan los datos crudos obtenidos del GPS, utilizando un color distinto para cada modo de operación de este último.

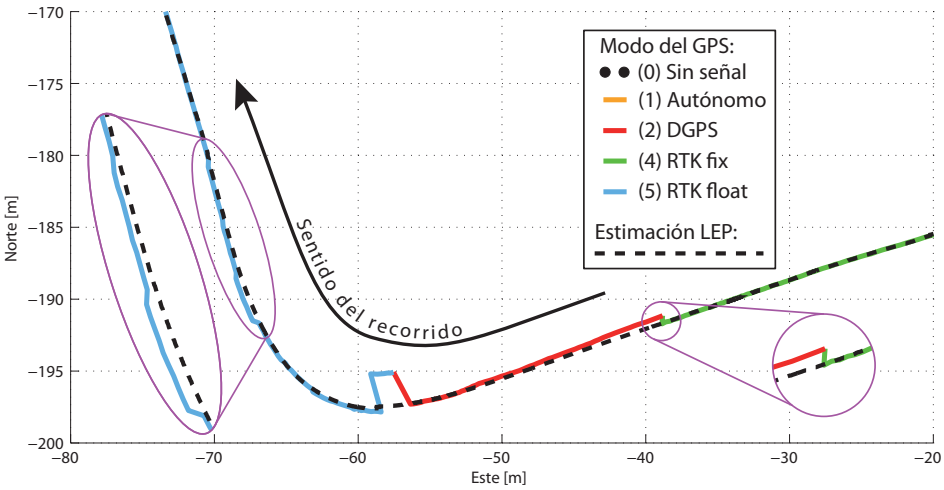


Figura 4.17: Resultados del algoritmo LEP ante una obstrucción parcial de los satélites.

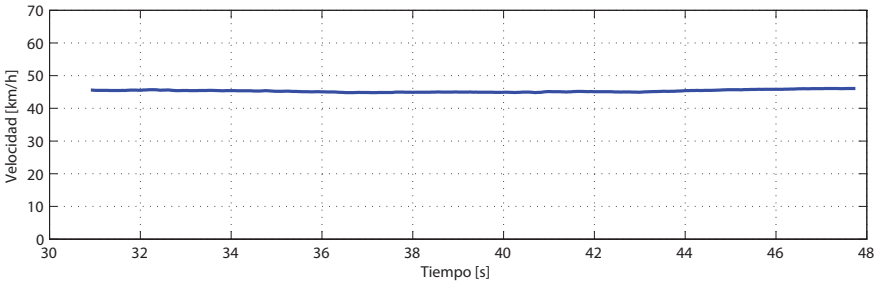
Se puede apreciar en la gráfica que, al entrar en la zona, los obstáculos afectan la recepción de las señales de los satélites, provocando que el receptor GPS cambie del modo de operación RTK *fix* —modo 4— al modo de operación DGPS —modo 2—, en el que la precisión de la medida es menor. No obstante, en ningún momento ocurren vacíos totales de mediciones —modo 0—.

Continuando con el recorrido podemos ver que, justo antes de cambiar al modo RTK *float* —modo 5—, se produce un desplazamiento de aproximadamente tres metros hacia la derecha de la trayectoria, ocurriendo a su vez el desplazamiento opuesto luego de un par de ciclos. Además de este salto, se observan otros más pequeños en las zonas ampliadas. Para todos estos casos, y en general para toda la zona de obstrucción, podemos apreciar que el algoritmo ha conseguido generar una trayectoria mucho más suave al combinar las medidas del GPS con la información de la IMU y del CAN del coche. Esta mejora es muy importante para el sistema de control del vehículo, ya que las variaciones radicales en la trayectoria se reflejan directamente como cambios bruscos en las acciones de control del volante (Milanés, 2010).

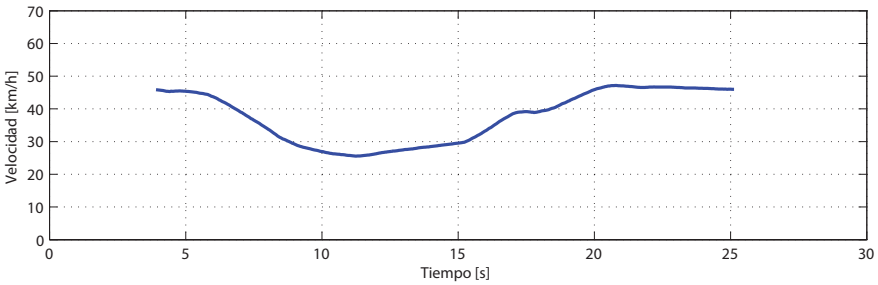
En cuanto al escenario interurbano, realizamos dos recorridos diferentes con el objetivo de comparar la respuesta del algoritmo ante variaciones de la velocidad del vehículo durante los períodos de obstrucción. Como se puede apreciar en las gráficas de la figura 4.18, la velocidad del vehículo se mantuvo en torno a los 45 km/h durante el primer recorrido; mientras que para el segundo, se realizó una variación de 45 a 25 km/h a la entrada del primer túnel, acelerando nuevamente hasta los 45 km/h a la salida del mismo.

Como demuestran las gráficas presentadas en la figura 4.19, el resultado obtenido es similar para ambos casos; con la excepción de que durante el segundo recorrido las mediciones del GPS se interrumpieron totalmente en dos oportunidades. Esto es producto de la variación de velocidad en el interior del túnel, lo que provocó que el recorrido fuese más lento y, por lo tanto, más larga la duración de la obstrucción.

Para estas pruebas, el punto más crítico lo representa el primer túnel, el cual tiene una



(a) Recorrido 1.

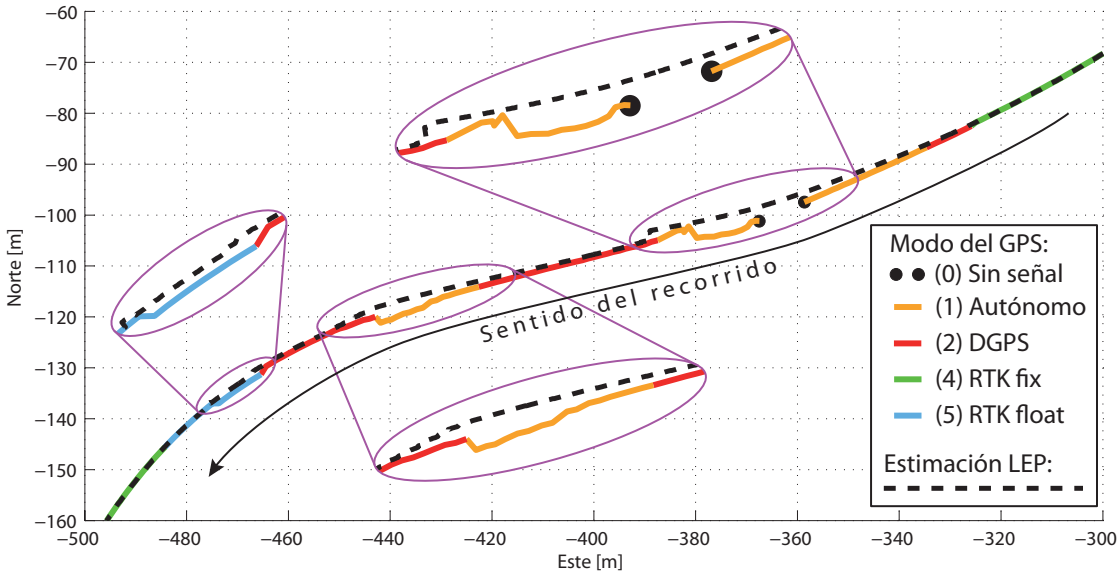


(b) Recorrido 2.

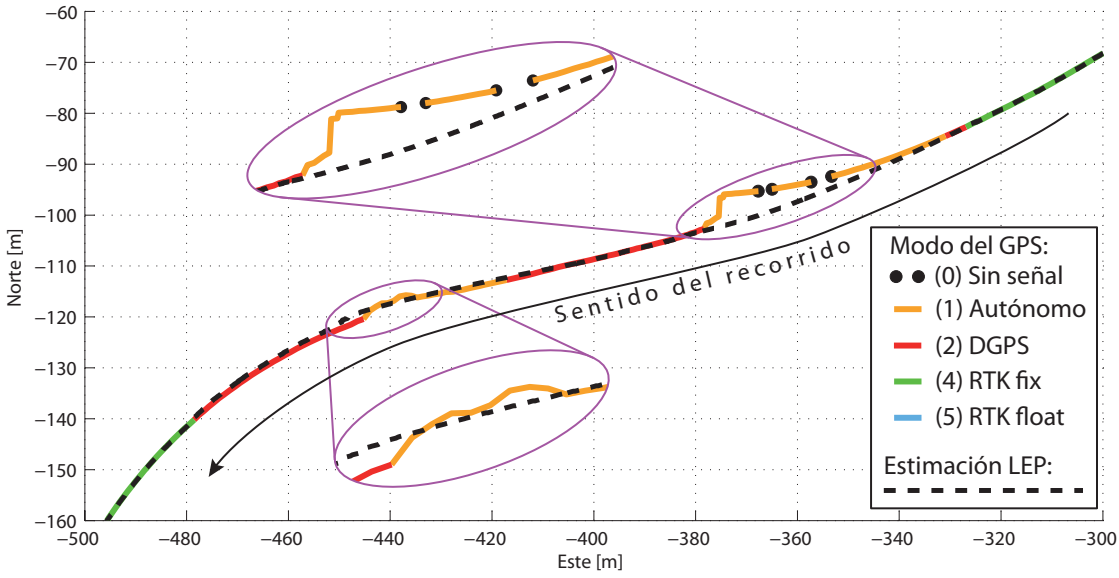
Figura 4.18: Velocidad durante los recorridos interurbanos.

longitud en torno a los 50 metros y genera una obstrucción de las señales durante un mínimo de 4 segundos. Durante este período el modo de operación del GPS no sólo disminuye, sino que también se pierde totalmente su capacidad de medida. Por otro lado, podemos observar que a la salida de este túnel es donde la desviación de las medidas alcanza su máximo valor, en torno a los 4 y 6 metros para el primer y segundo recorrido respectivamente. No obstante, para los dos recorridos el algoritmo LEP ha conseguido sortear todos estos inconvenientes, generando una trayectoria mucho más suave y acorde con el recorrido real del vehículo.

El conjunto de resultados que hemos obtenido con la implementación de este algoritmo en la arquitectura de AUTOPÍA es muy alentador; demostrando que el método desarrollado es adecuado para su aplicación en este tipo de escenarios donde la presencia de obstáculos en el entorno producen pérdidas considerables en la señal de los satélites. En cuanto al algoritmo en sí, consideramos que es posible mejorar aún más la calidad de su estimación incluyendo información de sensores adicionales en casos como estos, donde la información de los mapas digitales no se encuentra disponible. Por ejemplo, a través de la detección de líneas en la calzada.



(a) Manteniendo velocidad constante



(b) Con variación de velocidad en el interior del túnel.

Figura 4.19: Resultados del algoritmo ante una obstrucción total de los satélites.

CAPÍTULO 5

CONTROL BASADO EN APRENDIZAJE Y ADAPTACIÓN

Hoy en día, existen diversas técnicas que facilitan el diseño de controladores para casos complejos —como la conducción autónoma de un vehículo—. No obstante, gran parte de estas se basan en la existencia de un modelo preciso del sistema; lo cual, en algunos casos, es un requisito que no es posible satisfacer dada la alta complejidad asociada con la identificación del sistema. En este sentido, las técnicas de inteligencia artificial han abierto el camino al desarrollo de controladores a través de métodos de aprendizaje y/o adaptación que aprovechan el conocimiento y la experiencia humana —e.g. aprendizaje por imitación (Nehaniv y Dautenhahn, 2007) y aprendizaje supervisado (Mohri et al., 2012)—, o bien son capaces de generar su propia base de conocimiento a partir de ciertos parámetros de recompensa establecidos por el usuario —e.g. aprendizaje no supervisado (Hinton y Sejnowski, 1999) y aprendizaje por refuerzo (RL por sus siglas en inglés) (Sutton y Barto, 1998)—. En este capítulo trataremos dos enfoques de aprendizaje que fueron aplicados, durante el desarrollo de la presente tesis doctoral, al control lateral y longitudinal del vehículo respectivamente. En ambos casos, el objetivo ha sido el evaluar la capacidad de estas técnicas para mejorar la respuesta del vehículo ante las posibles incertidumbres que puedan afectarlo; como son el ruido en las mediciones de los sensores o la falta de un modelo preciso.

Este capítulo se divide en dos partes. En la primera se presenta el enfoque empleado para el control lateral, el cual se basa enteramente en el aprendizaje por refuerzo y permite determinar, sin ningún tipo de conocimiento previo, cual de las acciones disponibles sobre la dirección es la más adecuada de acuerdo al estado del vehículo. En la segunda parte, se presenta un método de aprendizaje y adaptación para controladores borrosos, el cual se ha aplicado al control longitudinal del vehículo. Para ambos enfoques, la fase de diseño y desarrollo se ha realizado en simulación. En el caso del control longitudinal se ha incluido además una fase de experimentación utilizando un vehículo real.

5.1. Control lateral y aprendizaje por refuerzo

De acuerdo a (Sutton y Barto, 1998), las técnicas modernas de RL tienen su origen en la combinación de tres ramas que fueron consideradas independientes hasta inicios de los años 80: el aprendizaje por ensayo y error, basado en el análisis del aprendizaje de los animales; el control óptimo y su solución a través de funciones de valor y programaciones dinámica; y, por último, los métodos de diferencia temporal, que consideran la correlación entre predicciones subsecuentes. De las tres ramas mencionadas, la más característica es la del ensayo y error, proveniente del concepto de "ley del efecto" manejado en el estudio del comportamiento animal (Sutton y Barto, 1998; Thorndike, 1911). De acuerdo a esta ley, de entre todas las posibles respuestas ante una determinada situación, aquellas que generen la satisfacción del animal serán mayormente vinculadas a la situación, y por lo tanto tenderán a repetirse más que aquellas que, por el contrario, generan insatisfacción o molestias en el animal.

Para hacernos una visión más clara de esta idea, supongamos que colocamos a una persona frente a un tablero con 2 botones, uno blanco y otro negro, y le indicamos que debe presionar uno. Sin embargo, no le explicamos que al presionar el botón blanco, la silla donde está sentado le dará un breve masaje —refuerzo positivo—; mientras que si presiona el botón negro, recibirá una descarga eléctrica nada confortable —refuerzo negativo—. Al cabo de un par de interacciones con el tablero, es de esperar que la persona haya entendido el comportamiento del mismo, tendiendo a presionar únicamente el botón blanco. Dicho esto, el concepto de aprendizaje por refuerzo dentro de la inteligencia artificial se refiere al traslado de esta conducta al desarrollo de sistemas que sean capaces de aprender a partir de la realimentación que reciben del entorno.

La figura 5.1 muestra un esquema del modelo que representa este enfoque, siendo el agente el equivalente al animal que está aprendiendo. En cada instante de tiempo, el agente recibe información sobre el estado del entorno. A partir de esta, el agente selecciona una acción a ejecutar, la cual modifica el estado del entorno y finalmente provoca una determinada recompensa para el agente. Así podemos resaltar entonces que el problema de RL se compone de los siguientes elementos: un conjunto S de posibles estados s del entorno, un conjunto A de posibles acciones a y un conjunto R de recompensas r (Kaelbling et al., 1996). No obstante, es necesario resaltar que, a pesar de la recompensa inmediata r, el verdadero objetivo de esta técnica consiste en encontrar una política o táctica de actuación que permita maximizar la recompensa final r_f.

Actualmente, existen numerosos algoritmos presentados como solución al problema del aprendizaje por refuerzo. Sin embargo, la comparación de estos métodos excede el alcance del presente trabajo de tesis doctoral, donde únicamente se plantea la aplicación de esta técnica al control lateral de un vehículo. Para alcanzar este objetivo, se ha seleccionado uno de los métodos más conocidos y aplicados para RL: el *Q-learning* (Watkins y Dayan, 1992). Se puede encontrar un análisis más detallado de otros métodos de RL en (Sutton y Barto, 1998).

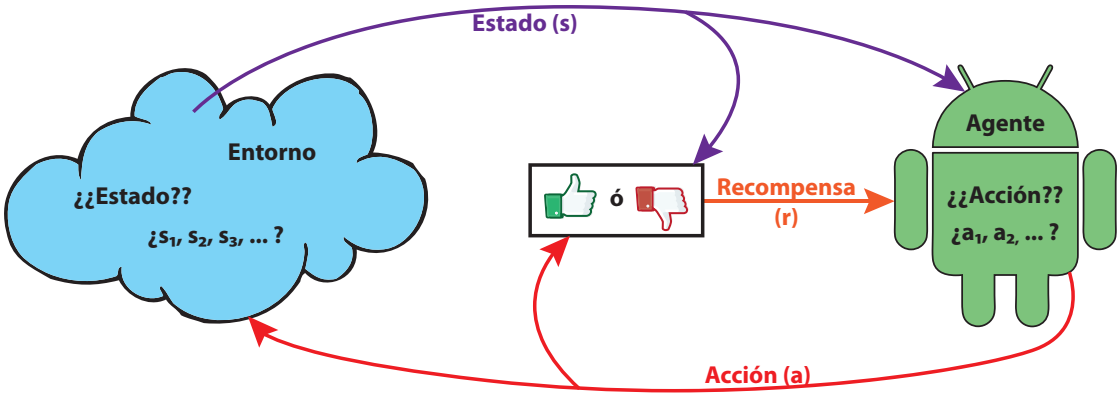


Figura 5.1: Esquema del modelo de aprendizaje por refuerzo.

5.1.1. *Q-learning*

Q-learning es un método de aprendizaje por refuerzo que se basa en el cálculo de un valor de calidad Q asociado a las acciones disponibles en un determinado estado s (Watkins y Dayan, 1992). Esta estimación de calidad se realiza de forma iterativa; es decir, actualizando el valor de $Q(s, a)$ cada vez que se selecciona la acción a en dicho estado. La actualización se realiza a través de la siguiente ecuación:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \cdot \left(r_t + \gamma \max_{a_{t+1}} (Q(s_{t+1}, a_{t+1})) - Q(s_t, a_t) \right) \tag{5.1}$$

donde s_t y a_t son el estado y la acción tomada en el instante t y r_t la recompensa obtenida por dicha acción. La variable α es un parámetro conocido como la tasa de aprendizaje, el cual se refiere a la rapidez con la que el algoritmo ajusta su conocimiento en función de la experiencia. Su valor oscila entre 0 y 1, siendo el valor proporcional a la rapidez deseada. Por su parte, γ es el factor de descuento del algoritmo, que establece la prioridad de la recompensa a largo plazo sobre la recompensa inmediata. Su valor está comprendido entre 0 y 1, donde 1 representa la mayor prioridad a la recompensa a largo plazo.

Una vez que la función o tabla Q converge a su valor final —hecho que ha sido demostrado en (Watkins y Dayan, 1992)—, se puede obtener la política óptima para el sistema seleccionando en cada estado la acción con el valor máximo de Q. Esta política está reflejada además en la ecuación 5.1 a través del término $\max_{a_{t+1}}(Q(s_{t+1}, a_{t+1}))$.

La principal ventaja de este enfoque de RL es que no depende de la definición de un modelo de evolución o de una política de control previamente definida por el usuario, sino que se basa directamente en la interacción del sistema con su entorno y de la recompensa recibida en cada instante (Lauer, 2011). No obstante, esta característica es a su vez una de las debilidades del algoritmo, ya que la experiencia del sistema se limita únicamente a aquellas situaciones que han sido exploradas previamente, careciendo de habilidad para afrontar nuevos escenarios aprovechando la experiencia aprendida ante situaciones similares. Por otro lado, el algoritmo ha sido pensado para problemas con espacios de estados finitos —discretos—, por lo que su aplicación a problemas con variables de entrada continuas requiere la partición apropiada

de las mismas. Sin embargo, es imposible conocer de primera mano el número de divisiones adecuadas para cada variable, lo que puede ocasionar la generación de espacios de estados tan grandes que la solución del problema se torna imposible, tanto por el tiempo de ejecución para resolver el mismo como por la memoria computacional necesaria.

Como solución a estos inconvenientes, se han planteado distintos métodos de generalización que permiten reducir la dimensión del problema y/o aprovechar el conocimiento adquirido en situaciones similares (Kaelbling et al., 1996). Dependiendo de su objetivo, estos se pueden clasificar en dos grupos: generalización de la entrada —estado del sistema— y generalización de la salida —valor de la función Q—. Un ejemplo del primer tipo es el enfoque utilizado por (DeLooze y Viner, 2009) en su aplicación de RL al juego de *Ms. Pac-Man*. En dicho trabajo, se empleó un método de agregación por lógica borrosa, en la que el rango continuo de posibles valores para las variables de entrada se dividió a través de su *fuzzificacion* a conjuntos borrosos. De esta forma, se transformó el espacio continuo en un espacio de tan sólo 27 estados borrosos. Por su parte, el empleo de redes neuronales para la generalización de la función Q es uno de los ejemplos más conocidos del segundo caso (Kaelbling et al., 1996). El objetivo de esta técnica consiste en aproximar el valor de esta función empleando una parte previamente explorada del espacio de estados. No obstante, la convergencia de este método no ha sido demostrada, por lo que es posible encontrar en la literatura distintas posiciones a favor (Lauer, 2011) y en contra de la misma (Boyan y Moore, 1995).

Para nuestra aplicación en particular, hemos optado por la generalización de la entrada. Para ello seleccionamos la técnica de agregación por lógica borrosa debido a su similitud con los controladores borrosos previamente desarrollados por AUTOPIA. En la siguiente sección presentamos una descripción detallada de nuestro escenario de aplicación.

5.1.2. Descripción del problema

La aplicación de controladores borrosos al problema del control lateral del vehículo es una tarea que ha sido abordada previamente por AUTOPIA utilizando distintos enfoques. En el primero de ellos (Naranjo et al., 2005; Naranjo, 2005), se empleó una combinación de dos controladores —uno para tramos rectos y otro para tramos curvos— que consideraban como variables de entrada el error lateral y angular del vehículo respecto a la trayectoria de referencia. Estos controladores fueron sintonizados a partir de la experiencia humana y del ensayo y error; presentando resultados adecuados para el control de una furgoneta eléctrica a velocidades inferiores a los 15 km/h. Posteriormente, (Onieva, 2011) empleó un método basado en algoritmos genéticos para la sintonización de los controladores. Con esto se aprovechó la experiencia de conductores humanos al capturar sus datos de conducción y utilizarlos como base de conocimiento durante la etapa de optimización; obteniendo finalmente controladores que emulaban la conducción humana. Por último, (Pérez, 2012) desarrolló un único controlador borroso, capaz de conducir en tramos rectos y curvos, eliminando la necesidad de diferenciarlos. Para ello implementó una arquitectura de control en cascada que incluía la velocidad de giro del volante como variable de control en el bajo nivel y, en el alto nivel, la velocidad del vehículo y la distancia a la siguiente curva.

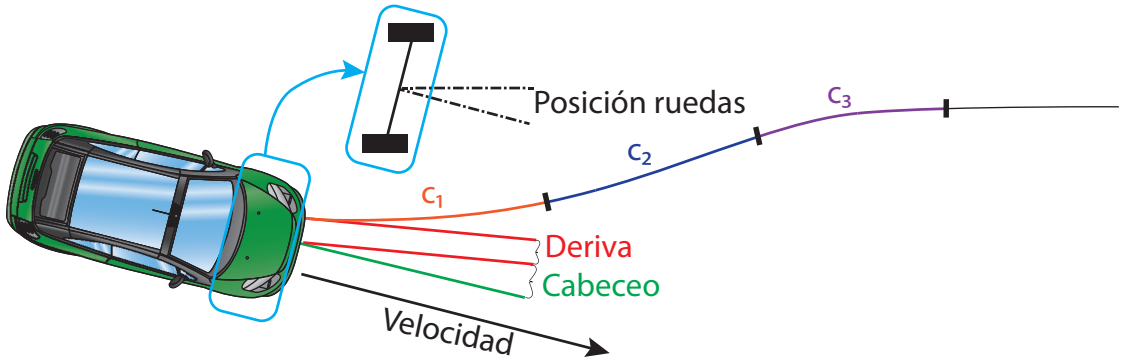


Figura 5.2: Variables para el control lateral.

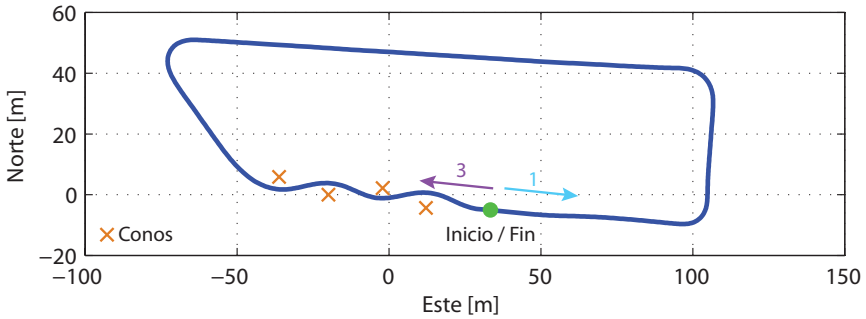
En este trabajo, planteamos la utilización de las técnicas de aprendizaje por refuerzo para el control lateral del vehículo. El objetivo es generar un controlador adecuado sin utilizar ningún tipo de experiencia de un conductor humano, bien sea directamente como base de conocimiento para el controlador o, indirectamente, en el ajuste manual del mismo. De igual forma, se busca analizar la contribución de otras variables para la mejora del control lateral del vehículo.

La figura 5.2 muestra una imagen del problema de control lateral y de las variables que han sido utilizadas. En primer lugar tenemos las variables comunes a los trabajos previos: la deriva, o error lateral; y el cabeceo, o error angular. Ambas calculadas respecto a la trayectoria de referencia utilizando el punto delantero del eje central del vehículo. Por otro lado, encontramos la velocidad del vehículo, considerada por el enfoque más reciente de AUTOPIA (Pérez, 2012). Finalmente, tenemos 2 variables que no han sido consideradas hasta ahora: la posición de las ruedas y la curvatura de la trayectoria de referencia c_i. En el caso de esta última, se calcula su valor promedio dentro de un tramo de longitud l_c inmediatamente siguiente a la posición del vehículo, pudiéndose utilizar varios tramos ($c1$, $c2$ y $c3$) para aumentar la previsión del vehículo ante cambios bruscos en la referencia.

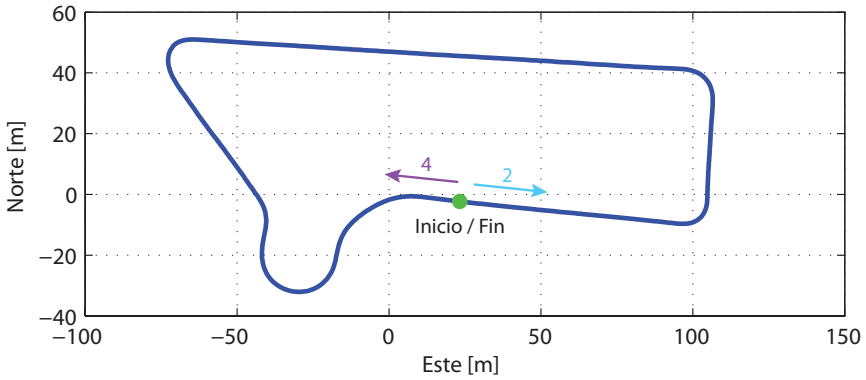
A simple vista podría parecer que, con un máximo de 5 variables de entrada, las dimensiones del problema no son muy complejas para la aplicación del aprendizaje por refuerzo sin ningún método de agregación. Sin embargo, tan sólo realizar una discretización tosca del 5 % sobre el rango de cada variable —20 segmentos— resultaría en un total de $3{,}2 \cdot 10^6$ estados posibles; requiriendo una tabla Q de $9{,}6 \cdot 10^6$ casillas para un mínimo de 3 acciones por estado. De utilizar campos *floats* para esta tabla, necesitaríamos una capacidad de 38,4 gigabytes para su almacenamiento, lo cual es inviable.

5.1.3. Implementación

Dada la complejidad y alto tiempo de ejecución que suponen los experimentos de aprendizaje en general, hemos decidido emplear un entorno simulado para el desarrollo y validación de esta aplicación de RL. La totalidad de la programación se ha realizado utilizando MatLab, con el fin de aprovechar las herramientas disponibles para la fuzzificación de variables, ela-



(a) Recorridos 1 y 3.



(b) Recorridos 2 y 4.

Figura 5.3: Recorridos utilizados para el aprendizaje.

boración de gráficos y manejo e indexación de tablas de datos.

Como trayectorias de referencia para el aprendizaje, empleamos dos recorridos realizados con un vehículo real en las instalaciones del CAR. Para el primero de ellos —figura 5.3a— se inicia el recorrido en un tramo recto de la pista y luego continuamos con una sucesión de 4 giros a la izquierda; los dos primeros de 90º, seguidos de uno de 120º y un giro final de 60º que nos permitió volver al mismo tramo recto donde se inició el recorrido. En esta última recta realizamos además un zigzag entre 4 conos, para llegar finalmente al punto de inicio. Para la segunda trayectoria —figura 5.3b—, se comienza de forma similar, manteniendo los tres primeros giros. Sin embargo, en el cuarto giro en vez de volver al tramo inicial nos incorporamos, con un leve giro a la derecha, a la rotonda del circuito. En esta realizamos la vuelta casi entera, abandonándola en la salida anterior a la entrada para llegar de nuevo al punto de inicio.

Durante el aprendizaje, ambas trayectorias de referencia se utilizarán en los dos sentidos posibles, evitando así que el vehículo realice únicamente giros a la izquierda. A continuación se describe el modelo empleado para simular el movimiento del vehículo y el algoritmo de aprendizaje.

a_2	0,1283	a_0	$-6{,}346 \cdot 10^{-5}$	b_1	0,8614	c_0	0,6321
a_1	$-0{,}116$	b_2	$-1{,}849$	b_0	$-3{,}604 \cdot 10^{-19}$	d_0	$-0{,}3679$

Tabla 5.1: Variables de los modelos discretizados.

5.1.3.1. Modelo de evolución

Para emular el comportamiento real de un vehículo, se ha empleado un modelo aproximado de uno de los prototipos del grupo. Este se ha estimado a partir de una identificación realizada como parte de trabajos previos (de Torres et al., 2010; Tejado et al., 2011), siendo adecuado únicamente para dinámicas lentamente cambiantes y superficies planas. Como es usual, el modelo empleado está desacoplado, describiendo de forma independiente el comportamiento longitudinal y lateral del vehículo. La componente longitudinal está representada por la siguiente ecuación:

$$G_{long}(s) = \frac{V(s)}{U(s)} = \frac{K\omega_n^2}{s^2 + 2\eta\omega_n s + \omega_n^2} \tag{5.2}$$

donde $V(s)$ es la velocidad del vehículo y $U(s)$ es la variable de entrada del sistema, es decir la acción de aceleración o frenado. Los valores de las constantes son $K = 25{,}14$, $\eta = 160$ y $\omega_n = 55{,}87$. Para este caso, como el objetivo de la aplicación está relacionado únicamente con el control lateral, se ha añadido un controlador PI para regular la velocidad del vehículo automáticamente de acuerdo a la referencia dada. Así, la función de transferencia final es:

$$G_{long}(s) = \frac{V(s)}{V^r(s)} = \frac{K\omega_n^2(K_P s + K_I)}{s^3 + 2\eta\omega_n s^2 + \omega_n^2 s} \tag{5.3}$$

con $K_P = 0{,}3$, $K_I = 0{,}1$ y $V^r(s)$ el valor de referencia para la velocidad. Por su parte, el comportamiento de la dirección se describe a través de la siguiente función de transferencia:

$$G_{lat}(s) = \frac{\phi(s)}{\phi^r(s)} = \frac{1}{0{,}1s + 1} \tag{5.4}$$

donde $\phi(s)$ es la posición angular de las ruedas delanteras y $\phi^r(s)$ el valor de la referencia de posición.

Ambas funciones de transferencia —ecuaciones 5.3 y 5.4— se llevan al ámbito discreto aplicando la transformada bilineal $s \to \frac{2}{T}\frac{z-1}{z+1}$, resultando en:

$$H_{long}(z) = \frac{a_2 z^2 + a_1 z + a_0}{z^3 + b_2 z^2 + b_1 z + b_0} \tag{5.5}$$

para la componente longitudinal, y

$$H_{lat}(z) = \frac{c_0}{z + d_0} \tag{5.6}$$

para la componente lateral. Los valores de las constantes se muestran en la tabla 5.1.

Finalmente, las funciones de transferencia discretizadas —ecuaciones 5.5 y 5.6— se reescriben de forma lineal, quedando:

$$\begin{aligned} V_k =& a_2 V_{k-1}^r + a_1 V_{k-2}^r + a_0 V_{k-3}^r \\ & - b_2 V_{k-1} - b_1 V_{k-2} - b_0 V_{k-3} \end{aligned} \tag{5.7}$$

para la evolución de la velocidad del vehículo y,

$$\phi_k = c_0 \phi_{k-1}^r - d_0 \phi_{k-1} \tag{5.8}$$

para la evolución de la posición de las ruedas. En la ecuación 5.7 los términos V_k y V_k^r se refieren respectivamente a la velocidad del vehículo y la referencia de la velocidad en el instante k . De igual forma, los términos ϕ_k y ϕ_k^r de la ecuación 5.8 se refieren a la posición de las ruedas y el ángulo de referencia en cada instante k.

Teniendo ya las ecuaciones que describen la evolución de la velocidad del coche y la posición de las ruedas, el último paso es describir movimiento del vehículo. Esto se hace a través del siguiente modelo:

$$\begin{cases} x_k = x_{k-1} + V_k \cdot t_s \cdot \cos(\psi_{k-1}) \\ y_k = y_{k-1} + V_k \cdot t_s \cdot \sin(\psi_{k-1}) \\ \psi_k = \psi_{k-1} + \dfrac{V_k \cdot t_s \cdot \tan(\phi_k)}{L} \end{cases} \tag{5.9}$$

donde t_s es el tiempo de muestreo —establecido en 0,1 s—, ϕ_k es el ángulo de las ruedas delanteras, V_k es la velocidad del vehículo y L es la longitud del mismo.

Con el entorno de simulación presentado, pasamos a describir en la siguiente sección el algoritmo implementado para el programa de aprendizaje.

5.1.3.2. Programa

Antes de iniciar con la descripción detallada del programa de aprendizaje, presentamos en la figura 5.4 el esquema general del mismo. Como se puede apreciar en la imagen, el primer bloque del programa se refiere a su inicialización. En esta fase se definen todos y cada uno de los parámetros que se utilizarán a lo largo del programa, como son las constantes de aprendizaje (parámetros α y γ en la ecuación 5.1), la duración máxima del programa, los mapas a utilizar como trayectorias de referencia y las variables de entrada utilizadas por el aprendizaje, así como sus respectivas funciones de pertenencia. De ahora en adelante, el número de variables de entradas utilizadas por el programa de aprendizaje se denotará como $nVar$

Para la agregación de cada variable de entrada, se asocia a la misma un conjunto de n funciones de pertenencia definidas a través de un arreglo de la forma $\{a, b, c, ...m, n\}$ donde $a < b < c < ... < m < n$. Cada uno de los valores del arreglo define una función triangular, siendo el valor en cuestión la posición central de la función —máximo de pertenencia (1)— y sus dos vecinos más cercanos los extremos de la misma —mínimo de pertenencia (0)—.

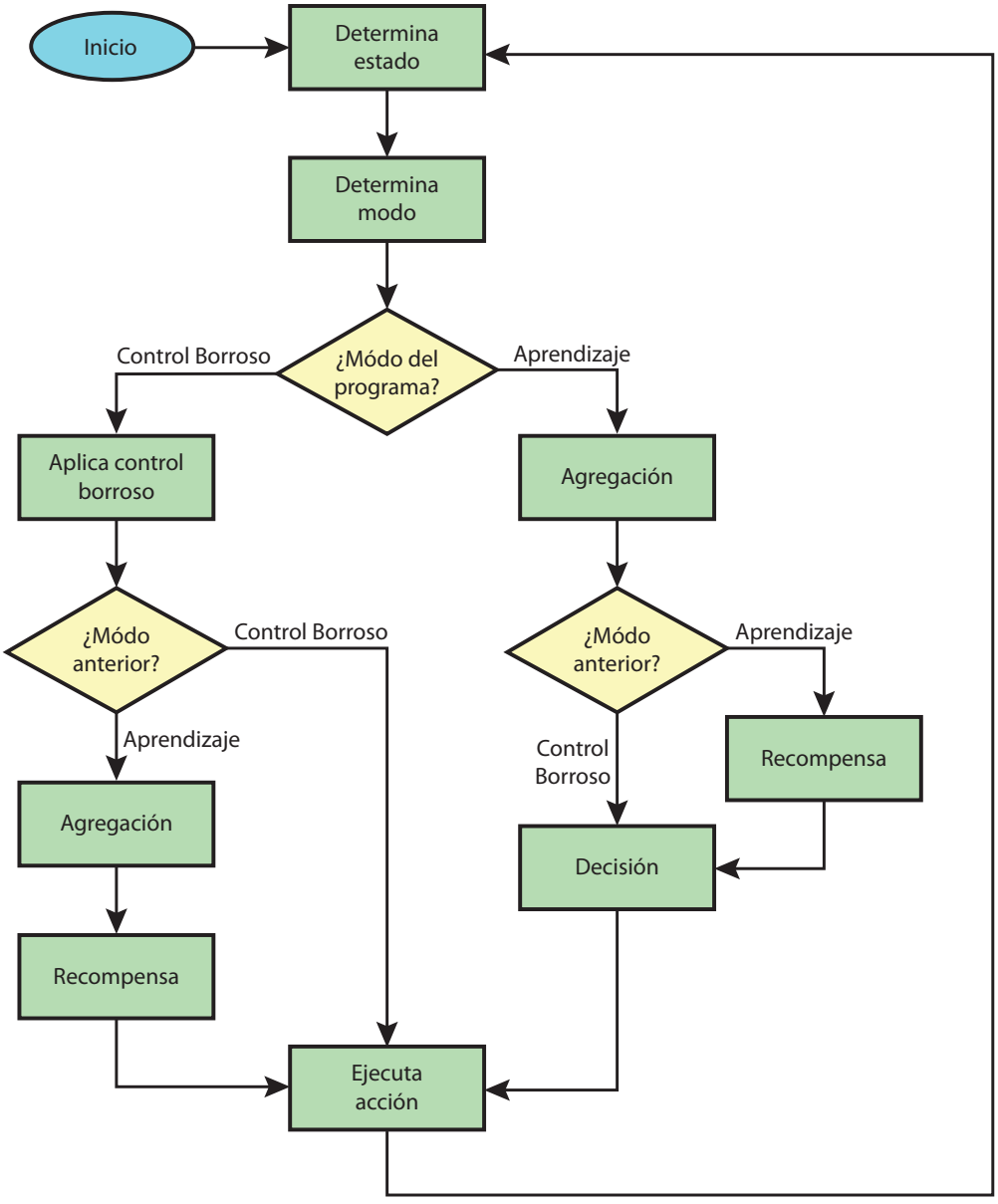


Figura 5.4: Esquema general del programa de aprendizaje.

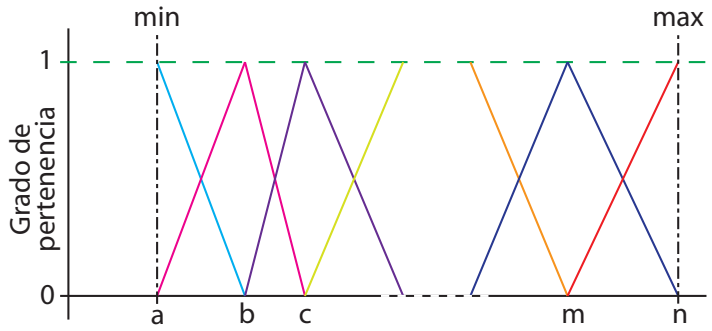


Figura 5.5: Ejemplo de funciones de pertenencia.

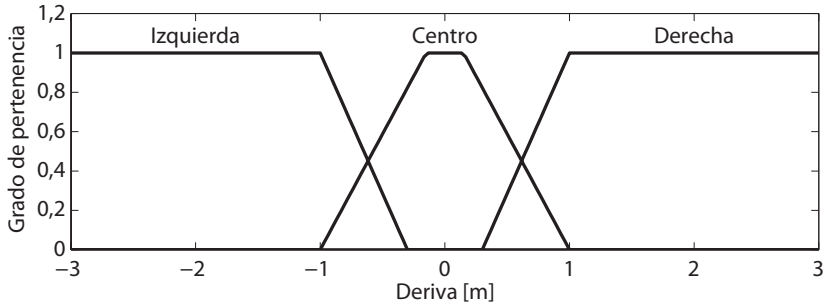
Los valores mínimo y máximo del arreglo se utilizan además para definir los límites de la variable, por lo que no es necesario definir más que uno de sus extemos. La figura 5.5 muestra un ejemplo gráfico de este método.

Una vez declaradas las funciones de pertenencia para cada entrada, se definen los posibles estados del sistema como el producto cartesiano de todos los conjuntos de entrada. De esta forma, si se implementa el aprendizaje utilizando sólo dos variables de entrada V_a y V_b, con funciones de pertenencia $\{a_1,\ a_2,\ a_3\}$ y $\{b_1,\ b_2\}$ respectivamente, tendremos un total de 6 estados posibles $\{(a_1, b_1), (a_2, b_1), (a_3, b_1), (a_1, b_2), (a_2,b_2), (a_3\ ,b_2)\}$. Finalmente, se describe la función Q utilizando una tabla donde el número de filas se refiere al número de estados del sistema, y el número de columnas a las acciones disponibles. Al inicio del aprendizaje, todos los valores de la tabla se establecen en 0.

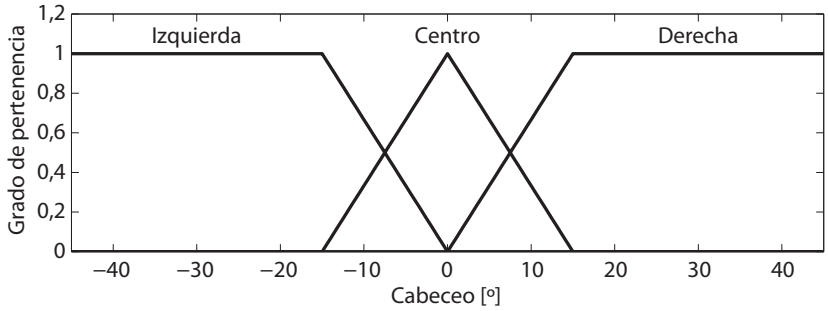
Luego de inicializar el programa y todos sus parámetros, se procede con el ciclo de ejecución del mismo. En cada ciclo, la primera acción consiste en determinar el estado del vehículo. Esto es, calcular la nueva posición y orientación del coche, los errores laterales y angulares respecto a la trayectoria de referencia y la(s) curvatura(s) de la misma; tal y como se presentó en la sección 5.1.2.

Después de calcular el estado del vehículo, se determina el modo de control a ejecutar: control borroso o control por aprendizaje. En el primer modo, el vehículo es controlado por un controlador borroso que ha sido ajustado de forma manual para el modelo de evolución del vehículo. Por su parte, en el segundo modo de control es donde se ejecuta la tarea de aprendizaje por refuerzo. Esta división ha sido necesaria para limitar los efectos del comportamiento errático del coche sobre el aprendizaje, especialmente en la fase inicial, donde se dispone de muy poco conocimiento para abordar el problema adecuadamente y se podrían producir situaciones en las que el vehículo deambula continuamente sin acercarse a la referencia. Así, el controlador borroso cumple el papel de socorrista, retornando el vehículo a una zona cercana a la trayectoria siempre que éste se encuentre fuera de los límites establecidos para el aprendizaje. La conmutación entre ambos modos se realiza de forma automática de acuerdo al estado del vehículo —e.g. rescatándolo cuando se desplace a más de 6 metros de la trayectoria y volviendo al aprendizaje una vez que está a menos de 3 metros de la misma—.

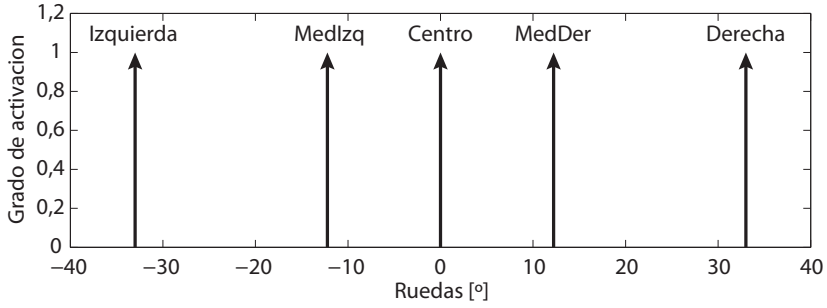
En el modo de control borroso, el control del vehículo se realiza únicamente utilizando



(a) Error lateral.



(b) Error angular.



(c) Posicion de las ruedas.

Figura 5.6: Funciones de pertenencia del controlador borroso auxiliar.

dos variables de entrada: la deriva y el cabeceo. A la salida del controlador se obtiene sólo la posición de las ruedas delanteras del coche, ya que el control borroso se realiza a una velocidad constante, establecida durante la inicialización del programa. La figura 5.6 muestra las funciones de pertenencia definidas para cada una de las entradas y la salida del controlador. A partir de estas funciones, las reglas que rigen el comportamiento del vehículo son las siguientes:

SI *Deriva* **Izquierda** Y *Cabeceo* **Izquierda** ENTONCES *Ruedas* **Derecha**
SI *Deriva* **Izquierda** Y *Cabeceo* **Centro** ENTONCES *Ruedas* **Derecha**
SI *Deriva* **Izquierda** Y *Cabeceo* **Derecha** ENTONCES *Ruedas* **Centro**
SI *Deriva* **Centro** Y *Cabeceo* **Izquierda** ENTONCES *Ruedas* **MedDer**
SI *Deriva* **Centro** Y *Cabeceo* **Centro** ENTONCES *Ruedas* **Centro**
SI *Deriva* **Centro** Y *Cabeceo* **Derecha** ENTONCES *Ruedas* **MedIzq**
SI *Deriva* **Derecha** Y *Cabeceo* **Izquierda** ENTONCES *Ruedas* **Centro**
SI *Deriva* **Derecha** Y *Cabeceo* **Centro** ENTONCES *Ruedas* **Izquierda**
SI *Deriva* **Derecha** Y *Cabeceo* **Derecha** ENTONCES *Ruedas* **Izquierda**

En cuanto al modo de aprendizaje, este se divide en tres fases: agregación, recompensa y decisión. La fase de agregación, como su nombre lo indica, se refiere a la fuzzificación de las variables de entrada y del estado del sistema. Para ello, primero se determina el grado de pertenencia asociado a cada una de las funciones definidas para las entradas. Acto seguido, se calcula la activación μ de cada uno de los estados como el mínimo valor de activación obtenido por sus respectivas funciones de pertenencia. Teniendo en cuenta la forma en que se han definido los conjuntos borrosos para cada variable —figura 5.5—, sólo 1 ó 2 funciones de pertenencia de cada variable estarán activas en cada ciclo, por lo que el número máximo de estados activados por ciclo es igual a 2^{nVar}.

Una vez que se ha calculado el grado de activación μ_s para cada estado agregado s^f, procedemos con el cálculo del vector Q de acuerdo al estado actual del vehículo. Sin embargo, dado que con este método de agregación se activan varios estados agregados de forma simultánea, el vector Q final se obtiene a través de la suma ponderada de los vectores Q de cada estado activado s^a; utilizando como factor de ponderación el grado de activación de los estados, tal y como se muestra en la ecuación 5.10.

$$V_q = \frac{\sum\limits_{\forall s^f \in S^a} \mu_s V_q^s}{\sum\limits_{\forall s^f \in S^a} \mu_s} \tag{5.10}$$

Alcanzado este punto, se verifica el modo de ejecución empleado en el ciclo anterior. Si este ha sido un ciclo de aprendizaje, se procede con la fase de recompensa; en caso contrario, se omite la misma y continuamos con la fase de decisión. En la fase de recompensa, se actualizan los valores de la tabla Q de acuerdo a la variación del estado del vehículo, la recompensa asociada y el valor del vector Q en el nuevo estado. Esto se hace de acuerdo a la ecuación 5.1, presentada en la sección 5.1.1. Sin embargo, debido a que el vector Q se calcula como una contribución de todos los estados activados, modificamos la ecuación a fin de otorgar la recompensa a cada estado de forma proporcional a su grado de activación. Así, la ecuación quedaría:

$$Q(s_t^f, a_t) = Q(s_t^f, a_t) + \alpha \cdot \mu_s \cdot \left(r_t + \gamma \max_{a_{t+1}} (Q(s_{t+1}, a_{t+1})) - Q(s_t, a_t) \right) \tag{5.11}$$

En este caso, la principal diferencia respecto a la ecuación 5.1 recae en la diferenciación entre el estado ponderado del vehículo s_t y cada uno de los estados agregados s_t^f.

Antes de concluir la fase de recompensa, calculamos nuevamente el valor del vector Q de acuerdo a la ecuación 5.10. Este paso es necesario debido a que las modificaciones realizadas sobre la tabla Q pudieron haber afectado los valores utilizados previamente para el cálculo de este vector. De esta forma se garantiza que la información obtenida del aprendizaje —i.e. la recompensa— se utilice tan pronto como esté disponible.

Finalmente, en la fase de decisión se determina la acción a tomar de acuerdo al estado del vehículo. Para ello, lo más sencillo es emplear una política *greedy* o voraz, seleccionando siempre aquella acción que corresponde al valor más alto dentro del vector Q; tal y como supone el algoritmo básico del *Q-learning*. No obstante, esta política se enfoca en la explota-

ción del conocimiento actual y la búsqueda de la máxima recompensa inmediata, impidiendo la selección de acciones que si bien pueden parecer inferiores, podrían conducir a estados con una recompensa mayor (Sutton y Barto, 1998). Por esta razón, hemos empleado una política conocida como ϵ-*greedy*. Esta es una variación de la política *greedy*, donde existe una probabilidad ϵ de seleccionar una acción de forma aleatoria, independientemente de los valores obtenidos para el vector Q (Sutton y Barto, 1998; Lauer, 2011).

A continuación se listan algunas consideraciones adicionales que se han tenido en cuenta durante la implementación del programa:

- Como se puede apreciar en el esquema de la figura 5.4, en el modo de control borroso también se verifica el modo ejecutado por el programa en el ciclo anterior. Esto se ha hecho con el objetivo de otorgar una recompensa negativa a todas aquellas acciones que conduzcan a estados donde el vehículo deba ser rescatado, permitiendo así que el sistema aprenda a evitarlos.

- La velocidad de referencia del vehículo durante el aprendizaje varía de forma aleatoria, dentro del rango [1,5, 8,5] m/s, establecido durante la inicialización del sistema. Así, provocamos que el vehículo se exponga a diferentes situaciones generadas por la variación de velocidad. Por defecto, hemos establecido que la velocidad cambie con cada vuelta completa a un mapa.

- Los cambios de mapas, o trayectorias de referencia, se realizan después de completar un determinado número de vueltas l_m en el mismo. Estos se seleccionan de forma secuencial entre la lista de trayectorias, de forma tal de que todos los mapas son utilizados el mismo número de veces a lo largo del aprendizaje. Una ejecución entera de la lista de mapas se denomina de ahora en adelante "etapa de aprendizaje".

- En el momento de realizar el cambio de referencia, el programa pasa al modo de control borroso hasta que se ubica al vehículo en la zona cercana a la nueva referencia. De estarse ejecutando el modo de aprendizaje antes del cambio de mapa, no se aplica ninguna recompensa al sistema debido a que el cambio de estado no ha sido provocado por una acción del sistema.

Una vez descritos los detalles de la implementación del programa, presentamos los resultados que se han obtenido con esta técnica de aprendizaje por refuerzo.

5.1.4. Experimentos y resultados

A fin de validar el correcto funcionamiento de la propuesta de control lateral basado en aprendizaje por refuerzo, hemos realizado una serie de pruebas con distintas configuraciones de partida. La idea tras estos experimentos es comparar el rendimiento del algoritmo de acuerdo a las variables de entrada empleadas para el controlador, así como sus respectivas funciones de pertenencia. De igual forma, utilizamos dos configuraciones de salida para el control: incremental, donde la acción de control se suma al último valor de referencia para las

ruedas; y absoluta, donde la referencia de las ruedas toma el valor de salida del controlador. Esto se hace en concordancia con los trabajos anteriores de AUTOPIA, donde se han tenido en cuenta ambas posibilidades de salida (Naranjo, 2005; Onieva, 2011; Pérez, 2012).

Antes de iniciar el análisis detallado de los resultados, debemos mencionar algunos parámetros de configuración que se han empleado por igual para todas las pruebas:

1. Están limitadas por el número de etapas de aprendizaje, no por el tiempo de ejecución. Cada prueba completa consta de un total de 50 etapas.

2. Se ha establecido el valor 5 para el parámetro l_m, siendo este valor el número de vueltas que deben completarse en cada mapa antes de cambiar al siguiente.

3. A fin de obtener una velocidad moderada de adaptación, empleamos una tasa de aprendizaje α de 0,4. Por otro lado, establecimos un factor de descuento γ de 0,99, dándole más prioridad a la recompensa a largo plazo sobre la recompensa inmediata.

4. Por último, en la política de selección de la mejor acción empleamos $\epsilon = 50$; lo cual a la frecuencia de muestreo del programa —10 Hz— corresponde a una acción aleatoria cada 5 s.

Habiendo mencionado los factores comunes a las pruebas, continuamos con los resultados de las mismas, agrupados en función de la configuración de salida del controlador.

5.1.4.1. Controlador con salida incremental

Para el controlador con salida incremental, definimos 3 acciones que modifican la referencia del ángulo de las ruedas a partir de su último valor: (i) desplazar 5 grados a la izquierda, (ii) no modificar y (iii) desplazar 5 grados hacia la derecha. Así mismo, los límites de la referencia los hemos fijado en ±30 grados, correspondiendo los valores negativos con posiciones angulares a la derecha del eje central del vehículo —giro en sentido horario—.

En la tabla 5.2 presentamos las 10 configuraciones de entrada que hemos sometido a consideración, resaltando las características que se han añadido/modificado a partir de configuraciones previas. Esto se resume en:

- Para la configuración 1, consideramos únicamente 3 variables de entrada: la deriva, el cabeceo y la posición del volante con 5, 5 y 7 funciones de pertenencia respectivamente. Esto nos deja un escenario con 175 estados posibles.

- En la segunda prueba, agregamos la velocidad del vehículo como cuarta variable de entrada, empleando únicamente 3 funciones de pertenencia para un rango de entornos urbanos, es decir, de bajas velocidades. En este caso se incrementa a 525 el número de estados.

- Para las configuraciones 3 y 4, consideramos además la curvatura promedio del segmento inmediato a la posición del vehículo —c_1 en la figura 5.2—. Para ello configuramos el

parámetro l_c a un valor de 4 metros. La diferencia entre ambas pruebas recae únicamente en la amplitud de las funciones de pertenencia empleadas para la agregación de la variable c_1. Para estas configuraciones, el número de estados asciende a 2 625.

- En las pruebas 5 y 6, agregamos dos segmentos adicionales de curvatura a las variables de entrada para el control —c_2 y c_3—. Ambas variables emplean la mismas funciones de pertenencia empleadas para c_1 en las pruebas 3 y 4. Con estas 7 variables, el número de estados llega a 65 625.

- Para las pruebas 7 y 8, realizamos una leve modificación a partir de las pruebas anteriores, reduciendo el rango de las funciones de pertenencia empleadas para el cabeceo del vehículo.

- Por último, las pruebas 9 y 10 hacen lo contrario a las dos pruebas anteriores, aumentando el rango de las funciones de pertenencia del cabeceo.

Para cuantificar los resultados de todas las pruebas y facilitar el proceso de análisis, hemos empleado dos criterios: el error absoluto medio (*MAE - Mean Absolute Error*) —en metros— respecto a la trayectoria de referencia y el número de rescates realizados por parte del controlador borroso en cada etapa. La evolución de estos valores en función de la etapa de aprendizaje se muestra en las tablas 5.3 y 5.4, respectivamente.

Al comparar los resultados de las configuraciones 1 y 2, observamos una clara mejora de ambos criterios de evaluación en esta última. Esto es gracias a la inclusión de la velocidad como variable de entrada del sistema, lo cual le permite discernir mejor la situación del vehículo a fin de determinar la mejor acción de control. En lo que respecta a la incorporación de la curvatura de la trayectoria inmediata —pruebas 3 y 4—, no se aprecia una mejora considerable en el rendimiento del controlador. Esto se puede verificar con las gráficas de la figura 5.7, las cuales muestran la evolución del MAE por etapa de aprendizaje para las configuraciones 2, 3 y 4. Las imágenes muestran claramente una evolución más lenta del proceso de aprendizaje para el caso de las dos últimas configuraciones mencionadas. Esto es producto del incremento del número de estados posibles del sistema.

La misma situación se ha presentado con la inclusión de las tres curvaturas inmediatas de la trayectoria —pruebas 5 y 6—. No obstante, se puede apreciar una mejor evolución en la fase intermedia —20, 30 y 40 etapas— para las configuraciones con un rango más amplio para las curvatura sobre el caso contrario —i.e. pruebas 4 y 6 respecto a las pruebas 3 y 5—. Esto lo confirmamos con las pruebas 7 y 8, donde a pesar de no mejorar el rendimiento del controlador respecto a la configuración 2, si notamos un mejor rendimiento global en la prueba 8 sobre la 7.

En lo que respecta a las últimas dos pruebas con funciones de pertenencia más amplias para el cabeceo, tampoco se ha conseguido mejorar el resultado obtenido en cuanto al error lateral con la configuración 2, aunque si se ha reducido el número de intervenciones necesarias por parte del controlador borroso en la prueba 10. Por este motivo consideramos que el rendimiento global de la configuración 10 es superior al de la configuración 2. Al igual que

Configuración	Entradas	Agregación	Estados
1	Deriva [m]	{-3, -1, 0, 1, 3}	175
	Cabeceo [º]	{-30, -15, 0, 15, 30}	
	Volante [º]	{-30, -20, -10, 0, 10, 20, 30}	
2	Configuración 1 +		525
	Velocidad [km/h]	{5, 15, 30}	
3	Configuración 2 +		2625
	Curvatura 1 [m^{-1}]	{-1/10, -1/30, 0, 1/30, 1/10}	
4	Configuración 2 +		2625
	Curvatura 1 [m^{-1}]	{-1/6, -1/12, 0, 1/6, 1/12}	
5	Configuración 3 +		65 625
	Curvatura 2 [m^{-1}]	{-1/10, -1/30, 0, 1/30, 1/10}	
	Curvatura 3 [m^{-1}]		
6	Configuración 4 +		65 625
	Curvatura 2 [m^{-1}]	{-1/6, -1/12, 0, 1/6, 1/12}	
	Curvatura 3 [m^{-1}]		
7	Configuración 5 +		65 625
	Cabeceo [º]	{-20, -10, 0, 10, 20}	
8	Configuración 6 +		65 625
	Cabeceo [º]	{-20, -10, 0, 10, 20}	
9	Configuración 5 +		65 625
	Cabeceo [º]	{-40, -20, 0, 20, 40}	
10	Configuración 6 +		65 625
	Cabeceo [º]	{-40, -20, 0, 20, 40}	

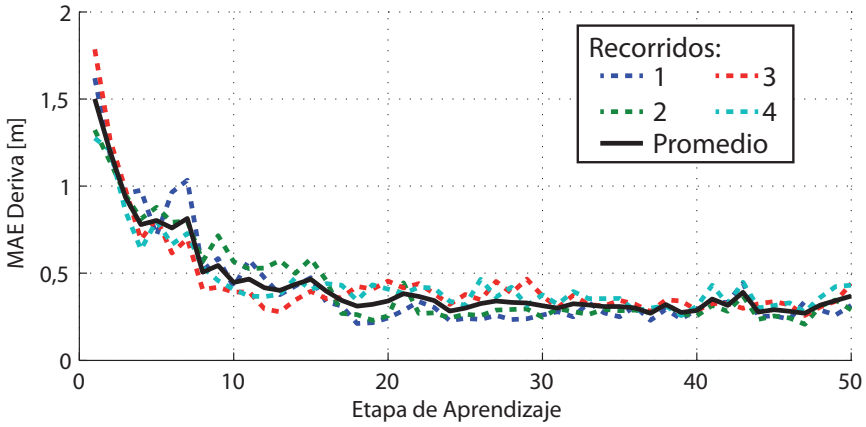
Tabla 5.2: Configuraciones de entrada empleadas para el aprendizaje por refuerzo.

Configuración	Etapa de aprendizaje					
	1	10	20	30	40	50
1	1,281 [m]	0,321 [m]	0,322 [m]	0,402 [m]	0,662 [m]	1,447 [m]
2	1,500 [m]	0,447 [m]	0,341 [m]	0,313 [m]	0,286 [m]	0,369 [m]
3	1,382 [m]	1,304 [m]	1,129 [m]	0,880 [m]	0,722 [m]	0,617 [m]
4	1,657 [m]	1,083 [m]	0,890 [m]	0,803 [m]	0,770 [m]	0,726 [m]
5	1,467 [m]	1,214 [m]	1,060 [m]	0,897 [m]	0,817 [m]	0,768 [m]
6	1,390 [m]	1,165 [m]	0,916 [m]	0,659 [m]	0,650 [m]	0,629 [m]
7	1,412 [m]	1,319 [m]	1,014 [m]	0,969 [m]	0,718 [m]	0,653 [m]
8	1,344 [m]	1,311 [m]	0,962 [m]	0,613 [m]	0,565 [m]	0,479 [m]
9	1,561 [m]	0,726 [m]	0,595 [m]	0,597 [m]	0,542 [m]	0,515 [m]
10	1,301 [m]	0,625 [m]	0,459 [m]	0,496 [m]	0,416 [m]	0,410 [m]

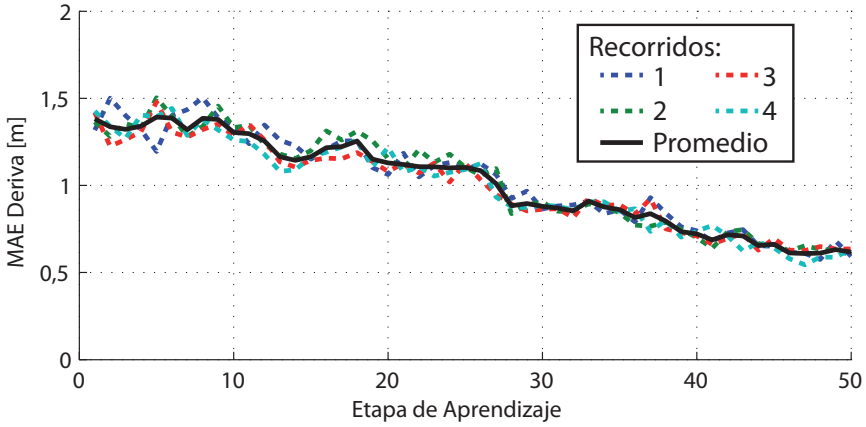
Tabla 5.3: Evolución del MAE lateral para las configuraciones de aprendizaje con control incremental.

Configuración	Etapa de aprendizaje					
	1	10	20	30	40	50
1	297	40	52	55	75	346
2	414	49	25	42	42	37
3	386	378	212	102	53	38
4	423	146	48	62	58	85
5	396	313	160	136	101	97
6	347	248	68	69	80	118
7	427	314	232	157	120	112
8	389	329	78	67	107	100
9	420	44	13	31	21	25
10	293	30	12	9	6	9

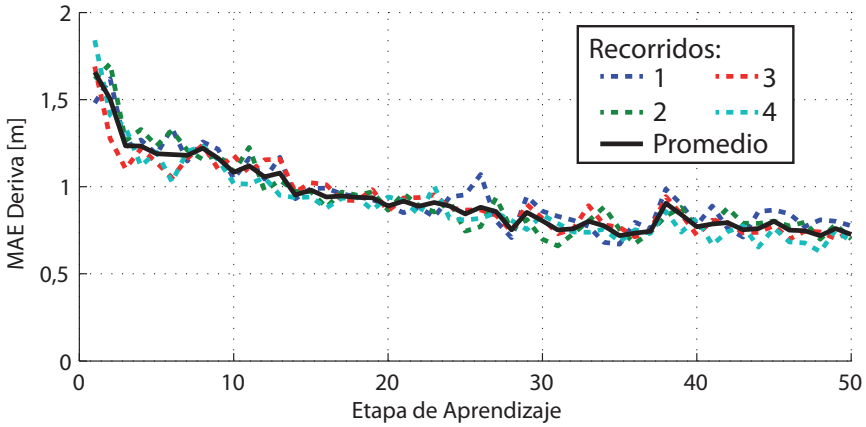
Tabla 5.4: Evolución del número de rescates para las configuraciones de aprendizaje con control incremental.



(a) Configuración 2

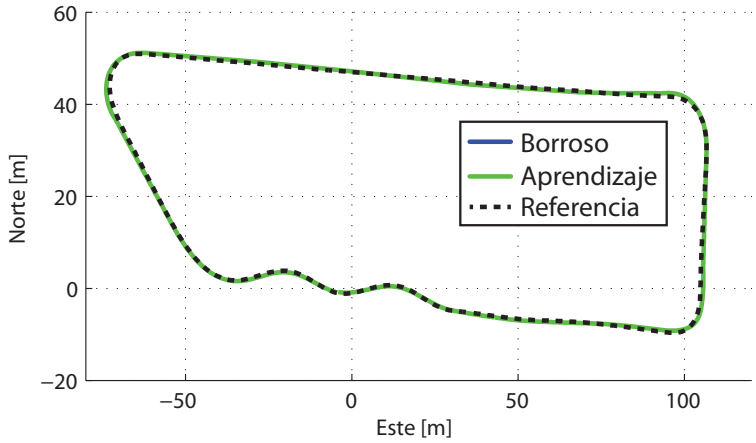


(b) Configuración 3

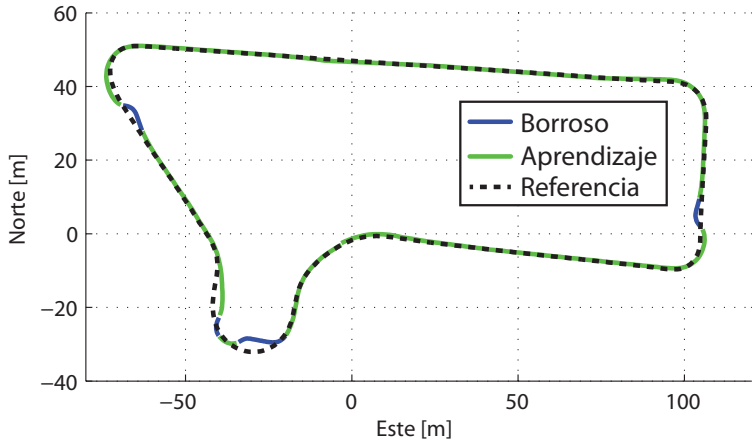


(c) Configuración 4

Figura 5.7: Comparación del error lateral para las configuraciones 2, 3 y 4.



(a) Recorrido 1.



(b) Recorrido 2.

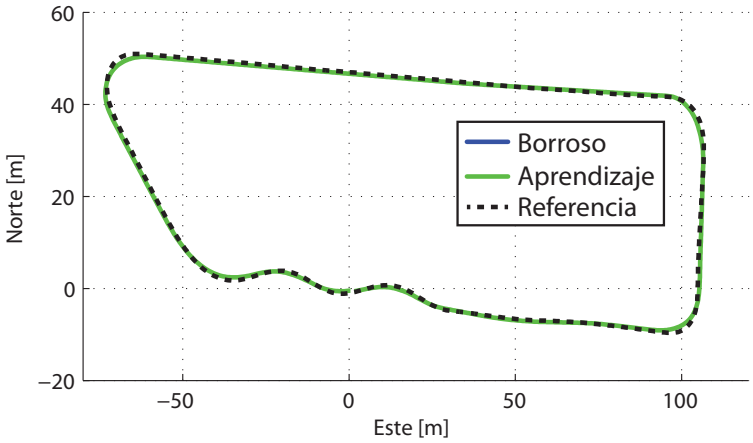
Figura 5.8: Quinta vuelta de la configuración 2 en los recorridos 1 y 2.

para las pruebas 7 y 8, también observamos un mejor rendimiento al emplear funciones de pertenencia más amplias para la curvatura. Para finalizar, en las figuras 5.8 y 5.9 se presenta la última vuelta de cada configuración para las trayectorias de referencia 1 y 2.

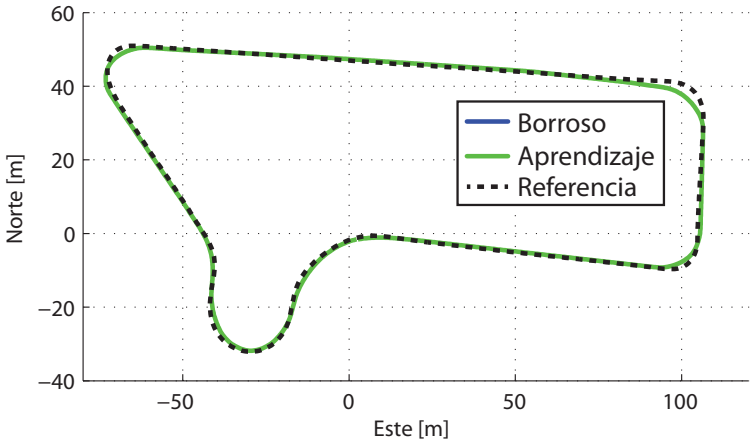
5.1.4.2. Controlador con salida absoluta

A diferencia del controlador anterior, en este caso las acciones definen directamente la referencia para el control de posición de bajo nivel de la dirección. Para ello, empleamos un conjunto de 5 acciones posibles: -30, -10, 0, 10 y 30 grados; correspondiendo los valores negativos con giros hacia la derecha del eje central del vehículo; es decir, en sentido horario. Estos valores los hemos seleccionado de forma tal de que el vehículo sea capaz de abordar tanto giros abiertos, con un radio de curvatura amplio; como giros muy cerrados, con el valor máximo permitido para el controlador incremental.

Para esta configuración de control, la validación del método de aprendizaje la realizamos a través de las mismas pruebas definidas para el controlador incremental —ver tabla 5.2—. Sin embargo, dado que la salida de este controlador es absoluta, las funciones de pertenencia



(a) Recorrido 1.



(b) Recorrido 2.

Figura 5.9: Quinta vuelta de la configuración 10 en los recorridos 1 y 2.

Configuración	Etapa de aprendizaje					
	1	10	20	30	40	50
1	0,961	0,185	0,141	0,110	0,102	0,076
2	0,705	0,282	0,232	0,229	0,211	0,203
3	1,452	0,434	0,453	0,383	0,343	0,375
4	1,449	0,484	0,374	0,375	0,354	0,349
5	0,758	0,556	0,487	0,401	0,393	0,375
6	1,453	0,715	0,496	0,469	0,462	0,468
7	1,331	0,759	0,762	0,671	0,513	0,453
8	1,065	0,746	0,696	0,632	0,570	0,546
9	0,706	0,251	0,281	0,269	0,294	0,262
10	0,841	0,369	0,286	0,285	0,309	0,269

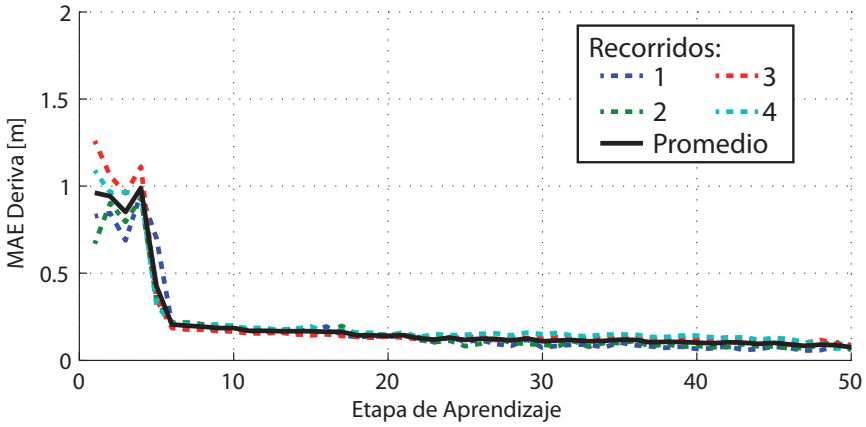
Tabla 5.5: Evolución del MAE lateral para las configuraciones de aprendizaje con control absoluto.

relacionadas con la posición del volante no son necesarias y no se tienen en cuenta en ninguno de los ensayos, reduciendo así el número de estados posibles del sistema.

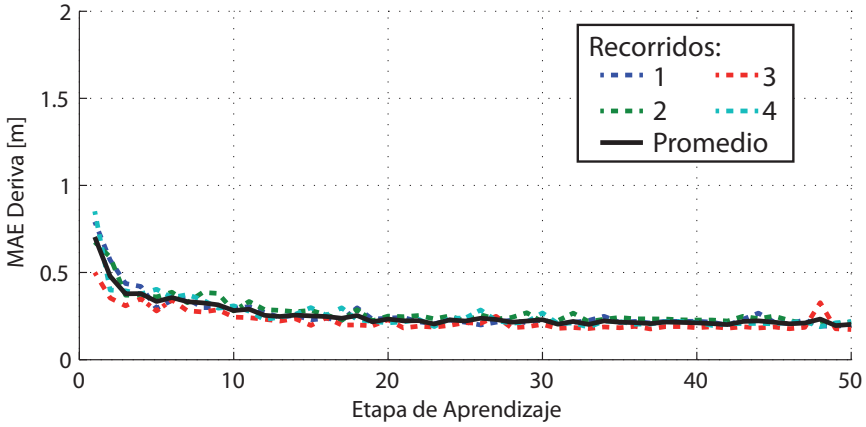
Las tablas 5.5 y 5.6 muestran los resultados obtenidos en cada una de las pruebas, empleando los mismos criterios de comparación seleccionados para el controlador incremental. En este caso, observamos una relación inversa en la evolución de ambas variables, siendo las configuraciones 1 y 2 las que arrojan los mejores resultados tanto en el error lateral como en el número de rescates necesarios. Es en esta última variable donde se aprecia la mayor diferencia respecto al controlador incremental, puesto que incluso desde la primera etapa de aprendizaje el número de rescates realizados es inferior al menor valor obtenido con el controlador previo.

La figura 5.10 muestra las gráficas de evolución del error lateral para las configuraciones 1, 2 y 10. En ellas se puede apreciar que la evolución del aprendizaje presenta un patrón similar en los tres casos, adaptándose rápidamente durante las primeras etapas y presentado una evolución lenta en las etapas finales.

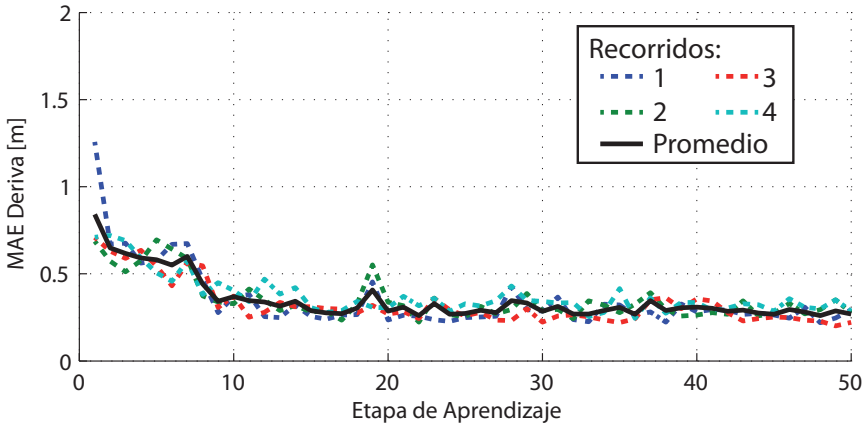
A partir de estos resultados concluimos que, para el caso del control con salida absoluta, no compensa de ninguna manera el aumentar el número de variables del sistema, puesto que con una configuración de tan sólo 75 estados y 5 acciones se logra obtener el mejor resultado en cuanto a los dos criterios de comparación. De igual forma apreciamos que el rendimiento del controlador absoluto es claramente superior al del controlador incremental, requiriendo además una tabla Q de menores dimensiones, lo que simplifica tanto el ajuste como la aplicación del controlador. No obstante, si no se limita adecuadamente la salida del controlador absoluto, se pueden generar variaciones muy bruscas de la consigna de referencia, afectando tanto el rendimiento como la estabilidad del sistema.



(a) Configuración 1



(b) Configuración 2



(c) Configuración 10

Figura 5.10: Comparación del error lateral para las configuraciones 1, 2 y 10.

Configuración	Etapa de aprendizaje					
	1	10	20	30	40	50
1	12	1	2	0	0	0
2	8	0	2	0	0	0
3	152	0	2	0	0	2
4	137	0	0	0	0	6
5	46	0	1	0	0	2
6	131	15	0	4	10	11
7	145	1	0	0	5	3
8	57	0	1	0	0	3
9	22	0	0	1	1	2
10	49	2	3	5	1	8

Tabla 5.6: Evolución del número de rescates para las configuraciones de aprendizaje con control absoluto.

5.1.5. Aplicación a un vehículo real

A pesar de que con esta propuesta de control lateral hemos alcanzado resultados adecuados y esperanzadores en entornos simulados, no ha sido posible satisfacer las expectativas en su aplicación a un vehículo real. Esto se debe a: (i) el ajuste de un controlador con el vehículo prototipo requiere de un elevado tiempo de ejecución, equivalente a varios días de conducción continuada; (ii) durante la fase inicial del ajuste, se producen movimientos bruscos en la dirección del vehículo, los cuales pueden llegar a producir daños en la plataforma de experimentación; y (iii) la transferencia de un controlador ajustado por simulación al vehículo real no es inmediata, puesto que si bien el modelo es una representación bastante apropiada del vehículo, este no es lo suficientemente exacto. En este sentido, consideramos que el desarrollo de un modelo más preciso permitiría obtener resultados de transferencia inmediata.

5.2. Control longitudinal y adaptación de controladores

Las técnicas de aprendizaje, como la descrita en la sección anterior para el control lateral, son muy útiles a la hora de abordar un problema en el que se carece de conocimiento previo o bien no se desea condicionar la solución a la experiencia del programador. No obstante, una desventaja de las mismas es que dependen en gran medida de la exploración amplia del espacio de estados, lo cual se refleja en la necesidad de un elevado tiempo de ejecución para converger a una solución adecuada. Esto, si bien puede sortearse en entornos virtuales, es una limitación a la que nos tenemos que enfrentar necesariamente en su aplicación final en entornos reales, como sería el caso de la conducción autónoma del vehículo.

Por otro lado, ha quedado demostrado que la complejidad del problema de aprendizaje aumenta de forma exponencial a medida que se incrementa el número de variables de entrada,

por lo que es prácticamente imposible, e inviable, tener en cuenta todas las variables que de una u otra forma afectan el comportamiento del sistema. De igual forma, aunque no existiese esta limitación, determinar la influencia de algunas variables sobre el sistema puede ser una labor bastante compleja. La presión de los neumáticos, la relación de las marchas, el peso de los ocupantes o el estado del motor son algunos ejemplos de variables que, si bien afectan el control longitudinal del vehículo, no suelen tomarse en cuenta en el desarrollo de controladores. En el caso de un conductor humano, estas variables tampoco se consideran directamente, sino que el conductor se adapta a medida que se familiariza con la respuesta del vehículo. Por ejemplo, ante una cuesta pronunciada, el conductor selecciona una marcha u otra en función de la respuesta del vehículo durante la subida.

En esta sección presentamos el desarrollo de un controlador longitudinal borroso adaptativo. La idea es que dependiendo del rendimiento del vehículo, el controlador sea capaz de adaptar su estructura en tiempo de ejecución, mejorando finalmente la respuesta del vehículo ante variaciones impredecibles del entorno. Durante el desarrollo del controlador nos hemos enfocado en los entornos urbanos, ya que es en estos donde el efecto de las perturbaciones externas es mayor. A continuación presentamos una breve descripción del problema del control longitudinal del vehículo y el enfoque aplicado en la solución planteada.

5.2.1. Descripción del problema

Hoy en día, el exceso de velocidad sigue siendo una de las causas recurrentes en la mayoría de los accidentes de tráfico (Dirección General de Tráfico, 2012; International Traffic Safety Data and Analysis Group, 2011). Por este motivo una de las principales líneas de investigación presentes actualmente en la industria automotriz es el desarrollo de sistemas para el control automático de la velocidad. Además de mejorar la seguridad en las carreteras, la aplicación de este tipo de sistemas también busca mejorar el confort del usuario, facilitándole el proceso de conducción.

Actualmente, existen diferentes enfoques en cuanto al control automático de la velocidad. Los más conocidos y comercializados son los sistemas de control de crucero, los cuales permiten regular la velocidad del vehículo manteniendo una velocidad de referencia previamente establecida por el conductor. Dada la amplia aceptación de estos últimos, se continuó su desarrollo hasta llegar a los sistemas de control de crucero adaptativo. Estos últimos, además de mantener la velocidad fijada por el usuario, tienen en cuenta la presencia de otros vehículos en la vía. Gracias a un conjunto de sensores instalados a bordo —cámaras y/o radares— estos sistemas son capaces de mantener una distancia de seguridad con el vehículo más próximo, reduciendo temporalmente la velocidad del control de crucero en caso de ser necesario (Naranjo et al., 2003).

Más recientemente, y gracias a los avances en cuanto a sistemas de comunicaciones entre vehículos, se ha canalizado esta línea de investigación al desarrollo de sistemas de ACC con cooperación (*cooperative adaptive cruise control - CACC*) (Desjardins y Chaib-draa, 2011). En este caso, el sistema de control emplea la información compartida a través de un enlace de comunicaciones, lo cual permite aumentar la seguridad y el rendimiento del mismo (Van Arem

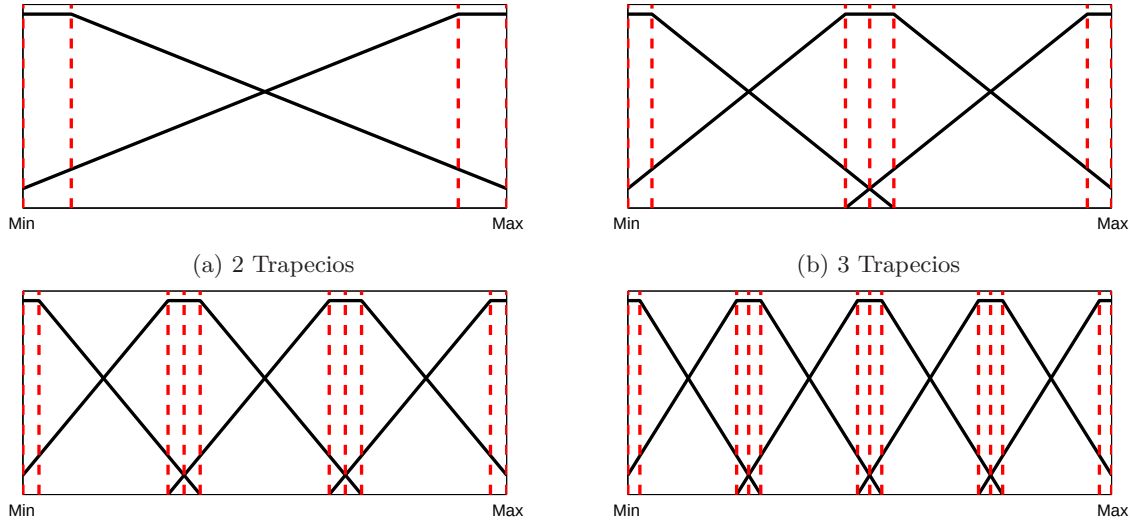
et al., 2006). Por último, algunos sistemas de ACC cuentan con capacidad *Stop&Go*, es decir, son capaces de funcionar incluso en situaciones en las que el vehículo debe detenerse por completo; como por ejemplo, durante un atasco (Naranjo et al., 2007; Martinez y Canudas-de Wit, 2007).

A pesar de que muchos fabricantes ya incorporan los sistemas CC o ACC en sus vehículos, estos suelen tener restricciones de velocidad mínima para su activación. Esto se debe a que, a baja velocidad, las acciones que se ejecutan sobre el acelerador y/o el freno afectan en mayor grado la dinámica del vehículo, dificultando el control del mismo (Davis, 2004). Así mismo, se incrementa el efecto de ciertas perturbaciones externas impredecibles como pueden ser el peso de los pasajeros o las variaciones en la superficie y/o inclinación de la carretera. Por lo tanto, para que un sistema CC o ACC sea viable a bajas velocidades, debe ser capaz de adaptarse a estas condiciones en tiempo de ejecución.

En este sentido, nos planteamos como objetivo el desarrollar un sistema CC con capacidad de adaptación, a fin de regular la velocidad del vehículo de forma precisa, independientemente de la dinámica del mismo o de las condiciones del camino. Para ello, presentamos un método de adaptación que permite evolucionar una estructura inicial de controlador borroso para que esta satisfaga ciertos requisitos establecidos. La evolución del sistema se realiza en tiempo de ejecución, empleando únicamente los datos de rendimiento del controlador, por lo que no es necesario implementar ningún tipo de modelo del vehículo.

Como base para el ajuste empleamos un controlador Takagi-Sugeno con dos variables de entrada y una única salida. Las variables de entrada son: (i) `ErrorVel`, que representa la diferencia entre la referencia de velocidad y la velocidad actual del vehículo —en m/s—; y (ii) `Aceleración` que, como su nombre indica, se refiere a la tasa de cambio de la velocidad —en m/s^2—. La codificación de cada entrada i se realiza mediante un número inicial de trapecios n_t^i definido por el usuario para cada una de ellas. Los trapecios son generados de manera uniforme, distribuyendo sus centros a lo largo de todo el rango de la entrada y definiendo el tamaño de la base menor como el 10 % de la base mayor. La figura 5.11 muestra un ejemplo de la distribución de los trapecios para $n_t^i = 2, 3, 4$ y 5. Se puede apreciar en la imagen que, en todos los casos, la parte inferior de los trapecios se solapa con la de sus vecinos. Esto se ha hecho con el fin de garantizar que todo valor de entrada active al menos dos funciones de pertenencia de la variable.

En cuanto a la salida del controlador, definimos la variable `Pedal` para codificar simultáneamente las acciones sobre el acelerador y el freno del vehículo. Esta variable está dentro del rango $[-1, 1]$, donde los valores positivos representan acciones sobre el acelerador con ningún tipo de acción sobre el freno, y los valores negativos el caso contrario, es decir, acciones sobre el freno con acción nula sobre el acelerador. El número de *singletons* que componen la salida es igual al número de reglas del controlador. Inicialmente, todos los *singletons* tendrán valor 0, representando ninguna acción sobre los actuadores. Por su parte, la base de reglas se define mediante la combinación de todas las funciones de pertenencia de cada entrada, de forma tal de que se consideran todos los casos posibles. Así, para 2 entradas con 3 y 4 funciones de pertenencia, tendremos un total de 12 reglas borrosas y una salida con 12 *singletons*.



(a) 2 Trapecios

(b) 3 Trapecios

(c) 4 Trapecios

(d) 5 Trapecios

Figura 5.11: Ejemplos de distribución inicial de los trapecios para las entradas.

5.2.2. Adaptación

El método de adaptación propuesto para el ajuste del controlador se divide en dos etapas: (i) ajuste de consecuentes, donde se modifica la posición de los *singletons* que definen la salida del controlador; y (ii) ajuste estructural, donde se ajusta la estructura del controlador borroso modificando las funciones de pertenencia que definen las entradas. A continuación se describen detalladamente cada una de estas etapas, así como una serie de consideraciones finales para el ajuste del controlador.

5.2.2.1. Ajuste de consecuentes

En esta etapa se realiza el ajuste de los consecuentes de la base de reglas, a fin de mejorar el rendimiento del controlador; es decir, el seguimiento de la velocidad de referencia. El procedimiento empleado para el ajuste es similar al del aprendizaje por refuerzo, asignando una recompensa al controlador en función de las variables de entrada. No obstante, en este caso el valor de la recompensa repercute directamente en la posición de los *singletons* de salida, modificándolos de acuerdo al grado de activación de su regla asociada. Esto se hace aplicando la siguiente ecuación:

$$S_k^i = S_{k-1}^i + \mu_{k-1}^i R\left(E_k, A_k\right) \tag{5.12}$$

donde S_k^i es la posición del singleton i en el instante k, μ_{k-1}^i es el grado de activación de la regla i en el instante $k-1$ y $R\left(E_k, A_k\right)$ es la recompensa asignada en función de las dos variables de entrada en el instante k, siendo E_k y A_k los valores de `ErrorVel` y `Aceleración` respectivamente.

A fin de garantizar el confort de los pasajeros en todo momento, hemos establecido una serie de consideraciones para el control longitudinal, en base a las cuales realizaremos la

asignación de recompensas:

- La aceleración del vehículo está limitada por dos umbrales de confort: a_c^+ para el límite superior y a_c^- para el límite inferior. Para evitar posibles confusiones debemos aclarar que los signos + y − se refieren al signo de la variable `Aceleración`, positiva cuando el vehículo está aumentando su velocidad y negativa cuando el vehículo está frenando.
- Con objeto de evitar cambios bruscos en la aceleración, definimos un tiempo t_c tal que cuando la velocidad del vehículo se aproxime a la referencia, la aceleración se reduzca linealmente hasta alcanzar el valor 0 en mismo instante en que $E_k = 0$. La reducción se inicia una vez que se cumple la condición $|E_k| \leq a_c^{\pm} \cdot t_c$.
- Existe un error de medición de las variables, por lo que deben tenerse en cuenta ciertos umbrales al momento de calcular las recompensas. En concreto, el umbral T_a para la aceleración.

Así, consideramos los 9 casos presentados en la tabla 5.7 para la asignación de la recompensa. En concreto:

- El conjunto $\{C_1, C_2, C_3, C_4\}$ se refiere al caso en que la velocidad del vehículo es inferior a la referencia. En particular:
  - El conjunto $\{C_1, C_2\}$ agrupa los casos en los que el tiempo de establecimiento con la máxima aceleración (t_e) es superior a t_c, es decir que la velocidad del vehículo es considerablemente inferior a la referencia. En este caso deseamos que el vehículo mantenga la aceleración en el máximo posible, esto es en el valor de a_c^+. Los casos particulares de esta situación son:
    - C_1: La aceleración del vehículo es mayor que la aceleración de confort más el umbral de medición, por lo tanto se debe reducir el valor de los consecuentes.
    - C_2: La aceleración del vehículo es menor a la aceleración de confort menos el umbral de medición, por lo tanto se debe aumentar el valor de los consecuentes.
  - Por su parte, el conjunto $\{C_3, C_4\}$ representa la situación en la que el tiempo de establecimiento con la máxima aceleración es menor o igual a t_c, por lo que se debe iniciar la reducción lineal de la aceleración del vehículo. Los casos particulares son:
    - C_3: La aceleración del vehículo es mayor a la aceleración necesaria para eliminar el error de velocidad en el tiempo t_c, siendo la aceleración necesaria el cociente de E_k/t_c más el umbral de medición de la aceleración. En este caso debemos reducir el valor de los consecuentes.
    - C_4: La aceleración del vehículo no es suficiente para reducir el error en el tiempo t_c, por lo que se debe aumentar el valor de los consecuentes. En este caso también consideramos el umbral de medición, sin embargo limitamos la aceleración necesaria a 0 ya que un cambio de signo significaría frenar el vehículo, que no es lo deseado.

Caso	Condiciones			Recompensa
C_1	$E_k > 0$	$t_e = \frac{E_k}{a_c^+} > t_c$	$A_k > a_c^+ + T_a$	$-\alpha \cdot \lvert E_k \rvert$
C_2			$A_k < a_c^+ - T_a$	$\alpha \cdot \lvert E_k \rvert$
C_3		$t_e = \frac{E_k}{a_c^+} \leq t_c$	$A_k > \frac{E_k}{t_c} + T_a$	$-\alpha \cdot \lvert E_k \rvert$
C_4			$A_k < \text{máx}\left(0, \frac{E_k}{t_c} - T_a\right)$	$\alpha \cdot \lvert E_k \rvert$
C_5	$E_k < 0$	$t_e = \frac{E_k}{a_c^-} > t_c$	$A_k < a_c^- - T_a$	$\alpha \cdot \lvert E_k \rvert$
C_6			$A_k > a_c^- + T_a$	$-\alpha \cdot \lvert E_k \rvert$
C_7		$t_e = \frac{E_k}{a_c^-} \leq t_c$	$A_k < \frac{E_k}{t_c} - T_a$	$\alpha \cdot \lvert E_k \rvert$
C_8			$A_k > \text{mín}\left(0, \frac{E_k}{t_c} + T_a\right)$	$-\alpha \cdot \lvert E_k \rvert$
C_9	Ninguna de las anteriores			0

Tabla 5.7: Casos a considerar durante el ajuste de consecuentes del controlador.

- El conjunto $\{C_5, C_6, C_7, C_8\}$ agrupa los casos en los que la velocidad del vehículo es superior a la deseada. En concreto, se refiere a las situaciones totalmente opuestas del conjunto $\{C_1, C_2, C_3, C_4\}$, por lo que las condiciones y recompensas están reflejadas, teniendo en cuenta el valor de a_c^-.
- Por último, C_9 representa el caso en que no se satisface ninguna de las condiciones anteriores, por lo que no debe aplicarse ninguna modificación sobre el controlador puesto que los valores de las variables de entrada están dentro del rango deseado.

La constante α empleada en el cálculo de la recompensa permite regular el grado de adaptación —por ciclo— del controlador en función del error respecto a la referencia ($|e|$). Su valor por defecto es 0,01; pudiendo ser modificada posteriormente. De igual forma, a las constantes a_c^+, a_c^- y t_c les hemos asignado valores de 1 m/s^2, -2 m/s^2 y 1 s respectivamente. El valor de los umbrales de aceleración los hemos seleccionado de acuerdo a los valores empleados en la literatura para aplicaciones similares (Bechtel, 1993). No obstante, consideramos que el frenado del vehículo es una acción más crítica, por lo que el valor absoluto de su umbral es 2 veces el valor de a_c^-.

Con esta configuración, el conjunto de todos los casos define las zonas de recompensa mostradas en la figura 5.12, donde las zonas en rojo y verde representan las situaciones en las que el valor de los consecuentes debe reducirse o aumentarse, respectivamente; y el color azul representa los casos en los que no se realiza ninguna modificación sobre el controlador.

5.2.2.2. Ajuste estructural

En la sección anterior presentamos como se modifican los consecuentes del controlador en función de su rendimiento en cada ciclo. Ahora es el turno de describir el proceso de adaptación de su estructura; esto es, el ajuste de las funciones de pertenencia de cada una de las entradas. A diferencia de la etapa de ajuste de consecuentes, no ejecutamos esta etapa continuamente sino cada n_e ciclos. Esto lo hacemos con el fin de no modificar la estructura

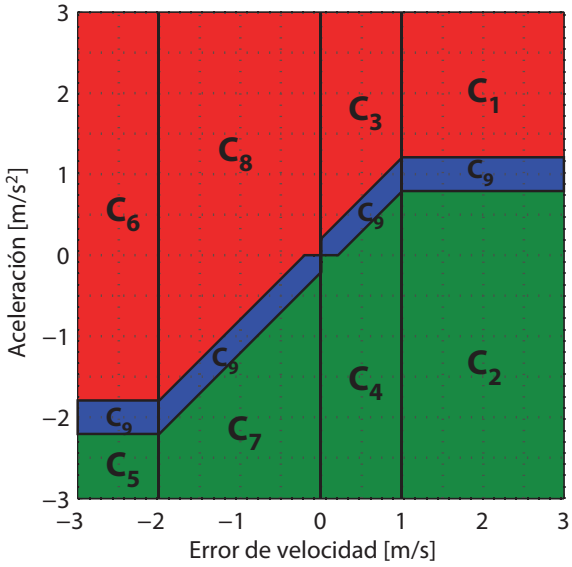


Figura 5.12: Zonas de recompensa definidas de acuerdo a las posibles situaciones.

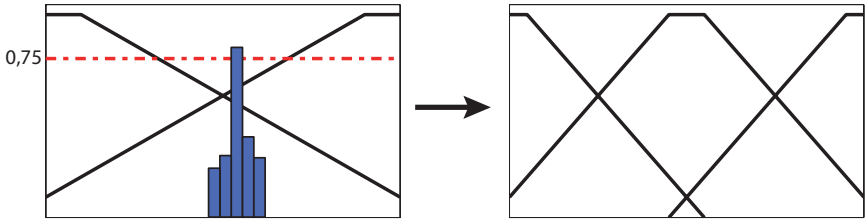
del controlador en base a datos individuales sino considerando un conjunto de datos.

En esta etapa, el proceso de ajuste se inicia con un análisis estadístico de los últimos valores de entrada del controlador, lo cual nos permite determinar que tipo de modificación se debe aplicar sobre las funciones de pertenencia —si es necesario—. Para ello elaboramos un histograma de los datos de entrada de los últimos n_e ciclos y analizamos la ubicación de los valores más frecuentes respecto a las funciones de pertenencia definidas. En este proceso tenemos en cuenta las siguientes consideraciones:

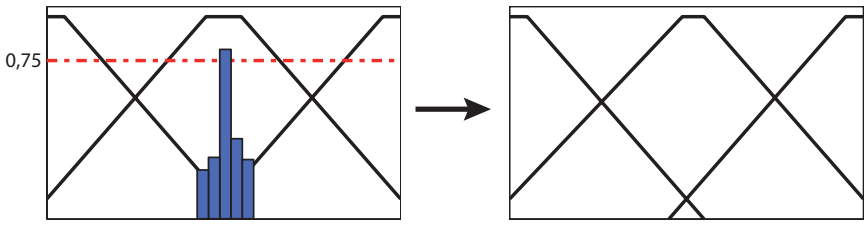
- Si la clase más frecuente del histograma produce una activación menor a 0,75; entonces el rango de la variable i no está cubierto adecuadamente, por lo que debemos agregar una nueva función de pertenencia. Esto se hace incrementando en una unidad el valor de n_t^i y reiniciando nuevamente la posición de todas las funciones, tal y como se hace en la inicialización del algoritmo. De igual forma, es necesario reiniciar a 0 la posición de los *singletons* ya que se han modificado las reglas del controlador.

- Si las dos clases más frecuentes del histograma producen una activación superior a 0,75 de la misma función de pertenencia, entonces el área abarcada por esta función es muy amplia, por lo que se procede con la reducción de su base menor. En este caso los *singletons* no son reiniciados puesto que la base de reglas del controlador no ha sido modificada.

La figura 5.13 muestra un ejemplo de ambas adaptaciones. Al agregar una nueva función de pertenencia a una determinada entrada —figura 5.13a— incrementamos el grado de activación que provocan los valores de entrada más frecuentes, a fin de poder generar una acción de control más específica a los mismos. Por su parte, la reducción de la base menor de una función de transferencia —figura 5.13b— permite que el controlador diferencie mejor los valores de entrada.



(a) Agregando una función. A la izquierda las funciones originales con el histograma de los datos y a la derecha las funciones adaptadas.



(b) Modificando una función. A la izquierda las funciones originales con el histograma de los datos y a la derecha las funciones adaptadas.

Figura 5.13: Ejemplos del ajuste estructural del controlador.

5.2.2.3. Consideraciones finales

Con objeto de mejorar el proceso de ajuste del controlador, hemos tenido en cuenta tres consideraciones adicionales:

1. Si la salida del controlador cambia de signo —i.e. se produce un cambio de actuador—, la misma se establece al valor 0 durante 0,5 s. Esto se hace a fin de limitar los efectos de una posible oscilación de la salida cuando el error se acerca a 0.

2. El proceso de ajuste del controlador se detiene durante 1 s cuando se produce un cambio en la velocidad de referencia del vehículo. Esto le da un tiempo de gracia al controlador para afrontar el cambio de referencia antes de iniciar nuevamente su ajuste. Así se evitan cambios drásticos en los *singletons*, producto del aumento repentino en el error de velocidad.

3. Por último, se establece un umbral mínimo de 2 % para las acciones sobre los actuadores; es decir, que cualquier acción en el rango $[-0{,}02,\ 0{,}02]$ se aproxima a 0. Esto se hace debido a que las acciones inferiores a este umbral no tienen un efecto real en los actuadores. Este valor fue determinado experimentalmente durante la instrumentación del vehículo.

Una vez descrito el método de adaptación para el controlador longitudinal, pasamos a presentar los resultados experimentales obtenidos durante su validación. Estos se han llevado a cabo en dos fases: la primera, utilizando un entorno de simulación con distintos modelos de vehículo; y la segunda, utilizando el vehículo real instrumentado con la arquitectura presentada en el capítulo 3.

5.2.3. Resultados experimentales en entornos simulados

Como primera fase para la validación del método propuesto, hemos implementado el controlador dentro de un entorno virtual. Esto lo hemos hecho a fin de verificar su funcionamiento de forma segura y para distintas condiciones. Para ello empleamos el software TORCS[1] (*The Open Racing Car Simulator*) como plataforma de prueba. TORCS es un simulador de carreras de coches, similar a un videojuego, disponible para las plataformas *Linux* y *Windows* bajo licencia GPL. Este ha sido utilizado previamente por AUTOPIA para el desarrollo del trabajo de tesis de (Onieva, 2011). Algunas de las características de este software que favorecen su aplicación en este ámbito son:

- Es altamente configurable y adaptable a las condiciones del usuario. Además, su código fuente está disponible por lo que es posible tener un mayor nivel de personalización, agregando las funcionalidades específicas que sean necesarias.
- Cuenta con un potente motor físico que simula adecuadamente parámetros como la aerodinámica, tracción y consumo de combustible de los vehículos.
- Es utilizado por una gran comunidad de desarrolladores, por lo que cuenta con un amplio abanico de vehículos, cada uno con un modelo específico con diversas características dinámicas. De igual forma dispone de varias pistas de prueba y un editor personalizado para las mismas.
- Su estructura facilita el desarrollo de "robots conductores", los cuales no son más que programas de ordenador capaces de conducir un vehículo a partir de la información proporcionada por el programa a través de una interfaz. Al estar desarrollados como programas independientes, es posible emplear cualquier técnica de control, aprendizaje u optimización para la conducción de los vehículos; siempre y cuando se respete el protocolo de comunicaciones con el programa principal.

Para el experimento, implementamos el controlador de crucero y su método de ajuste en distintos vehículos, verificando el rendimiento y adaptación del controlador ante distintas referencias de velocidad. En total, hemos utilizado 30 vehículos, cada uno de ellos con un comportamiento longitudinal específico. La figura 5.14 muestra una comparación estadística de algunos de los parámetros que influyen en la dinámica del modelo. En la imagen, las líneas rojas y los recuadros azules representan, respectivamente, el valor medio y la desviación típica de cada parámetro.

A fin de evitar perturbaciones a causa del control lateral, hemos empleado una pista con forma de óvalo para los experimentos. La ventaja de esta pista es que está formada por dos segmentos rectos de 1,6 km de largo unidos por dos semicírculos, por lo que el control lateral del vehículo es bastante simple y no afecta el rendimiento longitudinal. Por su parte, hemos simplificado un poco el control longitudinal del vehículo aplicando dos reglas simples para el

[1] http://torcs.sourceforge.net

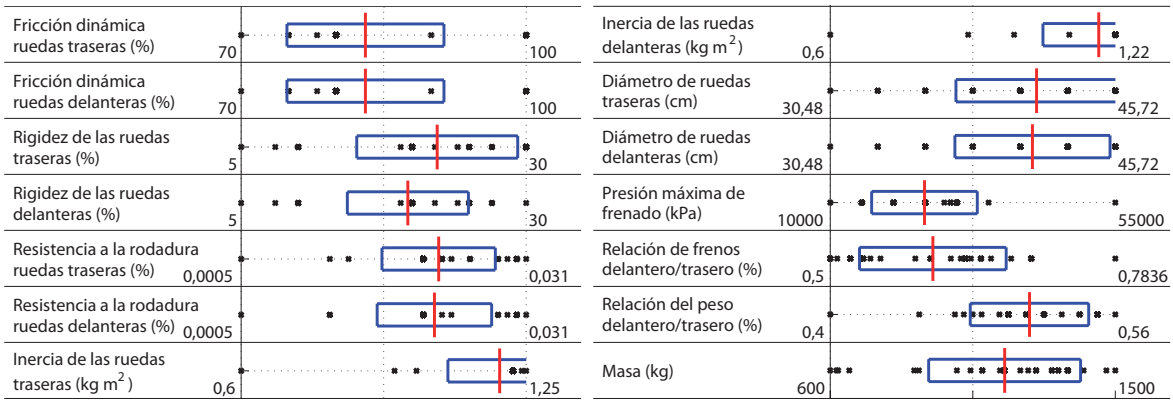


Figura 5.14: Comparación de parámetros de los diferentes vehículos empleados.

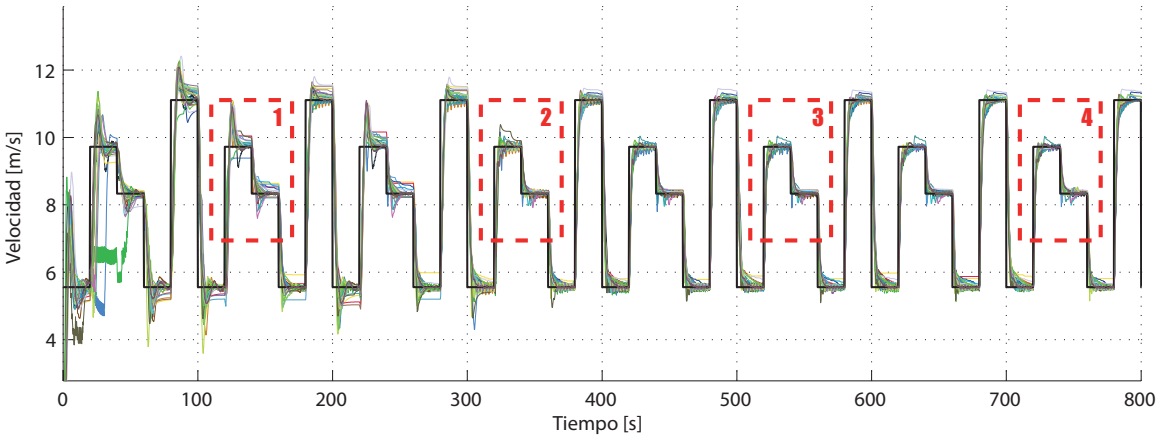
cambio de marchas: (i) aumentar la marcha cuando las revoluciones del motor superen las 4000 rpm y (ii) disminuirla cuando estén por debajo de las 2500 rpm.

Como referencia de velocidad, empleamos la secuencia $\{20, 35, 30, 20, 40\}$ km/h; estableciendo cada valor como referencia durante un periodo de 20 s y repitiendo la secuencia 8 veces, para un total de 800 s por cada test. En cuanto al controlador base, limitamos el error de velocidad dentro del rango $[-6{,}94, 6{,}94]$ m/s ($[-25, 25]$ km/h) y la aceleración en el rango $[-2{,}22, 2{,}22]$ m/s^2 (variación de 8 km/h en 1 s). Para ambas variables, el número de trapecios iniciales es 2. Por su parte, el ajuste estructural se programó para ejecutarse al final de cada 2 secuencias completas (n_e=2000).

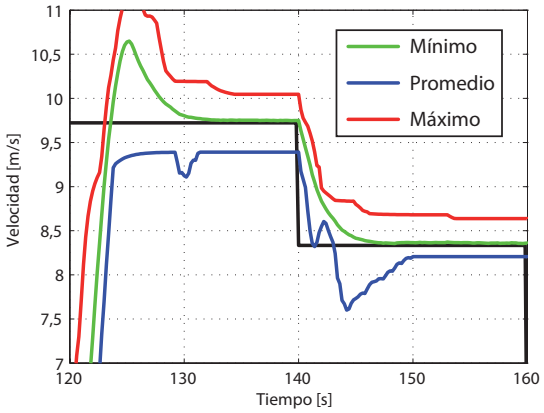
En la figura 5.15 presentamos los resultados obtenidos en esta fase de pruebas. La imagen 5.15a muestra la evolución de la velocidad para todos los vehículos junto a la referencia. Podemos apreciar claramente en esta superposición como, a nivel general, se reducen los picos de velocidad en todas las gráficas a medida que pasa el tiempo y el controlador se adapta. Es importante mencionar que, para todos los casos, el ajuste estructural se ejecutó una única vez y siempre en la primera verificación (t = 200), agregando una función de pertenencia a cada variable de entrada.

Las zonas de la gráfica que están delimitadas por rectángulos rojos e identificadas con los números 1, 2, 3 y 4 corresponden con las figuras 5.15b, 5.15c, 5.15d y 5.15e respectivamente. En estas últimas sólo presentamos las gráficas correspondientes a los errores máximo y mínimo, así como el promedio de todos los vehículos. El análisis secuencial de estas imágenes nos permite apreciar la evolución del controlador con un mayor nivel de detalle. Al inicio del test —figura 5.15b—, los cambios de referencia provocaban picos de hasta 1,5 m/s y un error estacionario en torno a los 0,4 m/s. No obstante, en la última vista ampliada —imagen 5.15e— se aprecia como el pico posterior al cambio de referencia no supera los 0,4 m/s y la diferencia estacionaria es inferior a 0,1 m/s.

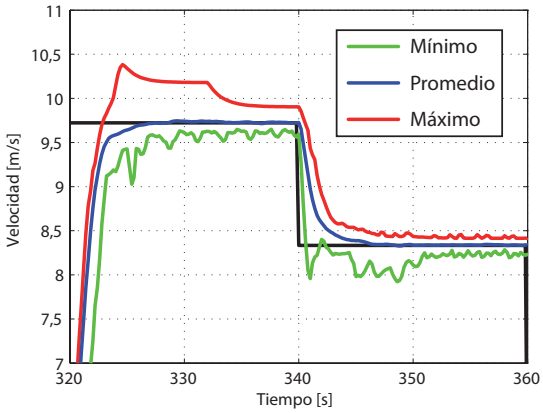
Habiendo obtenido resultados tan alentadores en entornos simulados, procedimos a verificar el funcionamiento del controlador y su método de ajuste en Platero, nuestra plataforma real de pruebas. Los detalles de esta implementación así como los resultados obtenidos se presentan en la siguiente sección.



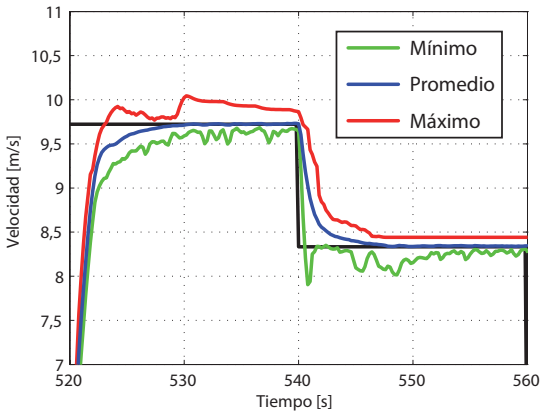
(a) Superposición de los resultados obtenidos con los 30 coches.



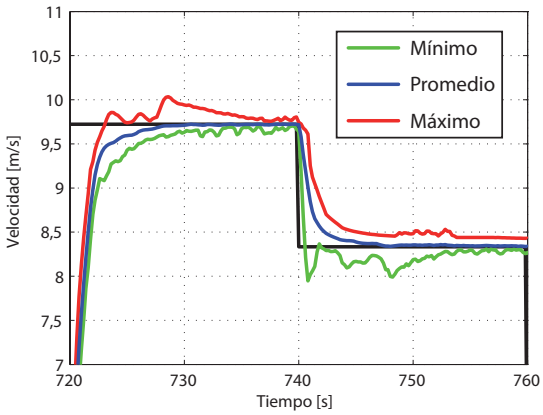
(b) Primer recuadro.



(c) Segundo recuadro.



(d) Tercer recuadro.



(e) Cuarto recuadro.

Figura 5.15: Resultados obtenidos en la simulación en TORCS.

5.2.4. Resultados experimentales con un vehículo real

Durante la segunda fase de experimentación se implementó el controlador y su método de ajuste en el vehículo de prueba, verificando su correcto funcionamiento a través de distintos test de seguimiento de la velocidad de referencia. Las pruebas se llevaron a cabo en la pista de AUTOPIA, en las instalaciones del CAR. Gracias a la arquitectura de control empleada en Platero, el traslado del controlador desde el entorno virtual al real fue casi directo. No obstante, realizamos algunas modificaciones respecto a la primera fase de experimentación, las cuales mencionamos a continuación:

- Como medida de seguridad, hemos optado por desactivar el control lateral del vehículo, dejándolo a cargo del conductor en todas las pruebas realizadas. Esto nos permite además maniobrar libremente por el circuito, adaptando el recorrido de acuerdo a la velocidad de referencia seleccionada para el vehículo.
- Dado que Platero cuenta con una caja de cambios robotizada, no hemos empleado ninguna regla de control sobre el cambio de marchas sino que hemos dejado que el propio sistema del vehículo decida cual es la marcha más adecuada.
- En el controlador base, limitamos la variable `ErrorVel` dentro del rango [−5,56**,** 5,56] m/s ([−20, 20] km/h) y la variable `Aceleración` al rango [−1,39**,** 1,39] m/s^2.
- La codificación inicial de las variables de entrada `ErrorVel` y `Aceleración` se hace con 4 y 2 trapecios respectivamente; para un total de 8 consecuentes. A partir de ahora, nos referiremos a la configuración del controlador como AxB, donde A y B corresponden al número de trapecios para las variables `ErrorVel` y `Aceleración`, respectivamente.
- Por último, se configuró el ajuste estructural cada 100 s (n_e=1000).

Con el sistema ya implementado en el vehículo, realizamos dos pruebas diferentes de control de crucero. En la primera prueba verificamos el comportamiento del controlador comparando su desempeño con el de un conductor humano, ambos frente a una referencia constante de velocidad. En cuanto a la segunda prueba, realizamos un procedimiento similar al del entorno virtual, variando la velocidad de referencia en distintos puntos de la pista de pruebas.

5.2.4.1. Comparación con conductor humano

Para la comparación con un conductor humano empleamos dos referencias de velocidad: 15 y 5 km/h (4,17 y 1,39 m/s respectivamente). El primer valor se seleccionó por su cercanía con el umbral de cambio de marchas del vehículo (de primera a segunda), determinado de forma experimental. Por su parte, el segundo valor nos permitió evaluar el rendimiento a muy bajas velocidades, donde la pendiente de la pista y/o los cambios en los actuadores afectan en mayor grado la velocidad del vehículo. En ambos casos, la duración de la prueba fue de 300 s. En el caso del conductor humano, le facilitamos la lectura del valor de la velocidad

del vehículo al proporcionarle este dato a través de la pantalla del ordenador a bordo. Lejos de determinar cual es la mejor opción para el control, el objetivo de esta prueba es verificar que el comportamiento del controlador desarrollado sea similar a la forma de conducir de una persona.

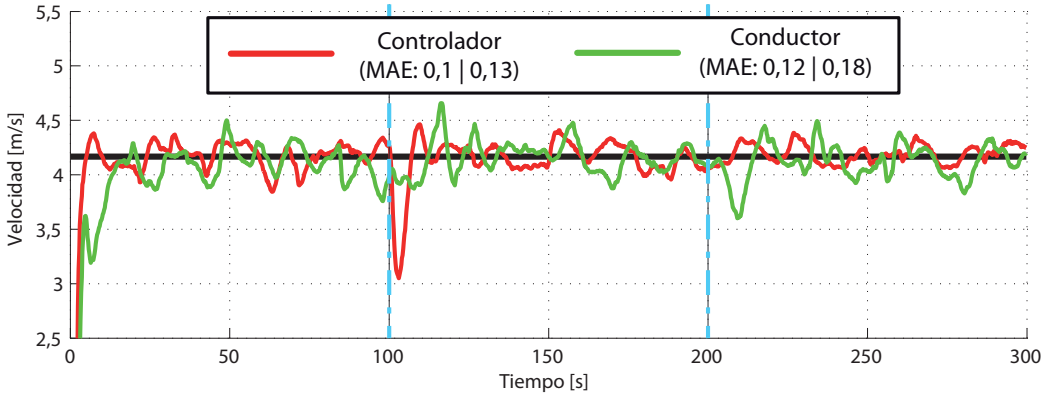
En la figura 5.16 presentamos los resultados obtenidos durante la primera prueba, con una velocidad de referencia de 4,17 m/s. Las gráficas de evolución de la velocidad para el controlador y el conductor humano se muestran en la imagen 5.16a en rojo y en verde, respectivamente. En la leyenda de esta imagen podemos apreciar además dos valores numéricos para cada gráfica. Ambos valores corresponden al MAE de la gráfica respecto a la referencia de velocidad, con la diferencia de que el primero sólo considera los datos a partir de t=25 s, despreciando la parte transitoria. Al comparar los valores obtenidos para cada caso, observamos que el rendimiento del controlador es equiparable al de un conductor humano, con un error levemente inferior.

Por otro lado, podemos apreciar que existe un pico negativo en la respuesta del controlador en la zona inmediatamente posterior a t=100s. Este pico se debe al ajuste estructural del controlador, donde se pasó de una configuración $4x2$ a una $5x3$. El efecto del ajuste lo podemos constatar en la gráfica de la salida del controlador —imagen 5.16b— donde se aprecia un salto al valor 0 en el mismo instante, provocado por el reinicio de los valores de los *singletons*. En lo que respecta a la imagen 5.16c, esta nos muestra la evolución de los valores asignados a cada consecuente de la variable de salida. En ella también se puede ver claramente el efecto del primer ajuste estructural en el instante t=100s. Aunque no se aprecia en ninguna de las gráficas anteriores, es importante mencionar que durante la segunda fase de ajuste estructural (t=200 s) se realizó la reducción de las funciones de pertenencia centrales de ambas entradas.

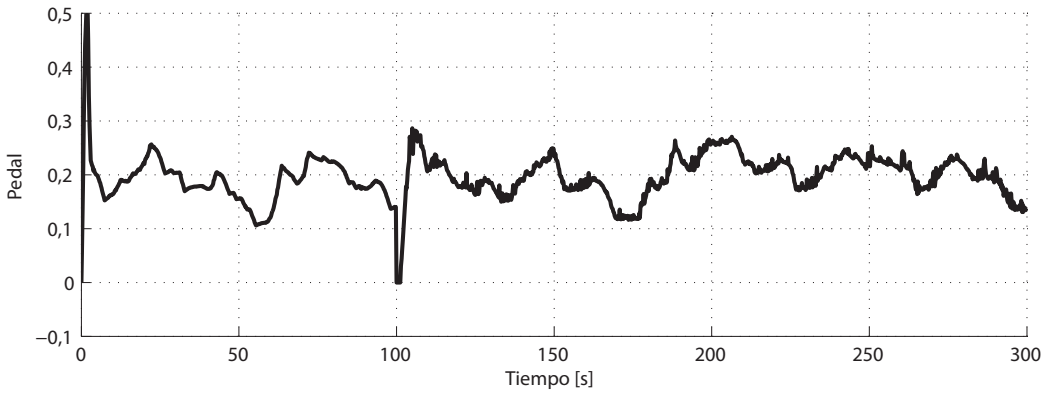
En lo que respecta a la prueba con la segunda referencia, los resultados se presentan en la figura 5.17. En la imagen 5.17a podemos apreciar como, a pesar del sobrepico inicial y la oscilación provocada por el primer ajuste estructural, el rendimiento del controlador se asemeja al del conductor humano, siendo levemente superior. Alrededor del instante t=150 s observamos una oscilación tanto en la evolución del controlador como la salida del mismo —imagen 5.17b—. Esta se debe a una pequeña zona de la pista de prueba en la que existe una inclinación de aproximadamente 3 %, lo cual provoca que el vehículo supere la velocidad de referencia sin ningún tipo de actuación sobre el acelerador, siendo incluso necesario actuar brevemente sobre el freno. Igualmente, podemos apreciar en la imagen 5.17c como el *singleton* conrrespondiente a la combinación de las funciones de pertenencia centrales —linea negra— presenta las mismas oscilaciones por debajo del valor 0. Una vez fuera de esta zona, su posición se ajusta nuevamente hasta alcanzar valores positivos, necesarios para mantener la velocidad del vehículo.

5.2.4.2. Variación de la velocidad de referencia

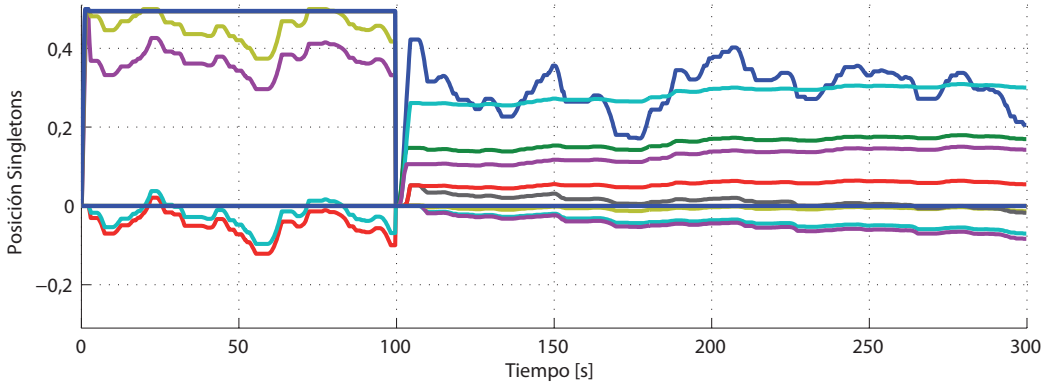
Para la segunda prueba con el vehículo real, variamos la referencia de velocidad crucero a su paso por distintas zonas de la pista. La figura 5.18 muestra una imagen de la trayectoria seleccionada, identificando sobre ella los puntos donde se producen los cambios. El punto



(a) Comparación de los resultados entre el controlador y el conductor humano.

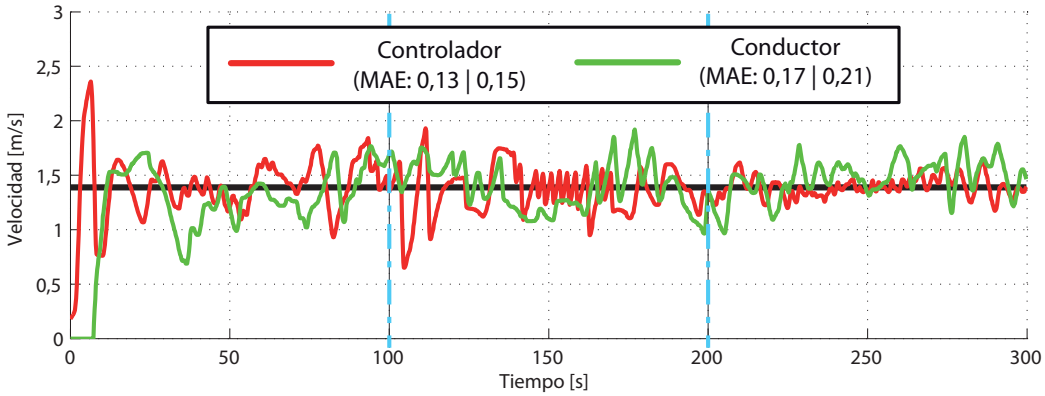


(b) Salida del controlador en función del tiempo.

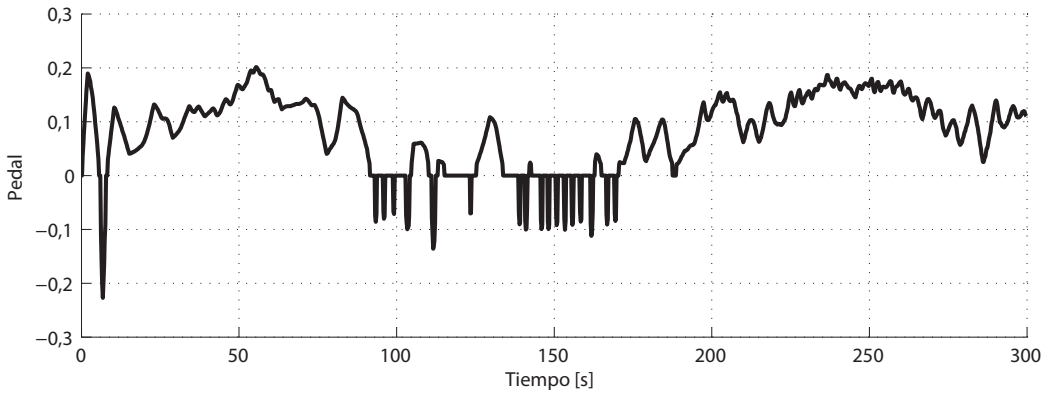


(c) Evolución de los *singletons* del controlador.

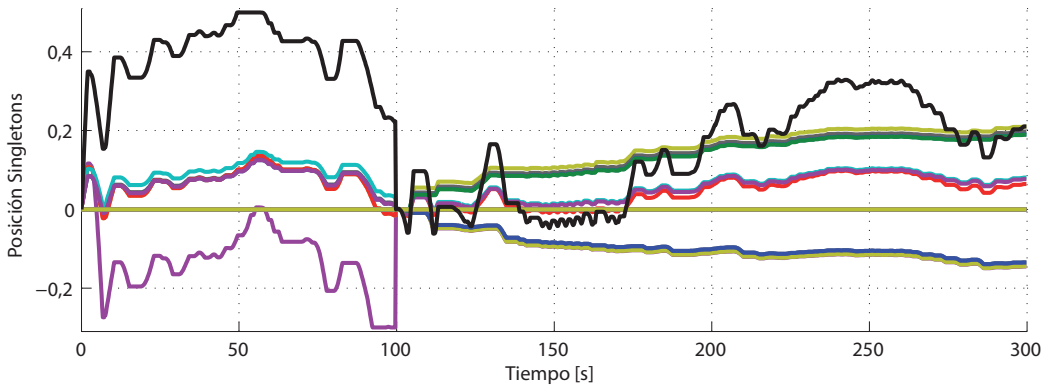
Figura 5.16: Resultados del control de crucero a velocidad constante (15 km/h).



(a) Comparación de los resultados entre el controlador y el conductor humano.



(b) Salida del controlador en función del tiempo.



(c) Evolución de los *singletons* del controlador.

Figura 5.17: Resultados del control de crucero a velocidad constante (5 km/h).

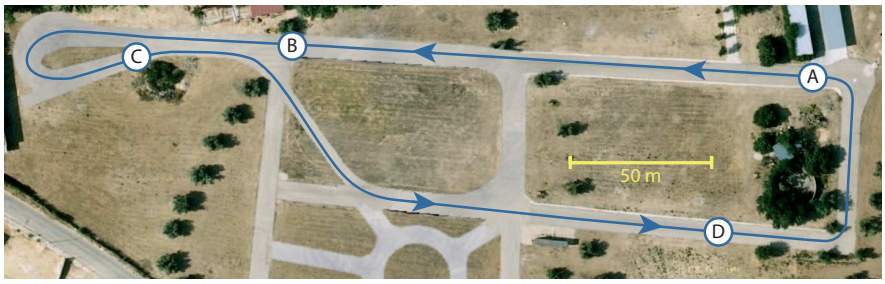


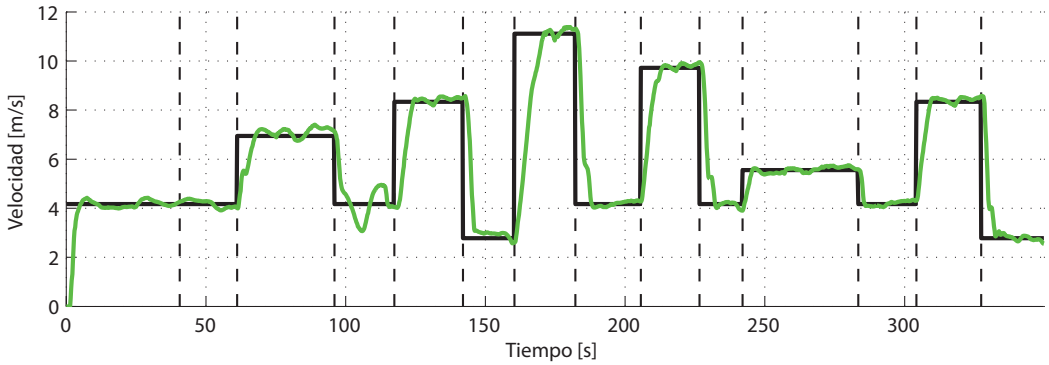
Figura 5.18: Recorrido y puntos de control para el experimento.

A se corresponde además con el punto de inicio de la prueba. Salvo en su primer paso por el punto B, variamos manualmente la velocidad del vehículo en cada punto, seleccionando valores entre 5 y 40 km/h (1,39 y 11,11 m/s).

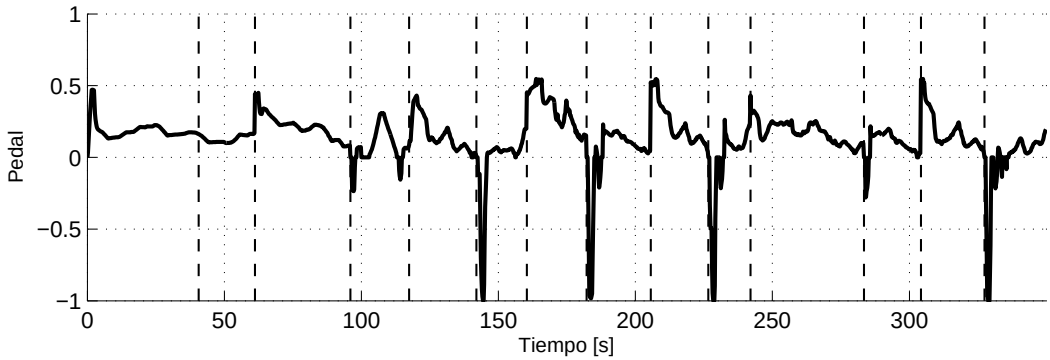
Los resultados obtenidos se presentan en la figura **??**. En cada gráfica, las líneas verticales discontinuas identifican el paso del vehículo por los puntos de control. En general, se observa que el vehículo es capaz de seguir de forma adecuada la referencia variable, disminuyendo notablemente el error de seguimiento luego de t=100s; es decir, luego del primer ajuste estructural. Al igual que en las pruebas realizadas con referencia constante, este ajuste modificó la configuración del controlador de $4x2$ a $5x3$. Por su parte, en el segundo ajuste (t=200s) sólo se modificó la función de pertenencia central de la variable `ErrorVel`.

A fin de facilitar el análisis de los resultados, presentamos en la figura 5.20 el error promedio obtenido con cada velocidad de referencia. En la gráfica superior agrupamos los valores asociados a los incrementos de velocidad y en la gráfica inferior los asociados a su reducción. Además del valor promedio para cada segmento, presentamos también los valores obtenidos durante la parte transitoria y estacionaria de cada seguimiento. En estas imágenes se puede apreciar una leve diferencia entre las fases de acelerado y frenado del vehículo, obteniendo una mejor respuesta en el último caso. Por un lado, esto se debe a que la capacidad de aceleración del vehículo es inferior a la de frenado; y por otro lado, a que el umbral de aceleración permitido para el frenado es superior. Por último, se puede ver como el error disminuye progresivamente desde el inicio de la prueba, llegando a alcanzar valores en torno a 0,1 m/s al final de la misma —en estado estacionario—.

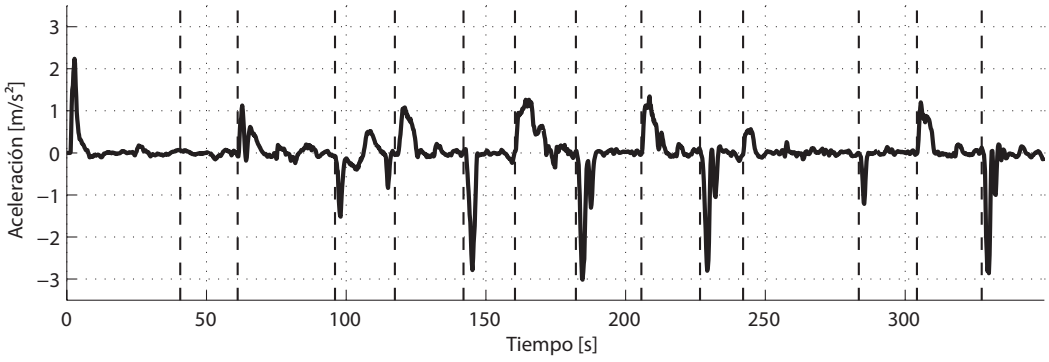
A partir de estos resultados obtenidos con el vehículo real, verificamos que el método de ajuste es adecuado para el control de crucero del vehículo, tal y como habíamos validado también durante la fase de experimentación en entornos virtuales.



(a) Evolución de la velocidad respecto a la referencia.

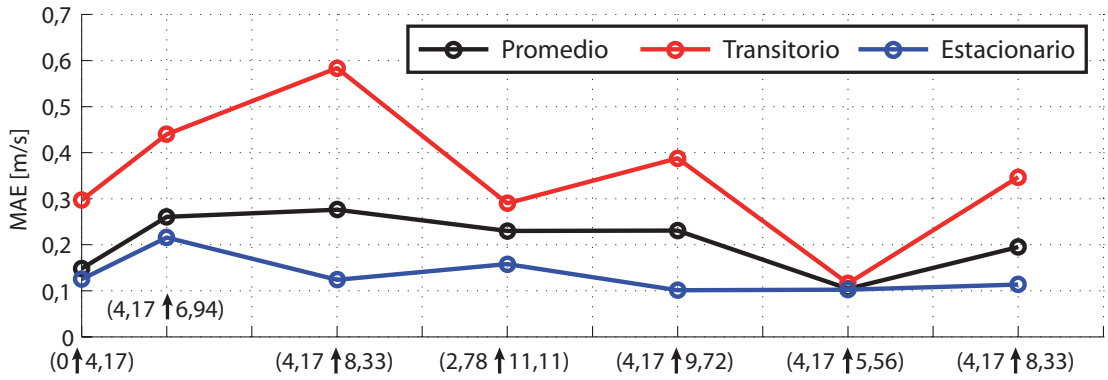


(b) Salida del controlador.

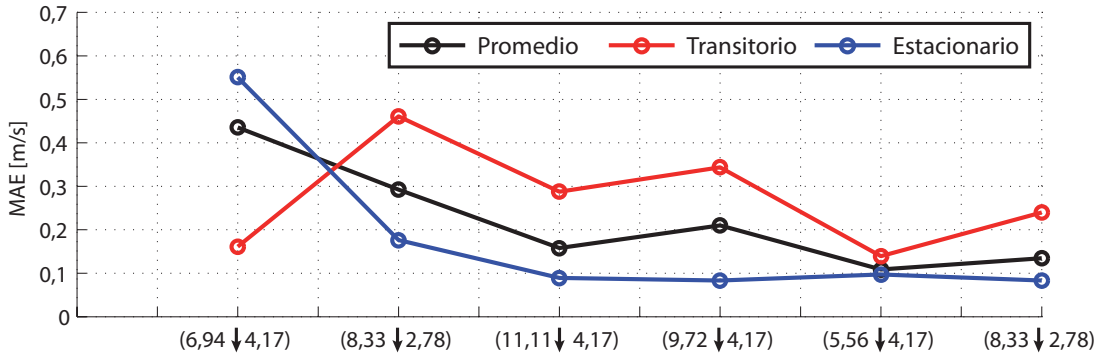


(c) Aceleración del vehículo.

Figura 5.19: Resultados del control de crucero con cambios en la referencia.



(a) Incrementos de velocidad consigna.



(b) Reducción de velocidad consigna.

Figura 5.20: Error promedio del control de crucero con cambios en la referencia.

CAPÍTULO 6

DEMOSTRACIÓN DE CONDUCCIÓN AUTÓNOMA

Con objeto de verificar el funcionamiento, viabilidad y robustez de la arquitectura, algoritmos y métodos presentados; se ha organizado una demostración en carreteras abiertas al tráfico. La idea tras esta prueba ha sido someter los elementos de nuestro sistema a condiciones más reales a las permitidas en cualquiera de los circuitos privados que han sido utilizados hasta ahora por AUTOPIA (Naranjo, 2005; Milanés, 2010; Pérez, 2012). Durante la demostración completamos un recorrido autónomo de aproximadamente 100 km de longitud, desde la población de San Lorenzo de El Escorial hasta las instalaciones del CAR en Arganda del Rey. Para ello implementamos un esquema lider–seguidor basado en comunicaciones, donde un primer vehículo transmite su trayectoria en tiempo real a la vez que un segundo vehículo emplea esta información como referencia para su recorrido.

El contenido de este capítulo está organizado de la siguiente manera: primero, presentamos los detalles de la demostración y la selección de la ruta más adecuada para la misma. En segundo lugar, describimos el sistema de comunicaciones empleado para la transmisión de la información. Posteriormente, procedemos a explicar el método empleado para la generación de la trayectoria y los detalles del control implementado en el vehículo. Por último, presentamos el análisis de los resultados obtenidos en la demostración.

6.1. La prueba y su organización

En el momento de plantear la demostración del sistema de conducción autónoma en carreteras abiertas, consideramos dos criterios que resultaron claves para la selección de la ruta: (i) su longitud debía ser suficiente para demostrar la robustez del sistema ante tiempos prolongados de funcionamiento y (ii) debía combinar tanto entornos urbanos como autovías, a fin de validar las capacidades del sistema en ambas situaciones. Así, nuestra primera opción fue realizar un recorrido de 140 km desde la población de Segovia hasta las instalaciones del CAR en Arganda del Rey. No obstante, esta ruta incluye carreteras pertenecientes a dos comunidades autónomas —Madrid y Castilla y León— y un segmento de una autovía de peaje

—38 km aproximadamente— cuya gestión pertenece a una empresa privada, dificultando la tramitación de los permisos y autorizaciones pertinentes para la realización de la prueba. Por este motivo, y a sugerencia de la Guardia Civil, optamos por trasladar el punto de partida a San Lorenzo del Escorial, al término de la autovía de peaje.

La figura 6.1 muestra la ruta definida finalmente para la demostración, con una longitud aproximada de 100 km. Además de la trayectoria resaltada con la línea azul, podemos apreciar en la imagen 6 puntos de control identificados con las letras de la A a la F, los cuales denotan hitos relevantes de la ruta:

- A: el punto de partida del recorrido, en la explanada del monasterio de San Lorenzo de El Escorial.
- B: la incorporación desde la carretera M-600 a la autovía A-6. En este punto concluye el primer segmento de recorrido en entornos urbanos, de aproximadamente 10 km de longitud.
- C: la salida 23 de la autovía A-6, donde realizamos la incorporación a la autovía M-50 a la altura del kilómetro 83. Este cambio a la circunvalación permite, por un lado, extender la longitud de la ruta bordeando la ciudad de Madrid y, por otro lado, evitar parcialmente el complicado nivel de tráfico de esta última. La distancia recorrida desde el punto anterior es de 25 km aproximadamente.
- D: la incorporación a la autovía A-3 desde la salida 31 de la M-50.
- E: la salida 22 de la autovía A-3, correspondiente a la población de Arganda del Rey y al último tramo de entorno urbano de la demostración. En torno a 8 km de distancia del punto anterior.
- F: las instalaciones del CAR en Arganda del Rey, punto final del recorrido y del último segmento de entornos urbanos, de aproximadamente 2 km de longitud.

Tanto la selección del itinerario como la organización general de la demostración se llevaron a cabo en colaboración con el ayuntamiento de San Lorenzo de El Escorial, la DGT y la Guardia Civil de Tráfico. La participación de todas estas entidades fue un requisito fundamental para la realización del experimento debido a que la legislación española vigente no permite la circulación de ningún tipo de vehículo sin conductor. Por lo tanto, fue necesario que los vehículos empleados en la demostración circularan bajo la condición de transporte especial, teniendo un límite de velocidad inferior al permitido normalmente y estando escoltados en todo momento por unidades de la Guardia Civil de tráfico.

6.2. Sistema de comunicaciones

En lo que respecta al intercambio de información entre los vehículos durante la prueba, hemos empleado el sistema de comunicaciones implementado para la participación del equipo

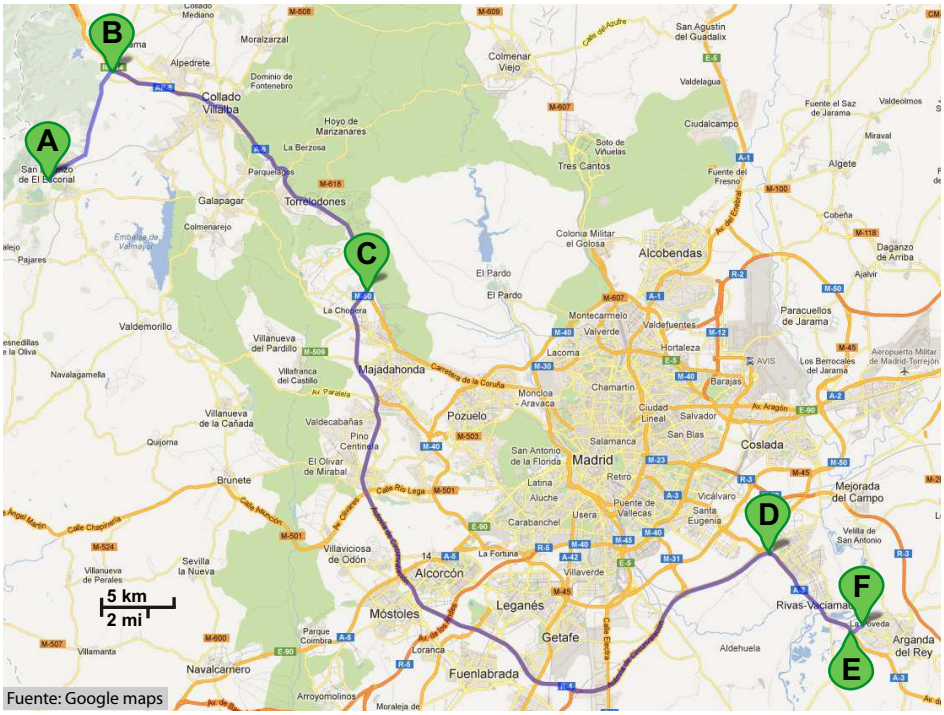


Figura 6.1: Itinerario de la demostración de conducción autónoma.

en la competición GCDC en 2011 (van Nunen et al., 2012; Pérez, 2012). Este sistema está basado en el estándar IEEE 802.11p (Jiang y Delgrossi, 2008) y el protocolo CALM (ISO, 2010; Williams, 2004), ambos diseñados especialmente para la comunicación entre vehículos.

A nivel de hardware, el sistema se compone de un ordenador embebido, una tarjeta Wi-Fi *Microtik RH52* y una antena externa cuya ganancia es de 9 dBi en la frecuencia de 5,9 GHz. Por defecto, la tarjeta empleada sólo tiene capacidad de operación bajo las especificaciones a, b y g del estándar IEEE 802.11. No obstante, a través de una modificación del controlador es posible ajustar sus parámetros de operación de acuerdo a la especificación p del estándar. Tanto este ajuste del controlador, como el desarrollo general del software de comunicaciones para el ordenador embebido, fueron tareas asignadas al consorcio encargado del GCDC (de Jongh, 2011). En la figura 6.2 se muestran cada uno los elementos de hardware mencionados.

Desde el punto de vista del ordenador de control, el funcionamiento de la unidad de comunicaciones es totalmente transparente; ya que esta última hace las veces de puerta de enlace entre la interfaz inalámbrica CALM/802.11p y la red cableada del vehículo. A través de una conexión UDP con el sistema de comunicaciones, cualquier programa ejecutado en el ordenador de control tiene la capacidad de intercambiar información con el resto de vehículos en la red inalámbrica, sin considerar ningún detalle relacionado con la codificación y transmisión de los datos.

Para nuestra aplicación diseñamos una estrutura de datos propia, declarada bajo el nombre de `sEstadoCoche` —Figura 6.3—. En esta estructura empaquetamos la información relacionada con el estado del vehículo, es decir, las coordenadas exactas de su posición, su

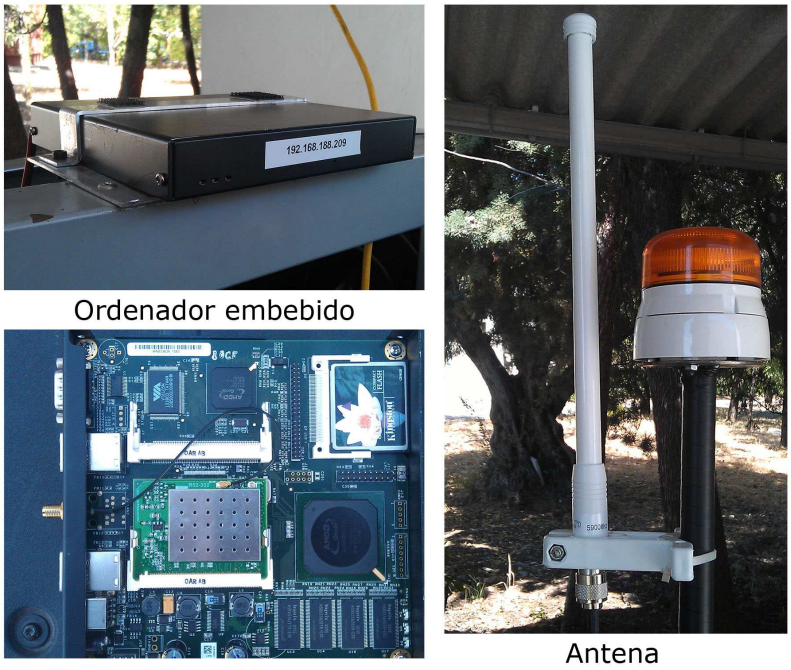


Figura 6.2: Elementos de hardware del sistema de comunicaciones.

velocidad, su orientación y la calidad del posicionamiento. Así mismo, incluimos un par de variables auxiliares que se emplean en función de la aplicación final del sistema. Por ejemplo, para transmitir la intención de giro en un cruce, el ID del coche líder en un *platooning* o cualquier otra información no contemplada directamente.

Por último, debemos mencionar que gracias a este sistema de comunicaciones hemos logrado salvar distancias de hasta 500 m entre los vehículos, sin llegar a tener pérdidas de paquetes de datos en ningún momento. No obstante, este rendimiento puede disminuir o aumentar en función del estado del entorno y del número de vehículos que transmiten información (Milanés et al., 2012b).

```
struct can_modulo_estado
{
    double UTMEste;             // Coord. Este
    double UTMNorte;            // Coord. Norte
    double Orientacion;         // Orientación
    double Velocidad;           // Velocidad
    double HoraGPS;             // Hora actual del GPS
    double Altura;              // Altura
    double Calidad;             // Calidad de posicionamiento
    double Auxiliar1;           // Variable auxiliar 1
    double Auxiliar2;           // Variable auxiliar 2
    unsigned int TCoche_seg;    // Tiempo del ordenador
    unsigned int TCoche_nseg;   // Tiempo del ordenador
    unsigned short Checksum;    // Checksum del mensaje
    unsigned char ID_Coche;     // ID del emisor
}
```

Figura 6.3: Estructura de datos definida para las comunicaciones.

6.3. Generación de trayectoria

Cómo se mencionó en el capítulo 3, la arquitectura planteada cuenta con dos módulos que son claves para la planificación de la trayectoria del vehículo: el módulo misión, que define una lista de objetivos, y el módulo planificador, encargado de convertir la misión en una trayectoria de referencia para el control. En el caso de esta aplicación de seguimiento, utilizamos el módulo misión para almacenar un registro de la trayectoria seguida por el vehículo líder. La información la extraemos a partir de los mensajes intercambiados continuamente por el sistema de comunicaciones de los vehículos. En concreto, introducimos las posiciones del vehículo líder como objetivos del vehículo seguidor. En paralelo a esto, el módulo planificador procesa los datos almacenados por la misión y genera una trayectoria adecuada para el control.

A continuación presentamos dos consideraciones que tuvimos en cuenta a la hora de definir la misión del vehículo seguidor:

- Durante la inicialización del sistema incluimos la posición del vehículo seguidor y del vehículo líder como los dos primeros puntos de la misión. De esta forma el planificador puede iniciar la generación de la trayectoria de referencia desde el momento en que se recibe el primer mensaje desde el vehículo líder. Como medida de seguridad, el sistema impide su activación si no se recibe ningún tipo de información del vehículo líder, notificando además al conductor en el momento en que se cumple esta condición.
- A fin de reducir el número de puntos almacenados en la misión y facilitar la tarea de planificación, consideramos una separación mínima d_{min} entre dos puntos de misión consecutivos. Así, descartamos todos aquellos mensajes donde la posición no cumple esta condición respecto al último punto añadido. El valor asignado a d_{min} supone un compromiso entre el número de puntos a procesar y la precisión con la que se registra la trayectoria del líder. Para la demostración le asignamos el valor de 0,15 m, la décima parte del valor por defecto asignado a la separación de los puntos de planificación (sección 3.7.2).

En la figura 6.4 mostramos un ejemplo de la generación de la trayectoria de referencia. En la imagen, la línea azul representa el recorrido real del líder y las marcas verdes los puntos que definen la trayectoria de referencia para el vehículo seguidor. En lo que respecta a la vista ampliada, las marcas azules representan los puntos de misión una vez considerados los criterios de separación y velocidad mínima. En la imagen podemos apreciar que, independientemente del número de puntos de misión, el planificador extrapola la información para mantener la equidistancia de los puntos de referencia. Los datos presentados en la gráfica se corresponden a una prueba real en entornos urbanos, con velocidades entre 10 y 30 km/h.

6.4. Control del vehículo

Como ha sido habitual durante el desarrollo de este trabajo, dividimos el control del vehículo en dos secciones: longitudinal y lateral. A continuación presentamos los detalles de

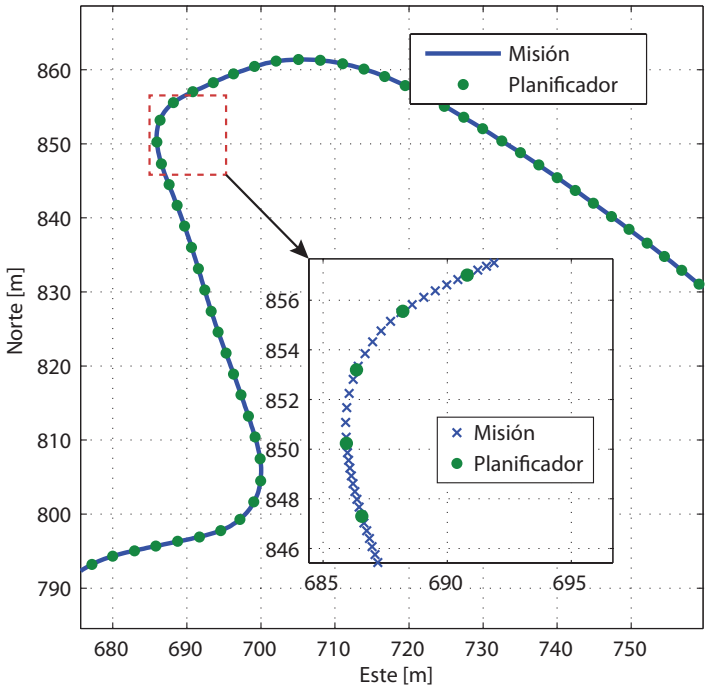


Figura 6.4: Ejemplo de la planificación de trayectoria.

cada una de ellas.

6.4.1. Control Longitudinal

Desde el punto de vista del control longitudinal, el comportamiento requerido para el vehículo seguidor durante la demostración corresponde al de un sistema CACC con capacidad *Stop&Go*, ya que mientras el vehículo líder indica la trayectoria de referencia, este debe seguirlo manteniendo una separación prudencial con el mismo. No obstante, dado que la prueba incluye tanto entornos urbanos como autovías, el sistema de control debe ser capaz de funcionar dentro de un rango de velocidad de 0 a 100 km/h. Así mismo, debemos garantizar el comportamiento adecuado en todo momento, reduciendo el efecto de ciertas perturbaciones externas como la pendiente del camino o cambios en la resistencia aerodinámica.

Ante estos requisitos, hemos particularizado el controlador CC desarrollado en la sección 5.2, adaptándolo a la tarea del control CACC. Para ello, modificamos su configuración de entradas, considerando (i) la velocidad relativa entre los vehículos (V_r) y (ii) el error de separación entre ellos (d_e). Este último se calcula como la diferencia entre la distancia medida entre los coches d_m y la distancia de referencia d_r. Para la salida del controlador empleamos la misma configuración del CC: una única variable de salida `Pedal` definida en el rango $[-1, 1]$, la cual regula simultáneamente las acciones sobre el freno y el acelerador. Es importante mencionar que, en este caso, el objetivo no es generar un nuevo controlador sin ningún tipo de intervención, sino mejorar el rendimiento de un controlador CACC que ha sido ajustado manualmente para la aplicación.

Por razones de seguridad, la distancia d_r varía en función de la velocidad del vehículo

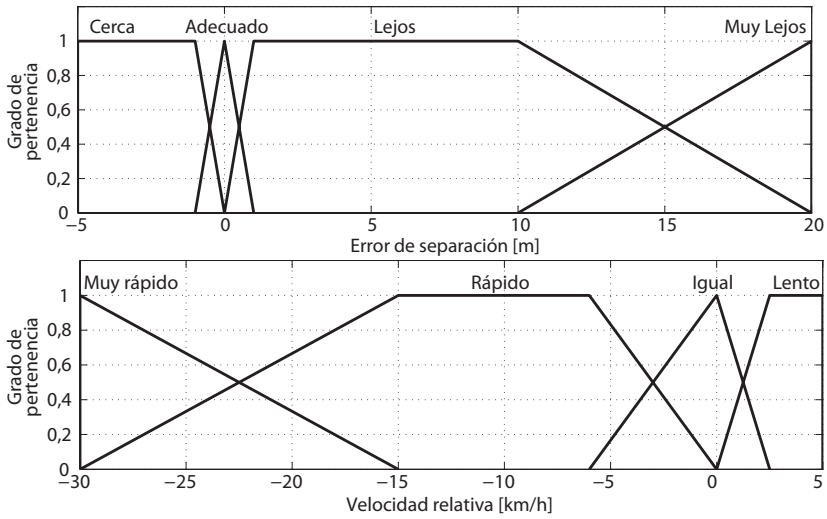


Figura 6.5: Funciones de pertenencia para las entradas del controlador CACC.

líder, de forma tal que:

$$d_r = d_o + \Delta_t \cdot v_t \tag{6.1}$$

donde d_o es la distancia mínima que debe existir entre los vehículos cuando estos están detenidos —en metros—, v_t es la velocidad del vehículo líder —en m/s— y Δ_t es la separación temporal requerida entre los vehículos —en s—. Los valores de la distancia mínima d_o y la separación temporal Δ_t los hemos ajustado de acuerdo a las normas de seguridad que se emplearon durante el GCDC: 10 m y 0,6 s respectivamente (Pérez, 2012). Por otro lado, la distancia medida d_m la calculamos como la separación real que existe entre los vehículos, medida sobre la trayectoria de referencia. De esta forma evitamos el cálculo euclidiano de la distancia, ya que en curvas muy cerradas este valor difiere en gran medida de la distancia real y generaría la respuesta inadecuada del controlador.

Para el ajuste inicial del controlador, realizamos varias pruebas de seguimiento con velocidades comprendidas en el rango de 10 a 90 km/h. En cada prueba, el vehículo lider mantenía una velocidad crucero y se observaba el rendimiento general del controlador CACC en el vehículo seguidor, ajustando las funciones de pertenencia de las entradas y/o la posición de los consecuentes en caso de ser necesario. Este proceso se repitió hasta conseguir un error de seguimiento inferior a 2 m en todo el rango de velocidades. La figura 6.5 muestra la configuración final de las funciones de pertenencia definidas para las entradas. Por su parte, la posición final de los consecuentes se muestra en la tabla 6.1.

En cuanto al método de adaptación, realizamos dos modificaciones respecto a su aplicación al control de crucero: (i) se omite la fase estructural y (ii) restringimos el máximo desplazamiento de los consecuentes del controlador, tanto por ciclo como de forma absoluta. Con la primera modificación evitamos alterar las funciones de pertenencia que han sido ajustadas manualmente para cada entrada. Así mismo, la segunda consideración limita el rango de variación permitido para cada consecuente, a fin de que estos no se alejen excesivamente desde

Error de separación	**Velocidad relativa**			
	Lento	*Igual*	*Rápido*	*Muy Rápido*
Cerca	0	-0,6	-1	-1
Correcto	0,25	0	-0,6	-1
Lejos	0,8	0,5	0	-1
Muy Lejos	0,8	0,8	0,8	0,215

Tabla 6.1: Posición inicial de los *singletons* del controlador CACC.

su posición inicial ni modifiquen bruscamente su posición de un ciclo al siguiente. El máximo desplazamiento por ciclo lo limitamos a ±0,05 en torno a la posición original; mientras que el máximo desplazamiento absoluto es de ±0.2 —10 % del rango de la variable—.

Para el ajuste de consecuentes aplicamos un criterio de recompensa similar al empleado para el CC, ajustando la posición de los *singletons* de acuerdo al grado de activación de su regla asociada. Así, en cada ciclo k la nueva posición se calcula de acuerdo a:

$$S_k^i = S_{k-1}^i + \mu_{k-1}^i R\left(d_{c_k}, v_{r_k}\right) \tag{6.2}$$

donde S_k^i es la posición del consecuente i en el instante k, μ_{k-1}^i el grado de activación en el instante anterior y $R\left(d_{c_k}, v_{r_k}\right)$ la recompensa asignada en función de ambas variables de entrada del controlador. Este último valor se calcula a través del siguiente razonamiento:

- Si el error d_c es menor al umbral $-u_d$, el vehículo está mas cerca de lo recomendado. En este caso:
  - Si la velocidad relativa v_r es mayor al umbral u_v, el vehículo circula más lento que el líder, por lo que no modificamos los consecuentes ya que ambos errores se compensan (Zona I).
  - No obstante, si v_r es menor al umbral $-u_v$, el vehículo tiene una velocidad mayor a la del líder, así que debemos desplazar sus consecuentes hacia el lado negativo (frenado): $R = C \cdot v_r$ (Zona II).
- Si d_c es mayor al umbral u_d, el vehículo está más lejos de lo recomendado. Ahora:
  - Si v_r es mayor al umbral u_v, el vehículo circula más lento que el líder, por lo que directamente aumentamos los consecuentes del controlador. En este caso $R = C \cdot d_c$ (Zona III).
  - Si v_r es menor al umbral $-u_v$, circulamos más rápido que el líder. No obstante:
    - Si el error d_c es mayor a d_o, entonces estamos muy lejos del líder, por lo que intentamos darle alcance con velocidad relativa $v_a < 0$. Ahora:
      - Si circulamos con $v_r < v_a - u_v$, la velocidad del seguidor sigue siendo excesiva, así que reducimos los consecuentes con $R = C \cdot (v_r - v_a)$ (Zona IV).

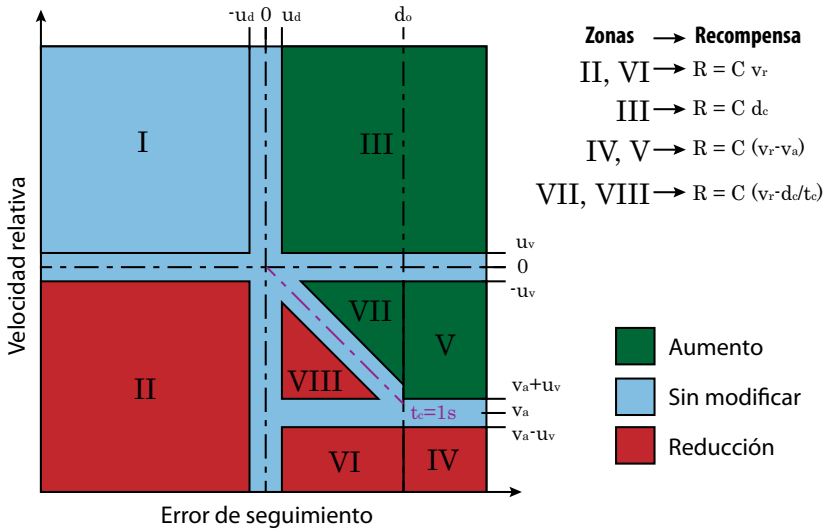


Figura 6.6: Zonas de recompensa definidas para el controlador CACC.

- ◇ Y si circulamos con $v_r > v_a + u_v$, entonces no hemos llegado al valor de la velocidad de alcance, así que aumentamos los consecuentes con $R = C \cdot (v_r - v_a)$ (Zona V).

- Si el error d_c es menor a d_o, ya nos estamos aproximando al líder, por lo que debemos reducir la velocidad relativa a medida que nos acercamos, hasta alcanzar $v_r = 0$ cuando $d_c = 0$. Procedemos a verificar:
  - ◇ Si $v_r < v_a - u_v$, la velocidad relativa es excesiva, así que directamente reducimos la acción de los consecuentes con $R = C \cdot v_r$. (Zona VI).
  - ◇ En caso contrario, verificamos si $v_r > -d_c/t_c + u_v$ o $v_r < -d_c/t_c - u_v$ entonces la velocidad no permite compensar el remanente del error. O circulamos más lento que el líder estando aún mas lejos de lo indicado por la referencia —primer caso (Zonas VII)— o bien más rápido cuando estamos más cerca de lo debido —segundo caso (Zonas VIII)—. En esta situación compensamos con $R = C \cdot (v_r + d_c/t_c)$.

donde t_c es el tiempo de confort establecido para la reducción de la velocidad relativa en el caso del alcance. La figura 6.6 muestra una imagen de la distribución de las zonas de recompensa para $t_c = 1$ s. Los umbrales u_d y u_v los configuramos en 0,2 m y 0,5 km/h respectivamente.

6.4.2. Control Lateral

En el momento de abordar el problema del control lateral para la demostración, el sistema presentado en la sección 5.1 se encontraba aún en fase de desarrollo y, a pesar de los alentadores resultados obtenidos en simulación, su aplicación directa al vehículo real resultó inviable desde el punto de vista práctico y de seguridad —ver sección 5.1.5—. Por otro lado, previamente el Programa AUTOPIA había demostrado de forma exitosa sus sistemas de control

en entornos totalmente controlados, destacando la demostración final del proyecto CyberCars2 (Milanés et al., 2011a) y la participación en el GCDC (Pérez, 2012). No obstante, esta es la primera demostración orientada a validar el funcionamiento de la nueva arquitectura propuesta en cuanto a fiabilidad y robustez en entornos semicontrolados. Así, el controlador lateral implementado parte de los resultados previos de AUTOPIA, realizando diversas modificaciones en el diseño a fin de cubrir un mayor rango de velocidades y curvaturas, permitiendo la circulación tanto en entornos urbanos como en autovías. En este sentido, hemos considerado la diferencia de comportamiento que existe entre los escenarios a baja velocidad y aquellos a alta velocidad. Si bien en entornos urbanos es necesario que el vehículo pueda maniobrar con un rango amplio de la dirección, a medida que aumentamos la velocidad se requieren movimientos más suaves y con un rango de desplazamiento menor. Teniendo esto en cuenta, definimos dos controladores laterales: uno para el caso de velocidades bajas y otro para situaciones con velocidad moderada-alta. Ambos están basados en lógica borrosa y se combinan en mayor o menor grado en función de la velocidad del vehículo.

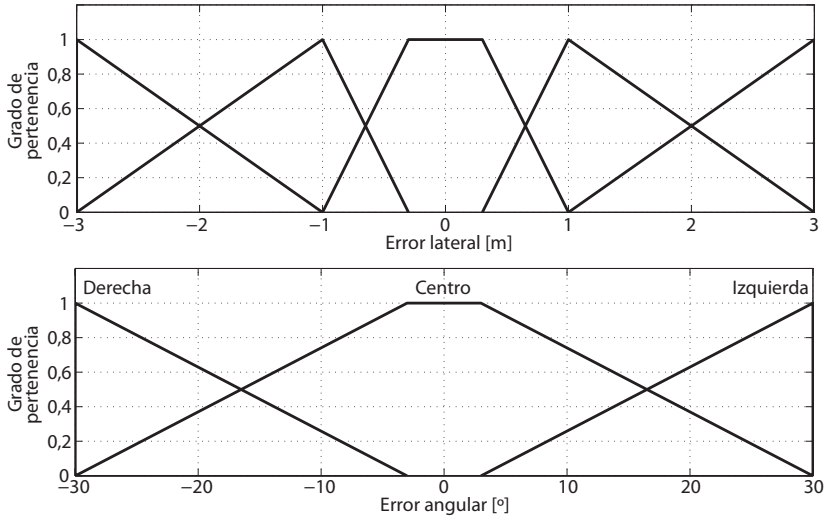
Hasta ahora, el enfoque habitual de AUTOPIA para el control lateral consistía en clasificar los tramos rectos y curvos de la trayectoria, empleando una configuración de control específica para cada caso (Naranjo et al., 2005). Esto se hacía a fin de mejorar la estabilidad del vehículo en las secciones rectas, sin comprometer el rendimiento en las secciones curvas de la trayectoria. No obstante, el problema principal de esta configuración es la discontinuidad que produce en la salida, provocada por el cambio de controlador de un tramo a otro. Así mismo, los tramos curvos presentaban oscilaciones de la referencia, causadas por la corrección total de ambos errores. Esto último se debe a que cuando los errores se acercan a cero, la respuesta del controlador es centrar el volante, lo cual en una sección curva conlleva nuevamente al aumento del error. Para corregir esto, el controlador desarrollado para esta aplicación determina una referencia inicial en función de la curvatura de la trayectoria, ajustándola posteriormente de acuerdo a los valores de error obtenidos.

La curvatura γ asociada a cada segmento de referencia la estimamos aproximando la trayectoria a una circunferencia. Para ello usamos la información de tres puntos: el inicio del segmento anterior, el fin del segmento actual y el punto común a ambos segmentos. Una vez estimada la curvatura, determinamos la referencia inicial δ_0 con la siguiente ecuación:

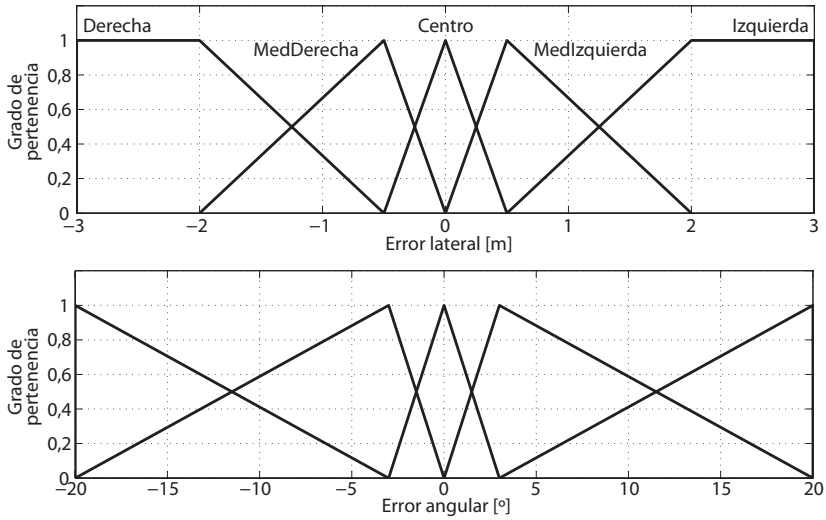
$$\delta_0 = \alpha \cdot \arctan(\gamma B_C) \tag{6.3}$$

donde B_C es la batalla del vehículo —distancia entre los ejes delantero y trasero— y α es la relación de transmisión entre el volante y las ruedas del vehículo. En el caso de Platero, determinamos experimentalmente que $\alpha = 16{,}36$.

En lo que respecta a los controladores borrosos implementados, ambos consideran el error lateral y angular del vehículo respecto a la trayectoria —deriva y cabeceo— como variables de entrada y el ángulo de referencia para el volante como variable de salida. La deriva está definida como la desviación del punto del eje delantero respecto a la trayectoria —en metros— y el cabeceo como la diferencia entre el ángulo de la trayectoria y la orientación del vehículo —en grados—. Por su parte, la salida la codificamos dentro del rango $[-1, 1]$, donde los



(a) Controlador baja velocidad.



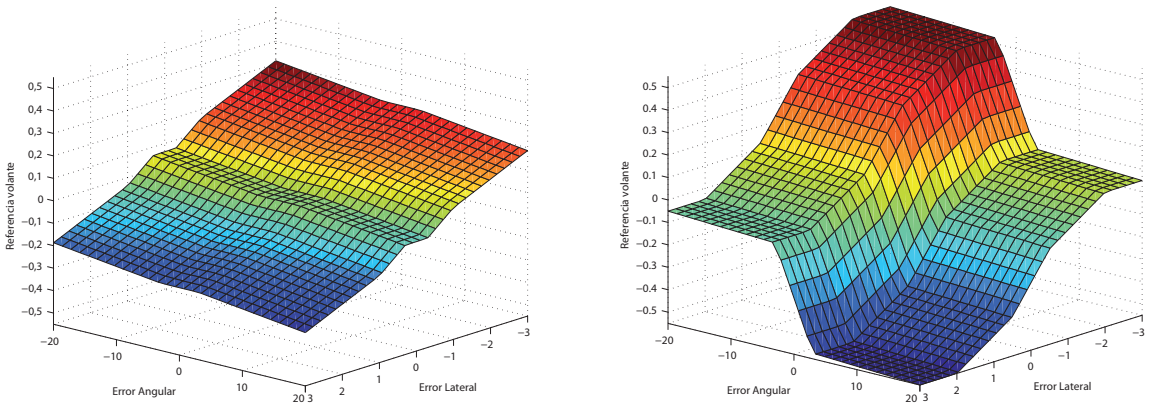
(b) Controlador alta velocidad.

Figura 6.7: Funciones de pertenencia asociadas a los controladores laterales.

valores negativos producen giros hacia la derecha del vehículo y los valores positivos el caso contrario. Como se menciona en la sección 3.6.2, el máximo giro del volante hacia cada lado es de 540 grados respecto a su posición central.

El ajuste de las funciones de pertenencia y los consecuentes del controlador se realizó manualmente en función de su rendimiento. Para ello se llevaron a cabo varias pruebas en los circuitos de prueba del CAR en Arganda del Rey y del Instituto Nacional de Técnica Aeroespacial (INTA) en Torrejón de Ardoz. La figura 6.7 muestra las funciones de pertenencia que definen cada controlador. En la imagen podemos apreciar que las funciones de pertenencia del controlador de baja velocidad están ajustadas a un rango más amplio. Esto se debe a que los errores alcanzan valores más elevados en entornos urbanos —e.g. en una curva cerrada—.

La figura 6.8 muestra la superficie de control obtenida para cada controlador. En esta imagen observamos una evolución más suave de la acción para el escenario a baja velocidad,



(a) Controlador baja velocidad.

(b) Controlador alta velocidad.

Figura 6.8: Superficies de control de los controladores laterales.

lo cual se debe a la amplitud de las funciones de pertenencia asociadas. En el caso del control a alta velocidad, la reacción ante los cambios es más rápida, a fin de corregir rápidamente cualquier desviación. No obstante, esto no quiere decir que el controlador provoque giros bruscos del volante, sino que su tiempo de respuesta es menor.

La referencia final δ la calculamos como una combinación de la salida de ambos controladores en función de la velocidad del vehículo:

$$\delta = \begin{cases} \delta_l, & \text{si } v_c \in [0, V_l) \\ \delta_l * \frac{V_h - v_c}{V_h - V_l} + \delta_h * \frac{v_c - v_l}{V_h - V_l}, & \text{si } v_c \in [V_l, V_h] \\ \delta_h, & \text{si } v_c \in (V_h, \infty] \end{cases} \tag{6.4}$$

donde δ_l es la referencia del controlador de baja velocidad, δ_h la referencia del controlador de alta velocidad, v_c es la velocidad del vehículo y V_l y V_h son dos umbrales que establecen los límites de operación individual para cada controlador. En el caso concreto de la demostración, estos umbrales se han ajustado a los valores de 20 y 50 km/h respectivamente.

Por último, hemos establecido dos condiciones de seguridad, limitando el movimiento del volante en función de la velocidad del vehículo. Esto se ha hecho, por un lado, para evitar cambios bruscos en la posición del volante y, por otro lado, para evitar que el vehículo derrape como consecuencia de un giro cerrado y una velocidad elevada. Para la primera verificación limitamos la tasa de cambio —por ciclo— de la referencia a ±0,066, lo cual equivale a una vuelta por segundo del volante. En cuanto a la segunda condición, restringimos la máxima referencia a ±0,5 cuando la velocidad es inferior a 30 km/h y a ±0,2 cuando la velocidad supera los 80 km/h. Para el rango entre ambas velocidades, reducimos el límite máximo de forma proporcional a la velocidad del vehículo. Los valores empleados se determinaron experimentalmente, comparando la amplitud de los movimientos del volante con conductores humanos a distintas velocidades.

6.5. Resultados

La demostración de conducción autónoma se llevó a cabo el día domingo 10 de Junio de 2012, con una duración aproximada de 92 minutos y una velocidad promedio de 66 km/h. Desde el punto de vista del tiempo de ejecución, se alcanzó un 99,03 % de conducción autónoma, ya que la presencia de un túnel de 900 metros de largo nos forzó a pausar el modo autónomo y reiniciar el algoritmo de localización, puesto que la incertidumbre alcanzada en el mismo no era suficiente para garantizar el comportamiento adecuado del vehículo.

La figura 6.9 muestra dos fotografías realizadas durante el recorrido. En la primera imagen se observan los vehículos Clavileño y Platero en la autovía, a la salida de San Lorenzo de El Escorial. La segunda imagen es una captura del interior del vehículo seguidor. En esta última fotografía se observa al autor leyendo un libro —sujetado con ambas manos— mientras que el vehículo circula automáticamente por la autovía. La presencia del conductor fue un requisito establecido por la DGT para garantizar la seguridad tanto de los vehículos de la demostración como del resto de vehículos que circulaban en la vía pública.

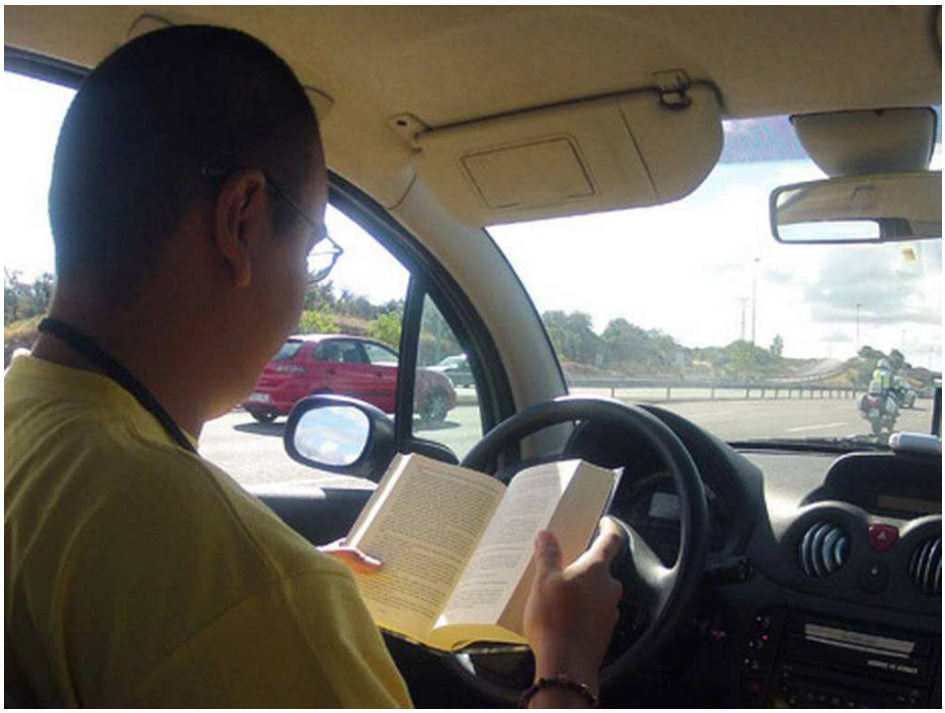
A fin de ilustrar mejor los resultados de la demostración, presentamos el comportamiento del vehículo en dos sectores de la trayectoria. El primer sector corresponde a un entorno urbano al inicio del recorrido, específicamente entre los minutos 6 y 9 de la prueba. Por su parte, el segundo sector corresponde al recorrido sobre la autovía entre los minutos 57 y 58 de la prueba.

La figura 6.10 muestra la trayectoria del vehículo en el sector urbano. Durante esta parte del recorrido, de aproximadamente 2 km de longitud, el vehículo circuló a través de dos rontondas, en torno a los minutos 6,5 y 7,9 respectivamente. Como se puede apreciar en la figura 6.11, la velocidad de ambos vehículos en este sector varía entre los 30 km/h, al paso por las rotondas, y los 55 km/h en los sectores rectos del recorrido.

En lo que respecta al control longitudinal, la figura 6.12 muestra la evolución de las dos variables de entrada y la salida del controlador. En la gráfica 6.12a observamos que el sistema de CACC implementado es capaz de mantener la distancia de referencia al coche líder con un error de ± 1 metro. Por su parte, la gráfica 6.12b nos muestra la evolución de la velocidad relativa entre los vehículos, la cual se adapta de forma adecuada para corregir el error de separación. Como se aprecia en la gráfica 6.12c, la salida del controlador corresponde mayormente a acciones sobre el acelerador, con excepción de la zona cercana al minuto 7,5 donde la reducción de velocidad del vehículo líder provocó oscilaciones en torno a 0.

En el caso del control lateral, la figura 6.13 muestra las entradas y la salida del controlador dual. Al igual que para el control longitudinal, podemos apreciar claramente en las gráficas correspondientes a las variables de entrada el paso del vehículo a través de las rotondas. En el momento de incorporarse a cada rotonda, tanto el error lateral como el angular se incrementan, indicando que la trayectoria se desplaza a la derecha del vehículo. Una vez dentro de la rotonda, ambas variables toman valores negativos debido al giro constante en sentido antihorario, hasta el momento de la salida donde nuevamente toman valores positivos. Por su parte, en los segmentos rectos el error lateral permanece dentro del rango de $\pm 0{,}5$

(a) Vehículos en la autovía.



(b) Interior del coche seguidor.

Figura 6.9: Fotografías realizadas durante la demostración de conducción autónoma.

Figura 6.10: Trayectoria del vehículo durante los minutos 6 y 9 de la prueba.

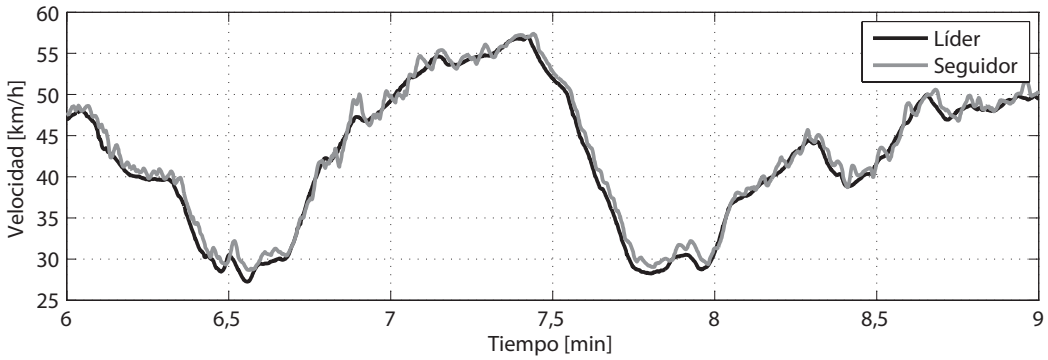
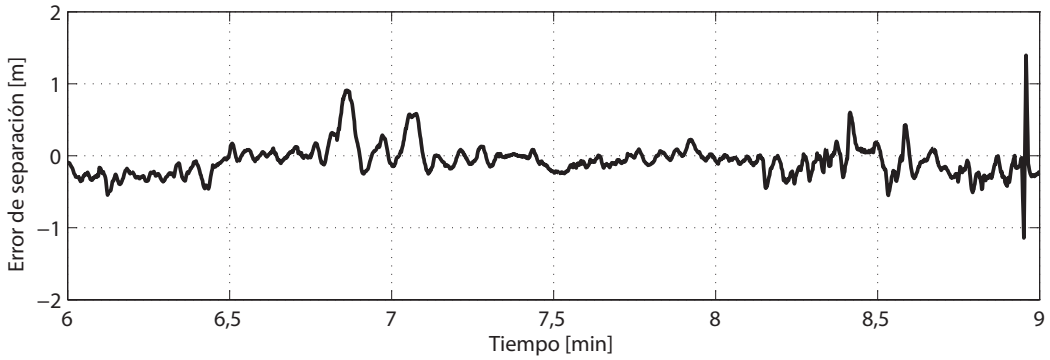
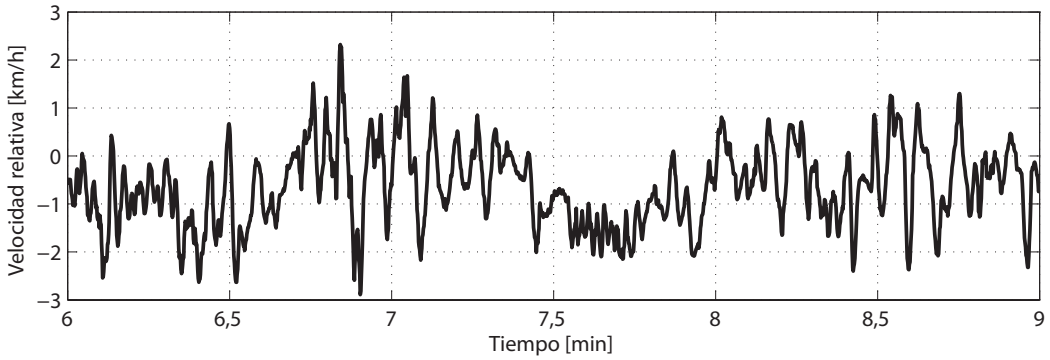


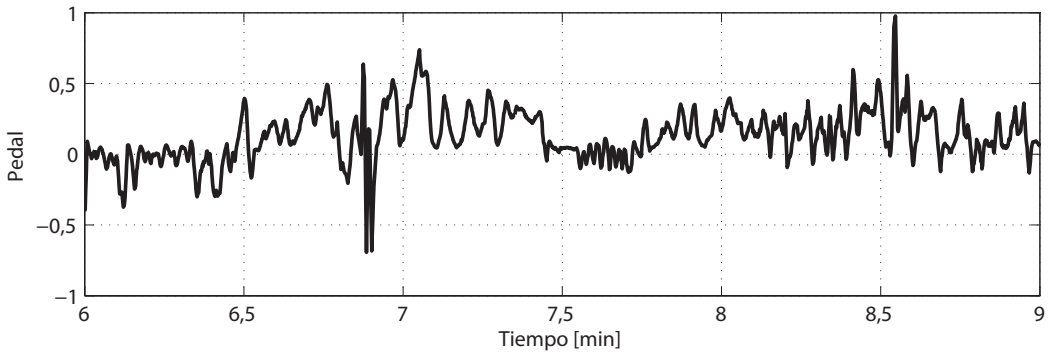
Figura 6.11: Evolución de la velocidad en el sector urbano.



(a) Error de separación



(b) Velocidad relativa.



(c) Salida del controlador

Figura 6.12: Evolución de las variables del control longitudinal en el sector urbano.

metros, mientras que el error angular está en el rango ±4 grados. Es interesante remarcar que, gracias al algoritmo de posicionamiento y al adecuado sistema de control, el vehículo se encuentra siempre dentro del carril —cuyo ancho es de 3 metros—, lo que demuestra la fiabilidad del sistema desarrollado para conducir el vehículo de forma segura por carreteras reales.

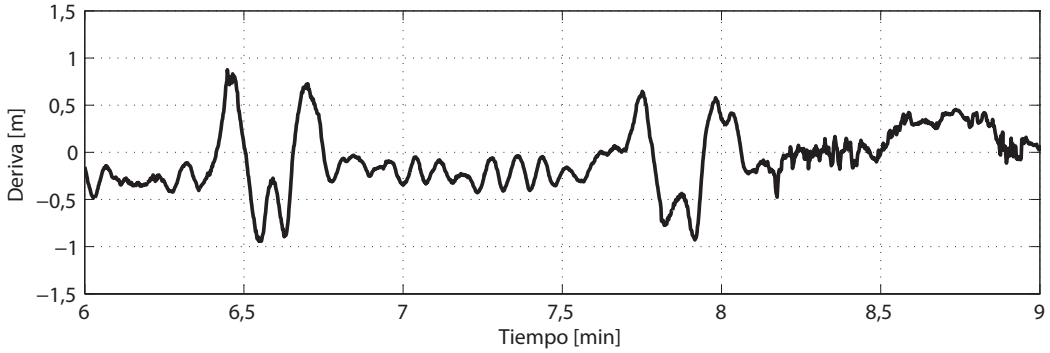
En la gráfica de la salida —imagen 6.13c— se observa como el controlador responde adecuadamente a los valores de entrada. En los segmentos rectos se mantiene el valor de referencia en torno a los cero grados con muy pocas variaciones, mientras que se indican giros derecha-izquierda-derecha para circular a través de las rotondas.

En cuanto al sector correspondiente al recorrido sobre la autovía, el comportamiento general del vehículo es similar al sector urbano. Mientras que la velocidad del vehículo líder se incrementa desde los 70 a los 80 km/h —figura 6.14— el controlador longitudinal mantiene el error de distancia respecto al líder en ±1 m. En la gráfica correspondiente a este error —imagen 6.15a— observamos que su valor se mantiene principalmente en torno a cero, con excepción de tres picos claramente apreciables. Estos picos corresponden a aumentos de la velocidad del líder, los cuales observamos también en la gráfica correspondiente a la velocidad relativa entre los vehículos —figura 6.15b—.

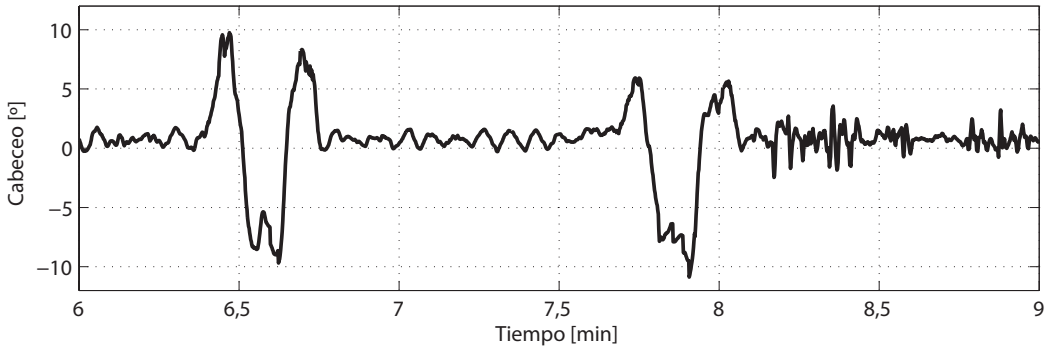
Finalmente, en las gráficas correspondientes a las entradas del control lateral —figura 6.16— observamos que ambos errores se mantienen dentro de un rango menor al del sector urbano. Esta es la respuesta normal del controlador debido a que la curvatura de la trayectoria en la autovía es considerablemente inferior a los valores que se alcanzan en entornos urbanos.

En general, observamos durante la demostración un comportamiento adecuado para el control autónomo del vehículo en entornos urbanos y autovías. En primer lugar, debemos destacar el buen comportamiento del controlador longitudinal desarrollado, el cual permitió realizar el seguimiento con errores inferiores a un metro. Esta conducta da al conductor la sensación de que el vehículo se encuentra acoplado a su predecesor, aumentando la sensación de seguridad para todos los ocupantes del vehículo (Nowakowski et al., 2010). Por otro lado, el controlador lateral mantuvo al vehículo dentro del carril, demostrando su capacidad para funcionar de manera autónoma siguiendo a un vehículo líder.

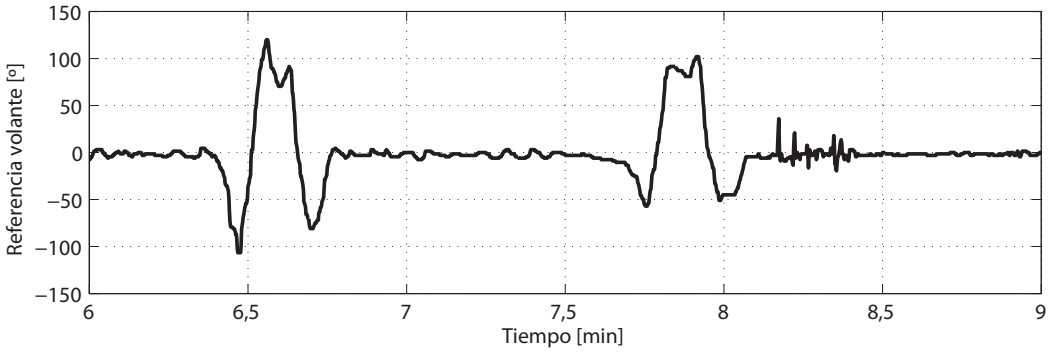
Antes de finalizar este capítulo, es importante mencionar que a pesar de no haber alcanzado el 100 % de ejecución autónoma, esta prueba nos ha permitido demostrar la funcionalidad y robustez general de la arquitectura propuesta a lo largo de esta tesis doctoral. Los ocupantes del vehículo seguidor destacaron la fiabilidad y seguridad del sistema propuesto, resaltando el comportamiento del mismo, próximo al que llevaría a cabo un conductor experimentado en condiciones similares. Gracias a la nueva arquitectura se han evitado fallos técnicos como los ocurridos durante la participación en el GCDC (Pérez, 2012), donde se observaron numerosos problemas de localización y lectura/escritura de periféricos. En el caso del control longitudinal, consideramos que se puede perfeccionar más su rendimiento con la integración de sistemas de visión artificial, los cuales nos proporcionarían información adicional sobre el comportamiento de los vehículos. Así, podríamos considerar criterios adicionales de seguridad, como por ejemplo que el seguimiento se realizase bajo la condición de que el vehículo



(a) Error lateral



(b) Error angular



(c) Referencia de posición del volante

Figura 6.13: Evolución de las variables del control lateral en el sector urbano.

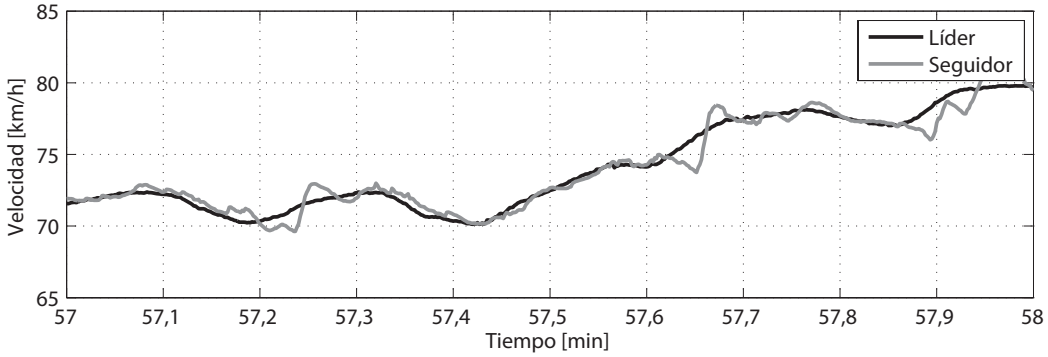
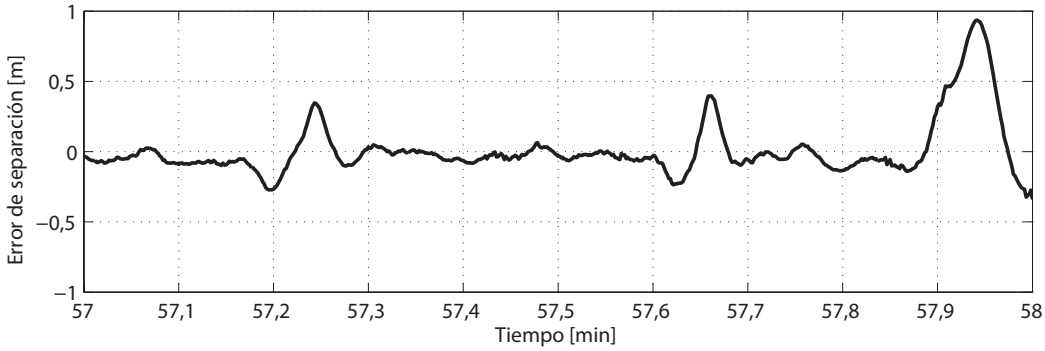
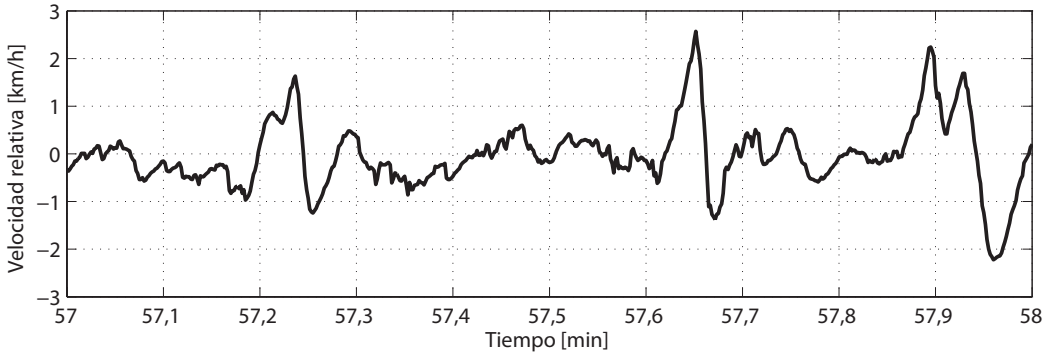


Figura 6.14: Evolución de la velocidad en un sector de autovía.

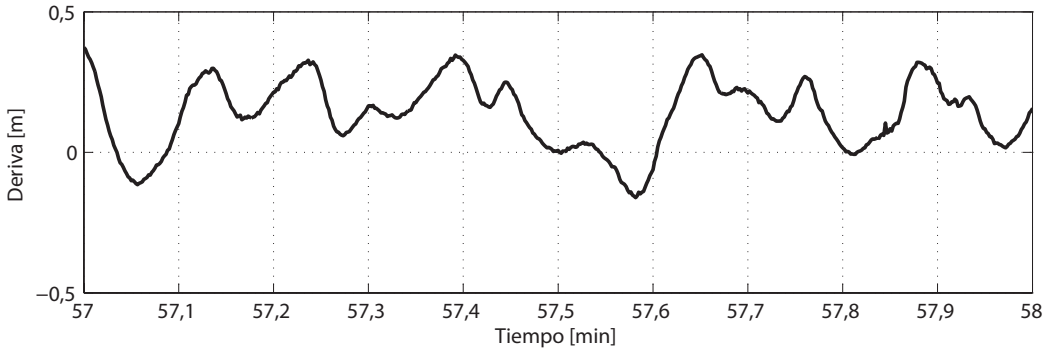


(a) Error de separación

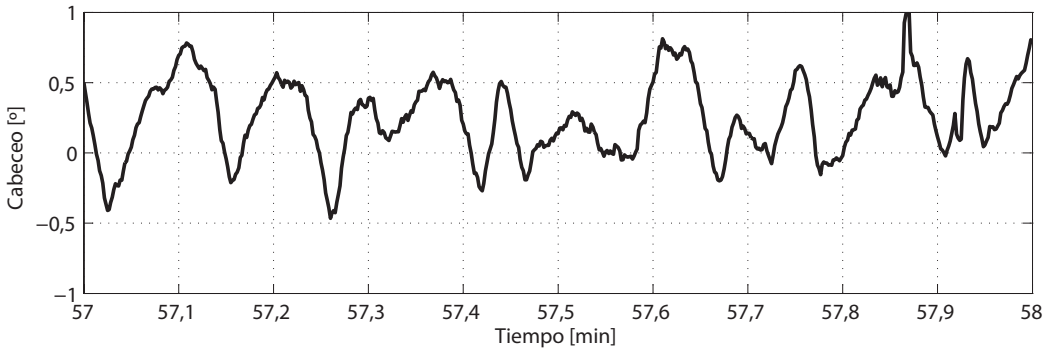


(b) Velocidad relativa.

Figura 6.15: Evolución de las entradas del control longitudinal sobre la autovía.



(a) Error lateral



(b) Error angular

Figura 6.16: Evolución de las entradas del control lateral sobre la autovía.

líder no abandone el carril de forma involuntaria.

CAPÍTULO 7

CONCLUSIONES Y TRABAJO FUTURO

El trabajo presentado en esta tesis doctoral continua la línea de investigación del Programa AUTOPIA, referente a la conducción autónoma de vehículos. A lo largo de su desarrollo hemos presentado una serie de aportaciones a la arquitectura de control del grupo, las cuales nos han permitido avanzar aún más hacia la consecución de este objetivo. Como primer paso, se han realizado profundas modificaciones en la estructura de la arquitectura de control, priorizando el manejo de recursos y la robustez ante posibles fallos de los componentes. Así pues, la arquitectura propuesta se compone de un conjunto de elementos de software que si bien son ejecutados de forma independiente, trabajan de forma conjunta.

Además de la nueva estructura de la arquitectura de control, en este trabajo se ha desarrollado un algoritmo de localización basado en enjambres de partículas. A través de este método se realiza la estimación del estado del vehículo a partir de la fusión de la información proveniente de distintos sensores embarcados, como son el GPS, la unidad inercial y la información de velocidad del vehículo. El algoritmo tiene un funcionamiento similar al de los ya conocidos y utilizados filtros de partículas, con la salvedad de que incorpora un comportamiento conjunto en el proceso de evolución de las mismas, emulando la conducta de los enjambres naturales durante su desplazamiento. El diseño de este método de localización parte del conocimiento adquirido por el autor durante la estancia realizada en el LIVIC, adaptándolo posteriormente para su aplicación en los vehículos del Programa AUTOPIA. Su correcto funcionamiento ha quedado validado a través de una serie de pruebas realizadas en entornos urbanos e interurbanos, donde se ha conseguido superar situaciones donde se hacía imposible localizar el vehículo empleando únicamente un receptor GPS. Así mismo, este algoritmo ha sido fundamental para la realización del último experimento en carreteras reales, donde los receptores GPS estaban sometidos a continuas interferencias ocasionadas por el entorno.

Superado el problema de la localización, se ha continuado con el análisis de la capacidad de dos técnicas de inteligencia artificial para el aprendizaje y adaptación de controladores. En una primera fase se ha aplicado el método de aprendizaje por refuerzo *Q-learning* al

control lateral del vehículo, obteniendo resultados alentadores sin proporcionar ningún tipo de conocimiento previo. Posteriormente, se ha desarrollado y aplicado un método de adaptación *on-line* al control longitudinal, obteniendo finalmente un sistema de control de crucero que es capaz de adaptarse rápidamente para sortear las posibles perturbaciones del entorno.

Todas estas aportaciones se han realizado en pro del desarrollo de sistemas de conducción autónoma para vehículos. Por este motivo, el trabajo concluye con la realización de un experimento de conducción en carreteras reales, donde se ha verificado el funcionamiento conjunto de la arquitectura de control desarrollada.

A continuación se exponen en detalle las principales conclusiones derivadas del trabajo realizado en esta tesis doctoral:

Arquitectura de control

- Se ha desarrollado e implementado una arquitectura de control para la conducción autónoma de vehículos, la cual se basa en una estructura modular de software. Al evitar la centralización del control en un proceso maestro, implementándolo como una combinación de varios módulos independientes, se consigue aprovechar en mayor medida los recursos computacionales del sistema, además de minimizar el riesgo de bloqueo del software y retrasos de procesamiento.

- Gracias a la segmentación del software es posible minimizar el impacto que tiene la falla de un componente sobre el rendimiento global del sistema. Así, un software de supervisión sería capaz de recuperar el sistema inmediatamente, reiniciando únicamente aquellos componentes que presenten problemas en su ejecución, sin necesidad de detener la totalidad del sistema.

- A pesar de que esta reestructuración del software hace necesaria la implementación de un método de comunicación entre procesos —no requerida anteriormente—, las herramientas utilizadas mejoran la capacidad para depurar y detectar fallas en el sistema. Así mismo, su opción de captura de datos permite el desarrollo y validación de los distintos métodos y algoritmos *off-line*, facilitando la posterior transferencia de los componentes al sistema real.

- La arquitectura no está limitada al conjunto de módulos que se presentan en este trabajo. Por el contrario, se pueden agregar y eliminar módulos en cada etapa, expandiendo las capacidades de las mismas. Esto es posible siempre y cuando los nuevos módulos no interfieran con los mensajes y canales de comunicación definidos previamente. De igual forma, la nueva arquitectura permite expandir la capacidad de cómputo del sistema a través de la ejecución de los elementos de software en varios ordenadores conectados en red.

Localización

- Se ha diseñado, desarrollado e implementado un algoritmo de localización que fusiona la información obtenida a través de distintos sensores embarcados, mejorando significa-

tivamente el proceso de estimación del estado de los vehículos, altamente dependiente de los receptores GPS.

- Se ha validado el funcionamiento del algoritmo a través de varios experimentos con distintas configuraciones de obstáculos en el entorno, demostrando su capacidad para mantener la coherencia en la medición incluso en aquellos escenarios donde se producen vacíos en la medición del GPS.
- Se ha verificado la capacidad del algoritmo para trabajar con diferentes arquitecturas, probando su rendimiento en vehículos cuya instrumentación difiere en varios aspectos como la frecuencia de medición y resolución de los sensores.

Aprendizaje y adaptación

- Se ha analizado la aplicación de técnicas de aprendizaje y adaptación al control autónomo de vehículos.
- Se ha diseñado un controlador lateral capaz de determinar las acciones a tomar sobre la dirección del vehículo sin ningún tipo de conocimiento previo.
- Se ha desarrollado un sistema de control de crucero con capacidad de ajuste, lo que le permite adaptarse rápidamente para sortear las perturbaciones impredecibles del entorno. El funcionamiento del sistema ha sido verificado experimentalmente a través de su implementación en un vehículo real.

Demostración

- Se ha realizado la primera prueba de conducción autónoma del Programa AUTOPIA en carreteras reales, estableciendo un antes y un después en la historia del Programa con el experimento que, hasta ahora, ha alcanzado la mayor velocidad, distancia recorrida y tiempo de ejecución.
- Este experimento ha permitido demostrar la funcionalidad conjunta de los sistemas que componen la arquitectura propuesta a lo largo de esta tesis doctoral, resaltando los niveles de fiabilidad y seguridad que se han alcanzado. Así mismo destaca la capacidad del sistema para llevar a cabo la labor de conducción de forma similar a la de un conductor experimentado.
- Se ha verificado la robustez general del sistema, eliminando los fallos técnicos presentados por la arquitectura previa en pruebas similares, como los ocurridos en el 2011 durante el GCDC.

Más allá de los avances y aportaciones realizados por el presente trabajo, debemos recordar que aún existe un largo camino por recorrer hasta que la conducción autónoma sea una realidad sobre ruedas en todas las carreteras. Por ahora, el principal obstáculo sigue siendo la

seguridad y robustez de los sistemas desarrollados, los cuales si bien presentan un rendimiento aceptable en ciertas condiciones, no son capaces de adaptarse aún a la totalidad de situaciones a las que se enfrenta un conductor humano. No obstante, hoy en día podemos encontrar una gran diversidad de soluciones basadas en la automatización parcial de los vehículos, los cuales si bien sólo están pensados para una situación en particular, como aparcar o mantener la velocidad, nos dejan ver que cada día estamos más cerca de alcanzar esta idea hasta ahora utópica.

Independientemente del enfoque de investigación que se mantenga, a partir de ahora es sumamente importante tener en cuenta los avances tecnológicos en materia de la comunicación V2V y V2I. Tan sólo hace falta dar una mirada rápida a la literatura para ver que, a nivel mundial, la mayoría de los fabricantes y grupos de investigación están apostando por soluciones donde la cooperación e intercambio de información entre vehículos e infraestructura es la base del desarrollo. Al fin y al cabo, no debemos olvidar que los vehículos no circulan solos en la carretera. Como continuación de este trabajo, es necesario profundizar en la capacidad de percepción del sistema, incorporando nuevos sensores —e.g. cámaras de visión y sensores de proximidad— a fin de ampliar la visión general del entorno. Esto último deberá ir de la mano de la fusión sensorial, de forma tal que los distintos sensores embarcados se complementen entre sí, aumentando la robustez del sistema y las soluciones desarrolladas. En cuanto al control del vehículo, se debe llevar el desarrollo a un nivel más elevado, enfocándonos en el análisis del comportamiento colectivo de varios vehículos en un área determinada y desarrollando soluciones de control cooperativo. Por supuesto, todo esto sin olvidar nunca el control individual de los mismos. En este sentido, es necesario continuar el desarrollo de sistemas inteligentes, capaces de adaptarse rápidamente a las distintas situaciones que se producen día a día en las carreteras, manteniendo siempre la seguridad y eficiencia como principales prioridades. Para esto último, es requisito indispensable contar con una plataforma de simulación más adecuada, a fin de facilitar tanto el proceso de diseño y validación de los sistemas en entornos virtuales como su transferencia a los vehículos reales.

CONCLUSIONS AND FUTURE WORKS

The work carried out for this dissertation extends the research of the AUTOPIA Program, focused on the development of autonomous driving systems. Several contributions to the previous control architecture are presented, leading toward this main goal. First of all, the control scheme has been deeply modified, giving priority to the resource management and robustness against possible component failure. Therefore, the proposed architecture is made up by several software modules running individually, but working together.

In addition to the new control architecture, in this work a filter algorithm based on the behaviour of biological swarms has been designed. The main goal of the algorithm is to perform vehicle location by combining the data from different sensors. In this sense, the algorithm considers several solutions at the same time, with an operation scheme similar to the well-known Particle Filters. This design is related to the work carried out during a research stage at the LIVIC, being further modified to the AUTOPIA control scheme. Results obtained on tests made on several scenarios show a proper performance of the developed algorithm, which even manage to overcome scenarios where a GPS-based location is impossible. Furthermore, this solution has been essential for the development of the experiment on real roads, where GPS measurements are commonly affected by interference and occultation.

Further work carried out consists on the analysis of two AI techniques for the development of controllers through learning and adaptation approach. On a first stage, the Q-learning method –related to reinforcement learning– has been applied to the lateral control of the vehicle. Results with this approach are encouraging, considering the control actions are deduced only from the system interaction with the environment. Subsequently, an adaptive method for longitudinal control has been developed. This approach led to the development of a cruise control system able to overcome the possible disturbances on the environment.

The final objective of all these improvements has been to contribute to the development of autonomous driving systems for vehicles. In this sense, the work concludes with the results of a final experiment on open roads, verifying the joint operation of the control architecture developed.

Let us see now in detail the conclusions that arise from the work carried out for this dissertation:

Control architecture

- A new control architecture for autonomous driving has been developed, based on a modular software scheme. The software is not centralized in a master process, but it is composed of a set of smaller processes running simultaneously. This exploits in a better way the computation resources and minimizes the risk of deadlocks and processing delay.
- By dividing the software in several modules, it is possible to minimize the impact of a failure in one of the modules over the global performance of the system. Thus, if a supervisor program is implemented, only the crashed process is restarted as soon as the failure is detected, without being necessary to restart all the software.
- Although the software fragmentation in several processes required the usage of an IPC method – not necessary in previous works; the selected tool increases the capability of the developers for debugging and detecting system failures. Furthermore, thanks to the logging utility it is possible to develop and test all the systems off-line, but with a fast transition to the real platform.
- The architecture is not limited to the modules presented at this work. New modules can be easily added either to expand any of the stages or to replace deprecated modules. This is possible as long as the new modules do not interfere with the channels and messages previously defined. Likewise, should it be needed, it is possible to expand the processing capability of the system by using several networked computers.

Location

- A novel algorithm has been designed, developed and implemented for vehicle localization. It merges data from different on-board sensors, improving the estimation of the vehicle state.
- The performance of the algorithm has been validated through several tests on different scenarios. Results are encouraging, showing the capability to estimate the vehicle state even when there are gaps on the GPS measurement.
- Moreover, the location algorithm has been tested with different architectures and equipment, showing a proper performance in spite of all the differences on the instrumentation of vehicles, such as sample rate and sensors resolution.

Adaptation and learning

- The application of adaptation and learning techniques to the control of autonomous vehicles has been analysed on this work.
- A system able to learn from scratch has been design for the lateral control. It learns the best action to take on different situations, using only the experience obtained from the interaction with the environment.

- Likewise, an on-line adaptable cruise control system has been design and implemented. The controller has the capacity to evaluate the performance of the vehicle in real time, adapting the control consequents to reduce the tracking error.

Demonstration of autonomous driving

- The first test of the AUTOPIA's system on real roads has been carried out for this work, leaving a breaking point on the history of this research program. This experiment has set the distance, cruising speed and duration records among all the tests performed by AUTOPIA up to now.
- This experiment has showed the overall functionality of the set of systems composing the proposed control architecture, remarking both the safety and reliability levels reached by this new configuration. Moreover, it also highlights the system capability to perform the driving task with a human-like behaviour.
- The system robustness has been proved throughout this experiment. Compared to the performance of the previous architecture on the GCDC 2011, the common technical failures have been solved and the overall performance improved.

Beyond the contributions made by this work, there is yet a long path before the self-driving vehicles become a reality on all roads. So far, the safety and reliability of the developed systems remains one of the main obstacles. Although most the systems have a proper performance when addressing some scenarios, they are still far from having the capacity to overcome all the situations a human driver deals with. Nevertheless, today a large number of solutions based on the partial vehicle automation can be found on the market, such as the assisted parking systems, adaptive cruise control and collision avoidance systems among other. This leads to the idea that we are closer to reach this utopia.

From now on, in the development of ITS systems it is extremely necessary to keep track of the technological advances on V2V and V2I communications. A simple review of the literature shows a global trend to invest on the design of systems where the cooperation and information exchange among vehicles is the basis of the development. After all, a vehicle is never alone on the road. As future work, it is necessary to improve the perception capacity of the system, incorporating new sensors –such as vision cameras or proximity sensors– in order to get a more completed view of the vehicle surrounding. In parallel, further research is needed on the sensory fusion, so all the on-board sensors complement each other. Regarding vehicle control, the scheme has to be taken to a higher level, focusing on the analysis of the joint behaviour of the vehicles in a given area and in the development of cooperative solutions; all this without forgetting the control of single vehicles. In this sense, it is necessary to continue with the development of intelligent systems able to adapt to the different situations on roads, keeping both efficiency and safety as top priorities. Finally, a proper simulation platform is needed, so system design and validation can be easily performed on virtual environments with a fast transfer to real vehicles.

APÉNDICE A

SUMMARY

The proposal of this chapter is to briefly present the technical aspects of this dissertation for benefit of the readers of English language.

A.1. Control Architecture

Nowadays, the public interest in driverless vehicles has increased worldwide. This has been motivated by popular events as the DARPA Challenges, the GCDC and the Google driverless car. Although it is true that there is yet a long way for the entire automation of roads and cities, the advances and technologies developed up to now have shown the strong potential of vehicle automation for safety and efficiency improvement. In this sense, the AUTOPIA Program is one of the Spanish team focused on this research line, transferring control techniques from the mobile robots area to commercial vehicles.

AUTOPIA implements a control architecture based on the hierarchical control scheme: perception, decision and action; emulating the human behaviour. This architecture breaks down into three main stages: (i) Perception, including all the on-board sensors used to sense the environment; (ii) Planning, which chooses a driving mode according to the environment conditions and finally it determines the action to take over the vehicle's actuators; and (iii) Actuation, where the actions decided are actually applied over the actuators.

Up to now, different tests and experiment have been carried out using this control scheme (Milanés et al., 2008; Pérez et al., 2010), showing a proper behaviour for the autonomous vehicles guidance. However, several issues regarding the scalability and robustness of the system have also come to the surface (Pérez, 2012). Bearing this in mind, the first contribution of the present work is to design a new structure to the control architecture, prioritizing on the efficiency, feasibility and robustness of the new system.

The development and validation processes have been done using a prototype vehicle of the Program. The car, known under the codename *Platero*, is a conventional gas-propelled Citroën C3 which includes hardware modifications for the autonomous control of its actuators using an on-board computer. The car has a DGPS receiver Trimble BD960, an Inertial Measurement Unit (IMU) Crossbow VG440, and a device able to communicate with the CAN bus of the

car, accessing to the information of the sensors boarded by the manufacturer. Regarding the actuation, a PCI card installed on the board provides the analogue signals for controlling the throttle. For the brakes, an electrohydraulic system controlled by an I/O module has been installed in parallel to the original system. On its turn, the steering wheel is controlled by an external controller, coupled to the motor of the Electric Power Steering system.

Turning to the design of the new control scheme, one of the most important considerations taken into account has been the system modularity. Thus, the system has been decomposed in several hardware and software modules working together. The advantage of this fragmentation is twofold: on the one hand, several partners can work at the same time in different modules without interfering with each other, and on the other hand, in case of a system failure the problematic modules can be quickly identified. Moreover, if each software module is defined as an independent program, a failure in one module does not necessarily affect the whole system.

Bearing this in mind, a modular architecture for vehicle automation is proposed. This system is divided in three main stages: perception, decision and actuation; and an auxiliary one shared by the latter three: general I/O. Each stage is further divided on several modules. The perception stage is in charge of providing information about vehicle state and its surroundings – i.e. other vehicles or infrastructure – from the sensorial equipment. The decision stage includes goals, path planning, vehicle state estimation and determination of the control actions. The actuation stage is composed by the systems executing the control actions over the throttle, brake and steering wheel of the vehicle. Finally, the general I/O includes all the low-level devices – as communications and digital/analogue I/O modules – that can be used by any other module in the main layers. This removes the direct dependency between a hardware module and a unique software module, allowing that several software modules use this hardware resource.

From the software point of view, the modular outline of the architecture requires the implementation of a reliable inter-process communication (IPC) technique, allowing the information exchange among processes in real time. In this sense, several IPC solutions were considered, including main methods such as shared and mapped memory, pipes, FIFOs, files and sockets; and the recent API solutions developed on the robotic field such as CARMEN, ROS, LCM, and MOOS. At the end, the LCM API was selected over all, due to its features regarding the communication model, transport layer, message codification, supported programming languages and development tools.

One of the critical factors for this choice has been the communication protocol. LCM is based on a publish-subscribe scheme, without requiring a master process. Implementing this tool, each software module is able to define its own communication channels and message structures, providing information to other modules and subscribing to those required for its operation. Having solved the problem of the information exchange among the different software modules, the final structure defined for the control architecture is:

Three software modules on the perception stage, each one dedicated to a specific sensor,

namely GPS, IMU and CAN BUS adapter. The GPS process reads the receiver information through a network connection and with a 10-hz sampling rate. At each cycle, the software processes three data frames —GPGGA, GPVTG and GPGDT— extracting information about the current positioning coordinates, altitude, speed, heading, receiver resolution, and the measurement errors. As last step, the data is encapsulated on a structure called `gps_all` and sent it over a communication channel with the same name. The procedure is the same for the IMU, using a RS-232 interface and publishing the information over the `imu_all` channel. Regarding the CAN BUS adapter, the dedicated software reads only two frames, related to the speed and acceleration of the vehicle and to the steering wheel movement – position and turning speed. Since the sampling rate is not the same for both frames, this software uses two different channels: `can_coche_vel_acel` and `can_coche_volante`.

For the general I/O stage, two software modules are defined. The first one is in charge of configuring and managing the analogue PCI card. To that end, the program subscribes to the `control_analogico` channel, through which it receives the commands from other modules. After processing and verifying each message received, it sets the card output according to the command. A similar behaviour is implemented on the second program, which is in charge of the analogue-digital I/O module. In this case, the software receives the commands through the channel `modulo_can_control`. Besides this task, the program periodically reads the state of the hardware inputs, publishing the information on the `modulo_can_estado` channel.

Regarding the actuation stage, a software module is defined for each actuator. The three programs work the same way, interpreting the medium-level commands to actuate over the vehicle. Despite having a specific channel for each module – `orden_acelerador`, `orden_freno` and `orden_volante` – the commands for the three actuators use the same base structure – `orden_actuador`. The throttle and brake programs send commands to the general I/O modules, but the steering wheel software directly communicates with the external hardware controller. As safety measurement, the three programs implement a watchdog timer. When no messages are received, the timer triggers and the actuators outputs are reset. This prevents the actuators from keeping the last value in case the control program crashes or it is ended by the user.

On its turn, the decision stage is implemented as a single procedure, but with several modules defined as classes. The most relevant ones are the mission, planner and control modules. The mission modules is designed to keep a list of generic goals, defined either before running the program or even in running time, describing the major goals for the car. The planner module is in charge of generating an adequate trajectory for the car to perform the main task. To that end it continuously analyses the information stored on the mission, adapting the vehicle reference trajectory. Finally, the control module estimates the variables, mode and actions for a proper control of the vehicle. This last module includes a tool for the execution of fuzzy controllers, which are normally used

due to its similarity to the human reasoning. However, this is not a limiting factor since the architecture allows the implementation of any type of controller as long as the output is coherent with the normalized input for the low-level controllers

The implementation of the architecture on the prototype vehicle allowed verifying its performance, validating the improvements regarding the robustness, safety and feasibility of the system. Moreover, this proposal is the basis for the rest of the work that has been carried out on this dissertation.

A.2. Location

Vehicle location is one of the most essential task for Intelligent Transport Systems (ITS) applications development, but at the same time it is one of the most complex due to the requirements, both high robustness and high accuracy. Nowadays, Global Positioning Systems (GPS) are considered as the default solution to this problem. Nevertheless, it is also well known that GPS measurements are commonly affected by interference, occultation and reflection in satellite signal reception – e.g. trees, buildings, meteorological conditions, etc – reducing considerably its accuracy. For most of the cases, this issue can be corrected by implementing differential systems – DGPS; however, these solutions are quite expensive and could reduce the system autonomy. For these reasons, architectures combining both proprioceptive and exteroceptive data from different sensors – e.g. GPS, INS; LIDAR, cameras – are increasingly used. This work describes the development of a filter algorithm based Particle Swarm Optimization (PSO), emulating the behaviour of biological swarms. The main goal of the algorithm is to perform vehicle location by combining the data from several sensors and/or digital maps.

Up to now, different filter algorithms have been presented as solutions for merging data, being the Kalman Filter (KF) the most popular method. Extended (EKF) and Unscented (UKF) approaches allow removing some KF limitations, as model linearity, improving filtering performance and applicability. Despite its great acceptance, Gaussian noise assumption by KF is a limiting factor for some applications as vehicle location. This has resulted in the application of other filtering methods as Particle Filter (PF). Unlike KF, particle filters represent several possible states of the system at the same time. Each one of these states – called particles – is associated to a weight representing its degree of confidence of being the real state of the system. Nevertheless, a major drawback related to this method is the degeneration of particles, leading to situations where all but one particle have negligible weight. A common approach to avoid this problem is the Sampling Importance Resampling (SIR) PF. In a SIR filter, a minimum threshold is set to particles'weight, resampling all the particles which do not reach the minimum value.

On its turn, PSO is an optimization method based on the behaviour of biological swarms – e.g. flock of birds or schooling fishes. In this method a set of potential solutions – called particles – are initialized inside the search space. Each particle moves through the search space

seeking the optimal solution, considering both their own experience and the best experience of the swarm.

Our proposal is to adapt the PSO procedure to the estimation of vehicle states. As other location algorithms, this approach – henceforth called Particle Swarm Location algorithm (PSL) – merges the information obtained through different sensors in order to estimate vehicle state in real time. To that end, a set of N individuals or particles – called Location Swarm (LS) – is defined as representation of the possible vehicle state –position and orientation–, evolving step by step according to data received from sensors.

For the development of the algorithm, a set of data was captured using a prototype vehicle of the LIVIC. The set included information from a low-cost GPS, IMU, and from the CAN Bus of the car – steering position and vehicle speed. It also includes data combined from a RTK GPS and a high fidelity IMU, which was used as reference to validate the final results. From position point of view, the PSL algorithm works as follows: at first step, the particles are initialized inside an uncertainty area where the vehicle is supposed to be –e.g. around the GPS measurement. After that, every individual is evolved according to the following procedure:

1. As soon as the information is received from sensors, an a priori estimation is calculated according to the evolution model implemented.

2. To describe the movement of the particle, an a priori displacement vector is calculated as the difference between the a priori estimation and the previous particle state. The same procedure is applied over the last swarm estimated value.

3. The a posteriori displacement vector for the particle is then calculated as a combination of the a priori values of both its own displacement vector and the swarm's. The relationship between the values is determined by the particle weight.

4. Having the a posteriori displacement vector, the new particle state is calculated adding the vector value to the previous particle state.

5. As final step, the swarm estimation –and vehicle state– is calculated directly from the updated states of the particles after the evolution process. This estimation is made by the weighted addition of all particles states.

The weight value for all particles is defined as a combination of two auxiliary weights. The first one is related to the probability of the particle of being a solution according to the GPS information. For each particle, a temporal value indicates if the particle is inside the uncertainty area related to the last GPS measurement, being 1 when it is and 0 if it is not. This temporal value is then used to update the GPS weight, using an inertial relationship with the previous value. The second auxiliary value is related to digital maps information. It is calculated in a similar way to GPS weight, but consider is the particle is on the road –1– or not –0.

On the basis that the vehicle is always on the road, it is obvious that particles that satisfy both conditions –inside the uncertainty area and on the map– have a higher probability of being the real vehicle state. Thus, the particle weight is finally calculated as a combination of the auxiliary weights product and the individual values.

An additional consideration is taken into account for updating the GPS weight, called the GPS confidence. This is the correlation between the displacement measured from GPS and the one taken from odometry. If the difference among both values is higher than a defined threshold, then the algorithm considers the GPS has an error caused by external interferences. If this situation occurs, the GPS weight is no longer updated until the confidence on the GPS return to it default value.

As for the particles filters, a resampling method has been implemented on the PSL. This method is in charge of replacing low probability particles with new ones. To that end, two different thresholds are considered, for the GPS and Map weights respectively. Should the GPS weight be below the threshold, the individuals are replaced by a new particle around the last GPS measurement. Likewise, the low map weight particles are resampled around the last swarm estimation. Both resampling procedures are executed as soon as the weights are updated by the sensor information.

With the goal of evaluating the behaviour of the location algorithm according to the number of particles in the swarm, a first experiment was carried out using the same data but different lengths on the range from 50 to 1000 particles. Comparing both the average error and the execution time of the algorithm, the results showed the best configuration is around 250-300 particles. A 250-PSL was able to perform the vehicle location with an average error around 0.4 metres, with an execution time lower than 50 milliseconds, proper for the sample rate of the data. Compared to the 50-PSL, the average error is reduced around 50 %.

A second experiment was carried out with the goal of comparing the proposal with an EKF. Results showed a similar performance of both algorithms, proving the PSL is a viable solution for vehicle location.

From the results obtained with the data captured at LIVIC, the work continued with the application of the developed algorithm to the proposed architecture for AUTOPIA vehicles. However, due to the high resolution of the DGPS receivers used by AUTOPIA, the main goal of this implementation is to sort scenarios where the GPS is not available. To that end, some modifications were introduced on the method. As first change, the map weighting was removed, since information from digital maps is not used. In return, a particle confidence was added to replace the map weight gap. For each particle, the confidence value is estimated as the relation between the particle age –cycles without being resampled– and the oldest particle in the swarm.

In order to maintain harmony with the control architecture first developed, the algorithm is embedded in an individual software module. This module accesses the sensor information through the LCM channels defined for perception stage –i.e. `gps_all`, `imu_all`, `can_coche_vel_acel` and `can_coche_volante`. Likewise, the filtered data is published over the channel `filtro_lep`. In contrast with the implementation using LIVIC information, in

this case the algorithm has been optimized to run at the rate the IMU messages are received, increasing the sampling rate of the filter procedure.

The algorithm proficiency was once again validated through several experiments with real data. For the first one, the prototype vehicle circulated in an urban area, where the interference and reflection of the satellites'signals produced measurement errors along the trajectory, decreasing the receiver precision. In this test, the algorithm was able to overcome several errors on the GPS measurement, keeping a proper estimation of the vehicle location.

During the second experiment, the algorithm was tested for scenarios where the GPS is unable to measure since there is no a clear sky-view. To that end the prototype circulated along an interurban road, through two short tunnels. Two different tests were carried out on this scenario, one keeping the vehicle speed and the other modifying it on the tunnels, where there is no GPS measurement. For both tests, the vehicle location was properly estimated, even when there was no information from the GPS receiver. In this experiment the algorithm was able to overcome errors up to 6 metres, in addition to the GPS gaps.

Both the test with the LIVIC's prototype and the experiment with the AUTOPIA's platform have validated the proper performance of the designed algorithm for vehicle location. Moreover, its capability for operating on different platforms has been shown.

A.3. Learning and Adaptation

When driving, an average person considers a lot of variables to control the vehicle, such as its speeds, surrounding vehicles, state of the traffic lights, and so on. This complicates the development of controllers for autonomous driving, since the transference of this behaviour to computer systems is more complicated than it appears. First off all, it is almost impossible to have a realistic model of the driving problem, due to the great amount of singularities and situations to consider. Likewise, the high number of input variables hinders the adjustment of controllers. In this sense, the application of artificial intelligence techniques is one of the trends for the system development in the ITS field.

In this thesis, we have studied the capability of two learning and adaptation techniques for controlling autonomous vehicles. The first one selected has been the reinforcement learning, which allows the system to learn from scratch using the iteration with the environment. This technique is applied to the development of a lateral control for the vehicle. The second one is an adaptation method developed for its application with fuzzy controllers. The method modifies the controller membership functions defined for the inputs and its consequents by evaluating the performance of the system in real time. This method has been applied to the development of a cruise control system.

A.3.1. Lateral control and reinforcement learning

Reinforcement learning (RL) is an AI technique based on the animal learning theory. According to this theory, those actions generating a positive reward for the animal are more

likely to be taken again than those leading to negative rewards. Applied to the AI systems, the method is based on the continuous evaluation of the system performance, rewarding it positively or negatively according to the achievements reached.

Up to now, there are several approaches for the application of reinforcement learning on the AI field. One of the widely applied methods is the Q-learning. This method is based on the estimation of a quality value (Q) associated to each available action. To calculate this value, the system needs to continuously interact with the environment, evaluating the capacity of the available actions to improve its own performance. The main advantage of this method is that it does not require an evolution model nor a control policy defined by the developer. It only uses the information about the rewards obtained through the interaction with the environment. However, this feature is also a drawback for the method, since its experience is limited by the explored situations. This means the previous knowledge is not exploited when a new situation arises. On the other hand, it is a method for discrete problems, meaning the application to continuous state problems requires a space discretization. Depending of the problem, the discretized space could be very large, requiring a huge execution time and computer resources to deal with the problem.

A common solution to this problem is the input aggregation. Through the aggregation methods the space of the problem can be generalized, reducing the number of possible states. Moreover, it is possible to exploit the experience obtained from similar situations when dealing with new ones. In this work, we apply a fuzzy aggregation technique to the problem of lateral control of the vehicle. Fuzzy logic was selected due to the similarity with the previous controllers developed by AUTOPIA.

Among the variables commonly used for the lateral control of vehicle one can found the lateral and angular errors, the vehicle speed, the steering wheel position and the trajectory curvatures. A simple reinforcement learning approach with these 5 variables, each one discretized with a 5 % resolution would lead to $3{,}2 \cdot 10^6$ states. From the point of view of the required memory, the problem would be unviable to solve. However, applying the input aggregation method the number of states is considerably reduced.

For the implementation of learning on the lateral control, a virtual environment was created using Matlab. This has been done with the goal of reducing the required time for the controller adjustment. Two models are used for describing both the lateral and longitudinal evolution of a virtual vehicle. As reference for the control, four different trajectories are used. These have been recorded with the prototype vehicle on the test track. The controller output is defined as the reference position for the steering wheel. A fuzzy controller has been tuned for the vehicle model, designed to help the learning controller when it goes far away from the reference –e.g. more than 5 metres – bringing the vehicle closer to the reference.

Several learning configurations were tested, comparing the obtained results for both absolute and incremental outputs controllers. All configurations run along 50 learning stages, being each stage composed of 5 laps to each reference trajectory. The worst result for the incremental controller was obtained with the simplest input configuration: steering wheel position and lateral and angular errors. At the final stage, this configuration still requires 346

rescues by the fuzzy controller, with an average error around 1.4 metres. On the other hand, the best result was achieved by one of the complex input configuration, which considers 7 input variables including the previous three plus the vehicle speed and three trajectory curvatures. In this case, the error at the final stage was around 0.4 metres, with only 9 rescues performed by the fuzzy controller.

In contrast to these result, the simplest configuration of the absolute output controller was also the best one. At the last stage, the controller was able to guide the car without any fuzzy rescue, keeping the average error under 0.8 metres. In this case, the incorporation of additional input variables did not help to reduce the error, but increasing it as well as the number of rescues.

The results obtained for this learning controller are encouraging. However, due to the amount of time required for the controller adjustment and the harsh usage of the testing platform on the process, it was decided not to run the learning procedure on the prototype vehicle.

A.3.2. Fuzzy controller adjustment for longitudinal control

The learning techniques as the one presented for the lateral control are an option to address problems when there is no previous information about it or conditioning the system performance by the developer approach is not desired. However, learning from scratch is a long procedure, requiring a wide exploration of the space of the problem. This can be done using virtual environments, but in real ones it is a big challenge. Moreover, it has been shown how the problem complexity expands exponentially as the number inputs increases. This makes impossible to take into account all the possible variables for the problem. Nevertheless, even if this last limitation would not exist, determining the influence of some variables on the system performance could be a really complex task.

The objective of this task herein described was to create a system capable of allowing the evolution of fuzzy rules for the management of the pedals of a vehicle in urban driving contexts. The use of fuzzy logic for control systems has two main advantages: (i) Fuzzy logic obviates the need to use complex approximate models that are either computationally efficient if they are realistic, or unrealistic if they are computationally efficient. (ii) The aim is not to represent the system mathematically, but to emulate the behaviour and experience of human drivers. There is no systematic approach to the design of fuzzy controllers. Instead, how they are designed depends on the knowledge available about the system to be controlled.

On this basis, an algorithm for on-line adaptation of fuzzy controller has been designed and implemented on the prototype vehicle for cruise control. The goal is to evaluate the vehicle performance in real time, adapting the controller configuration in order to overcome any change on the environment conditions, such as road slopes, gear changes, weight of the occupants or other unpredictable parameters. To this end, one defines a zeroth-order TSK fuzzy controller with trapezia for codifying inputs and singletons as consequents. An initial fuzzy controller with all consequents set to zero (this implying the pedals are not acted

upon) evolves over time, adapting both the position of the singletons and the granularity of the trapezia.

For the initial empty controller to evolve, a first module is designed that adapts the positions of the singletons defining the consequents of the system depending on the speed and the acceleration of the vehicle. After a certain amount of time, a second, structural learning module takes responsibility for adding or modifying the trapezia that codify the input variables of the system. Finally, a third module is in charge of filtering the pedal actions, with the aim of remove possible output oscillations.

The system was tested under stepwise changes of the desired speed of the vehicle in two different experiments: (i) over 30 different vehicles in a simulated environment, and (ii) in a real vehicle. The simulations showed that the system is able to provide similar behaviour in different vehicles. The real environment results showed the suitability of the system for real applications, that it had remarkable precision, and was comparable with a human driver.

The adaptation method here developed has been the basis for the implementation of the cooperative adaptive cruise control (CACC) system used on the final experiment, described on the next section.

A.4. Demonstration on open roads

The last part of this dissertation deals with a final experiment performed on open roads. The main goal was to validate the behaviour, viability, robustness and feasibility of the control architecture, as well as the algorithms and methods developed. In this experiment the prototype vehicle performed a 100-km route in driverless mode. To that end, a leader-follower scheme was used along a V2V communication system. Within this scheme, the leading vehicle –another AUTOPIA prototype manually driven – only task is to transmit its trajectory in real time so the follower can use this information as a reference.

For the route selection, two criteria were taken into account: (i) the route length should be enough to validate the robustness of the system to deal with long running time; and (ii) both urban and motorway environments should be present, validating the system capabilities in these two main scenarios. Thus, the first choice was a 140-km route from *Segovia* to the CAR facilities in *Arganda del Rey*. However, several factors hindered the administrative requirements for this route. Therefore, the route was modified as suggested by the traffic patrol of the Civil Guard, setting as start point the town of San Lorenzo de El Escorial. Finally, the route length was around 100 km, being 12 of them in urban environments.

This experiment was organized and performed in collaboration with the city council, the general directorate of traffic (DGT, in Spanish) and the traffic patrol of the Civil Guard. The participation of these entities was fundamental for the experiment since the Spanish regulation on traffic does not authorize the circulation of autonomous vehicles on roads. For this reason, the two vehicles were treated by the authorities in accordance with the regulations for special transport, being escorted by traffic units throughout the test.

Regarding the information exchange, the same communication hardware implemented for

the AUTOPIA participation on the GCDC was used. This system is based on the standard IEEE 802.11p and the CALM protocol, both specially designed for communication among vehicles. This system is embedded as an independent hardware, using a network connection to communicate with the main on-board computer. From the latter's point of view, the communications system is totally transparent, since it behaves just as a gateway among the CALM/802.11p and the network interfaces. For the data transmission, a self-made binary message is used. On it, the system packages the information related to the state of the vehicle, i.e. the position, speed, orientation and positioning quality among others.

Based on the data received, the trailing vehicle is able to keep a real-time trace of the trajectory followed by the leader. In particular, the position data is stored by the mission module as checkpoints for the car. In parallel, the planner module processes the mission data and generates a suitable reference path for the trailing vehicle. For the inclusion of checkpoints in the mission list, the system addresses two additional issues: (i) When it is initialized, the system sets the current position of the trailing vehicle as the first checkpoint in the mission module. The planner is then able to start generating the route as soon as the first message from the leading vehicle is received since it requires a minimum of two points to run. (ii) To reduce the number of checkpoints added, a minimum distance is required between the last added point and a new one. This value is set to be an appropriate trade-off between restricting the number of checkpoints to process and maintaining resemblance of the trace with the real trajectory of the leader.

For the longitudinal control of the vehicle, a CACC controller has been implemented. The controller uses the same adaptation method developed for the CC controller on the previous section. In this case, the input variables are the distance error with the leading car and the relative speed among vehicles. On its turn, the output remains a unique signal controlling both throttle and brake. For the controller adaptation, the learning method evaluates the evolution of the two input variables, deciding whether the controller's consequents need to be augmented or reduced. As a safety measure, the distance reference is calculated proportionally to the vehicle speed. To that end a time gap of 0.6 seconds is considered. This value is taken from the safety requirements set for the GCDC. Likewise, a 10-m minimum distance was set.

Regarding the lateral control, a dual controller has been implemented. In this case, the learning approach was not used since it was still under development. The dual controller combines two independent ones according to the vehicle speed. For the implementation of the two controllers, the same input-output configuration is used: two input variables: the vehicle's angular and lateral errors vehicle; and a single output: the reference position for the steering wheel. The output is codified by means of singletons distributed in the [-1, 1] range, with -1 being the maximum possible turn of the steering wheel to the right and 1 the maximum possible turn to the left. For the low-speed speed scenarios, the controller considers a wider range for the inputs, since they reach greater values without the vehicle being off the trajectory –e.g., in sharp turns. Both controllers also consider the trajectory curvature, estimated by approximating the immediate trajectory to a circumference.

In general, an appropriate behaviour for autonomous control of the vehicle on motorways

and urban environments was observed along the entire route. First of all, it is necessary to remark the proper performance of the longitudinal controller developed, which allowed tracking the leader with less than one-meter errors. This behaviour gives the feeling that the trailing vehicle is virtually couple to its predecessor, increasing the sense of safety for all the occupants of the vehicle. On the other hand, the lateral control system was able to keep the vehicle within the lane, demonstrating its ability to work autonomously following a leading vehicle.

BIBLIOGRAFÍA

Arulampalam, M. S., Maskell, S., Gordon, N. y Clapp, T. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, vol. 50(2), páginas 174–188, 2002.

Bak, A., Gruyer, D., Bouchafa, S. y Aubert, D. Multi-sensor localization-visual odometry as a low cost proprioceptive sensor. En *2012 15th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, páginas 1365–1370. IEEE, 2012.

Bechtel. Compendium of executive summaries from the maglev system concept definition final reports. Informe técnico, U.S. Department of Transportation, 1993.

Bergenhem, C., Huang, Q., Benmimoun, A. y Robinson, T. Challenges of platooning on public motorways. En *17th World Congress on Intelligent Transport Systems*. 2010.

Bevly, D. y Parkinson, B. Cascaded kalman filters for accurate estimation of multiple biases, dead-reckoning navigation, and full state feedback control of ground vehicles. *IEEE Transactions on Control Systems Technology,*, vol. 15(2), páginas 199–208, 2007. ISSN 1063-6536.

Boyan, J. y Moore, A. W. Generalization in reinforcement learning: Safely approximating the value function. *Advances in neural information processing systems*, páginas 369–376, 1995.

C2C-CC. Car 2 car communication consortium manifesto. Informe técnico, CAR 2 CAR Communication Consortium, 2007.

Comisión Europea. *Política de transportes por carretera - Una Europa de carreteras abiertas*. Dirección General de Energía y Transportes, 2006.

Crossbow. *440 Series Inertial Systems – User Guide*. Crossbow, 2009.

Davis, L. Effect of adaptive cruise control systems on traffic flow. *Physical Review E*, vol. 69(6), página 066110, 2004.

DeLooze, L. L. y Viner, W. R. Fuzzy Q-learning in a nondeterministic environment: developing an intelligent ms. pac-man agent. En *IEEE Symposium on Computational Intelligence and Games, 2009. CIG 2009.*, páginas 162–169. IEEE, 2009.

Desjardins, C. y Chaib-draa, B. Cooperative adaptive cruise control: A reinforcement learning approach. *IEEE Transactions on Intelligent Transportation Systems*, vol. 12(4), páginas 1248–1260, 2011. ISSN 1524-9050.

Dirección General de Tráfico. *Las principales causas de la Siniestralidad Vial España 2011*. Dirección General de Tráfico, 2012.

DOT. Achieving the vision: From VII to IntelliDrive. Informe técnico, Research and Innovative Technology Administration, 2010.

Doucet, A., Godsill, S. y Andrieu, C. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, vol. 10, páginas 197–208, 2000. ISSN 0960-3174.

Ernst, T., Nebehaj, V. y Srasen, R. CVIS: CALM proof of concept preliminary results. En *Proc. Intelligent Transport Systems Telecommunications 9th Int. Conf*, páginas 80–85. 2009.

García, R. y De Pedro, T. First application of the ORBEX coprocessor: Control of unmanned vehicles. *Mathware and Soft Computing*, vol. 7, páginas 265–273, 2000.

García Rosa, R. y De Pedro, T. Modeling a fuzzy coprocessor and its programming language. *Mathware & soft computing*, vol. 5(2-3), páginas 167–174, 1998.

Geiger, A., Moosmann, F., Lauer, M., Ranft, B., Rapp, H. y Ziegler, J. Team AnnieWAY's entry to GCDC 2011. 2011.

Godoy, J., Milanés, V., Rastelli, J. P., Villagrá, J., De Pedro, T., González, C. et al. Implementación de un sistema de localización para vehículos sin conductor. En *Seminario Anual de Automática, Electrónica Industrial e Intrumentación*. 2010.

Gordon, N., Salmond, D. y Smith, A. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings F - Radar and Signal Processing*, vol. 140(2), páginas 107 –113, 1993. ISSN 0956-375X.

Hiblot, N., Gruyer, D., Barreiro, J.-S. y Monnier, B. Pro-sivic and roads. a software suite for sensors simulation and virtual prototyping of adas. En *Proceedings of DSC*. 2010.

Hinton, G. E. y Sejnowski, T. J. *Unsupervised learning: foundations of neural computation*. MIT press, 1999.

Huang, A., Olson, E. y Moore, D. LCM: Lightweight communications and marshalling. En *International Conference on Intelligent Robots and Systems*, páginas 4057 –4062. 2010. ISSN 2153-0858.

Huang, A. S., Olson, E. y Moore, D. Lightweight communications and marshalling for low latency interprocess communication. Informe técnico, Massachusetts Institute of Technology, 2009.

International Traffic Safety Data and Analysis Group. *IRTAD – Road Safety Annual Report 2011*. 2011.

ISO. Intelligent transport systems – Communications Access for Land Mobiles (CALM) – architecture. Informe Técnico ISO 21217:2010, International Organization for Standardization, 2010.

Jiang, D. y Delgrossi, L. IEEE 802.11p: Towards an international standard for wireless access in vehicular environments. En *Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE*, páginas 2036–2040. 2008. ISSN 1550-2252.

de Jongh, J. GCDC communication stack – building and installing the calm fast router. Informe técnico, SPITS/GCDC, 2011.

Julier, S. J. y Uhlmann, J. K. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, vol. 92(3), páginas 401–422, 2004.

Kaelbling, L. P., Littman, M. L. y Moore, A. W. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, vol. 4, páginas 237–285, 1996.

Kalman, R. E. A new approach to linear filtering and prediction problems. *Transaction of the ASME – Journal of Basic Engineering*, vol. 82(1), páginas 35–45, 1960.

Kaplan, E. D. *Understanding GPS: Principles and applications*. Artech House, 1996.

Kelly, A. A 3D state space formulation of a navigation Kalman filter for autonomous vehicles. Informe técnico, DTIC Document, 1994.

Lauer, M. A case study on learning a steering controller from scratch with reinforcement learning. En *Intelligent Vehicles Symposium (IV), 2011 IEEE*, páginas 260–265. IEEE, 2011.

Li, W. y Leung, H. Constrained unscented Kalman filter based fusion of GPS/INS/digital map for vehicle localization. En *Proc. IEEE Intelligent Transportation Systems*, vol. 2, páginas 1362–1367. 2003.

Llorca, D. F., Milanés, V., Alonso, I. P., Gavilán, M., Daza, I. G., Pérez, J. y Sotelo, M. A. Autonomous pedestrian collision avoidance using a fuzzy steering controller. *Intelligent Transportation Systems, IEEE Transactions on*, vol. 12(2), páginas 390–401, 2011.

Lu, H., Zhan-rong, J., Ming-ming, W. y Li-xin, Z. Adaptive extended Kalman filter based on genetic algorithm for tightly-coupled integrated inertial and GPS navigation. En *Second International Conference on Intelligent Computation Technology and Automation, 2009. ICICTA '09.*, vol. 1, páginas 520–524. 2009.

Martí, E. D., Martín, D., García, J., de la Escalera, A., Molina, J. M. y Armingol, J. M. Context-aided sensor fusion for enhanced urban navigation. *Sensors*, vol. 12(12), páginas 16802–16837, 2012.

Martinez, J.-J. y Canudas-de Wit, C. A safe longitudinal control for adaptive cruise control and stop-and-go scenarios. *IEEE Transactions on Control Systems Technology*, vol. 15(2), páginas 246–258, 2007.

Milanés, V. *Sistema de control de tráfico para la coexistencia entre vehículos autónosmos y manuales mediante comunicaciones inañámbricas*. Tesis Doctoral, Universidad de Alcalá. Escuela Politénica Superior. Departamento de electrónica, 2010.

Milanés, V., Alonso, J., Bouraoui, L. y Ploeg, J. Cooperative maneuvering in close environments among cybercars and dual-mode cars. *IEEE Transactions on Intelligent Transportation Systems*, vol. 12(1), páginas 15–24, 2011a. ISSN 1524-9050.

Milanés, V., Godoy, J., Villagrá, J. y Pérez, J. Automated on-ramp merging system for congested traffic situations. *IEEE Transactions on Intelligent Transportation Systems*, vol. 12(2), páginas 500 –508, 2011b. ISSN 1524-9050.

Milanés, V., Llorca, D. F., Villagrá, J., Pérez, J., Parra, I., González, C. y Sotelo, M. A. Vision-based active safety system for automatic stopping. *Expert Systems with Applications*, vol. 39(12), páginas 11234–11242, 2012a.

Milanés, V., Naranjo, J. E., González, C., Alonso, J. y de Pedro, T. Autonomous vehicle based in cooperative GPS and inertial systems. *ROBOTICA*, vol. 26(5), páginas 627–633, 2008.

Milanés, V., Onieva, E., Pérez, J., Simo, J., González, C. y de Pedro, T. Making transport safer: A V2V-Based automated emergency braking system. *Transport*, vol. 26(3), páginas 290–302, 2011c.

Milanés, V., Villagrá, J., Godoy, J., Simó, J., Pérez, J. y Onieva, E. An Intelligent V2I-Based Traffic Management System. *IEEE Transactions on Intelligent Transportation Systems*, vol. 13(1), páginas 49–58, 2012b.

Mitchell, M., Oldham, J. y Samuel, A. *Advanced Linux Programming*. Landmark Series. New Riders, 2001. ISBN 9780735710436.

Mohri, M., Rostamizadeh, A. y Talwalkar, A. *Foundations of machine learning*. The MIT Press, 2012.

Montemerlo, M., Roy, N. y Thrun, S. Perspectives on standardization in mobile robot programming: The carnegie mellon navigation (CARMEN) toolkit. En *In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, páginas 2436–2441. 2003.

Naranjo, J. *Sistema de Conducción Automática de Vehículos basado en Lógica Borrosa y Sistemas Globales de Posicionamiento por Satélite: Programa AUTOPIA.*. Tesis Doctoral, Universidad Politécnica de Madrid, 2005.

Naranjo, J., González, C., García, R. y de Pedro, T. ACC+Stop&Go maneuvers with throttle and brake fuzzy control. *IEEE Transactions on Intelligent Transportation Systems*, vol. 7(2), páginas 213 – 225, 2006. ISSN 1524-9050.

Naranjo, J., González, C., García, R., de Pedro, T. y Haber, R. Power-steering control architecture for automatic driving. *IEEE Transactions on Intelligent Transportation Systems*, vol. 6(4), páginas 406 – 415, 2005. ISSN 1524-9050.

Naranjo, J. E., González, C., García, R. y de Pedro, T. Cooperative throttle and brake fuzzy control for ACC + Stop&Go maneuvers. *IEEE Transactions on Vehicular Technology*, vol. 56(4), páginas 1623–1630, 2007.

Naranjo, J. E., González, C., Reviejo, J., García, R. y De Pedro, T. Adaptive fuzzy control for inter-vehicle gap keeping. *IEEE Transactions on Intelligent Transportation Systems*, vol. 4(3), páginas 132–142, 2003.

Ndjeng, A., Glaser, S. y Gruyer, D. A multiple model localization system for outdoor vehicles. En *Intelligent Vehicles Symposium, 2007 IEEE*, páginas 1050–1055. 2007. ISSN 1931-0587.

Ndjeng Ndjeng, A., Lambert, A., Gruyer, D. y Glaser, S. Experimental comparison of Kalman filters for vehicle localization. En *Proc. IEEE Intelligent Vehicles Symp*, páginas 441–446. 2009.

Nedjah, N. . y de Macedo, L. *Swarm Intelligent Systems*. Springer, 2006.

Nehaniv, C. L. y Dautenhahn, K. *Imitation and social learning in robots, humans and animals: behavioural, social and communicative dimensions*. Cambridge University Press, 2007.

Newman, P. M. MOOS-mission orientated operating suite. Informe técnico, Massachusetts Institute of Technology, 2008.

NMEA. *NMEA 0183 – Standard for Interfacing Marine Electronic Devices*. National Marine Electronics Association (U.S.), 2002.

Nowakowski, C., OConnel, J., Shladover, S. y Cody, D. Cooperative adaptive cruise control: Driver acceptance of following gap settings less than one second. En *Proc. Human Factors and Ergonomics Society, 54th annual Meeting.*. 2010.

van Nunen, E., Kwakkernaat, M. R. J. A. E., Ploeg, J. y Netten, B. D. Cooperative competition for future mobility. *IEEE Transactions on Intelligent Transportation Systems*, vol. PP(99), páginas 1 –8, 2012. ISSN 1524-9050.

Onieva, E. *Técnicas Difusas y Evolutivas para el Control de Vehículos en Entornos Reales y Virtuales.*. Tesis Doctoral, Universidad de Granada, 2011.

Parnas, D. L. On the criteria to be used in decomposing systems into modules. *Commun. ACM*, vol. 15(12), páginas 1053–1058, 1972. ISSN 0001-0782.

Pérez, J. *Agentes de Control de Vehículos Autónomos en Entornos Urbanos y Autovías.* Tesis Doctoral, Universidad Complutense de Madrid., 2012.

Pérez, J., González, C., Milanés, V., Onieva, E., Godoy, J. y de Pedro, T. Modularity, adaptability and evolution in the AUTOPIA architecture for control of autonomous vehicles. En *IEEE International Conference on Mechatronics (ICM)*, páginas 1 –5. 2009.

Pérez, J., Milanés, V., Alonso, J., Onieva, E. y De Pedro, T. Adelantamiento con vehiculos autónomos en carreteras de doble sentido. *Revista Iberoamericana de Automática e Informática Industrial RIAI*, vol. 7(3), páginas 25–33, 2010.

Pérez, J., Milanés, V. y Onieva, E. Cascade architecture for lateral control in autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, vol. 12(1), páginas 73 –82, 2011a. ISSN 1524-9050.

Pérez, J., Milanés, V., Onieva, E., Godoy, J. y Alonso, J. Longitudinal fuzzy control for autonomous overtaking. En *IEEE International Conference on Mechatronics (ICM), 2011*, páginas 188 –193. 2011b.

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R. y Ng, A. ROS: an open-source robot operating system. En *ICRA workshop on open source software*, vol. 3. 2009.

Rezaei, S. y Sengupta, R. Kalman filter-based integration of DGPS and vehicle sensors for localization. *IEEE Transactions on Control Systems Technology*, vol. 15(6), páginas 1080–1088, 2007. ISSN 1063-6536.

Shladover, S. Cooperative (rather than autonomous) vehicle-highway automation systems. *Intelligent Transportation Systems Magazine, IEEE*, vol. 1(1), páginas 10 –19, 2009. ISSN 1939-1390.

Soria, J. Factor humano. En *Tráfico y Seguridad Vial. Dirección General de Tráfico*. 2001.

Sotelo, M. A., Fernández, D., Naranjo, J. E., González, C., García, R. y de Pedro, T. Laser-based adaptive cruise control for intelligent vehicles. En *IEEE ICINCO 2004*. Setubal, Portugal, 2004.

Sugeno, M. On stability of fuzzy systems expressed by fuzzy rules with singleton consequents. *IEEE Transactions on Fuzzy Systems*, vol. 7(2), páginas 201 –224, 1999. ISSN 1063-6706.

Sutton, R. S. y Barto, A. G. *Reinforcement learning: An introduction*, vol. 1. Cambridge Univ Press, 1998.

Tan, H.-S., Rajamani, R. y Zhang, W.-B. Demonstration of an automated highway platoon system. En *American Control Conference, 1998. Proceedings of the 1998*, vol. 3, páginas 1823–1827 vol.3. 1998. ISSN 0743-1619.

Tejado, I., Milanés, V., Villagrá, J., Godoy, J., HosseinNia, H. y Vinagre, B. M. Low speed control of an autonomous vehicle by using a fractional pi controller. En *18th IFAC World Congress*, páginas 15025–15030. 2011.

Thorndike, E. L. *Animal Intelligence*. Hafner, Darien, 1911.

Thrun, S., Bennewitz, M., Burgard, W., Cremers, A. B., Dellaert, F., Fox, D., Hahnel, D., Rosenberg, C., Roy, N., Schulte, J. y Schulz, D. MINERVA: a second-generation museum tour-guide robot. En *Proc. IEEE Int Robotics and Automation Conf*, vol. 3, páginas 1999–2005. 1999.

Thrun, S. et al. Stanley: The robot that won the DARPA Grand Challenge. *Journal of Field Robotics*, vol. 23(9), páginas 661–692, 2006. ISSN 1556-4967.

Toledo-Moreo, R., Betaille, D., Peyret, F. y Laneurit, J. Fusing GNSS, dead-reckoning, and enhanced maps for road vehicle lane-level navigation. *IEEE Journal of Selected Topics in Signal Processing*, vol. 3(5), páginas 798–809, 2009. ISSN 1932-4553.

Tong, G., Fang, Z. y Xu, X. A particle swarm optimized particle filter for nonlinear system state estimation. En *Proc. IEEE Congress Evolutionary Computation CEC 2006*, páginas 438–442. 2006.

de Torres, E., Villagrá, J., Godoy, J. y de Pedro, T. Implementación del control lateral sobre un vehículo de serie con servodirección asistida. En *8vo Workshop Robocity2030-II – Robots de Exteriores*. 2010.

Toulminet, G., Boussuge, J. y Laurgeau, C. Comparative synthesis of the 3 main european projects dealing with cooperative systems (CVIS, SAFESPOT and COOPERS) and description of COOPERS demonstration site 4. En *Proc. 11th Int. IEEE Conf. Intelligent Transportation Systems ITSC*, páginas 809–814. 2008.

Trimble. *Trimble BD960 GNSS Receiver Module – User Guide*. Trimble, 2008.

Urmson, C., Anhalt, J., Bae, H., Bagnell, J. A. D., Baker , C. R., Bittner, R. E., Brown, T., Clark, M. N., Darms, M., Demitrish, D., Dolan, J. M., Duggins, D., Ferguson, D., Galatali, T., Geyer, C. M., Gittleman, M., Harbaugh, S., Hebert, M., Howard, T., Kolski, S., Likhachev, M., Litkouhi, B., Kelly, A., McNaughton, M., Miller, N., Nickolaou, J., Peterson, K., Pilnick, B., Rajkumar, R., Rybski, P., Sadekar, V., Salesky, B., Seo, Y.-W., Singh, S., Snider,

J. M., Struble, J. C., Stentz, A. T., Taylor , M., Whittaker, W. R. L., Wolkowicki, Z., Zhang, W. y Ziglar, J. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics Special Issue on the 2007 DARPA Urban Challenge, Part I*, vol. 25(1), páginas 425–466, 2008.

Uzcategui, R. y Acosta-Marum, G. Wave: A tutorial. *IEEE Communications Magazine*, vol. 47(5), páginas 126–133, 2009.

Van Arem, B., van Driel, C. y Visser, R. The impact of cooperative adaptive cruise control on traffic-flow characteristics. *IEEE Transactions on Intelligent Transportation Systems*, vol. 7(4), páginas 429–436, 2006. ISSN 1524-9050.

Watkins, C. J. C. H. y Dayan, P. Q-learning. *Machine learning*, vol. 8(3-4), páginas 279–292, 1992.

Welch, G. y Bishop, G. An introduction to the Kalman filter. Informe Técnico TR 95-041, University of North Carolina, Department of Computer Science., 1995.

Williams, B. The CALM handbook, continuous air-interface long and medium range. ISO TC204: ETSI ERM TG37, 2004.

Zadeh, L. Fuzzy sets. *Information and Control*, vol. 8(3), páginas 338–353, 1965.

Zadeh, L. Fuzzy logic = computing with words. *IEEE Transactions on Fuzzy Systems*, vol. 4(2), páginas 103–111, 1999.

Zhang, P., Gu, J., Milios, E. E. y Huynh, P. Navigation with IMU/GPS/digital compass with unscented Kalman filter. En *Proc. IEEE Int Mechatronics and Automation Conf*, vol. 3, páginas 1497–1502. 2005.