UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

DISEÑO DE UN SISTEMA DE GUIADO DE UN ROBOT MÓVIL MEDIANTE VISIÓN ARTIFICIAL CON CAPACIDAD DE TOLERANCIA A FALLOS

AUTOR: ADRIÀ MOMPÓ ALEPUZ

TUTOR: ÁNGEL RODAS JORDÁ

COTUTOR: JOAQUÍN GRACIA MORÁN

Curso Académico: 2016-17

Abstract

This project consists of the design and implementation of the guiding system with basic fault tolerance capabilities of a mobile robot based on computer vision and ultrasonic distance sensors. Additionally, a simple circuit will be designed, with certain singularities that the robot will be able to face. All this will constitute a simplified test platform for an autonomous vehicle that serves as a starting point for the further development of the adaptive fault tolerance project DINAMOS, started by the Fault Tolerance Group of the Computer and Systems Informatics Department at the Universitat Politècnica de València.

For the development of the project, the single-board computer Raspberry Pi will be used as the processing and control unit, as well as a camera module for Raspberry Pi, ultrasonic sensors, a mobile robot, the image processing library OpenCV and the matrix and array computing library NumPy. The programming language used will be Python.

Several control and image processing algorithms will be designed and implemented to guide the robot and recognize the unique elements of the circuit, such as intersections in the path, different models of traffic signals and multiple objects. Finally, the robot's perceptual system will be endowed with fault tolerance capabilities based on functional hardware redundancy and temporal redundancy.

Key words: *Mobile robot, Computer vision, Fault tolerance, Signal processing, Digital Controller, Optical Sensor, Ultrasonic Sensor, Electronics, Object oriented programming*

Resumen

El presente trabajo consiste en el diseño y puesta en marcha del sistema de guiado de un robot móvil basado en visión artificial y sensores de distancia ultrasónicos que presentará capacidades básicas de tolerancia a fallos. Se diseñará además un circuito simple con ciertas singularidades que el robot será capaz de afrontar. Todo esto constituirá una plataforma de pruebas simplificada de un vehículo autónomo que servirá como punto de partida para el posterior desarrollo del proyecto de tolerancia a fallos adaptativa DINAMOS, iniciado por el Grupo de Tolerancia a Fallos del Departamento de Informática de Sistemas y Computadores de la Universitat Politècnica de València.

Para el desarrollo del trabajo se empleará el computador de placa reducida Raspberry Pi como unidad de procesamiento y control, un módulo de cámara para la Raspberry Pi, sensores ultrasónicos, un robot móvil, la librería de procesamiento de imágenes OpenCV y la de cálculo matricial NumPy. El lenguaje de programación empleado será Python.

Se diseñarán e implementarán distintos algoritmos de control y procesamiento de imágenes para el guiado del robot y el reconocimiento de los elementos singulares del circuito, como serán las intersecciones en el trayecto, distintos modelos de señales de tráfico y múltiples objetos. Finalmente se dotará al sistema perceptivo del robot de capacidades de tolerancia a fallos basándose en la redundancia hardware funcional y la redundancia temporal.

Palabras clave: *Robot móvil, Visión Artificial, Tolerancia a fallos, Procesamiento de señal, Controlador Digital, Sensor Óptico, Sensor Ultrasónico, Electrónica, Programación orientada a objetos*

Resum

El present treball consisteix en el disseny i posada en marxa del sistema de guiat d'un robot mòbil basat en visió artificial i sensors de distància ultrasònics que presentarà capacitats bàsiques de tolerància a fallades. Es dissenyarà a més un circuit simple amb certes singularitats que el robot serà capaç d'afrontar. Tot açò constituirà una plataforma de probes simplificada d'un vehicle autònom que servirà com a punt de partida per al posterior desenvolupament del projecte de tolerància a fallades adaptativa DINAMOS, iniciat pel Grupo de Tolerància a Fallades del Departament d'Informàtica de Sistemas i Computadores de la Universitat Politècnica de València.

Per al desenvolupament del treball es farà us del computador de placa reduïda Raspberry Pi com a unitat de processament i control, un mòdul de càmera per a la Raspberry Pi, sensors ultrasònics, un robot mòbil, la llibreria de processament d'imatges OpenCV i la de càlcul matricial NumPy. El llenguatge de programació usat serà Python.

Es dissenyaran i implementaran distints algoritmes de control i processament de imatges para el guiat del robot i el reconeixement dels elements singulars del circuit, como seran les interseccions en el trajecte, distints models de senyals de tràfic i múltiples objectes. Finalment se dotarà al sistema perceptiu del robot de capacitats de tolerància a fallades basant-se en la redundància hardware funcional i la redundància temporal.

Paraules clau: *Robot mòbil, Visió Artificial, Tolerància a fallades, Processament de senyal, Controlador Digital, Sensor Òptic, Sensor Ultrasònic, Electrònica, Programació orientada a objectes*

Índice general

Abstract i

Resumen ii

Resum iii

Índice general v

Índice de figuras vii

Índice de tablas ix

Índice de ecuaciones x

Parte I Memoria 1

Capítulo 1 Introducción general y objetivos 2

1.1 Objetivos ... 2

1.2 Introducción al problema. Antecedentes, motivación y justificación ... 3

1.2.1 Introducción y antecedentes ... 3

1.2.2 Motivación y justificación ... 4

1.3 Estructura del trabajo ... 5

Capítulo 2 Materiales y métodos 7

2.1 Hardware del vehículo ... 7

2.2 Software ... 10

2.3 Entorno del vehículo ... 11

Capítulo 3 Plataforma de trabajo. Diseño inicial y calibración. 12

3.1 Pasos preliminares ... 12

3.2 Prueba de funcionamiento y calibrado ... 15

Capítulo 4 Visión con OpenCV. Fundamentos y entorno de programación. 18

4.1 Introducción a la visión por computador ... 18

4.2 Algoritmos básicos ... 19

Capítulo 5 Guiado básico. Diseño del controlador 24

5.1 Análisis de alternativas ... 24

5.2 Proceso y bucle cerrado. Sistema general ... 26

5.3 Diseño del controlador ... 30

5.3.1 Diseño del programa de control ... 30

5.3.2 Diseño y ajuste del controlador PID 34
Capítulo 6 Guiado y percepción avanzados. Demostradores **42**
6.1 Algoritmos de percepción del entorno 42
6.1.1 Detección de objetos 43
6.1.2 Identificación de cruces 47
6.1.3 Detección y clasificación de señales 54
6.2 Demostradores 59
6.2.1 Primer demostrador. Circuito cerrado, parada ante objetos 59
6.2.2 Segundo demostrador. Circuito con cruces 60
6.2.3 Tercer demostrador. Detección de señales 61
Capítulo 7 Tolerancia a fallos **64**
7.1 Introducción a la tolerancia a fallos 64
7.2 Aplicación al vehículo de pruebas 65
7.2.1 Tarea de reconocimiento de objetos 66
7.2.2 Tarea de reconocimiento de cruces y señales de tráfico 68
7.3 Implementación 70
7.4 Discusión 71
Capítulo 8 Conclusiones y trabajos futuros **74**
8.1 Conclusiones 74
8.2 Trabajos futuros 75
Capítulo 9 Bibliografía **76**
Parte II Presupuesto **1**

Índice de figuras

Figura 2.1. Esquema original del sistema electrónico del robot (SunFounder, s.f.). 8

Figura 3.1. Conexión sensores ultrasónicos – Raspberry Pi 12

Figura 3.2. Disposición de los sensores ultrasónicos y la cámara 13

Figura 3.3. Conexión remota VNC 14

Figura 3.4. Trazado del circuito de pruebas. 15

Figura 3.5. Diagrama de ejecución del hilo paralelo de obtención de distancias mediante los dos sensores ultrasónicos 16

Figura 3.6. Disposición angular de los sensores ultrasónicos. 17

Figura 4.1. Espacios de colores 19

Figura 5.1. Imagen de la pista con reflejos 25

Figura 5.2. Imagen de la pista, desplazamiento lateral respecto al centro 25

Figura 5.3. Disposición angular de la cámara 26

Figura 5.4. Proceso general 26

Figura 5.5. Proceso desglosado 27

Figura 5.6. Vista superior del vehículo en la pista. Relación entre la coordenada horizontal en la banda inferior de la imagen y la distancia lateral d del vehículo a la pista. 27

Figura 5.7. Proceso, dinámica del vehículo 27

Figura 5.8. Proceso, cinemática y transformación geométrica 28

Figura 5.9. No linealidad del proceso. Para una misma acción de control, se producen variaciones distintas en d, (relacionada directamente con la variable de control) según el estado previo de los parámetros cinemáticos del vehículo. 28

Figura 5.10. Detalle de proceso de transformación geométrica a partir del estado cinemático 29

Figura 5.11. Proceso de transformación óptica 29

Figura 5.12. Bucle cerrado 29

Figura 5.13. Imagen de pista, quinto inferior sin reflejos 31

Figura 5.14. Binarización por umbralización 31

Figura 5.15. Binarización por detección de bordes 32

Figura 5.16. Controlador PID en Python 33

Figura 5.17. Control Anti-Windup 34

Figura 5.18. Respuesta sostenida, método ZN 35

Figura 5.19. Controlador P, resultados 37
Figura 5.20. Controlador PD, método ZN 39
Figura 5.21. Controlador PD Ajustado 39
Figura 5.22. Controlador PID, método ZN 41
Figura 5.23. Controlador PID Ajustado 41
Figura 6.1. Casos extremos de variación de tonalidad y brillo en el color de la pista 45
Figura 6.2. Resultado de la detección de objetos por diferencia de colores ante distintos ejemplares. 47
Figura 6.3. Falso negativo en el reconocimiento de un objeto con color similar al fondo. 47
Figura 6.4. Histogramas de ángulos de bordes detectados 49
Figura 6.5. Gradientes calculados en rectas, curvas y cruces, con variaciones de iluminación 51
Figura 6.6. Vectores y valores propios en rectas, curvas, y cruces 52
Figura 6.7. Resultado del algoritmo de identificación de cruces en diferentes situaciones. 'rel' indica la relación dada por el cociente del menor autovalor entre el mayor, y 'sum' indica la suma de los autovalores. 53
Figura 6.8. Falso positivo, reconocimiento de cruces 54
Figura 6.9. Falso negativo, reconocimiento de cruces. 54
Figura 6.10. Señales de tráfico usadas 56
Figura 6.11. Señales de tráfico con bordes 56
Figura 6.12. Resultados del algoritmo de detección y clasificación de señales de tráfico 58
Figura 6.13. Circuito modificado para el primer demostrador. 59
Figura 6.14. Demostrador 1, diagrama de flujo. Detección de objetos. 60
Figura 6.15. Demostrador 2, diagrama de flujo. Identificación de cruces. 61
Figura 6.16. Demostrador 3, diagrama de flujo. Reconocimiento de señales. 62
Figura 7.1. Rangos de medida del sensor ultrasónico 67
Figura 7.2. Señales de tráfico detectadas en el borde del rango máximo (izquierda), y no detectadas por estar situadas fuera del rango (derecha) 69
Figura 7.3. Módulo de tolerancia a fallos, flujo de operación 70
Figura 7.4. Demostrador 4, diagrama de flujo. Tolerancia a fallos 71

Índice de tablas

Tabla 5.1. Parámetros del método Ziegler-Nichols 35

Tabla 6.1. Rangos HSL del modelo de fondo 46

Índice de ecuaciones

Ecuación 5.1. Parámetros obtenidos del método Ziegler-Nichols 35

Ecuación 5.2. PID – Ziegler-Nichols 36

Ecuación 5.3. PID – tiempo discreto 36

Ecuación 5.4. Relación de parámetros entre la ecuación PID de Ziegler-Nichols y la ecuación PID de tiempo discreto empleada 36

Ecuación 5.5. Controlador P, método ZN 37

Ecuación 5.6. Controlador PD, método ZN 38

Ecuación 5.7. Controlador PD Ajustado 38

Ecuación 5.8. Controlador PID, método ZN 40

Ecuación 5.9. Controlador PID Ajustado 40

Parte I
Memoria

CAPÍTULO 1

INTRODUCCIÓN GENERAL Y OBJETIVOS

1.1 OBJETIVOS

El objetivo de este trabajo es establecer un prototipo funcional simplificado de un vehículo autónomo basado en un robot móvil guiado mediante visión por computador. Dicho sistema servirá para evaluar posteriormente técnicas de tolerancia a fallos desarrolladas en el marco del proyecto DINAMOS de la Universitat Politècnica de Valènica (UPV).

Esta plataforma debe presentar una serie de características que permitan tanto acercarse a la realidad de forma técnica y funcional, simulando los sistemas de guiado de un vehículo autónomo, como ofrecer versatilidad a la hora de realizar las pruebas, estableciendo un entorno dinámico y ajustable a distintos niveles de complejidad. Todo esto teniendo siempre en cuenta el coste y la limitación de recursos, puesto que, al no disponer de un vehículo autónomo comercial, se debe optar por el uso de prototipos, técnicas y materiales al alcance.

Del balance de estos requisitos se escoge la vigente opción, que se trata de usar un robot móvil controlado por la computadora Raspberry Pi haciendo uso de la librería de procesamiento de imágenes *OpenCV*. El entorno de pruebas se sitúa en el laboratorio de Tolerancia a Fallos de la Escuela de Informática de la UPV.

Para poder llevar a cabo el trabajo se deben alcanzar una serie de objetivos específicos previos diferenciables.

- Entorno de pruebas: diseñar un circuito que presente elementos singulares útiles para los ensayos, como intersecciones, señales de tráfico u objetos.
- Electrónica: ampliar las capacidades del robot añadiendo módulos de detección de distancia por ultrasonidos, así como también la adaptación a las necesidades del trabajo del software que proporciona la interfaz a los distintos actuadores.
- Control del guiado del vehículo: evaluar distintas alternativas en función de las capacidades que presentan los medios técnicos usados, para a continuación implementar y ajustar el controlador seleccionado.
- Procesamiento de imágenes y visión por computador: diseñar y evaluar distintos algoritmos que puedan hacer frente a múltiples situaciones de percepción del entorno.
- Tolerancia a fallos: evaluar la aplicabilidad de un diseño tolerante a fallos al sistema de guiado del robot y, en función de ciertos criterios, dotar de redundancia al sistema perceptivo formado por la cámara y los sensores de distancia.

A lo largo del trabajo se establecen una serie de hitos, llamados demostradores, que en su conjunto dan lugar a la plataforma de pruebas. Estos demostradores, que ofrecen distintos niveles de complejidad, están constituidos por un programa y un circuito completos y válidos por sí mismos para efectuar ensayos de tolerancia a fallos en el ámbito del proyecto DINAMOS.

1.2 INTRODUCCIÓN AL PROBLEMA. ANTECEDENTES, MOTIVACIÓN Y JUSTIFICACIÓN

1.2.1 INTRODUCCIÓN Y ANTECEDENTES

Con el avance de la computación en las últimas décadas, ha sido posible desarrollar sistemas capaces de procesar información cada vez más compleja de forma más eficiente y asequible. Uno de los campos que más se ha beneficiado de esto ha sido el de la robótica, en concreto la rama encargada de la percepción y el control. Gracias a técnicas de localización en el entorno, visión por computador o aprendizaje automático, entre otras, ha sido posible dotar a los diferentes sistemas robotizados de habilidades antes inimaginables, como la navegación en entornos desconocidos y variables o el reconocimiento visual de formas complejas.

La aplicación más directa de estos avances al mundo cotidiano, y que puede tener un gran impacto sobre la sociedad en los próximos años, es la de los vehículos con capacidades de navegación autónoma. Pese a que hoy en día empiezan a ser viables técnicamente, dada la gran complejidad tecnológica que implican, el entorno impredecible en el que se desenvuelven y, sobre todo, el hecho de que involucran la vida de personas, se requieren unos altos estándares de seguridad y fiabilidad operativa. Es por ello que todavía se encuentran en fases de desarrollo por parte de distintas empresas. Aunque en condiciones normales de operación ciertos subsistemas pueden presentar dificultades en la percepción del entorno (como una cámara en condiciones atmosféricas adversas), esto generalmente no supone un problema debido a, por un lado, los diseños redundantes del sistema de percepción y guiado y, por otro, la capacidad de aprendizaje del software encargado de interpretar la información adquirida del entorno, que evoluciona hacia niveles mayores de fiabilidad. Sin embargo, en situaciones en las que cambia la configuración nominal del sistema general del vehículo, ya sea por fallo de alguno de sus componentes o por actualizaciones software o hardware intencionadas, el sistema en su conjunto deberá ser capaz de adaptarse de la mejor forma posible sin comprometer el desempeño eficiente del guiado autónomo y la seguridad de los ocupantes del vehículo.

En este ámbito se enmarca el proyecto DINAMOS (*Dependable adaptIve mechaNisms for AutonoMous and cOnnected vehicleS*, o en español, Mecanismos de adaptación confiable para vehículos autónomos y conectados), promovido por el Grupo de Tolerancia a Fallos de la UPV, y con periodo de ejecución 2017 - 2020. Expresado de forma breve, bajo este proyecto se pretenden “diseñar e implementar estrategias adaptativas de tolerancia a fallos basándose en hardware reconfigurable, permitiendo a los diseños cambiar las capacidades en función de las necesidades sin tener que reprogramar todo el sistema cada vez”. Para iniciar el proyecto, sin embargo, se requiere disponer de una plataforma de pruebas que simule en la medida de lo posible las capacidades de un vehículo autónomo. Es en la consecución de esta plataforma donde se centra este trabajo, en el que se diseñará un primer prototipo funcional que dará pie al comienzo de las primeras pruebas del proyecto DINAMOS.

Antes de continuar es importante señalar que, la parte de tolerancia a fallos (TaF) desarrollada en este trabajo es independiente de la TaF adaptativa que se pretende diseñar en el proyecto DINAMOS. En este trabajo simplemente se dotará de redundancia al sistema de guiado para fines meramente funcionales, mientras que en el proyecto DINAMOS se pondrán en práctica otras técnicas que podrán involucrar cualquier otro subsistema del robot, y tendrán como finalidad la propia evaluación de dichas técnicas y el impacto sobre las capacidades del vehículo.

Los ensayos en el ámbito de los vehículos autónomos se han llevado realizando durante años tanto en laboratorios como en escenarios reales. La mayoría de empresas o centros de investigación emplean coches comerciales adaptados o diseñados expresamente para la consecución de estas pruebas. En el caso de este proyecto, sin embargo, se parte de un estado inicial en que se dispone de recursos limitados que hacen necesario el uso de alternativas más asequibles económicamente para dar comienzo a las primeras pruebas. Debido al ámbito de aplicación específico y al carácter abierto en las especificaciones técnicas requeridas para la primera plataforma de pruebas, en este trabajo se llega a una solución que no presenta antecedentes similares identificables. Esto, sin embargo, no significa de ningún modo que el trabajo suponga un sistema o solución innovadora. Más bien implica que, haciendo uso de técnicas de distintas disciplinas, tanto estudiadas en el grado como adicionales, propone un resultado particular que cumple con el objetivo definido.

1.2.2 MOTIVACIÓN Y JUSTIFICACIÓN

Puesto que este trabajo se desarrolla como un Trabajo Final de Grado para la titulación de Ingeniería en Tecnologías Industriales de la Escuela Técnica Superior de Ingeniería Industrial de la UPV, su carácter será académico y la principal motivación para la realización de este será la obtención del título del grado.

En cuanto a la motivación personal del alumno, cabe destacar el gran interés mostrado por los sistemas electrónicos y de control y, en particular, por las técnicas que dotan a los distintos sistemas (en este caso un robot móvil) de capacidades más avanzadas de percepción y procesamiento de información, como es el caso presente de la visión por computador. Además, dada la previa experiencia con el lenguaje de programación Python, el uso de la librería de procesamiento de imágenes OpenCV (con interfaz en este lenguaje) ha resultado ser un incentivo adicional para decantarse por este trabajo. Por último, y no menos importante, la posibilidad de enfocar el trabajo realizado a un caso práctico, en que se presta un servicio a un grupo de investigación, ha sido el motivo decisivo para escoger el presente trabajo.

La labor llevada a cabo en este trabajo se justifica atendiendo al interés que surge de dos ámbitos distintos: el académico y el industrial.

En cuanto al interés académico, se han podido poner en práctica distintos conocimientos adquiridos en diversas asignaturas a lo largo del grado, así como también se han adquirido y consolidado otros nuevos. Para obtener estos últimos ha sido necesaria, además, la investigación y puesta al día por parte del alumno mediante la consulta de múltiples fuentes:

- Para el acceso a los **actuadores y sensores** del robot móvil desde la interfaz del programa ha resultado de gran utilidad disponer de las nociones de programación a bajo nivel aprendidas en Informática Industrial, Tecnología Electrónica y Programación de Sistemas Embebidos en C.
- Durante el **diseño del controlador** encargado del guiado del vehículo, ha sido esencial contar con el conocimiento y las habilidades adquiridas en las asignaturas Sistemas Automáticos y Tecnología Automática y en el Laboratorio de Automatización y Control.
- Atendiendo al **desarrollo del programa**, aunque ha sido necesario contar con experiencia previa en el lenguaje de programación Python, se han ampliado los conocimientos sobre este. Para ello ha resultado de gran ayuda la destreza obtenida en las asignaturas Desarrollo de Aplicaciones para Dispositivos Móviles y Programación de Sistemas Embebidos en C, ya que se han aprovechado las similitudes con Python que presentan los distintos lenguajes impartidos en estas (Java y C, respectivamente).
- En el ámbito del **procesamiento de imágenes** se ha requerido el aprendizaje de nuevas técnicas y la familiarización con las librerías *OpenCV* y *NumPy*. Sin embargo, han resultado fundamentales los conocimientos proporcionados por el seminario de filtrado de señales del Laboratorio de Automatización y Control y por Matemáticas III para ciertas operaciones matemáticas con imágenes, así como también las nociones de operaciones matriciales y álgebra lineal aprendidas en las asignaturas de Métodos Matemáticos y Matemáticas II. La experiencia previa con MATLAB ha facilitado el uso de la librería *NumPy* dadas las similitudes que presentan. También ha sido relevante la intuición aportada por Física I sobre los momentos para el desarrollo de ciertos algoritmos.

Respecto al interés industrial, los proyectos de investigación y desarrollo son los precursores de avances tecnológicos que potencialmente acaban siendo implementados en una gran variedad de industrias que hacen uso de estos. En este caso, el proyecto DINAMOS tiene un vínculo directo con la industria del automóvil y, aunque el presente trabajo realizado no se involucra directamente con la tecnología a desarrollar en dicho proyecto, dado el servicio que se ha prestado al grupo de investigación, se justifica la relación con este ámbito industrial. Por otro lado, las distintas técnicas empleadas y los diseños desarrollados tienen paralelismos de uso y aplicación en actividades de una gran variedad de industrias. Así, por ejemplo, el diseño de controladores PID es ampliamente usado en procesos industriales, así como la visión por computador en inspección visual automatizada de componentes, entre otros casos.

1.3 ESTRUCTURA DEL TRABAJO

Con lo expuesto, la memoria se estructurará de modo que represente en la mayor medida posible el orden en que se ha llevado a cabo el trabajo, teniendo en cuenta la secuencia lógica de las tareas realizadas.

Se dedicarán los dos primeros capítulos a definir y diseñar el punto de partida de la plataforma de pruebas.

Así, en el **capítulo 2** se presentarán los principales elementos que componen el sistema de pruebas, tanto por parte del vehículo (hardware y software) como del entorno.

En el **capítulo 3**, se detallarán los pasos seguidos para la preparación y el diseño inicial del hardware y del entorno, así como la instalación del software necesario y la realización de las primeras pruebas de funcionamiento y ajuste de los componentes del robot.

Con esto, el sistema estará listo para desarrollar los programas encargados del guiado autónomo del vehículo y de la percepción de los elementos del entorno. Puesto que todos estos programas se basarán en la información obtenida con la cámara, se dedicará un capítulo previo para exponer las funciones de OpenCV principales que se usarán en el resto del trabajo. Este constituirá el **capítulo 4**.

Tras esto, en el **capítulo 5** se definirá el controlador que permitirá guiar el vehículo por la pista y que resultará fundamental para poder llevar a cabo correctamente las distintas pruebas.

Finalmente, los dos últimos capítulos ocuparán el diseño y la mejora del sistema perceptivo que capacitará al robot móvil para hacer frente a los elementos singulares que presentará el circuito.

En el **capítulo 6** se desarrollarán los distintos algoritmos destinados a reconocer el entorno. Asimismo, se establecerán los tres primeros demostradores, cada uno agregando nuevas funcionalidades al anterior.

Por último, el **capítulo 7** se destinará al diseño redundante del sistema de interpretación del entorno planteado en el capítulo anterior, identificando en primer lugar las tareas que admiten este tipo de diseño, y se establecerá con esto el último demostrador.

CAPÍTULO 2

MATERIALES Y MÉTODOS

Una vez definidos los objetivos y conocido el alcance, se pasa a describir los elementos que componen el sistema de pruebas. Estos se pueden agrupar en tres ámbitos distintos, según su naturaleza y relación con el resto de elemento. Por una parte, se encuentra el hardware o soporte físico del vehículo, que incluye tanto la estructura mecánica como los componentes electrónicos. Por otra parte, se identifica el software que permitirá procesar la información adquirida y controlar el vehículo. Por último, se halla el entorno en que se desenvolverá el vehículo a la hora de hacer los distintos ensayos y pruebas.

2.1 HARDWARE DEL VEHÍCULO

Se parte de un kit de un robot móvil vendido por la empresa de tecnología *SunFounder*, dedicada al desarrollo de productos en código abierto. Este kit incluye la propia estructura del vehículo y las partes electrónicas que permiten el control y la conexión a una unidad de procesamiento, en el presente caso, a la Raspberry Pi. Los componentes del kit ofrecen una funcionalidad básica de control de la dirección y velocidad del vehículo. Sin embargo, por las necesidades del proyecto, se incluirán dos módulos de ultrasonidos que resultarán de ayuda en una gran variedad de pruebas. Además, aunque el kit dispone de una cámara web, se determina en primera instancia que esta no ofrece las prestaciones adecuadas para desarrollar correctamente un programa basado en el análisis de imágenes. Por ello se decide sustituir la cámara web por un módulo de cámara para la Raspberry Pi.

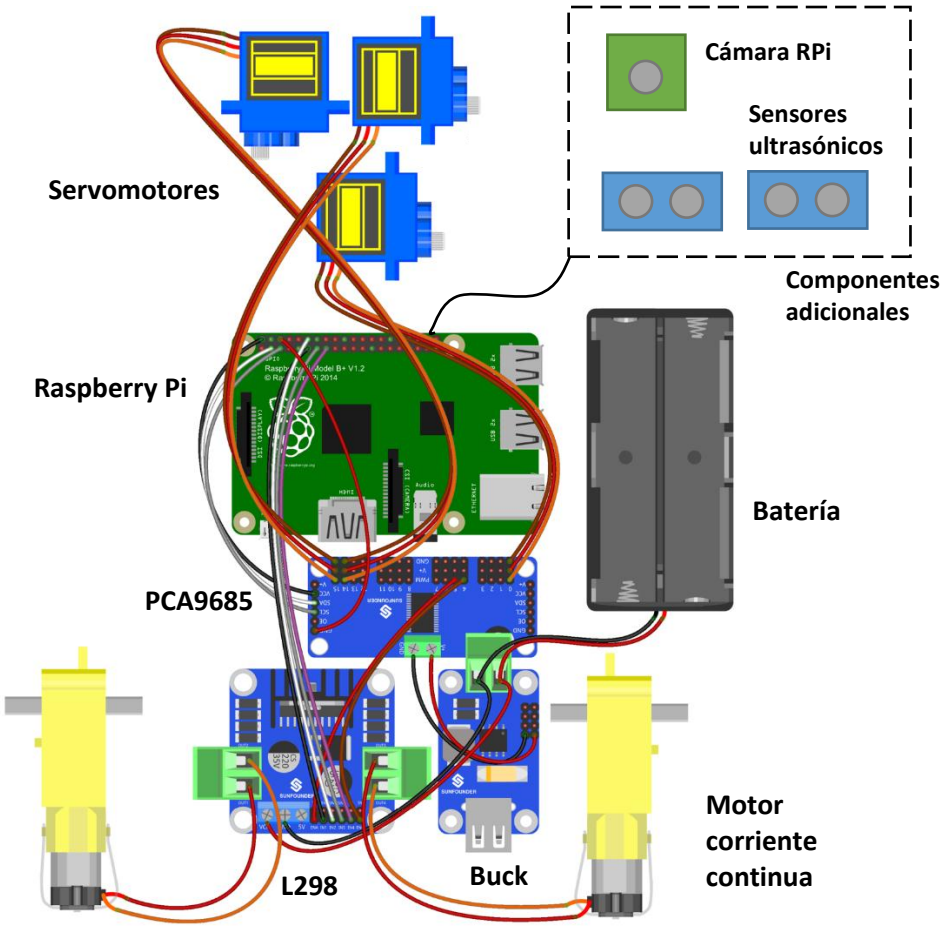


Figura 2.1. Esquema original del sistema electrónico del robot (SunFounder, s.f.).

A continuación se definirán las principales partes electrónicas que compondrán la versión final de pruebas del vehículo. Esto servirá para comprender el funcionamiento del conjunto del sistema y permitirá acceder al control de bajo nivel de cada componente si fuera necesario. Finalmente se establecerá la relación que existe entre los distintos componentes.

- Sistema de **alimentación** de **energía**: se dispondrá de dos baterías de ion de litio, con 3,7 V y 2600 mAh cada una. La tensión generada (7,4 V) será aprovechada de forma directa por los motores de corriente continua y, para el resto de componentes, se dispondrá de un reductor de tensión Buck que proporcionará 5 V a la salida.
- Unidad de **procesamiento**: se hará uso de la Raspberry Pi que, como se ha indicado anteriormente, es un computador de placa reducida. Como principales características técnicas se pueden destacar el procesador de 900 MHz y la memoria RAM de 1Gb. Debido a la variedad de periféricos de que dispone, resulta de gran utilidad para fines muy diversos. Dispone también de un conjunto de puertos de conexión con clavijas individuales destinadas tanto a entradas y salidas generales (General Purpose Input Output o GPIO) como a distintos protocolos de conexión como el I^2C, que será el usado para el control de la placa PWM. Estos puertos también incluirán salidas de tensión de 5 y 3,3 V y conexiones de tierra, lo cual

resultará útil a la hora de conectar distintos componentes como en este caso los sensores de ultrasonidos.

- Unidad de **almacenamiento**: se empleará una tarjeta SD donde se instalará el sistema operativo. Su capacidad será de 16 GB, lo que será suficiente teniendo en cuenta que OpenCV ocupará unos 5 GB.
- Unidad de **control** de **posición** de servomotores y de **velocidad** de motores de corriente continua: se dispondrá del módulo PCA9685, un versátil generador de 16 señales de onda de ancho de pulso modulado (PWM) que intercambia información mediante el protocolo I^2C. Conectándose con la Raspberry Pi, recibe órdenes para alterar el estado de los servomotores. Esencialmente, dispone de una serie de registros cuyo estado (modificado mediante la información recibida en el paquete de datos de la comunicación I^2C) determina los parámetros de la onda que se generará en cada una de las 16 salidas de la placa. Una de las ventajas de esta placa es que permite controlar cada salida de forma individual y generar una onda con frecuencia de 24 a 1526 Hz, aunque para el control de los servos bastará con unos 50 Hz. Además, disponiendo de una resolución de 12 bits en el contador del reloj interno, se puede ajustar el ciclo de trabajo de la onda PWM con una precisión de 10 a 0.15 µs, según las frecuencias mencionadas anteriormente.
- Unidad de **alimentación** y **control** de los motores de corriente continua: el circuito integrado L298 es un puente H doble que permitirá el control de los dos motores de corriente continua, modificando el sentido y la velocidad de giro de estos. La tensión de alimentación que será transmitida a los motores será la proporcionada directamente por las baterías. Este circuito dispone de dos entradas de 5 V que permiten habilitar cada puente H de forma independiente. En este caso el control de la velocidad se efectuará generando una onda PWM desde la placa PCA9685, que alternará la habilitación de cada puente. El sentido de giro de los motores se llevará a cabo invirtiendo la polaridad de la alimentación al motor cambiando el estado de los interruptores del puente H directamente desde los puertos de salida GPIO de la Raspberry Pi.
- Los **motores** de corriente continua incluirán un engranaje reductor de velocidad. Operan a tensiones de entre 3 y 12 voltios. En este caso, se proporcionarán los 7,4 voltios de las baterías.
- Los **servomotores** que se usarán serán los SG90. Estos operan a una tensión de 5 V, y la velocidad de operación nominal será de alrededor de 1,7 revoluciones por segundo. Dos de los servomotores se usarán para ajustar la posición de la montura de la cámara, y otro para controlar la dirección del vehículo.
- Los **sensores de distancia** ultrasónicos HC-SR04 tienen un rango de medida frontal de entre 2 cm y 4 m, con una precisión de hasta 3 mm. El ángulo de medida se extiende hasta 15° en cada lado del eje de simetría. La tensión de alimentación de estos será de 5 V. Su principio de funcionamiento se basa en la emisión de una señal acústica de 40 kHz por parte del emisor ultrasónico que es captada por el receptor tras un retardo de tiempo. Según la duración de dicho retardo, conociendo la velocidad del sonido en el aire se puede calcular fácilmente la distancia del objeto.

- La **cámara** usada será el módulo de cámara de la Raspberry Pi (Cámara RPi en Figura 2.1) presenta un alto grado de configurabilidad, contando con una librería escrita en Python, *picamera*, que proporciona una interfaz para acceder a las distintas funciones que la cámara ofrece. Tiene una resolución máxima de 3280×2464 píxeles, y permite la captura de vídeo con hasta 90 cuadros por segundo (con resolución de 640×480 píxeles). Con un campo de visión horizontal de 62,2 grados, resultará de gran utilidad para aplicaciones de seguimiento y reconocimiento de objetos en movimiento, como es el caso del presente trabajo.
- **Divisor de tensión**: se usarán resistencias de 1000 Ω y una regleta para conectarlas. Estas servirán para reducir el voltaje de entrada a la Raspberry Pi de aquellos periféricos que operen con 5 V, puesto que esta tan solo admite entradas de 3,3 V.

Considerando todas las partes, la relación entre estas se resume en el siguiente párrafo.

La batería alimenta directamente tanto los motores de corriente continua que proporcionarán tracción al vehículo como un convertidor Buck. El convertidor Buck reduce la tensión de 7,4 a 5 V, que es la tensión de alimentación de la Raspberry Pi y del módulo PCA9685. Este módulo recibe información de la Raspberry Pi y la procesa para controlar la velocidad de los motores de corriente continua y la posición de los tres servos presentes en el vehículo. Uno de ellos se destina al control de la dirección del vehículo y los dos restantes al posicionamiento angular de la cámara. La Raspberry Pi tendrá además una conexión directa (a través de los puertos GPIO) con el controlador L298 de los motores de corriente continua, para alterar el sentido de giro de estos. Finalmente, el módulo de la cámara y los dos sensores de ultrasonidos se conectarán también de forma directa a la Raspberry Pi. Estos últimos operarán con tensión de 5 V proporcionados por la Raspberry Pi, y requerirán por tanto el divisor de tensión para transmitir la señal de vuelta a esta.

2.2 SOFTWARE

Se parte de la librería proporcionada por *SunFounder* para el control del vehículo. Esencialmente proporciona una interfaz en Python para el acceso a los distintos componentes electrónicos del vehículo mediante el uso de las librerías *RPi.GPIO* (para los puertos de entrada y salida GPIO) y *smbus* (para la comunicación I2C) de Python. Esta interfaz incluye el control manual de la velocidad y dirección de los motores de corriente continua y del ángulo de los servomotores. Será de especial utilidad comprender el funcionamiento de las distintas partes de la librería para adaptarla a las necesidades del presente trabajo.

A esto se añaden las librerías necesarias para el procesamiento de imágenes que dotarán de un control autónomo al vehículo. Estas son *OpenCV* y *NumPy*.

- **OpenCV**: Es una librería de funciones de procesamiento de imágenes y visión artificial. Está publicada bajo la licencia BSD, que permite el uso libre tanto en el ámbito académico como en el comercial, sujeto a las condiciones que se especifican en esta. La librería fue desarrollada inicialmente por Intel, soportada posteriormente por el laboratorio de investigación de robótica *Willow Garage*, y mantenida actualmente por *Itseez*, una empresa de visión artificial con base en Rusia que fue adquirida en mayo de 2016 por Intel. OpenCV cuenta con una amplia

base de usuarios y es empleada en el sector industrial por empresas como Google, IBM, Honda o Toyota.

El núcleo de la librería está escrito en C++, y dispone de interfaces para C++, C, Java, Python y MATLAB. En este caso se hará uso de la interfaz para Python, que requiere de la disposición de NumPy para poder acceder a las matrices con las que trabaja OpenCV.

- **NumPy**: Es una librería de Python que permite trabajar con vectores y matrices multidimensionales de un modo similar a otros entornos como MATLAB.

Además, se hará uso de MATLAB para obtener gráficos de los datos obtenidos con OpenCV y NumPy.

También se hará uso de sistema de escritorio compartido VNC (Virtual Network Computing), que servirá para acceder de forma remota a la Raspberry Pi desde un ordenador personal. Las particularidades de la conexión se detallarán en el próximo capítulo.

2.3 ENTORNO DEL VEHÍCULO

Para poder realizar las pruebas en los distintos demostradores es esencial disponer de un entorno que presente una cierta cantidad de elementos singulares –dentro de las limitaciones– con los que poder simular las distintas facetas de un vehículo autónomo. Como requisitos básicos, la pista de pruebas deberá disponer de cruces en el trayecto y de señales de tráfico que se colocarán en las inmediaciones de estos. También se podrán añadir objetos al azar en distintas partes del circuito. Tanto las señales como los objetos deberán ser movibles y estarán sujetas a cambios de posición en el circuito.

Para el trazado del circuito se utilizará cinta adhesiva blanca, con tal de ofrecer contraste respecto al color oscuro del suelo y facilitar así el reconocimiento de la pista. Para las señales se usarán pequeños soportes de cartón sobre los que se adherirán los recortes de las señales impresas en papel.

Cabe destacar que el suelo en el que se situará la pista posee propiedades reflectantes al tratarse de una superficie plana pulida. Esto influirá en la elección de las técnicas de visión para el guiado y el reconocimiento del entorno.

CAPÍTULO 3

PLATAFORMA DE TRABAJO. DISEÑO INICIAL Y CALIBRACIÓN.

Habiendo definido ya todos los elementos que componen el sistema, se detallarán los pasos que permitirán, por un lado, establecer la infraestructura básica que servirá de base para el desarrollo del trabajo y, por otro, la verificación del correcto funcionamiento de los distintos agentes que tendrán un papel activo en el sistema, como son la visión, los sensores de ultrasonidos y los motores.

3.1 PASOS PRELIMINARES

Se empieza por el montaje del vehículo, comprendiendo la parte estructural y la electrónica. Se siguen las instrucciones proporcionadas por el vendedor (SunFounder, 2016). Esta tarea ha sido facilitada en su mayor parte por los profesores del laboratorio.

Al diseño de partida se le añaden los dos componentes adicionales presentados en el capítulo anterior. Estos son los dos módulos ultrasónicos y el módulo de cámara de la Raspberry Pi.

- Los sensores ultrasónicos se sitúan en la parte frontal inferior del vehículo. A la hora de conectarlos con la Raspberry Pi, se deberá tener en cuenta que esta tan solo admite entradas en los puertos GPIO con valores de tensión de nivel alto de 3,3 V. Puesto que el sensor de ultrasonidos opera con tensión de 5 V, será necesario el uso de un divisor de tensión. Se empleará una regleta para cada sensor a la que se conectarán tres resistencias en serie de 1000 Ω. Un extremo de las resistencias se conectará a tierra (compartida por el sensor y la Raspberry Pi), y el otro a la salida del sensor. El punto que proporcionará el voltaje deseado se conectará a la entrada GPIO de la Raspberry Pi. Se tomará cualquier combinación de puertos GPIO libres de la Raspberry Pi para conectar los terminales de *Trigger* y *Echo*. En concreto para este caso se eligen respectivamente los puertos GPIO23 y GPIO24 para el sensor de la derecha y GPIO5 y GPIO6 para el de la izquierda.

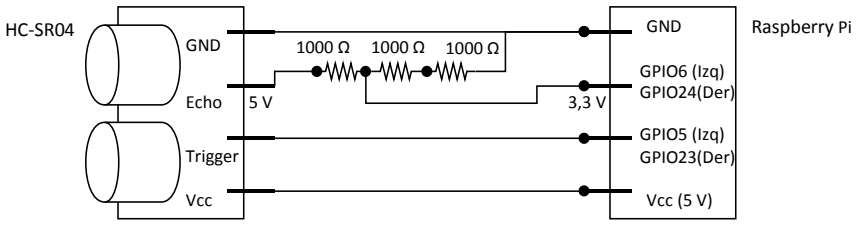


Figura 3.1. Conexión sensores ultrasónicos – Raspberry Pi

- La cámara se colocará en el soporte frontal del vehículo unido a los dos servomotores que determinan su posicionamiento. El ángulo se determinará más adelante, según las necesidades y limitaciones que se valorarán en el diseño del guiado del vehículo.

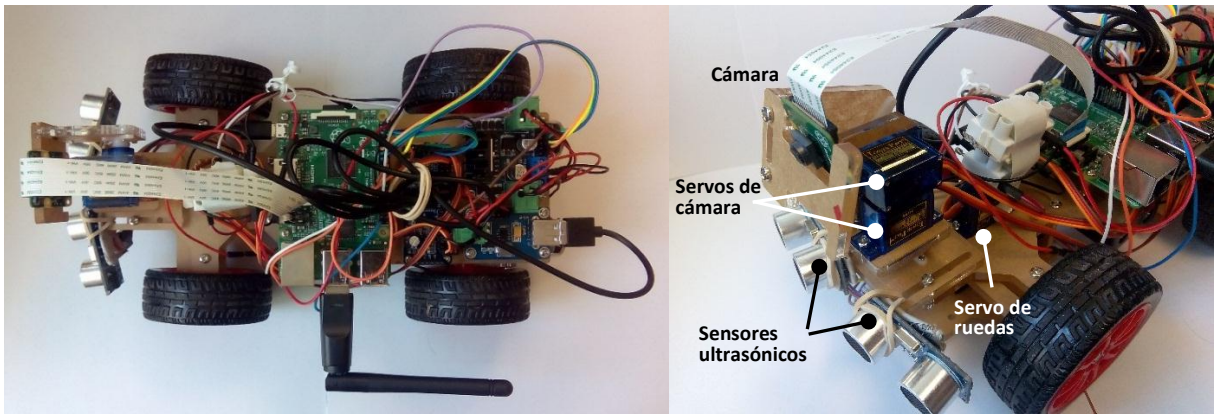


Figura 3.2. Disposición de los sensores ultrasónicos y la cámara

En cuanto al software, se empezará con la instalación del sistema operativo en la Raspberry Pi, seguido de las librerías requeridas para la visión y el control.

- El sistema operativo a utilizar será el distribuido oficialmente por la "Fundación Raspberry Pi", este es, *Raspbian*. Se seguirán las instrucciones indicadas en la web oficial, que incluirán la carga del archivo instalador "NOOBS" en la tarjeta SD desde un ordenador personal y la posterior instalación tras insertar la tarjeta en la Raspberry Pi.
- El siguiente paso será adquirir OpenCV y todas las librerías que este requiere para el correcto funcionamiento en la Raspberry Pi con interfaz en Python. Para ello se sigue una guía (Rosebrock, 2015) disponible en la web dedicada a temas de visión por computador, (Rosebrock, s.f.), que cubre todos los pasos necesarios, incluida la instalación de la librería NumPy y de la versión 3.4 de Python. Además, también se usará un módulo de Python (Rosebrock, 2016) escrito por el autor de la misma web cuyo propósito es la obtención de imágenes a través de la cámara de la Raspberry Pi usando un hilo de ejecución paralelo. Esto permitirá desacoplar la dependencia (asociada a la ejecución en serie) que el programa principal tendría si tuviera que esperar a que la cámara capturara una imagen. De este otro modo, el hilo principal ejecutará tan solo las instrucciones necesarias para tratar las imágenes y llevar a cabo el control del vehículo, y al inicio de cada iteración tan solo deberá tomar la última imagen guardada en el hilo de captura de imágenes. Con esto se logrará reducir el tiempo de ejecución del hilo principal, lo que mejorará las prestaciones del guiado del vehículo.
- Finalmente se cargarán los módulos de Python que permitirán la comunicación con la placa PCA9685 y el control de los actuadores. El código está disponible en la página web GitHub (SunFounder, 2016) del vendedor del kit del vehículo. Estos archivos fueron escritos en la versión de Python 2.7. Para pasarlos a la versión 3.4 se usa el conversor *2to3* incluido en las librerías oficiales de Python.

En referencia a lo anterior, puesto que *SunFounder* vende el vehículo ofreciendo la capacidad de ser controlado manualmente a distancia, el programa que proporciona la interfaz de control se basa en el protocolo TCP (Transmission Control Protocol), mediante el cual se establece un servidor (Raspberry Pi) y un cliente (ordenador personal) que comparten una dirección IP y se comunican mediante la transmisión de paquetes de datos. En este caso, el ordenador personal ejecuta un programa en Python que proporciona una interfaz gráfica y que envía comandos a la Raspberry Pi mediante dicho protocolo. En esta, se lanza un programa que recibe las órdenes y las procesa para comunicarse con los módulos de control de los actuadores del vehículo.

En lo que atañe al presente proyecto, se opta por disponer de la capacidad de acceder remotamente a la interfaz del propio sistema operativo, para así poder editar y ejecutar desde un ordenador personal los programas deseados de la Raspberry Pi. Por ello, se decide eliminar la funcionalidad de conexión vía TCP, ya que presenta el inconveniente de que la comunicación entre cliente y servidor se limita exclusivamente al ámbito de los programas que hacen uso del protocolo. Así pues, se modifican los programas proporcionados por *SunFounder* para unificar en un solo archivo de Python la interfaz gráfica y el acceso directo a los módulos de control de los actuadores.

Para garantizar la comunicación remota con el vehículo se usará el sistema de escritorio compartido VNC que, dados dos computadores conectados en una misma red, proporciona una interfaz gráfica para acceder al escritorio de trabajo de uno de los ordenadores (servidor) desde el otro (cliente), e interactuar con el entorno mediante un teclado y un ratón. En el caso presente se usará una red local sin acceso a internet, haciendo uso de un enrutador proporcionado por el Laboratorio de Tolerancia a Fallos. La Raspberry Pi –que se conectará a la red mediante conexión WiFi– ejercerá el papel de servidor, cediendo así el acceso a su escritorio al ordenador desde el que se operará –que se conectará directamente al enrutador mediante cable Ethernet.

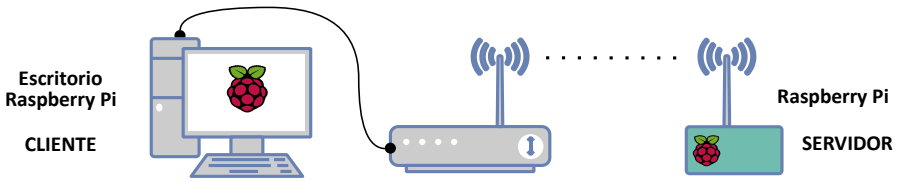


Figura 3.3. Conexión remota VNC

La última parte en la preparación preliminar del sistema de pruebas será el diseño de la pista o circuito que recorrerá el vehículo y de las señales de tráfico que deberá reconocer.

Para el soporte de las señales de tráfico se recorta una base de cartón que presente un mínimo de anchura de unos 2 cm para facilitar su detección con los sensores ultrasónicos.

Para el trazado del circuito se usará un área de unos cinco metros cuadrados, situada en el suelo del Laboratorio de Tolerancia a Fallos. Según el criterio del diseñador, el circuito dispondrá de tres cruces perpendiculares para ofrecer variedad en el entorno visual donde se deberán reconocer señales de tráfico, y de un tramo recto y largo que servirá para efectuar posibles ensayos de estabilidad de seguimiento básico de trayectoria. Destacar que, en cualquier momento, ciertas partes del circuito

podrán ser modificadas recolocando la cinta adhesiva, si alguna prueba precisase de una configuración distinta. En la Figura 3.3 se muestra la disposición del circuito final.

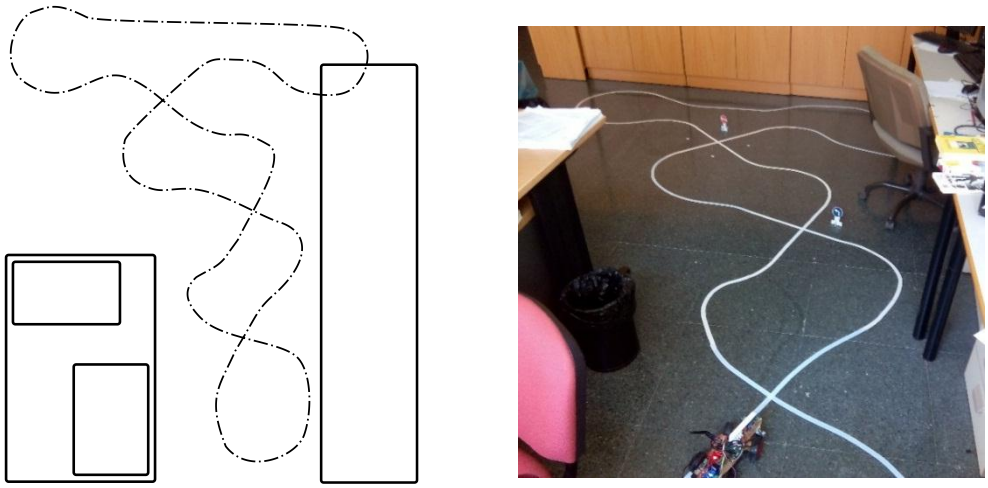


Figura 3.4. Trazado del circuito de pruebas.

3.2 PRUEBA DE FUNCIONAMIENTO Y CALIBRADO

Tras lo expuesto en el apartado anterior, la plataforma de pruebas se encuentra en disposición para verificar el correcto funcionamiento de todos los subsistemas y realizar el calibrado y ajuste de aquellas partes que lo requieran. A continuación, se detallarán los distintos puntos que se han tenido en cuenta para conseguir tales fines.

En primer lugar, se lleva a cabo la comprobación del funcionamiento de los motores y servomotores y el calibrado de estos últimos. Las librerías de Python proporcionadas por *SunFounder* cuentan con un programa destinado al calibrado de los servomotores que –tras las modificaciones mencionadas en el apartado anterior–, proporciona una interfaz para el control manual de los actuadores del vehículo. Este programa guarda en un fichero de texto la posición de partida de los servomotores que el usuario considere oportuna. Posteriormente, cada vez que se inicie un programa que haga uso de los módulos de control de los actuadores, los servomotores serán dirigidos a la posición a la que han sido ajustados. Con esto, para el calibrado del servo encargado de cambiar la dirección del vehículo, se ajusta el valor de su posición hasta que el vehículo consigue describir una línea recta. Los otros dos servos se ajustan para que la ranura del soporte donde se colocará la cámara quede enfrente del vehículo y alineado con su eje principal de simetría.

Resultará de utilidad en el Capítulo 5 saber que la interfaz de acceso a los servomotores acepta valores enteros positivos de 8 bits (*uint8*, de 0 a 255) para determinar la posición de estos. Para el caso particular del servomotor encargado de dirigir las ruedas delanteras, un valor de 0 asignará la posición máxima de giro a la izquierda, mientras que un valor de 255 se corresponderá con el máximo giro a la derecha. Del mismo modo, para el valor de 127, las ruedas tomarán su posición intermedia, que dirigirán el vehículo siguiendo una línea recta.

En segundo lugar, se comprueba el funcionamiento de la cámara y de OpenCV. Para ello se escribe un programa simple que captura imágenes con la cámara de la Raspberry Pi (usando la librería mencionada anteriormente que permite la obtención de imágenes en un hilo paralelo) y las muestra por pantalla con la función básica de OpenCV, `cv2.imshow(a,b)`. En esta función, 'a' es una variable tipo cadena de caracteres o *string* que guarda el nombre de la ventana donde se mostrará la imagen, y 'b' es la variable que almacena la imagen actual en forma matricial.

Por último, se efectúa la comprobación de la correcta captura de señal y el ajuste del ángulo de los ultrasonidos. El programa que se usará, escrito en Python, se encuentra disponible en la web (HC-SR04 Raspberri Pi, s.f.) (ver Capítulo 9). Se ejecuta y se comprueba que se obtienen medidas con errores inferiores a $1\ \mathrm{cm}$ para la mayor parte de los objetos usados en los ensayos. Se comprueba que, si un objeto plano se encuentra formando un ángulo lo suficientemente grande con el eje de simetría del sensor, este último tiende a perder la señal que indica la presencia del objeto, marcando distancias demasiado elevadas. Esto se debe a que la onda acústica emitida es reflejada en una dirección que no puede ser captada por el sensor. Este dato resultará útil en el momento de efectuar las pruebas de detección de objetos puesto que determinará el modo en que se deberán colocar si presentan superficies planas.

El programa original del sensor de ultrasonidos se modifica por un lado para que su ejecución se lleve a cabo en un hilo en paralelo al del programa principal, haciendo uso del módulo *Threading* de Python, y por otro lado para que soporte la medida de la distancia con dos ultrasonidos independientes. Con el fin de evitar interferencias, se introduce un desfase en la captura de medidas de modo que, para un periodo dado, al inicio de este opere un sensor y a mitad el otro. Una vez activos los dos sensores, se ensaya con distintos periodos de muestreo y se encuentra que, para periodos menores de 0,1 segundos, el uso de procesamiento de la Raspberry Pi escala rápidamente. Así pues, se determina que un periodo de 0,2 segundos proporciona una tasa de refresco de información adecuada con relación al consumo de CPU.

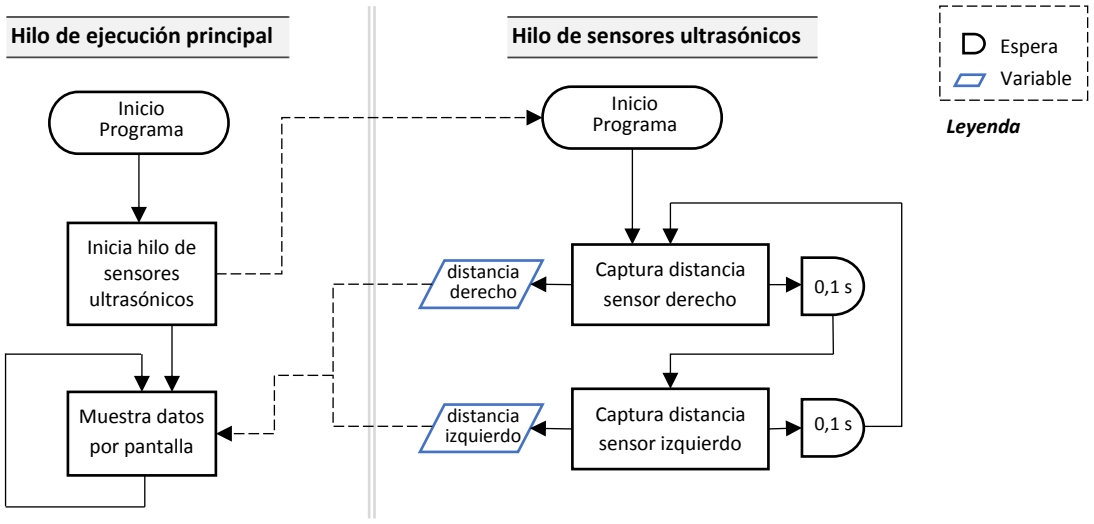


Figura 3.5. Diagrama de ejecución del hilo paralelo de obtención de distancias mediante los dos sensores ultrasónicos

Los sensores de ultrasonidos deberán ser capaces de detectar objetos laterales individualmente y, además, cualquiera de ellos deberá poder captar la presencia de un objeto en frente del vehículo. Dado el ángulo de percepción de aproximadamente 30° que estos presentan, se colocan de forma que las bandas se solapen parcialmente en el eje central, tratando al mismo tiempo de maximizar el barrido lateral. En la figura siguiente se muestra la configuración final que estos adoptarán, tras una serie de ensayos en los que se han posicionado objetos de tamaños variados a múltiples distancias y ángulos. En concreto se ha comprobado también la correcta detección de las señales de tráfico presentes en los laterales de la vía del circuito.

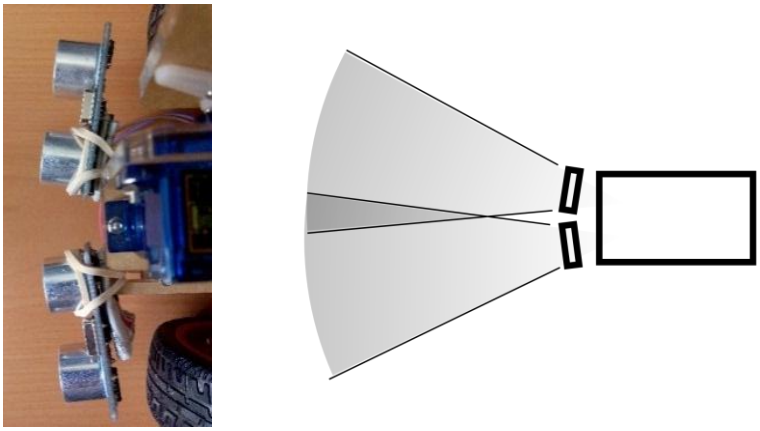


Figura 3.6. Disposición angular de los sensores ultrasónicos.

Una vez diseñada la base de partida del sistema de pruebas y comprobado el correcto funcionamiento de las partes que la integran, en los siguientes capítulos se procederá con el diseño de los sistemas de control y percepción, no sin antes definir las funciones básicas de OpenCV que se usarán más ampliamente durante el trabajo.

CAPÍTULO 4

VISIÓN CON OPENCV. FUNDAMENTOS Y ENTORNO DE PROGRAMACIÓN.

La librería OpenCV cuenta con una gran cantidad de algoritmos, que abarcan desde simples operaciones con imágenes como el cálculo de gradientes o momentos, hasta técnicas más avanzadas como la detección de características o el aprendizaje automático. En este trabajo se hará uso de una pequeña porción de ellos, la mayoría de ellos básicos. A continuación se describirán los que han resultado de mayor utilidad y se han usado ampliamente a lo largo del trabajo.

4.1 INTRODUCCIÓN A LA VISIÓN POR COMPUTADOR

Para poder tratar con imágenes con un computador, estas deben ser almacenadas en un formato cuantificable y discretizado en el espacio. Es por ello que en las imágenes en formato digital se habla de píxeles como unidades mínimas de representación. OpenCV tratará con matrices que guardarán en cada posición el valor de la intensidad de un píxel. Esta forma de almacenarlas presentará grandes ventajas a la hora de efectuar operaciones algebraicas sobre ellas para extraer información útil.

Las propiedades dimensionales de las imágenes digitales serán el tamaño y el número de canales.

- El tamaño vendrá dado por dos valores, que indicarán la cantidad de píxeles en el eje horizontal y en el vertical, respectivamente. Para las imágenes en OpenCV, el origen de coordenadas se situará en la parte superior izquierda, y para hacer referencia a una posición o píxel, se indicará en primer lugar la coordenada horizontal y en segundo la vertical.
- El número de canales variará según el tipo de imagen. Para las imágenes en color habrá tres canales, mientras que para las binarias (blanco y negro) o en escala de grises, solo se dispondrá de un canal.

En la mayor parte de los casos se trabajará con imágenes con tipo de datos *uint8*, esto es, valores enteros positivos, de 0 a 255. No obstante, como se verá, habrá algunos casos en los que se usará el tipo *float32* (valores reales desde $-3{,}4 \cdot 10^{38}$ hasta $+3{,}4 \cdot 10^{38}$) para disponer de un mayor número de valores y por tanto de una mayor resolución en el valor de la intensidad del píxel.

Generalmente las funciones de OpenCV operarán con una imagen origen o de entrada y retornarán otra imagen o valor que se asignará a una variable de salida o destino. Sin embargo, habrá también algunas funciones que usarán una sola imagen de entrada que será modificada y no retornarán ninguna salida.

```
salida = cv2.funciónEntradSalida(entrada, otrosParámetros)
cv2.funciónSinSalida(entradaParaModificar, otrosParámetros)
```

4.2 ALGORITMOS BÁSICOS

- **Cambio de color**

Las imágenes capturadas por la cámara se almacenan en formato RGB (Red Green Blue). Esto significa que, para su representación, se usan tres canales (o matrices) superpuestos, que guardan la intensidad de cada uno de los tres colores para cada pixel, que puede tomar valores entre 0 y 255. Ahora bien, este no es el único modo con el que se puede representar una imagen. Existen otras formas de organizar los colores e intensidades con las que se consigue cubrir el mismo rango de representación. Los distintos formatos son los llamados "espacios de colores". Ya que OpenCV permite la conversión entre varios de estos, se hará uso de los formatos de representación en coordenadas cilíndricas HSV y HSL, que permiten tratar con colores de un modo más intuitivo. En efecto, estos formatos cuantifican el valor de la tonalidad del color (*hue* en inglés) en coordenadas angulares (tomando valores de 0 a 180 en OpenCV para cubrir el espectro visible), lo cual permite tratar porciones independientes sin tener que buscar combinaciones de rojo, verde y azul. Las demás coordenadas se usan para representar valores de saturación y brillo, y toman valores de 0 a 255. También se hará uso del formato de escala de grises, que solo dispone de un canal en el que guarda la intensidad de cada píxel. Este resultará útil a la hora de tratar con formas y patrones espaciales en la imagen, y será usado por funciones que tratan con un solo canal.

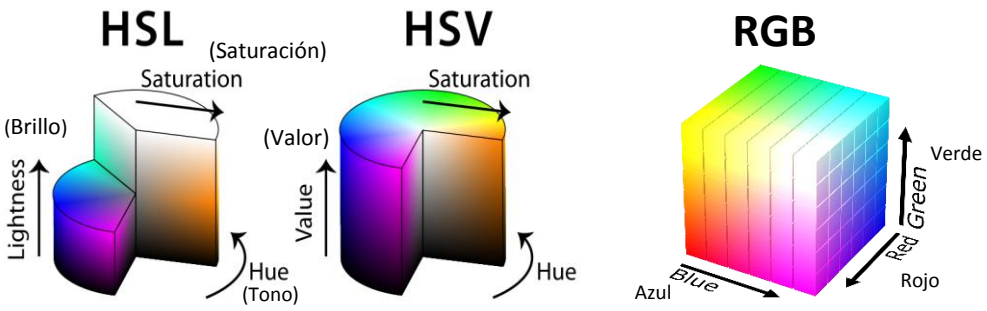


Figura 4.1. Espacios de colores

La función de OpenCV que permite cambiar entre espacios de colores es:

```
salida = cv2.cvtColor(entrada, cv2.ETIQUETA)
```

Donde `ETIQUETA` indica el tipo de conversión que se quiere efectuar. Las etiquetas que se usarán son las siguientes:

`COLOR_BGR2GRAY`	para cambiar de RGB a escala de grises,
`COLOR_BGR2HSV`	para cambiar de RGB a HSV,
`COLOR_BGR2HLS`	para cambiar de RGB a HSL.

- **Convolución de imágenes**

Dada una imagen en formato escala de grises $I(x, y) \in \mathbb{R}^2 \rightarrow \mathbb{R}$ (o de forma más general, dado un canal de una imagen) y dada una función (a veces referida como filtro) también en el dominio espacial bidimensional $F(x, y) \in \mathbb{R}^2 \rightarrow \mathbb{R}$ (con tamaño menor que el de la imagen), existen una serie de operaciones llamadas de convolución $I * F \in \mathbb{R}^2 \rightarrow \mathbb{R}^2$, resultantes de aplicar la función sobre la imagen, que producen como resultado una tercera imagen.

Las operaciones de convolución más comunes y ampliamente usadas en este trabajo serán la de suavizado, empleando una función gaussiana, y la de extracción de bordes, mediante el uso del operador de Sobel, basado en el cálculo de derivadas espaciales. La primera permite eliminar ruido de alta frecuencia, como pueden ser rayas o puntos no deseados en la imagen, mientras que la segunda permitirá un análisis posterior más profundo de las características extraídas, como la orientación de los gradientes o la presencia de ciertos patrones.

En sintaxis de OpenCV, podremos efectuar el suavizado de la imagen mediante la función:

```
salida = cv2.GaussianBlur(entrada, (tamañoX, tamañoY), 0)
```

Donde *tamañoX* y *tamañoY* son variables que indican el tamaño en píxeles del filtro o función a emplear, y el valor 0 en este caso indica que la desviación típica de la función gaussiana se ajustará al tamaño del filtro. Si se reemplaza el valor 0 por uno mayor, se tomará tal valor para la varianza de la función.

Para los operadores de Sobel se usará la función:

```
salida = cv2.Sobel(entrada, cv2.TIPODATO, dx, dy, ksize=3)
```

Donde *TIPODATO* indica el tipo de datos de la imagen de salida, *dx* y *dy* son las componentes del vector director que se usará para calcular la derivada, y el tamaño del filtro viene dado por el parámetro *ksize*. La etiqueta *TIPODATO* generalmente se establecerá a CV_32F o CV_64F, que indica que los valores en la salida se almacenarán en coma flotante (valor real), lo cual aportará un mayor grado de precisión. Las variables *dx* y *dy* se usarán con las combinaciones $(dx, dy) = (1,0)$ para la derivada de Sobel a lo largo del eje horizontal de la imagen y $(dx, dy) = (0,1)$ para la derivada en el eje vertical.

- **Binarización de imágenes**

Un formato que resultará especialmente útil para representar la presencia de ciertos patrones en la imagen es el binario. Las imágenes de este tipo almacenan tan solo valores binarios (cada píxel tomando el valor máximo –blanco– o mínimo –negro– de intensidad) en una sola matriz. Para obtener una imagen binaria se pueden emplear múltiples funciones de OpenCV, según el tipo de imagen inicial que se quiera transformar. En este caso se mostrarán los dos métodos que serán usados más frecuentemente.

Si se dispone de una imagen en escala de grises, la conversión de cada píxel a blanco o negro se efectuará mediante la elección de un valor umbral comprendido entre el máximo y el mínimo de intensidad representada por la imagen. Los píxeles que superen dicho valor se representarán con color blanco y los que queden por debajo con negro. La función de OpenCV es la siguiente, donde *valorMax* es el máximo valor de intensidad que se puede representar con el formato de imagen de entrada (255 para enteros de 8 bits), y *umbral* se situará por tanto entre 0 y este valor.

```
valorNoUsadoEnEsteCaso,salida =
        cv2.threshold(entrada,umbral,valorMax,cv2.THRESH_BINARY)
```

Disponiendo de una imagen en color, se puede obtener una imagen binaria tomando como blancos aquellos píxeles que estén dentro de un rango de valores, según el espacio de color que se use. Como se ha comentado anteriormente, los formatos HSV y HLS facilitan la discriminación de colores, por ello se deberá convertir las imágenes RGB obtenidas en la cámara a uno de estos dos formatos antes de proceder con la operación de binarización.

```
rangoInferior = numpy.array([inferior1,inferior2,inferior3])
rangoSuperior = numpy.array([superior1,superior2,superior3])

salida = cv2.inRange(entrada, rangoInferior, rangoSuperior)
```

El rango será definido por dos vectores, *rangoInferior* y *rangoSuperior*. Estos contienen tres elementos que especifican los valores límites del rango para cada uno de los tres canales de la imagen RGB, HSV o HLS.

- **Contornos**

Una operación exclusiva para imágenes binarias es la extracción de contornos. Esta consiste en el análisis de las formas o agrupaciones de valores binarios que se presentan en la imagen para determinar los puntos que componen la frontera de estas. Una vez obtenidas, se pueden aproximar a distintos perfiles poligonales para caracterizarlas y analizar sus propiedades con mayor facilidad. Esta función de OpenCV también presenta la capacidad de distinguir contornos encerrados dentro de otros y establecer jerarquías de pertenencia. Cabe mencionar que los contornos se obtienen en una lista que guarda las coordenadas de estos en la imagen.

```
salida,contornos,jerarquia =
cv2.findContours(entrada,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
```

Las etiquetas que se adoptarán serán `RETR_TREE` y `CHAIN_APPROX_SIMPLE`. La primera indicará el modo en que se devuelve la jerarquía (que no será usada en este trabajo), y la segunda el modo de aproximación de los contornos.

- **Obtención de momentos**

Puesto que cada canal de una imagen se trata en esencia de una distribución espacial de valores (intensidades), se puede aplicar el cálculo de momentos para extraer propiedades útiles. El cálculo de estos se podrá efectuar para cualquier rango de intensidades que presenten los píxeles, pero mayoritariamente se aplicará a imágenes binarias o a contornos, en los que ya ha sido extraído algún elemento característico como puede ser la forma de algún objeto. Los principales momentos que se usarán serán el área (momento de orden cero) y los centroides de las formas a partir de momentos de primer orden.

Para la obtención de estos se usará la siguiente función, que toma una imagen de entrada y devuelve una variable de Python tipo diccionario con los momentos calculados.

```
momentos = cv2.moments(entrada)
```

Para acceder a los momentos se usarán distintas claves que especificarán a qué momento del diccionario se quiere acceder. A modo de ejemplo se muestra cómo se obtiene el área y las coordenadas *X* e *Y* del centroide de la imagen procesada.

```
area = momentos['m00']
centroideX = momentos['m10']/area
centroideY = momentos['m01']/area
```

- **Template matching y operador Canny**

Estas dos funciones serán cruciales para poder desempeñar tareas más avanzadas.

El "template matching" o comparación de modelos se basa en la operación de convolución. En este caso, en lugar de usar una función matemática como filtro, se toma directamente como modelo una imagen de tamaño menor que el de la imagen sobre la que se efectuará la convolución. Si el modelo se encuentra en la imagen con tamaño y orientación muy similares, el resultado de la convolución presentará un valor elevado en el punto donde se encuentra dicho modelo. Por tanto, esta técnica permitirá localizar un modelo en una imagen con bastante precisión, siempre y cuando no varíe demasiado el tamaño y la orientación con respecto al usado en el modelo.

```
salida = cv2.matchTemplate(entrada, modelo, cv2.TM_CCOEFF)
```

El operador Canny se aplica a imágenes de un solo canal y tiene como objetivo hallar los bordes representativos de la imagen, mediante el ajuste de unos parámetros. Esta técnica se compone de varios pasos. En primer lugar, se efectúa un suavizado de la imagen con una convolución gaussiana de tamaño de filtro 5x5. A continuación, se detectan bordes mediante el operador de Sobel en dirección tanto horizontal como vertical, y se calcula la dirección e intensidad del gradiente para cada píxel. Después de esto se lleva a cabo la toma de los valores máximos locales de los gradientes, para obtener bordes "finos". Finalmente se aplica un filtrado de histéresis que mediante dos valores umbral decide qué bordes mantener y cuáles eliminar. El umbral superior indica a partir de qué valor de gradiente el píxel se considera incondicionalmente un borde, mientras que el umbral inferior rechaza como borde cualquier píxel cuyo valor de gradiente esté por debajo. Si el valor del gradiente se encuentra entre el máximo y el mínimo, el píxel se considerará borde si y solo si está conectado espacialmente a un borde que presente algún pixel con valor de gradiente mayor que el umbral superior.

El resultado de esta operación es una imagen binaria con los bordes detectados mostrados en blanco.

```
salida = cv2.Canny(entrada, umbralInferior, umbralSuperior)
```

CAPÍTULO 5

GUIADO BÁSICO. DISEÑO DEL CONTROLADOR

El primer requisito que debe cumplir el robot móvil para poder desarrollar todas las pruebas que se van a llevar a cabo en el trabajo es disponer de la capacidad de recorrer de forma autónoma el circuito diseñado. En particular, para el alcance del proyecto, deberá ser capaz de seguir una trayectoria cualquiera siempre que la curvatura de esta no supere el radio de giro máximo del vehículo. El robot obtendrá la información correspondiente al trazado de la pista únicamente a través de la cámara.

Así, el propósito de este capítulo será exponer los pasos que han permitido obtener el controlador que efectuará el guiado del vehículo. Para ello se evaluarán en primer lugar las alternativas posibles, para a continuación seleccionar y analizar aquella que mejor se adapte al ámbito del proyecto y finalmente establecer el diseño óptimo en base a la opción escogida.

5.1 ANÁLISIS DE ALTERNATIVAS

Dada la gran versatilidad que ofrece OpenCV en el procesamiento de imágenes, se presentan varias alternativas a la hora de diseñar el control del vehículo. Tanto es así, que se podría incluso implementar un control basado en algoritmos probabilísticos como las redes neuronales artificiales con el módulo de aprendizaje automático de OpenCV, para simular en la medida de lo posible las capacidades de un vehículo autónomo comercial y aprovechar las ventajas que ello ofrece. Sin embargo, esta opción se descarta rápidamente debido a la falta de capacidad de procesamiento de la Raspberry Pi para estas técnicas y a la elevada complejidad que supondría la puesta en marcha del sistema.

Por tanto, se debe optar por métodos más tradicionales de control. La opción ideal para un buen seguimiento de la trayectoria del circuito consistiría en un control cinemático en que se conocieran en cada momento los parámetros cinemáticos (posición lineal y angular, velocidad) que localizan el robot en el circuito. Sin embargo, el vehículo de pruebas no dispone de los sensores adecuados para conocer esta información, como serían tacómetros o giróscopos, entre otros. Aunque se podrían estimar ciertos parámetros posicionales a partir del procesamiento de la imagen de la pista captada por la cámara, estos datos serían poco fiables y susceptibles de incluir errores, debido a la falta de precisión en la medida óptica y al presente ruido en la imagen como son los reflejos u objetos del suelo.

Figura 5.1. Imagen de la pista con reflejos

Por ello, se decide efectuar un control tipo PID –Proporcional Integrador Derivativo– (Astrom, 2009), (Salcedo, 2016) siguiendo una referencia de fácil medida y robusta frente al ruido, como es la distancia lateral de la pista en frente del vehículo, captada en la banda inferior de la imagen, Figura 5.2. Efectivamente, esta señal, pudiéndose cuantizar con un solo dato (la desviación respecto del centro de la imagen), presenta menos errores de precisión en la medida. Además, situándose en la banda inferior de la imagen, se consiguen eliminar la mayor parte de los reflejos que, dada la naturaleza del entorno de pruebas, se presentan mayoritariamente para ángulos menores del eje normal a la cámara respecto del suelo.

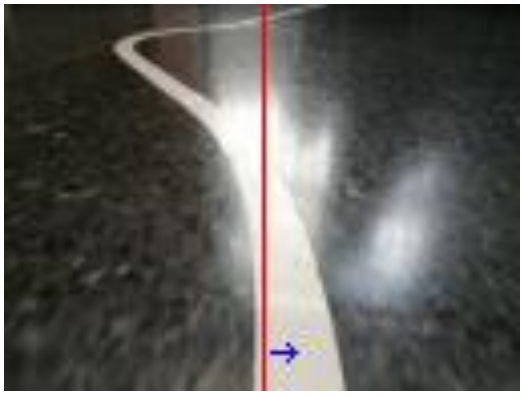


Figura 5.2. Imagen de la pista, desplazamiento lateral respecto al centro

De esta última idea se hace evidente la necesidad de establecer un ángulo adecuado de la cámara que consiga satisfacer dos requisitos con demandas opuestas. Por un lado, como se ha dicho, para ángulos mayores (con el máximo en 90°), se elimina una mayor cantidad de reflejos del suelo captados por la imagen. Por otro, con ángulos menores es posible tener una mayor distancia de visión y así obtener una mayor cantidad de información de los elementos presentes enfrente del vehículo. Esto resultará de gran utilidad a la hora de detectar elementos singulares como cruces u objetos. Tras varios ensayos posicionando el vehículo en distintos puntos del circuito y observando los reflejos presentes en la imagen, se determina que el ángulo que llega a un mejor compromiso entre los dos requisitos es el de aproximadamente 34°, como se muestra en la Figura 5.3.

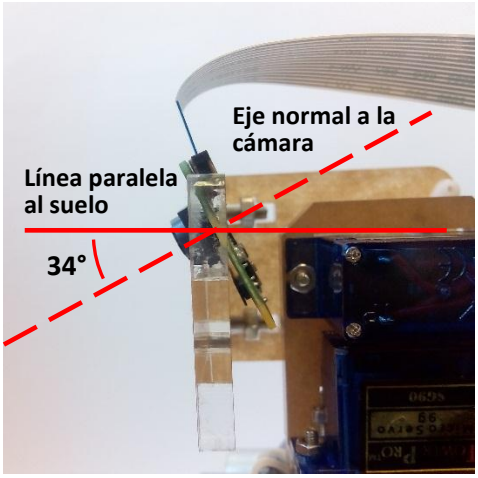


Figura 5.3. Disposición angular de la cámara

Con todo esto, se procederá a continuación con el análisis del sistema de control seleccionado (en concreto, el proceso que determina la relación de transformación entre una acción de control ejercida y la señal medida, así como también el sistema en bucle cerrado en su conjunto) para posteriormente diseñar el controlador.

5.2 PROCESO Y BUCLE CERRADO. SISTEMA GENERAL

El guiado del vehículo se rige por dos variables sobre las que se puede ejercer control: la velocidad (a través de los motores de corriente continua) y el radio de giro (a través del servomotor conectado a las ruedas). Dada una velocidad cualquiera no nula, la acción de control que permitirá el seguimiento del trayecto del circuito será el radio de giro del vehículo. Esta acción será por tanto la que ocupará el siguiente caso de estudio, en que se analizará el proceso mediante el cual evoluciona la variable medida (coordenada horizontal de la pista en la banda inferior de la imagen, sobre la cual se pretende ejercer un control manteniéndola en el centro de la imagen) en función de una acción de control en el radio de giro y del propio estado cinemático actual del vehículo. Se asumirá pues, que el control que se ejercerá sobre la velocidad del vehículo será de valor constante en el tiempo, esto es, se deseará que el vehículo mantenga una velocidad fija.

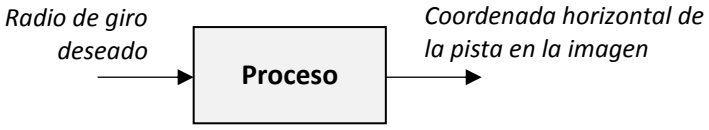


Figura 5.4. Proceso general

De un análisis de las fases que intervienen en el proceso se pueden diferenciar tres etapas principales, como se muestra en la figura Figura 5.5.

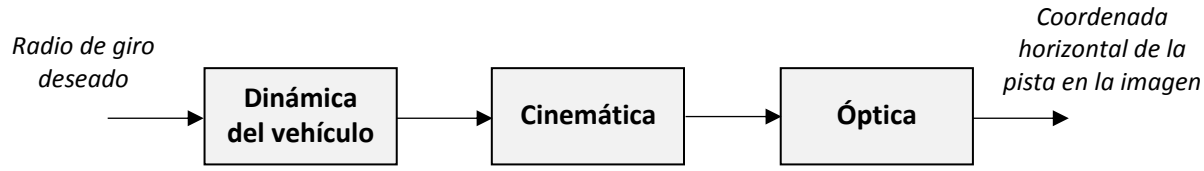


Figura 5.5. Proceso desglosado

Antes de analizar de forma independiente cada una de estas etapas en el orden en que se producen, se comentará brevemente el problema de forma inversa para mostrar de forma más clara cuál es la relación de las variables que intervienen en el proceso. Esto es, puesto que la variable de salida del proceso es la coordenada horizontal de la pista en la imagen, y dado que esta depende directamente de la distancia lateral del vehículo a la pista, d, el objetivo será comprender cómo varía esta última en función del estado –posición, velocidad y acción de control– en que se encuentre el vehículo.

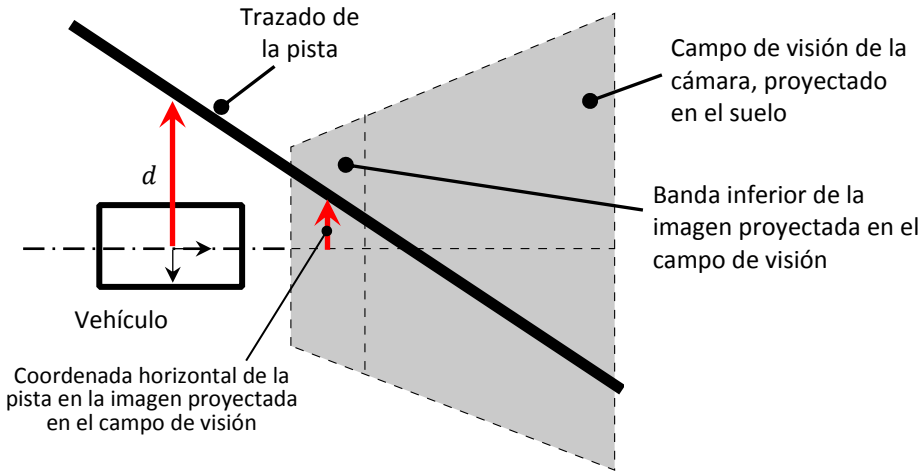


Figura 5.6. Vista superior del vehículo en la pista. Relación entre la coordenada horizontal en la banda inferior de la imagen y la distancia lateral d del vehículo a la pista.

Con esto, se pasa a evaluar cada una de las etapas del proceso que se han determinado.

En primer lugar, se encuentra la dinámica del vehículo. Dada una nueva orden en el programa de control, en este caso, el ángulo que se exige al servomotor (o lo que es lo mismo, el radio de giro del vehículo, ρ), existe un transitorio desde que esta se manda desde la unidad de procesamiento hasta que finalmente el mecanismo presenta dicho ángulo. En este transitorio influyen principalmente la fricción de las ruedas con el suelo y la inercia de estas, y en menor medida la inductancia del servomotor. La salida de este proceso será por tanto el radio de giro de las ruedas y también la velocidad v del vehículo, que se verá afectada ligeramente por la fricción del giro.

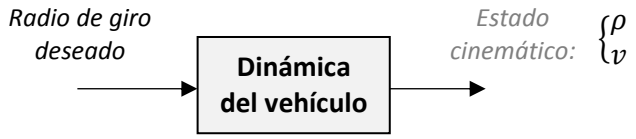


Figura 5.7. Proceso, dinámica del vehículo

En segundo lugar, se produce una transformación geométrica a causa de la cinemática del vehículo. En esta etapa del proceso, en función de la velocidad v y el radio de giro ρ del vehículo (que podrán ser modificados por la etapa anterior –dinámica del vehículo), así como también de la distancia actual d y ángulo del vehículo θ respecto de la pista (ver Figura 5.9), se producirá una variación de estos dos últimos parámetros. Esta variación dependerá en gran medida del estado anterior de las variables d y θ, como se muestra en la Figura 5.9. Por tanto, esto dotará al proceso de un carácter altamente no lineal. La variable de salida de esta etapa será la distancia lateral del vehículo a la pista, d, que será la que tras el último proceso (transformación óptica) determine la coordenada horizontal de la pista en la imagen.

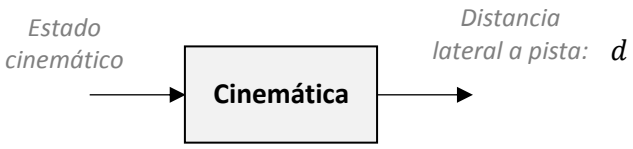


Figura 5.8. Proceso, cinemática y transformación geométrica

Cabe decir que en la Figura 5.9 se representa una acción en bucle abierto simplemente para ilustrar más claramente la no linealidad del proceso. Efectivamente, en el caso de la derecha el vehículo tomaría una acción de control opuesta a la mostrada si el control fuese en bucle cerrado.

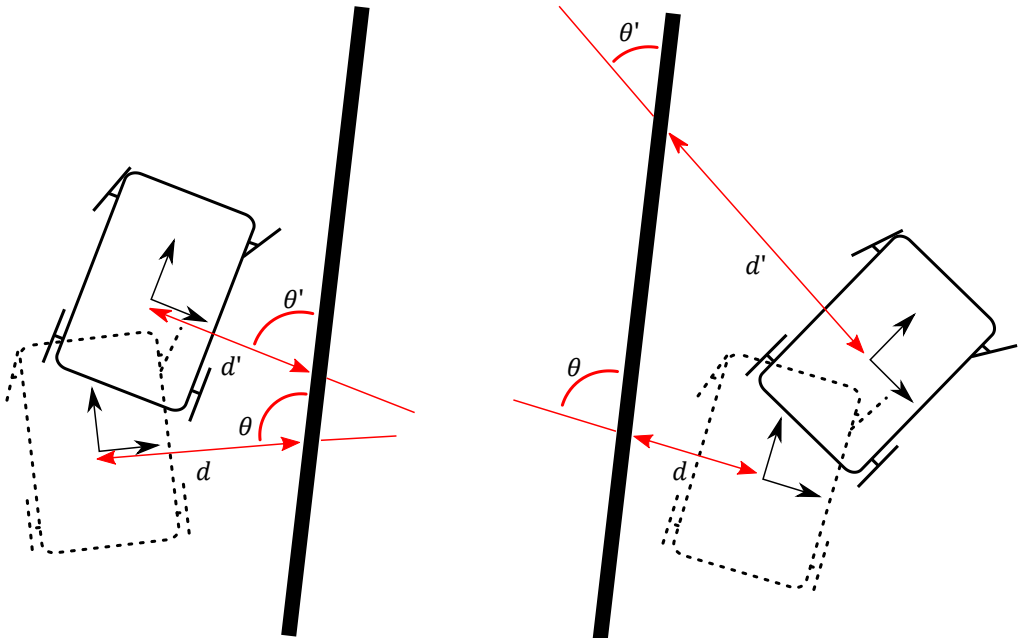


Figura 5.9. No linealidad del proceso. Para una misma acción de control, se producen variaciones distintas en d, (relacionada directamente con la variable de control) según el estado previo de los parámetros cinemáticos del vehículo.

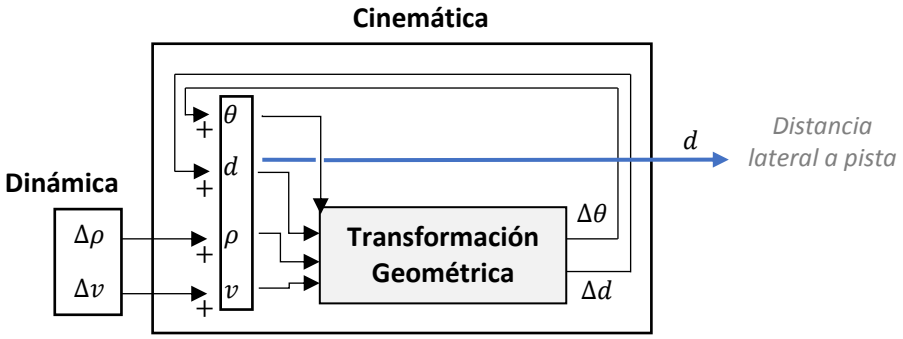


Figura 5.10. Detalle de proceso de transformación geométrica a partir del estado cinemático

Por último, tiene lugar la transformación óptica que asigna linealmente cada punto del mundo con uno de la imagen. La variable de salida de esta etapa es por tanto la coordenada en el eje horizontal de la pista en la imagen, que se usará para efectuar el control, y tendrá como unidad el píxel.

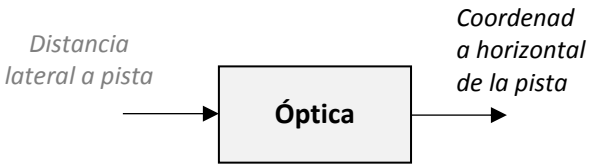


Figura 5.11. Proceso de transformación óptica

A continuación, se analiza el sistema en bucle cerrado, que será el que permitirá, dado un controlador adecuadamente ajustado, el seguimiento de una referencia mediante la realimentación negativa de la variable que se desea controlar.

Como se muestra en la Figura 5.12 el sistema se compone del controlador –discreto en este caso– y del proceso no lineal definido anteriormente. La referencia será el centro de la imagen y tendrá las mismas unidades que la señal medida, esto son, píxeles. Se cuenta con una serie de perturbaciones externas, que se corresponden con el cambio en el trayecto de la pista dado por las curvas. Estas alteran las variables que localizan el vehículo respecto de la pista, es decir, la distancia d y el ángulo θ y, por ende, también alterarán el valor de la señal medida.

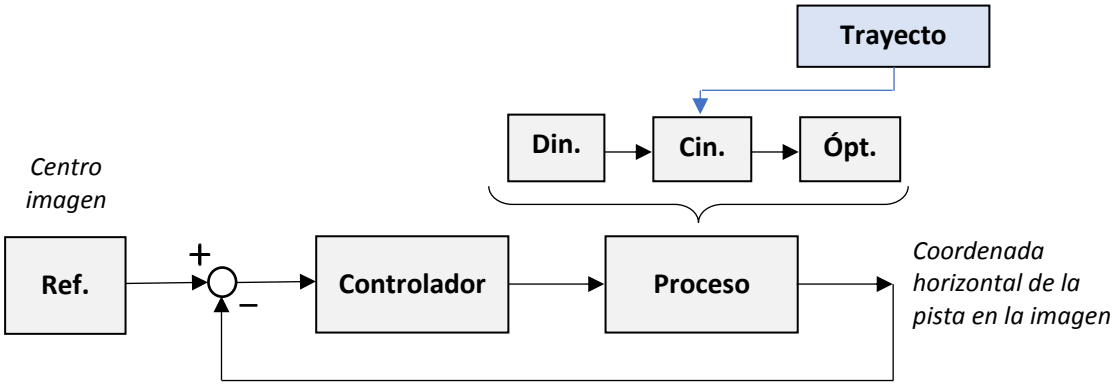


Figura 5.12. Bucle cerrado

Dada la falta de datos para modelizar la dinámica y la cinemática del vehículo y dada la complejidad que introduce la no linealidad de la transformación geométrica, se hace inviable pretender obtener un modelo linealizado fiable del proceso. Por ello será necesario recurrir a métodos empíricos de diseño y ajuste de controladores.

5.3 DISEÑO DEL CONTROLADOR

5.3.1 DISEÑO DEL PROGRAMA DE CONTROL

Antes de proceder con el diseño de las distintas partes del programa, será necesario establecer el tamaño de la imagen que se obtendrá con la cámara. El valor de este tendrá grandes implicaciones en múltiples aspectos tanto del diseño como del rendimiento del programa. En cuanto al diseño del programa, habrá ciertos parámetros, como el tamaño de los filtros usados, que serán dependientes del tamaño de la imagen. Esta circunstancia se podrá resolver en cierta medida estableciendo relaciones que ajusten el valor de los parámetros en proporción al tamaño de imagen que se establezca. En referencia al rendimiento, una mayor resolución de imagen permitirá una mejor distinción de los elementos que se quieran reconocer, como son los cruces. Pero a medida que esta aumente, el tiempo de procesamiento que requerirán las funciones para operar con las imágenes se incrementará en gran medida y el programa perderá capacidad de respuesta. Es por ello que será crucial la elección de un tamaño de imagen que llegue a un equilibrio entre precisión y tiempo de ejecución.

Dicha elección no será fácil en primera instancia ya que, el tiempo de ejecución, principal aspecto a tener en cuenta para evaluar el rendimiento, dependerá directamente de la cantidad de funciones que se hayan implementado en el programa hasta un momento dado. Puesto que se partirá de un estado del programa muy básico para el diseño del controlador, el tiempo de ejecución de este no será representativo de el del estado final. Para tratar de evitar este problema, en este trabajo se ha optado por implementar un controlador sencillo (sin seguir una metodología rigurosa) que ofreciera unas capacidades básicas de seguimiento de línea durante el desarrollo de los algoritmos de visión –que, aunque limitadas, serían suficientes para hacer pruebas en tramos específicos–, para en una fase más adelantada del trabajo efectuar un diseño más exhaustivo del controlador que caracterizara mejor el estado final del programa. Así pues, el diseño experimental del controlador que se presentará a continuación se realiza cuando ya se han desarrollado los principales algoritmos de visión y han sido implementados en el programa.

Con todo esto, se encuentra que el tamaño de la imagen que consigue balancear mejor la precisión en la visión y el tiempo de ejecución, es la de $112\text{x}80$ px. Los tiempos medidos para este tamaño se encuentran en torno a los 95 ± 15 ms. La variación se debe a la distinta carga de trabajo que pueda tener la Raspberry Pi en cada momento según los procesos activos en el sistema, y a las diferencias de contenido de las imágenes, que pueden ralentizar algunas operaciones como la detección de contornos.

Para la obtención de la variable sobre la que se efectuará el control (el centro de la pista en la banda inferior de la imagen), se consideran dos alternativas posibles. Ambas tendrán en común el cálculo de momentos sobre una imagen binaria para la obtención de la coordenada horizontal del centro de la pista, dada por el centroide de la imagen. Sin embargo, diferirán en el modo en que se obtendrá dicha imagen binaria. En cualquier caso, se ha establecido que la banda inferior sobre la que se calcularán los momentos será la que se corresponde con un quinto de la altura de la imagen. Esto permite eliminar prácticamente todos los reflejos de la zona de medida manteniendo un margen para cubrir posibles defectos en la pista que puedan introducir errores en la medida. A continuación se mostrarán los resultados de ambas alternativas aplicadas sobre la imagen con reflejos presentada anteriormente.

Figura 5.13. Imagen de pista, quinto inferior sin reflejos

En la primera alternativa se evalúa la umbralización (20) de la imagen, puesto que la pista y el suelo presentan una clara diferencia en el valor de la intensidad. Para ello se convierte en primer lugar la imagen a escala de grises. A continuación, se ensaya en condiciones de iluminación óptimas y se elige el umbral de 180 (recordar que la intensidad toma valores entre 0 y 255) para binarizar la imagen con la función *threshold* de OpenCV. Este método proporciona resultados aceptables cuando las condiciones de iluminación son las adecuadas. Sin embargo, en presencia de reflejos o variaciones de intensidad de la imagen en conjunto (por ejemplo, zonas del circuito con poca luz), se hace inviable la obtención de información fiable.

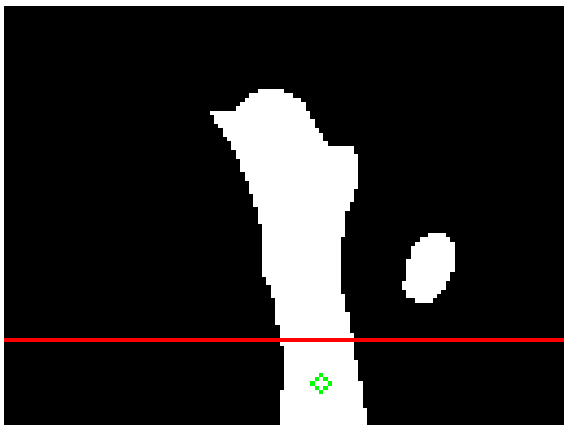


Figura 5.14. Binarización por umbralización

La segunda alternativa considera el uso de la detección de bordes ya que se presenta un gran contraste de intensidad entre la pista y el suelo, mostrando líneas de separación bien definidas. En este caso se convierte también a escala de grises para a continuación aplicar el detector de bordes Canny (22). Pese a que este método aplica una operación de suavizado previo, se comprueba que, aplicando un difuminado con un filtro gaussiano de tamaño 9 (con relación al tamaño de la imagen, un 11,25% de la altura), se consiguen eliminar mejor los reflejos manteniendo una medida fiable del borde de la pista. Los parámetros del detector Canny que producen mejores resultados son 120 para el límite inferior y 195 para el superior. Cabe decir que, frente a variaciones de iluminación del conjunto de la imagen, este método sigue proporcionando buenos resultados, ya que el contraste entre la pista y el suelo sigue siendo suficiente.

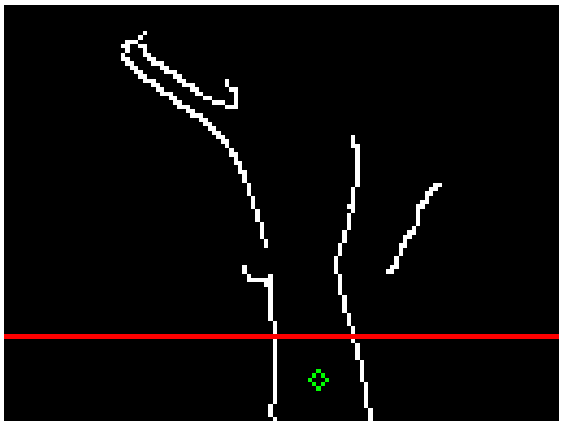


Figura 5.15. Binarización por detección de bordes

Es preciso destacar que, aunque en el diseño del método de obtención de la variable controlada solo se ha tenido en cuenta la calidad de la medida de esta, la binarización por bordes también proporcionará mejores resultados a la hora de detectar cruces, como se verá en el más adelante.

El controlador PID discreto escrito en Python se muestra más adelante. Este se sitúa tras la parte del programa encargada de procesar la imagen. Cabe decir que todos los nombres de las variables no coinciden exactamente con los presentes en el programa global, pues han sido modificados para mejorar la comprensión. El diseño mostrado es el de un controlador completo proporcional-integrador-derivativo (PID), aunque servirá también para implementar cualquier otro (P, PD, o PI) forzando a ser nulas las acciones de control pertinentes.

El periodo t se calcula al inicio de cada iteración. Se toma la diferencia entre el último instante de tiempo en que se ejecutó la acción de control, $t1$, y el instante de tiempo posterior tras haber ejecutado la parte de procesamiento de imagen, $t2$.

La referencia ref se corresponde con la coordenada horizontal que marca el centro de la imagen. En este caso, dado un ancho de imagen de 112, su valor será 56.

La variable controlada y, tomará valores que variarán dentro del ancho de la imagen. Teniendo en cuenta que el primer píxel se enumera como el 0, los valores enteros que podrá presentar esta variable estarán comprendidos entre 0 y 111 (112 valores contando en total)

El error se calcula restando la referencia (el centro de la imagen), ref, a la variable controlada, y. A continuación, dicho error se normaliza dividiendo entre la referencia, para obtener un valor decimal entre -1 y 1, lo cual permitirá que los parámetros de control sean independientes del tamaño de la imagen. Cabe destacar que el error se calcula con signo opuesto al método habitual, $ref-y$. Esto no supone ninguna ventaja en especial; simplemente permitirá usar ganancias positivas en los parámetros del controlador.

```
1.    # cálculo del periodo
2.    t2 = cv2.getTickCount()
3.    t = (t2-t1)/cv2.getTickFrequency()
4.
5.    # error: variable - referencia
6.    e = (y-ref)/ref
7.
8.    # parte Proporcional
9.    uP = e*(Kp)
10.   # parte Derivativa
11.   uD = (e - e_prev)*(Td)/t
12.   # parte Integral
13.   e_i += e*t
14.   uI= (e_i)*(1/Ti)
15.
16.   # acción de control
17.   U = int(127 + uP + uD + uI)
18.
19.   # actualización de variables para la siguiente iteración
20.   e_prev = e
21.
22.   t1 = cv2.getTickCount()
```

Figura 5.16. Controlador PID en Python.

La acción integral se calcula a partir del error acumulado e_i. Para evitar la acumulación del error cuando la acción de control está saturada, se realiza un control *Anti-Windup*, donde se toma el valor del error acumulado de la iteración anterior (restando el valor que se había sumado en la actual) si la acción de control ha superado los límites superior o inferior. Recordar que estos límites son 0 para el máximo giro de las ruedas a la izquierda y 255 para la derecha, según lo expuesto en el Capítulo 3, página 15.

```
# Limitación de la acción de control y Antiwindup
if U>255:
    U=255
    if e_i>0:
        e_i -= e*t
elif U<0:
    U=0
    if e_i<0:
        e_i -= e*t
```

Figura 5.17. Control Anti-Windup

Finalmente, se efectúa la acción de control usando la función que sitúa en la posición deseada el servomotor conectado a las ruedas delanteras.

```
control_direccion_vehiculo.gira(U)
```

5.3.2 DISEÑO Y AJUSTE DEL CONTROLADOR PID

Tal y como se ha razonado anteriormente, la opción más factible dada la información medible de la que se dispone, es realizar un ajuste de controlador mediante técnicas empíricas. En este caso se opta por el método heurístico de Ziegler-Nichols (Astrom, 2009) (Salcedo, 2016) (Ziegler–Nichols method, s.f.). Este se basa en el uso de un controlador proporcional cuya ganancia se incrementa progresivamente hasta provocar una respuesta oscilatoria sostenida en el sistema. A partir de la ganancia que provoca dicha respuesta y del tiempo de las oscilaciones se obtienen los parámetros iniciales de diseño del PID. Para realizar las pruebas en el laboratorio, se emplea un tramo del circuito que describe una recta de unos 1,5 metros.

Puesto que tan solo se efectuará el control sobre el ángulo de las ruedas delanteras, se deberá establecer una velocidad fija a la que se realizarán los ensayos. Se determina que, tras cierto tiempo de operación del vehículo, la velocidad que este presenta disminuye para un mismo valor exigido en el programa, debido al agotamiento de las baterías recargables. Se estima que, por debajo de aproximadamente un 30% de la acción de control de la velocidad, el vehículo presenta una excesiva lentitud en durante los giros. Por ello, se establece una velocidad del 40% para el diseño del PID, ya que velocidades demasiado elevadas dotan de una gran inestabilidad en el control.

Efectuando el método experimental de Ziegler-Nichols (de ahora en adelante, abreviado ZN), se estima tras varios ensayos que la ganancia que provoca una respuesta sostenida oscilatoria es de 160. Dicha respuesta del sistema se muestra en la siguiente figura. A partir de los tiempos medidos entre dos picos consecutivos se encuentra un periodo de oscilación de aproximadamente 1,18 segundos.

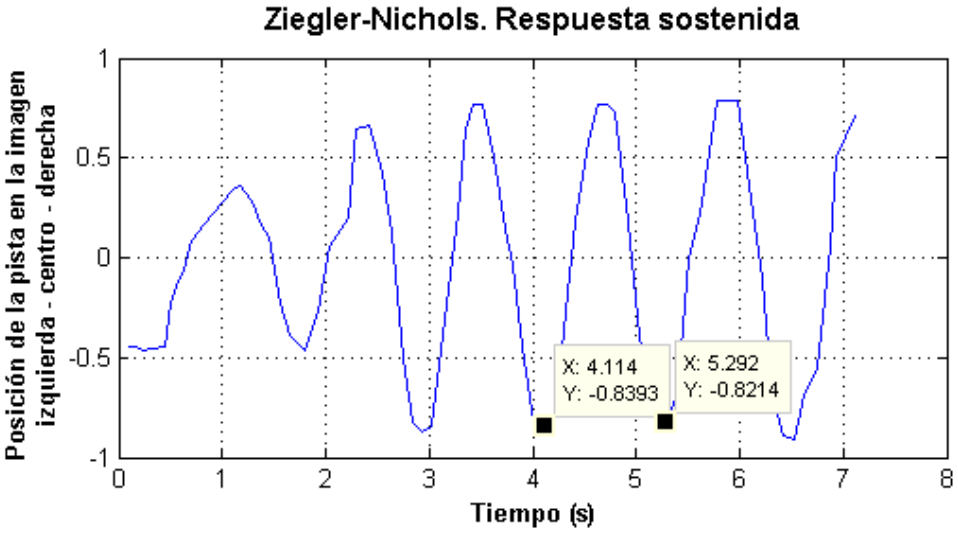


Figura 5.18. Respuesta sostenida, método ZN

Para la obtención de los parámetros del controlador PID se hará uso de la Tabla 5.1 de Ziegler-Nichols.

Tipo de Control	K_p	T_i	T_d
P	$0{,}5K_u$	-	-
PI	$0{,}45K_u$	$T_u/1{,}2$	-
PD	$0{,}8K_u$	-	$T_u/8$
PID clásico	$0{,}6K_u$	$T_u/2$	$T_u/8$
Regla Integral de Pessen	$0{,}7K_u$	$T_u/2{,}5$	$3T_u/20$
Poca Sobreoscilación	$0{,}33K_u$	$T_u/2$	$T_u/3$
Sin Sobreoscilación	$0{,}2K_u$	$T_u/2$	$T_u/3$

Tabla 5.1. Parámetros del método Ziegler-Nichols

Donde los valores de K_u y T_u serán los determinados anteriormente:

$$\begin{cases} K_u = 160 \\ T_u = 1{,}18\ \text{s} \end{cases} \quad (1)$$

Ecuación 5.1. Parámetros obtenidos del método Ziegler-Nichols

Estos parámetros se corresponden con la Ecuación 5.2. Sin embargo, el controlador discreto se implementará con ganancias independientes para cada término, para poder evaluar mejor la contribución de cada término en función del valor de su ganancia, tal y como se muestra en la Ecuación 5.3.

$$u(t) = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{de(t)}{dt} \right) \quad (2)$$

Ecuación 5.2. PID – Ziegler-Nichols

$$u(t_k) = K_p\, e(t_k) + \frac{1}{T_i}\, T \sum_{t=t_o}^{t_k} e(t) + T_d \frac{e(t_k) - e(t_{k-1})}{T} \quad (3)$$

Ecuación 5.3. PID – tiempo discreto

Con esto, los valores de los parámetros para el controlador discreto en función de los obtenidos en la tabla de Ziegler-Nichols, serán:

$$\begin{cases} K_p = (K_p)_{ZN} \\ T_d = (K_p\, T_d)_{ZN} \\ T_i = \left(\frac{T_i}{K_p} \right)_{ZN} \end{cases} \quad (4)$$

Ecuación 5.4. Relación de parámetros entre la ecuación PID de Ziegler-Nichols y la ecuación PID de tiempo discreto empleada

A continuación, se realizan una serie de ensayos para valorar y ajustar experimentalmente los parámetros de los controladores. Para este propósito se utiliza un tramo del circuito que presenta una sucesión de curvas tras las cuales se presenta una recta de 1,5 metros de longitud. Con este trayecto corto se podrán evaluar distintas prestaciones de los controladores, como la respuesta del sistema ante cambios pronunciados en la variable de control –los cambios de curvatura–, la aptitud en el seguimiento de curvas cerradas o prolongadas y finalmente, con el tramo recto, el tiempo de establecimiento y la magnitud de las sobreoscilaciones. Este último punto será critico a la hora de abordar los cruces, ya que la mayor parte de estos son precedidos por curvas.

Se optará en primer lugar por diseñar el controlador más simple, este es, el proporcional P. Analizando las limitaciones que presenta la respuesta se justificará el uso de un controlador PD. Finalmente, se justificará la necesidad de usar un controlador completo PID.

Todos los gráficos mostrarán por un lado la evolución de la posición del centro de la pista en la imagen (variable controlada, azul), y por otro la acción de control o variable manipulada, en rojo. El tramo del gráfico resaltado con trazado grueso se corresponde con el trayecto en que el vehículo recorre la recta del circuito. Destacar que, en todos los casos, como es propio del método de Ziegler-Nichols, se obtendrá una respuesta agresiva con notables sobreoscilaciones. Así pues, uno de los objetivos de estos ensayos será paliar dicho comportamiento, que generalmente se conseguirá reduciendo la ganancia del controlador.

Controlador P

Controlador proporcional obtenido con el método de Ziegler-Nichols mediante las ecuaciones Ecuación 5.1 y Ecuación 5.4:

$$P_{ZN}: \{K_p = (K_p)_{ZN} = 0{,}5K_u = 80 \quad (5)$$

Ecuación 5.5. Controlador P, método ZN

El controlador proporcional conseguido con el método de ZN no tiene la ganancia suficiente como para superar la curva larga del trayecto de pruebas. Efectivamente, con una ganancia de tan solo 80, debido al propio diseño del controlador y a la limitación de rango de medida de la imagen, este solo será capaz de proporcionar valores en la variable manipulada entre 127-80 = 57 y 127+80 = 207, respecto de los límites 0 y 255. Esto no proporciona al vehículo de un radio de giro lo suficientemente grande como para superar correctamente las curvas. Por ello, se muestra la Figura 5.19 correspondiente a un controlador proporcional con ganancia de 110. Esta es suficiente para superar la curva, pero presenta claramente un comportamiento oscilatorio no deseado, con un tiempo de establecimiento demasiado elevado. Para tratar de reducir tanto la magnitud como la duración de las oscilaciones, se opta por ensayar con el controlador proporcional-derivativo. Se considera que la parte derivativa será especialmente útil en este ámbito debido a las constantes variaciones de la señal medida dada la forma curvilínea del trayecto. Puesto que la acción derivativa actúa en oposición a las variaciones de la variable controlada, el uso de esta permitirá al vehículo adaptarse mejor a los cambios de en dicha variable inducidos por la presencia de curvas en el trayecto.

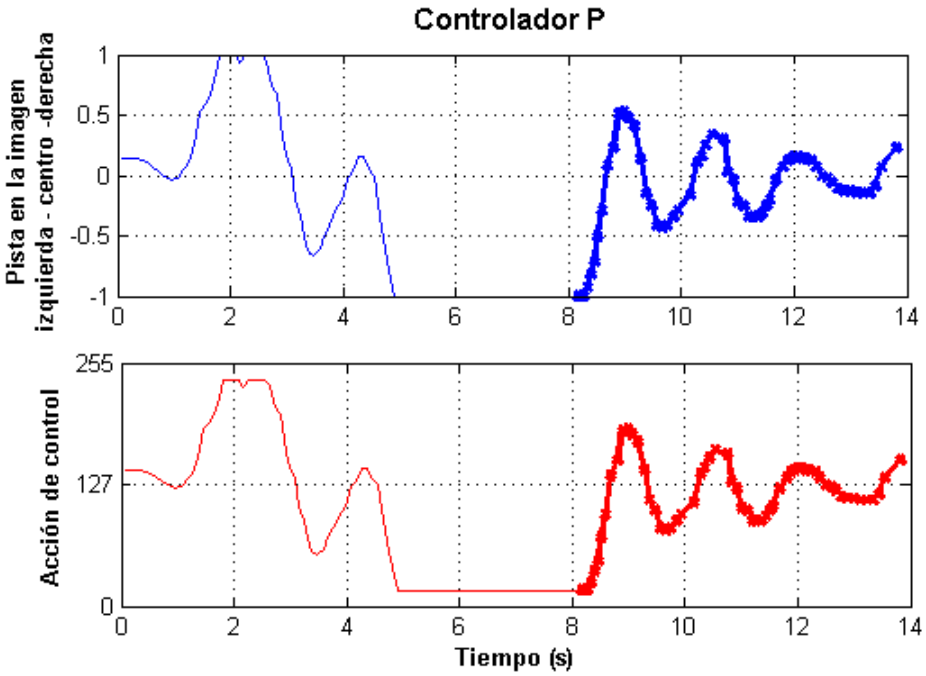


Figura 5.19. Controlador P, resultados

Controlador PD

Controlador proporcional-derivativo obtenido con el método de Ziegler-Nichols mediante las ecuaciones Ecuación 5.1 y Ecuación 5.4:

$$PD_{ZN}: \begin{cases} K_p = (K_p)_{ZN} = 0{,}8K_u = 128 \\ T_d = K_d \ = \ (K_p\, T_d)_{ZN} = 0{,}8K_u \dfrac{T_u}{8} = 18{,}88 \cong 19 \end{cases} \quad (6)$$

Ecuación 5.6. Controlador PD, método ZN

El controlador PD proporcionado por el método de ZN presenta un claro exceso de sobreoscilaciones, como se aprecia en la Figura 5.20. Con una ganancia inicial en la parte proporcional de 128, si se reduce demasiado podrá presentar problemas en el seguimiento de curvas, tal y como sucedía con un controlador proporcional. Por esta razón y por la presencia de una componente de alta frecuencia en la variable manipulada, apreciable en los primeros segundos de la señal registrada, se decide reducir en mayor medida la ganancia de la parte derivativa. Se llega finalmente a los valores mostrados a continuación, correspondientes con el gráfico de la Figura 5.21.

$$PD_{Ajustado}: \begin{cases} K_p = 110 & (-14\%) \\ T_d = K_d = 11 & (-42\%) \end{cases} \quad (7)$$

Ecuación 5.7. Controlador PD Ajustado

Como se puede observar, se ha conseguido eliminar la mayor parte de la componente de alta frecuencia en la acción de control. Sin embargo, no ha sido posible eliminar las sobreoscilaciones. Se ha reducido el tiempo de establecimiento de estas, no sin aumentar la magnitud del primer pico, correspondiente con la incorporación del vehículo a la recta. Se podría tratar de solucionar esto reduciendo la velocidad de los motores, que dotarían al vehículo de una dinámica más lenta y presentaría un comportamiento menos agresivo, con acciones de control menos acusadas. No obstante, como se ha mencionado anteriormente, esto puede presentar problemas con niveles de batería bajos, por lo que preferiblemente se desea mantener la velocidad actual. Es por ello que primero se tratará de obtener un controlador PID que consiga eliminar dicho comportamiento no deseado.

Hay que notar que se ha podido confirmar experimentalmente que, como se había indicado, el controlador PD, además de reducir el tiempo de establecimiento de las oscilaciones, dota al vehículo de una respuesta más rápida ante cambios de curvatura del trayecto, que le permiten anticipar en cierto grado la presencia de curvas y mejorar la incorporación a estas.

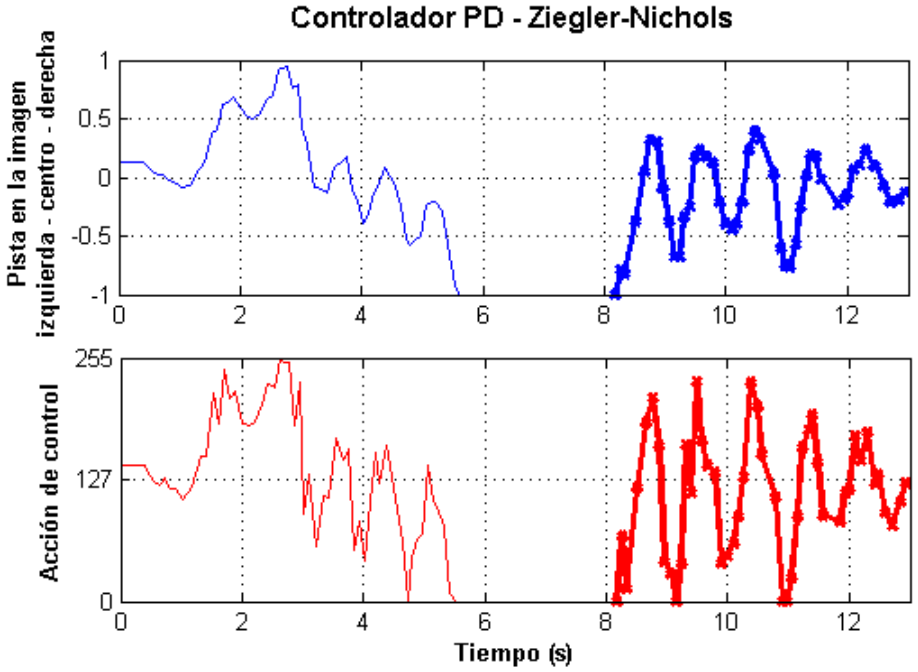


Figura 5.20. Controlador PD, método ZN

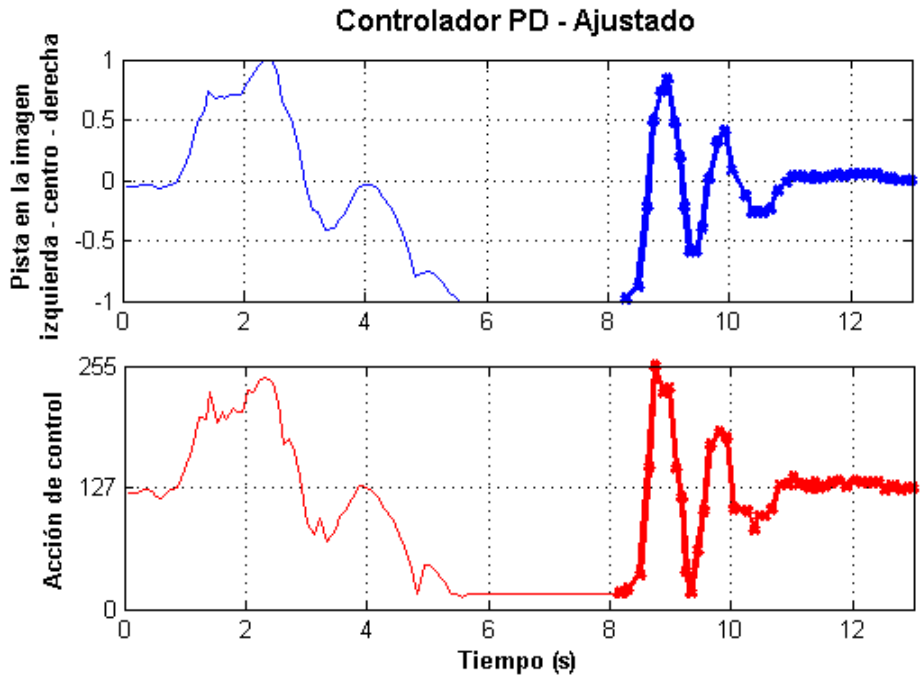


Figura 5.21. Controlador PD Ajustado

Controlador PID

Controlador proporcional-integrador-derivativo obtenido con el método de Ziegler-Nichols mediante las ecuaciones Ecuación 5.1 y Ecuación 5.4:

$$PID_{ZN}: \begin{cases} K_p = (K_p)_{ZN} = 0{,}6K_u = 96 \\ T_d = K_d = (K_p\, T_d)_{ZN} = 0{,}6K_u \dfrac{T_u}{8} = 14{,}16 \cong 14 \\ T_i = \left(\dfrac{T_i}{K_p}\right)_{ZN} = \dfrac{T_u}{2 \cdot 0{,}6K_u} = 6{,}15 \cdot 10^{-3} \cong 6 \cdot 10^{-3} \rightarrow K_i = \dfrac{1}{T_i} = 166{,}67 \end{cases} \quad (8)$$

Ecuación 5.8. Controlador PID, método ZN

Con el controlador PID obtenido por el método empírico de ZN, se consigue una respuesta que presenta notables mejoras respecto a los controladores anteriores, incluso después de haber sido ajustados. En primer lugar, se reduce significativamente la magnitud de las sobreoscilaciones, lo que se traduce en una incorporación más suave a la recta. En segundo lugar, se mejora el seguimiento de la curva prolongada (segundos 5 a 7.5 en la Figura 5.22), pues la acción integral permite ejercer un control que aprovecha toda la capacidad del actuador cuando el error se mantiene constante, sin tener que recurrir a aumentar la ganancia de la parte proporcional, que resultaba en un comportamiento del sistema más inestable.

A pesar de las mejoras introducidas, el controlador presenta aún una respuesta lejos de la ideal. Es por ello que, se tratará de ajustar los parámetros del PID para reducir en la medida de lo posible la magnitud de las sobreoscilaciones y el tiempo de establecimiento en la recta. Realizando varios ensayos, se observa que, reduciendo la ganancia de la parte integral, se consigue reducir el tamaño de las sobreoscilaciones, a costa de aumentar ligeramente el tiempo de establecimiento. Para compensar esto se incrementa levemente la ganancia de la parte proporcional. Finalmente, se reduce también la contribución de la parte derivativa, principalmente para reducir la componente de alta frecuencia en la señal de la acción de control. Con esto, se llega al siguiente conjunto de parámetros, con la respuesta del sistema de la Figura 5.23.

$$PID_{Ajustado}: \begin{cases} K_p = 105 & (+9\%) \\ T_d = K_d = 10 & (-29\%) \\ T_i = 25 \cdot 10^{-3} \rightarrow K_i = \dfrac{1}{T_i} = 40 & (-76\%) \end{cases} \quad (9)$$

Ecuación 5.9. Controlador PID Ajustado

Atendiendo a los resultados del controlador PID ajustado, es conveniente destacar una serie de observaciones. Se ha logrado el objetivo principal de reducir la magnitud y el tiempo de establecimiento de las sobreoscilaciones. Sin embargo, se ha introducido un ligero desvío de la variable controlada respecto de la referencia, que tiene un tiempo de establecimiento mayor que el de las oscilaciones. Esto se aprecia claramente en los segundos 9,5 a 12 en la Figura 5.23, donde la señal medida se sitúa desplazada levemente a la derecha de la imagen. Esto se debe a que el error de la acción integral acumulado durante el tramo de la curva requiere de un periodo más extenso para restablecerse, puesto que se ha incrementado el tiempo integral Ti. No obstante, esto no supone un

problema más allá de la precisión del seguimiento de la trayectoria del circuito, pues la desviación no es lo suficientemente elevada como para afectar negativamente a la detección de cruces. Así pues, se decide emplear el control PID ajustado para el resto de pruebas y demostradores.

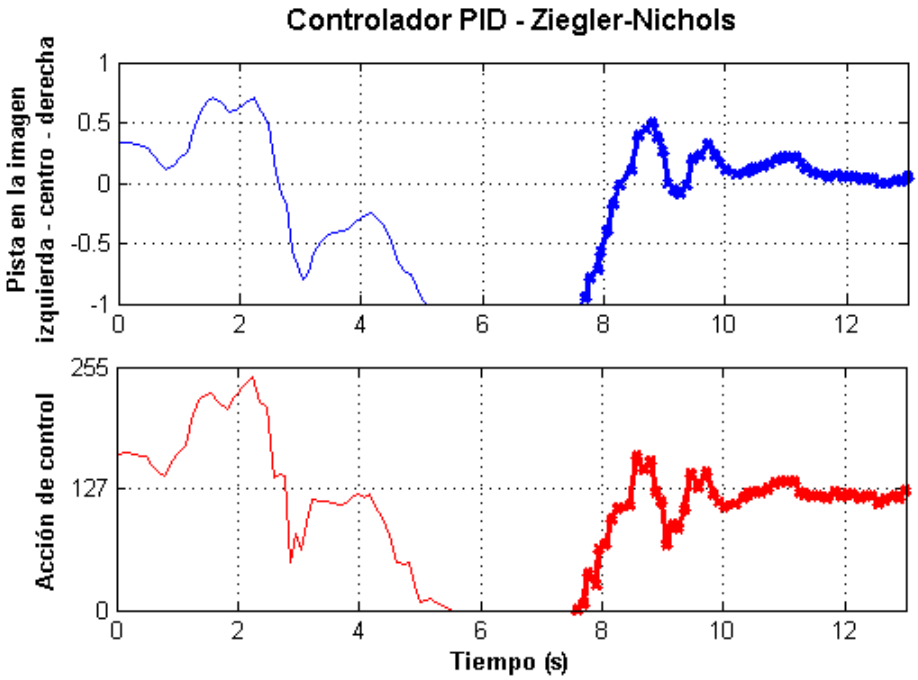


Figura 5.22. Controlador PID, método ZN

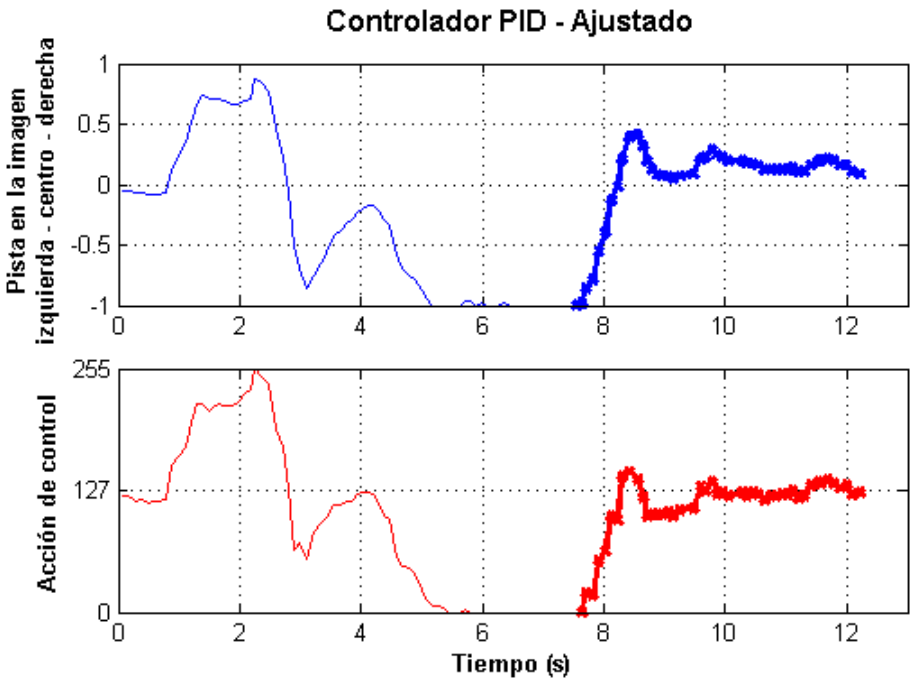


Figura 5.23. Controlador PID Ajustado

CAPÍTULO 6

GUIADO Y PERCEPCIÓN AVANZADOS. DEMOSTRADORES

En este capítulo se desarrollará el sistema perceptivo basado visión artificial que dotará al vehículo de las capacidades requeridas para interpretar su entorno a un nivel más avanzado que el mostrado en el capítulo anterior, en que se disponía de un guiado por seguimiento de línea. Para ello se añadirán gradualmente una serie de funciones al sistema que implementarán dichas capacidades, y que requerirán también de un cambio en la propia estructura del programa para adaptar la lógica encargada de tomar las decisiones en base a la información procesada. Estas funciones serán definidas al inicio del capítulo, para posteriormente ser integradas ordenadamente en programas que, junto con el entorno adecuado, darán lugar a los distintos demostradores. Finalmente se evaluará el desempeño de estos para justificar el uso de técnicas redundantes que dotarán al sistema de una aptitud básica de tolerancia a fallos.

6.1 ALGORITMOS DE PERCEPCIÓN DEL ENTORNO

Ya en capítulos anteriores se ha mencionado la importancia de disponer en el vehículo de la suficiente diversidad de funcionalidad como para constituir una plataforma sobre la cual poder ensayar la tolerancia a fallos en el ámbito del proyecto DINAMOS de la manera más cercana a la realidad posible y ofreciendo múltiples casos de pruebas. En este apartado se presentarán los distintos métodos y algoritmos diseñados a lo largo del trabajo que cubrirán dichas funcionalidades. A modo de repaso, se indicarán los distintos requisitos que se consideraron oportunos, para a continuación pasar a definir las soluciones a las que se ha llegado.

Los elementos fundamentales que debe presentar el entorno del vehículo son:

- **Objetos** en las inmediaciones del trayecto, que podrán simbolizar tanto la presencia de otros vehículos como la de peatones, o a efectos prácticos, la de un objeto genérico que invade el trayecto de la carretera.
- **Cruces** en el trayecto, que representarán intersecciones de carreteras que admitirán un sentido de circulación determinado.
- **Señales de tráfico**, que podrán estar presentes o no en los cruces, e indicarán el sentido de circulación o acción que se requiere del vehículo en el cruce.

Dadas las limitaciones de material y recursos, y puesto que se puede trabajar con tolerancia a fallos con distintos niveles de complejidad ofreciendo igualmente resultados útiles para el ámbito de este

proyecto, no se exigirá el uso de algoritmos del estado de arte que ofrezcan las máximas prestaciones. Se podrán usar técnicas que, si bien más sencillas, serán capaces de ofrecer al vehículo la capacidad de desenvolverse en este entorno de pruebas simplificado, pudiendo simular cada una de ellas un conjunto de técnicas más avanzadas que darían lugar a una funcionalidad muy similar.

Dicho esto, seguidamente se presentarán los algoritmos desarrollados para cubrir las tareas de percepción de los distintos elementos planteados del entorno.

Terminología

Cuando se habla de reconocimiento de imágenes, se pueden distinguir tres tareas principales.

1) **Detección** de objetos o patrones dentro de una imagen. En este caso, se analiza una imagen para tratar de hallar la presencia de un patrón especifico, y la posición de este dentro de la imagen. Una aplicación común es la videovigilancia, en que se debe detectar la presencia de personas en una zona determinada.
2) **Identificación** de un patrón. Dada una imagen de una instancia concreta, como puede ser un objeto o forma aislados, el programa debe decidir si esta se trata o no patrón que se quiere reconocer en particular. Como ejemplos se pueden destacar la identificación de una huella dactilar o de la cara de una persona.
3) **Clasificación** de objetos. Dada una imagen genérica, un sistema que ha sido programado o entrenado para distinguir entre varias clases de objetos debe categorizar los distintos elementos que aparecen en esta. Se puede entender como una generalización de la identificación aplicada a múltiples categorías de objetos, que pueden presentarse en condiciones (posición, tamaño, iluminación) diversas dentro de la imagen. Este tipo de reconocimiento se aplica a casos en que se quiere extraer un significado de un entorno complejo y variable. El ejemplo más conveniente para este trabajo es el de los coches autónomos, que deben ser capaces de interpretar distintos elementos en la imagen como son las señales de tráfico, los letreros, las marcas de la carretera, los peatones u otros vehículos.

A continuación se muestran los algoritmos desarrollados. Cada apartado empezará con una introducción a la tarea de reconocimiento que se desea abordar para posteriormente exponer la implementación en el sistema de la solución adoptada y finalmente mostrar los resultados obtenidos seguidos de una discusión acerca de estos.

6.1.1 DETECCIÓN DE OBJETOS

El reconocimiento de objetos en la imagen para el presente caso se podría abordar de distintas maneras. Si se debiesen llevar a cabo acciones diferentes en función de qué objetos se presentasen en la vía, sería imprescindible disponer de un clasificador de objetos. Sin embargo, para los fines de este trabajo, bastará con diseñar un algoritmo que detecte la presencia o ausencia de objetos genéricos en la imagen (y, en la medida de lo posible, su localización dentro de esta).

Inicialmente, se plantea aprovechar el detector de bordes Canny implementado en el guiado del vehículo, con el que, a partir de la imagen binaria que se obtiene como resultado, se podrían hallar los contornos de los bordes detectados del objeto para a continuación realizar un análisis sobre estos en que se determinasen propiedades como el área, la relación de aspecto, o posición. Esta sería una

alternativa muy conveniente dados los buenos resultados que suele ofrecer el operador de Canny sobre una imagen con presencia de objetos aislados. Sin embargo, puesto que los propios bordes de la pista serían detectados, así como también los distintos reflejos que pueden aparecer con ciertas orientaciones del vehículo en la pista, se obtendrían un gran número de falsos positivos no deseados que harían poco fiable este método. Con esto se hace evidente la dificultad de hallar un objeto cualquiera en un entorno variable.

Existen distintos métodos desarrollados en el ámbito de la visión artificial destinados a hallar la posición en la imagen de un objeto cualquiera en un entorno determinado observado por una cámara. Puesto que en gran parte de las aplicaciones de detección de objetos al mundo real la cámara se sitúa de forma estática respecto al entorno o con variaciones de posición en el tiempo muy sutiles, (como es el caso de las cámaras de videovigilancia), la mayoría de estos algoritmos toman provecho de esta propiedad. Así, un método comúnmente empleado se basa en la comparación de imágenes tomadas en distintos instantes de tiempo (como es el caso de un vídeo) para determinar las variaciones que aparecen en la imagen. Esta técnica se conoce como modelado o extracción de fondo (Yannick Benezeth, 2012), que también cuenta con la correspondiente implementación en OpenCV.

La forma más básica de estos algoritmos utiliza un modelo de fondo constante y determina la detección de un objeto cuando se presenta una diferencia notable en el tiempo de la intensidad de los píxeles de alguna zona de la imagen. Esto puede presentar inconvenientes en entornos sujetos a cambios de iluminación durante el periodo de operación de la cámara. Para ello, se han desarrollado métodos que actualizan continuamente el modelo de fondo sumando de forma ponderada las variaciones de la iluminación general en el entorno respecto al estado anterior. Con una ponderación muy pequeña, el modelo de fondo se corresponde prácticamente al caso del modelo constante. Para una ponderación elevada, se obtiene un modelo de fondo que se actualiza rápidamente en el tiempo, lo que implica que se admitirá movimiento de la cámara y variación de iluminación hasta cierto punto, aunque no será capaz de percibir variaciones sutiles en los elementos del entorno como pueda ser la aparición de objetos con colores similares al entorno.

Con esto, cabe plantearse si se podrían aprovechar estos métodos para el caso presente del proyecto. En un primer análisis se podría proponer un modelo de fondo que se actualizara rápidamente en el tiempo dado el continuo movimiento del vehículo. Sin embargo, puesto que el entorno presenta altos contrastes en los valores de intensidad (dados por los colores oscuros del suelo y el color claro de la pista), resultaría poco práctico ya que la elevada tasa de actualización necesaria adoptaría como modelos de fondo gran cantidad de casos en los que un objeto se presentase en la imagen, impidiendo así su detección.

Por ello, para el ámbito del trabajo se propone el uso de un método alternativo que aprovecha las particularidades del entorno de pruebas. Se determina que, puesto que el suelo (junto con la pista y los posibles reflejos) presenta un rango de colores muy reducido, se podría tratar de extraer un modelo de color que caracterizara el entorno sin objetos (o fondo). Así, se podría determinar fácilmente la presencia de cualquier objeto que presentase colores fuera del intervalo correspondiente al del suelo y la pista.

Aunque esto presentará limitaciones, siendo la principal de ellas la obtención de falsos negativos en presencia de objetos con colores similares al fondo, para el alcance del trabajo seguirá ofreciendo un

amplio margen de opciones a la hora de realizar pruebas. También cabe destacar que, según la zona del circuito en que se sitúe el vehículo se pueden presentar variaciones en el brillo y en la tonalidad del color de los reflejos si el trayecto transcurre cerca del mobiliario, como se muestra en la Figura 6.1.

Figura 6.1. Casos extremos de variación de tonalidad y brillo en el color de la pista

Para tratar la variación de la iluminación y del color de los reflejos, bastará con ampliar el intervalo de colores y brillos que pertenecerán al modelo del fondo. En general, los reflejos del mobiliario en el suelo serán lo suficientemente sutiles como para no alterar en gran medida el color base del suelo.

Implementación y resultados

Con todo esto, se expondrá a continuación el modo en que se ha implementado en OpenCV la detección de objetos basada en diferencias de color con el modelo del entorno, y se mostrarán los resultados aplicando esta técnica a imágenes con objetos distintos.

El primer paso será obtener la representación de la imagen en el espacio de color HSL o HSV, que como se indicó en el Capítulo 4, Figura 4.1, presentan una forma más intuitiva de diferenciar los colores y brillos que el formato común y obtenido por la cámara, RGB. Debido a que en el entorno de la pista se presentan múltiples tonalidades de colores con alto brillo, se elige el formato HSL por el mayor rango de valores dedicado a colores cercanos al blanco. Esto permitirá ajustar con mayor precisión la frontera que define el modelo de colores del entorno sin objetos. Con sintaxis de OpenCV, se usa la función presentada en el Capítulo 4:

```
imagenHLS = cv2.cvtColor(imagenRGB, cv2.COLOR_BGR2HLS)
```

El siguiente paso consiste en la elección del rango de valores de brillo, saturación y tono que permitan cubrir de forma adecuada todos los valores que podrá presentar el entorno sin objetos, incluyendo, como se ha dicho, los casos extremos de cambio de iluminación y tonalidad en la pista. Puesto que el objetivo final es obtener una imagen binaria que muestre en blanco la zona donde se ha detectado el objeto, se aplicará la binarización para imágenes de color expuesta en el Capítulo 4. Por tanto, será necesario especificar el rango de valores para cada porción de color que se quiera binarizar mediante dos matrices de NumPy que guardarán los límites inferior y superior de tal intervalo.

Dada la particularidad del problema, se opta por emplear un método heurístico. Se empieza por definir los rangos de colores que presentarán en uno de sus extremos algún valor límite de representación del formato HSL, para a continuación ajustar experimentalmente el otro extremo de modo que se consiga cubrir adecuadamente los posibles colores que podrá tomar la pista, invadiendo en la menor medida posible los colores que potencialmente podrá presentar un objeto. Por ejemplo, para cubrir los colores de alto brillo (blancos), se selecciona el rango de valores que tiene como límite superior el máximo valor de brillo (255) y como límite inferior un valor de brillo que incluya parcialmente los

colores claros, para contar con las distintas tonalidades que se pueden presentar en los reflejos. Del mismo modo, para los colores de bajo brillo (negros), se escoge un intervalo con 0 como límite inferior de brillo.

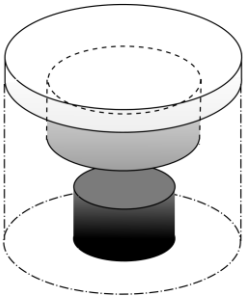
Tras varios ensayos se definen tres bandas o intervalos que representan los colores que puede tomar el entorno sin objetos. Siendo en OpenCV los valores que definen los límites de representación

$$[tono, brillo, saturación] = [0\,..\,180, 0\,..\,255, 0\,..\,255]$$

se muestran los extremos de los rangos en la tabla siguiente. La figura de la derecha representa de forma aproximada la distribución espacial de estos rangos en el espacio de color HSL (tan solo se muestran los valores de brillo; el tono y la saturación han sido omitidos por simplicidad de la representación)

Rango inferior	Rango medio	Rango superior
[0,0,0]	[0,160,0]	[0,255,0]
[180,125,90]	[180,225,175]	[180,225,255]

Tabla 6.1. Rangos HSL del modelo de fondo



Tras aplicar la operación de binarización por colores se obtienen tres imágenes binarias, cada una correspondiente a uno de los tres rangos de colores definidos. Estas representan en blanco la zona correspondiente al fondo o entorno modelizado, y en negro cualquier valor de color que se salga del rango definido. Para juntar estas imágenes en una sola en que se represente en blanco la zona del objeto, se utilizan operaciones lógicas para imágenes binarias. En primer lugar, se efectúa la operación de inversión de valores binarios (NOT) a cada una de las imágenes. En segundo lugar, se unen en una imagen mediante la operación AND, que mantendrá los valores blancos –ahora, correspondientes al objeto– de cada una de las imágenes. El resultado de estas dos operaciones suele incluir un leve grado de ruido (en forma de puntos blancos) en el fondo, correspondientes con valores del color de la textura del suelo no incluidos en el modelo. Esto se resuelve fácilmente usando operaciones morfológicas sobre las imágenes binarias, que consiguen eliminar correctamente el ruido usando un tamaño de filtro de 3 píxeles.

Finalmente, los resultados que se obtienen después de aplicar las operaciones descritas se muestran en las siguientes imágenes. Como se aprecia, el algoritmo es capaz de detectar la presencia de objetos de distintas tonalidades y brillos.

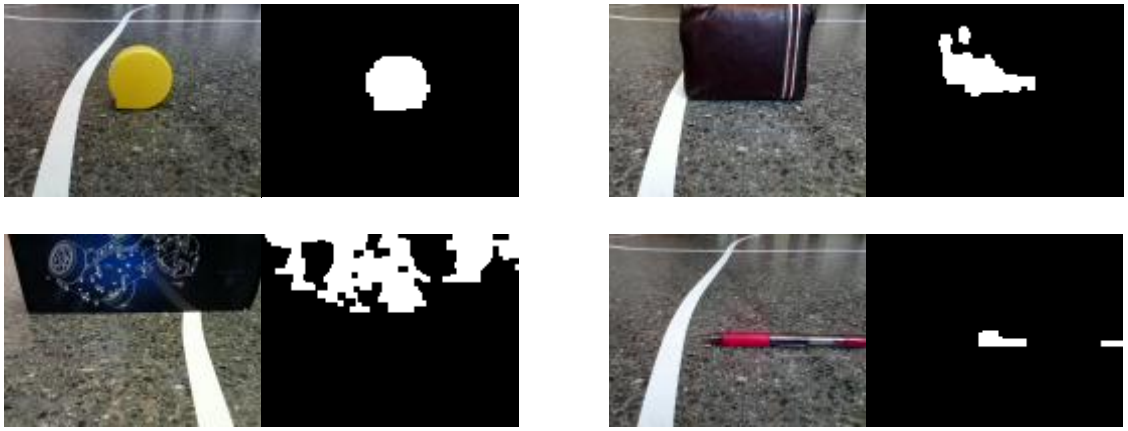


Figura 6.2. Resultado de la detección de objetos por diferencia de colores ante distintos ejemplares.

Discusión

Este método proporciona los resultados necesarios para realizar las pruebas del proyecto de detección de objetos mediante visión artificial. Aunque presenta limitaciones cuando se intentan detectar objetos con valores de colores similares a los del modelo de fondo (Figura 6.3), esto servirá también para justificar el uso de sistemas de percepción redundantes y para simular los casos reales en que un vehículo autónomo pueda presentar fallos también por visión debido principalmente a condiciones atmosféricas adversas.

Figura 6.3. Falso negativo en el reconocimiento de un objeto con color similar al fondo.

También habrá casos en que la iluminación varíe más de lo previsto e induzca falsos positivos en la detección de un objeto. No obstante, en el caso general se podrá ensayar en multitud de zonas de la pista –si no en todas– que presentarán unas condiciones de iluminación favorables.

Finalmente, dado el rango de colores que puede presentar un mismo objeto, perteneciendo algunos de estos al modelo del fondo, este método no permite obtener de forma fiable propiedades del objeto tales como el tamaño o la posición. Así, las acciones que efectuará el vehículo en las pruebas se regirán tan solo por la presencia de este, sin tener en cuenta sus características.

6.1.2 IDENTIFICACIÓN DE CRUCES

El reconocimiento de cruces se trata, en este ámbito, de una tarea con un solo objetivo. Se debe determinar, dada una imagen del circuito obtenida por el vehículo, si el trayecto presente en esta se corresponde con una intersección. Puesto que no se deben diferenciar entre distintas categorías ni se

pretende a priori conocer la localización del modelo en la imagen, se puede definir esta tarea de reconocimiento como identificación de un patrón.

En el diseño de este identificador se deberá hallar en primer lugar el conjunto de propiedades cuantificables que presente la mayor variación numérica posible entre cruces y curvas o rectas genéricas para poder facilitar su distinción. De un primer enfoque, se identifican un gran número de características espaciales que podrían definir un fragmento de trayecto visto en la imagen. Mediante el análisis ángulos, variaciones de curvatura a lo largo del trayecto, porción que este ocupa en la imagen e incluso mediante el cálculo de momentos, se podría diseñar un complejo identificador que ponderara todas estas cualidades para hallar la probabilidad de que el tramo de pista en la imagen en cuestión se tratase de un cruce. Sin embargo, tal y como se ha indicado en distintos puntos a lo largo del trabajo, tanto por la presencia de ruido en la imagen como por la necesidad de usar métodos de corto tiempo de ejecución (y también por razones prácticas), se rechaza cualquier alternativa demasiado compleja o sensible al ruido.

Tras múltiples observaciones se plantea que la propiedad cuyo valor se diferencia más claramente entre cruces y curvas, incluso en presencia de cierta cantidad de reflejos, es la distribución de los ángulos (píxel a píxel) de los bordes detectados en el trayecto. Esto es, para una forma de trayecto dada, si se calcula la orientación espacial de los bordes en cada píxel, el histograma de ángulos obtenidos presentará características bien distintas según el trayecto se trate o no de un cruce. Así, para el caso de un cruce, dicho histograma contará con dos picos destacados (distribución bimodal) correspondientes a los ángulos de las dos direcciones dominantes del cruce. En cambio, para el caso de una recta se presentará un solo pico, mientras que para una curva, se encontrará una distribución que podrá presentar uno o dos picos, pero con poca altura y en general mucha variabilidad en el rango de valores de ángulos.

Implementación y resultados

A la hora de implementar este método en el programa, se presentan una serie de opciones. La forma más sencilla y rápida se basaría en tomar las dos barras del histograma que presentasen un valor mayor de frecuencias de ángulos y, en función de las magnitudes de estos valores y de los ángulos a los que se correspondiesen, se podría tratar de distinguir los cruces. Esto, sin embargo, en la práctica, no resulta tan trivial puesto que, si los trazados de un cruce poseen cierta curvatura o se presentan reflejos en la imagen, las dos barras con frecuencias mayores podrían variar tanto en posición en el histograma (correspondiente al ángulo del borde) como en magnitud (cantidad o frecuencia de píxeles con tal ángulo), dificultando por tanto la distinción entre cruces y curvas. Visto de otro modo, si se analizan tan solo las dos barras con mayores frecuencias, se pierde la información del resto del histograma.

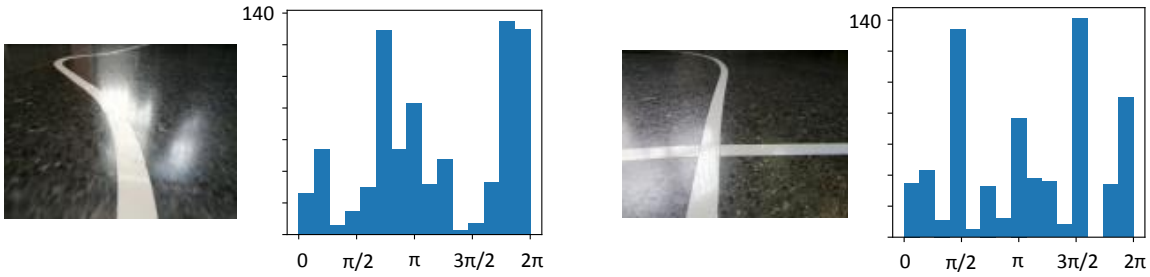
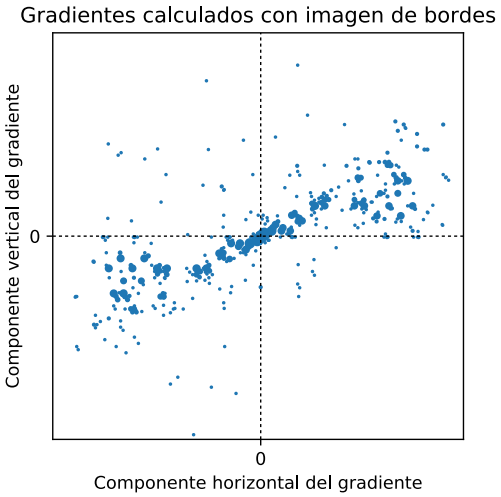
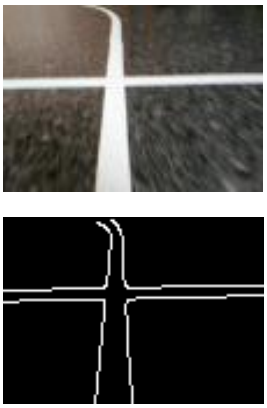


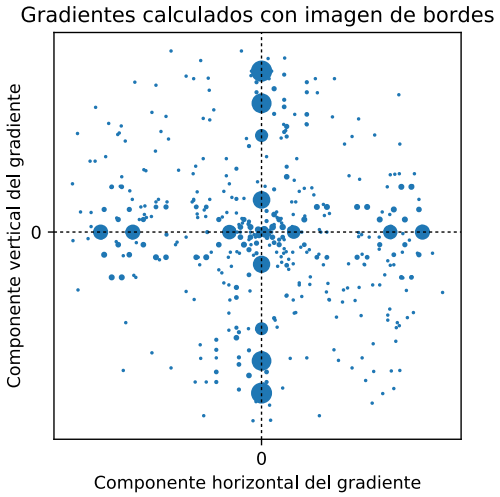
Figura 6.4. Histogramas de ángulos de bordes detectados

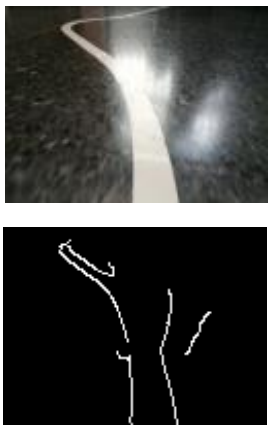
Para hacer frente a este problema, se plantea una solución en que se cuenta con todos los datos de los ángulos. No obstante, antes de exponerla será necesario comprender cómo se obtienen los ángulos de los bordes del trayecto con OpenCV. Como se ha dicho, se requiere en primer lugar una imagen en que se representen tan solo los bordes, con lo que será necesario usar el operador Canny. Se usarán los parámetros de este operador que se establecieron en el diseño del controlador para el guiado, que consiguen llegar a un mejor balance entre detección de bordes y eliminación de reflejos. A partir de la imagen binaria obtenida, se procederá con el cálculo de los gradientes espaciales mediante el operador de Sobel, con el que se obtendrán las derivadas en las direcciones horizontal y vertical de la imagen. Así, por cada píxel en la imagen se tendrá un gradiente expresado en coordenadas cartesianas. En el caso de que el píxel se encuentre en la proximidad de un borde, este gradiente será distinto de cero e indicará la dirección normal a dicho borde. Para los píxeles situados lejos de los bordes, el gradiente será nulo. Ahora bien, es fundamental saber que, puesto que el gradiente se obtiene por la variación de intensidad entre los píxeles pertenecientes al borde (blancos) y los de sus inmediaciones (negros), por cada línea que forme un borde se obtendrán gradientes no nulos a ambos lados de esta. Lo que es más, estos gradientes expresarán direcciones opuestas 180°.

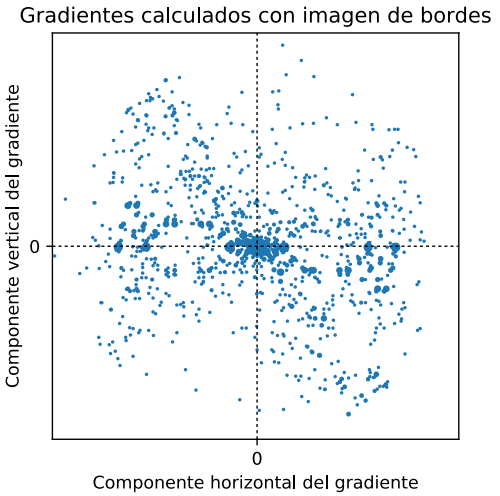
Sabiendo esto, si los gradientes (expresados por sus componentes vertical y horizontal en la imagen) se representan en un plano en coordenadas cartesianas, se mostrará una distribución espacial centrada en el origen (dado que cada ángulo tiene su opuesto 180°) que será distinta según la imagen sea de un cruce o de una recta o curva. En particular, para el caso de un cruce se presentarán dos direcciones principales claramente distinguibles en dicho plano que se corresponderán con los ángulos de las dos rectas que forman el cruce, y se encontrarán desfasadas un ángulo de aproximadamente 90°, admitiendo una ligera variación según la perspectiva desde la que se observe el cruce. A continuación, se muestran algunos de los resultados tras aplicar el cálculo de gradientes sobre imágenes de la pista binarizadas.


Gradientes calculados con imagen de bordes
Componente vertical del gradiente
0
0
Componente horizontal del gradiente




Gradientes calculados con imagen de bordes
Componente vertical del gradiente
0
0
Componente horizontal del gradiente




Gradientes calculados con imagen de bordes
Componente vertical del gradiente
0
0
Componente horizontal del gradiente

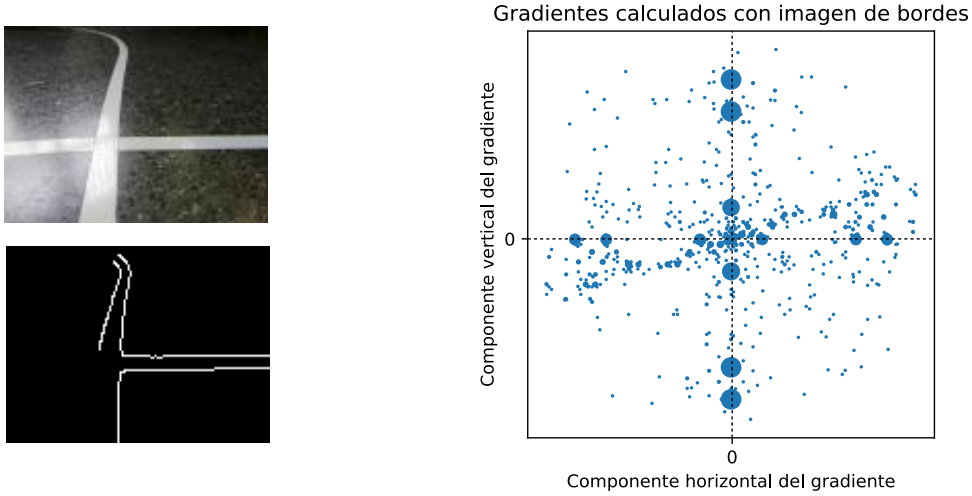


Figura 6.5. Gradientes calculados en rectas, curvas y cruces, con variaciones de iluminación

Así, se plantea que, mediante la obtención de los valores propios asociados a estas direcciones principales y tras un análisis de las magnitudes de estos, se podrá determinar si el trayecto de la imagen se trata de un cruce. Si así fuese, cabría esperar dos valores propios elevados. En cambio, si se tratase de una recta, se presentaría un valor propio elevado y otro mucho menor y, para una curva, se obtendrían dos valores propios distintos y de magnitud media, puesto que no se presentaría una clara dirección predominante.

En Python, se obtienen estos vectores propios efectuando dos operaciones, partiendo de una lista de puntos cuyas coordenadas en el plano cartesiano representan los componentes vertical y horizontal de los gradientes calculados con el operador de Sobel. Esta lista de puntos será en esencia una matriz de NumPy con tamaño 2xN, donde N representará número el total de píxeles de la imagen, y cada columna de esta matriz será un vector de tamaño 2x1 que guardará los dos componentes del gradiente.

Con esto, en primer lugar, se calcula la matriz de covarianzas de esta distribución de puntos con la función de NumPy

```
matrizCovarianzas = numpy.cov(listaDePuntos)
```

A continuación, a partir de esta matriz, mediante la operación “`eig`” del módulo de álgebra lineal de NumPy, se obtienen los autovalores y los autovectores.

```
autoValores,autoVectores = numpy.linalg.eig(matrizCovarianzas)
```

donde “`autovalores`” es un vector de Python de tamaño 2 que contiene los dos valores propios, y “`autoVectores`” es una matriz 2x2 que tiene como filas los dos autovectores.

Representando los autovectores en las imágenes anteriores, con la longitud de estos proporcional a sus respectivos autovalores, se obtienen los siguientes resultados. El vector con mayor autovalor se representa en azul y el menor en verde. El origen de estos vectores se sitúa en el centroide de la imagen binaria de bordes.

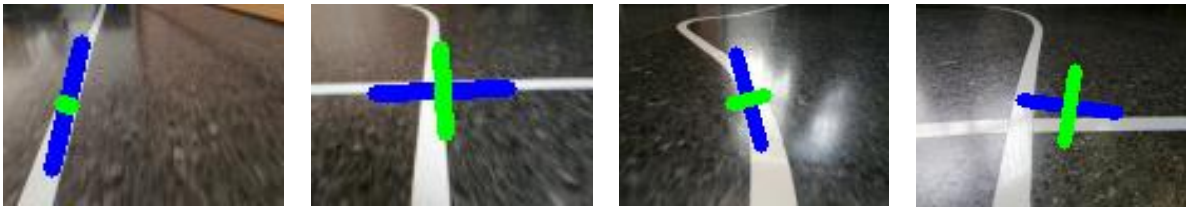


Figura 6.6. Vectores y valores propios en rectas, curvas, y cruces

Con esto, se pueden establecer dos criterios principales que permitirán identificar los cruces.

En primer lugar, puesto que la magnitud de los dos valores propios difiere en gran medida para imágenes con rectas o curvas suaves, se propone un primer filtro para descartarlas basado en esta propiedad. Para ello se calcula la relación existente entre los autovalores efectuando el cociente del menor entre el mayor. Se determina, tras una serie de análisis con distintas imágenes obtenidas, que la relación que pueden presentar los valores propios de un cruce puede llegar a mínimos cercanos a 0,5 en situaciones de medida degradada por reflejos o de desplazamiento del trazado de la intersección a un extremo de la imagen (ver Figura 6.7). Por tanto, como primer criterio se requerirá que la relación de los valores supere este umbral (criterio 1).

Sin embargo, puesto que se pueden presentar curvas muy pronunciadas en la imagen que superan dicho umbral, se determina otro factor diferenciador. Se halla que, la suma de los valores propios en el caso de los cruces supera incluso en las situaciones más desfavorables de iluminación (en que no se consigue distinguir parte del trayecto por la presencia de reflejos) el valor de K=650.000 mientras que la gran mayoría de las curvas no llegan a alcanzarlo. Estableciendo este valor como el mínimo que deben presentar (criterio 2), el criterio global para la identificación de cruces será, en sintaxis de Python:

```
cruce = (autoValMin / autoValMax > 0.5) and (autoValMin + autoValMax > K)
```

donde "`cruce`" será una variable binaria o booleana que será True (verdadera) si se cumple la condición y False (falsa) en caso contrario.

A continuación se presentan los resultados aplicando el algoritmo en distintos trayectos y condiciones de iluminación.

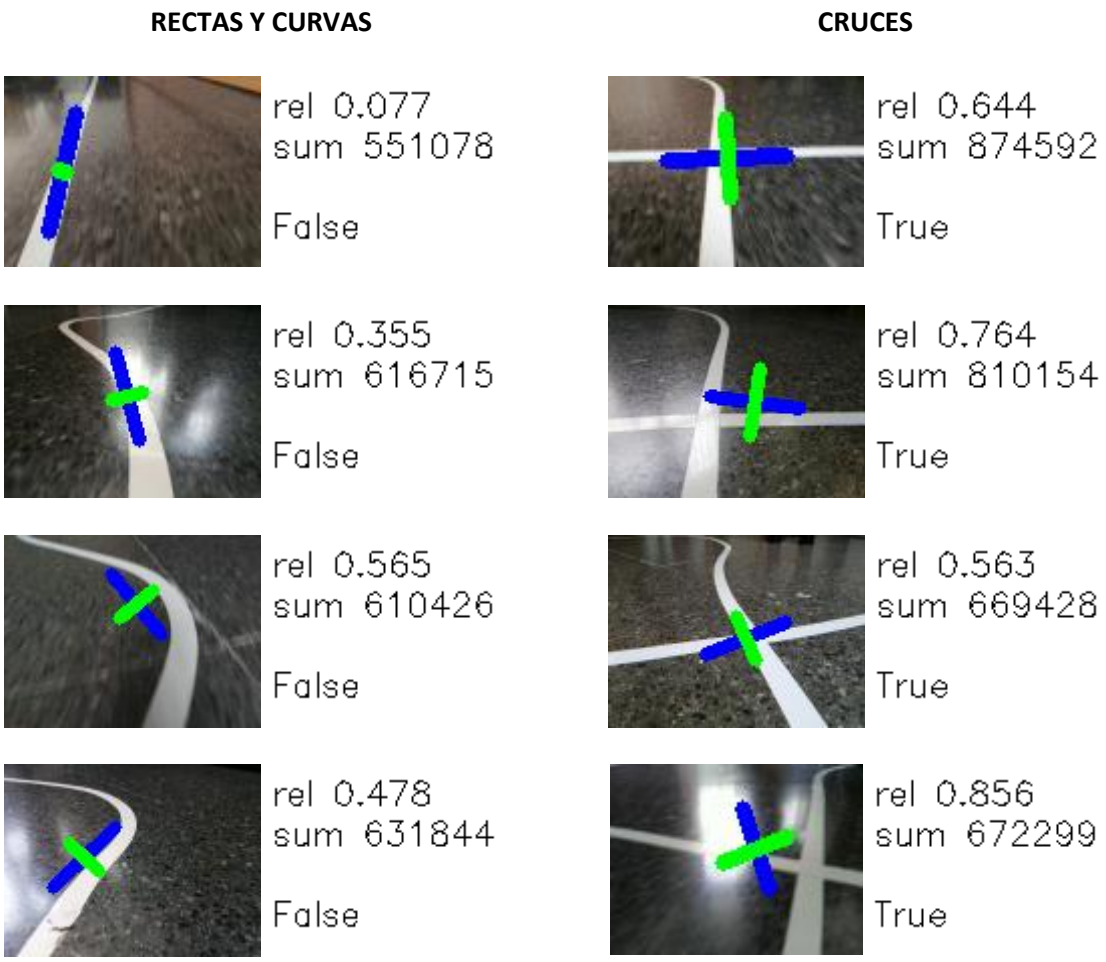


Figura 6.7. Resultado del algoritmo de identificación de cruces en diferentes situaciones. 'rel' indica la relación dada por el cociente del menor autovalor entre el mayor, y 'sum' indica la suma de los autovalores.

Discusión

Este método de identificación de cruces proporciona resultados favorables incluso en situaciones adversas de iluminación. Esto permitirá desarrollar las pruebas que se llevarán a cabo en el proyecto independientemente de las condiciones de reflejos que se presenten.

Como se ha mencionado, aunque muy aislados, existen casos en los que la suma de los autovalores obtenidos de una curva pueden superar el umbral de 650.000. Si esta curva además también es pronunciada y la relación de los valores propios supera el umbral de 0,5, se podrá identificar erróneamente como un cruce. En la práctica, tan solo se ha identificado una curva que presenta dichas cualidades. Para tratar de solucionar esto se podría rediseñar el circuito para cambiar la forma de la curva. Sin embargo, a la hora de realizar las pruebas de tolerancia a fallos, interesará disponer de casos que pongan al límite las capacidades de la visión para evaluar el grado en que los algoritmos que se desarrollen serán capaces de actuar ante anomalías.

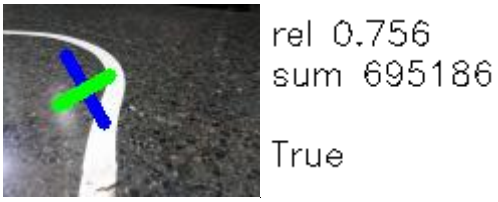


Figura 6.8. Falso positivo, reconocimiento de cruces

Por otro lado, también se pueden dar casos en que un cruce se presente en la imagen con una perspectiva que altere notablemente el ángulo de 90° que forman las rectas que lo componen en la realidad. Esto impedirá la obtención de unos valores propios cuyos autovectores representen las direcciones de los ángulos principales, y el criterio 1 no podrá ser alcanzado. No obstante, en la práctica esto no llega a suceder por el modo en que se ha diseñado el controlador PID del guiado. Por lo general, el vehículo siempre se mantendrá lo suficientemente centrado en el trayecto como para no observar el cruce desde una perspectiva que impida la identificación de este. En el ejemplo que se muestra abajo el vehículo se ha situado a propósito a una mayor distancia al centro del trayecto de la habitual. Aun así, si fuese necesario, se podría disminuir el umbral del criterio 1 y, aunque existiría un mayor riesgo de medir falsos positivos, haciendo uso de un diseño tolerante a fallos se podría tratar solventar, tal y como se ha mencionado en el párrafo anterior y se verá en el próximo capítulo.

Figura 6.9. Falso negativo, reconocimiento de cruces.

6.1.3 DETECCIÓN Y CLASIFICACIÓN DE SEÑALES

Idealmente, las señales de tráfico se situarían en cualquier lugar del circuito de pruebas para proporcionar distintas indicaciones al vehículo durante su marcha, como el ajuste de la velocidad límite del tramo de la carretera, el aviso de curvas cerradas o incluso el de la presencia de obstáculos en el camino. Esto, sin embargo, no es realizable del modo en que se ha diseñado el guiado, puesto que el ángulo que presenta la cámara respecto al suelo impide la percepción de información presentada a la altura de una señal. Se podría tratar de solucionar este problema con el uso de otra cámara adicional que proporcionara una visión más elevada y dirigida hacia el frente, pero disponiendo tan solo de una Raspberry Pi el tiempo de procesamiento aumentaría en exceso y afectaría negativamente al guiado.

Por tanto, se opta por usar señales que deban ser detectadas necesariamente cuando la cámara perciba algún tipo de información relevante que pueda estar relacionada con la presencia de estas.

Esto concuerda con el caso de los cruces, en que se requerirá que el vehículo tome una acción distinta según como se definan las señales el sentido de circulación de la intersección. Así pues, cuando se identifique la presencia de un cruce, el vehículo deberá parar, cambiar la posición de la cámara hasta formar un ángulo paralelo al suelo y, tras interpretar la información asociada al cruce dada por las señales (o por la ausencia de estas), deberá efectuar la acción correspondiente.

Esto puede quitar cierto realismo a la situación, puesto que no todas las intersecciones requieren la detención del vehículo si su carril tiene preferencia. No obstante, se asume que, para este caso práctico simplificado, la parada y el cambio de posición de la cámara simulan la presencia de otra cámara que actúa simultáneamente con el guiado.

La tarea de reconocimiento en este caso presentará dos fases. Por un lado, se deberá detectar la presencia de la señal en la imagen y, una vez localizada, deberá ser clasificada según los modelos de señales de que el programa disponga.

Existe una gran variedad de algoritmos complejos que permiten detectar una imagen modelo dentro de otra imagen genérica (por ejemplo, un coche específico –imagen modelo– en una carretera –imagen genérica). Estos se basan en la extracción de características del modelo que, por lo general, suelen ser las esquinas de las formas que se presentan en la imagen. Caracterizando la orientación de estas esquinas y la posición relativa que presentan entre ellas, se puede hallar el modelo en cuestión en otra imagen incluso habiéndose producido una transformación geométrica de éste como la rotación o la homografía.

A pesar del potencial que presentan estas técnicas, este no puede ser aprovechado cuando se dispone de un modelo con formas simples que presentan pocas esquinas, como es el caso de las señales de tráfico. Así, aunque OpenCV dispone de múltiples de estos algoritmos (tales como *SIFT, SURF, FAST, BRIEF* u *ORB*) y guías para implementarlos, se opta por buscar otra solución.

Teniendo en cuenta que las señales captadas en la imagen no sufrirán grandes transformaciones geométricas más allá de un escalado (o cambio de tamaño), se puede plantear el uso de la convolución del modelo de la señal sobre la imagen para hallar la zona que produce un resultado de mayor valor tras la operación. Esto se corresponde con la técnica de "template matching" o comparación de modelos, descrita en el Capítulo 4. Para tratar la variación del tamaño del modelo en la imagen en la que este se desea detectar, simplemente se efectuarán múltiples operaciones de convolución cambiando el tamaño del filtro o imagen de referencia del modelo.

Con todo esto, solo queda tener en cuenta otro aspecto para poder definir completamente el algoritmo que será usado en el programa. Mediante el uso de la técnica de comparación de modelos se obtiene una imagen de salida que presenta en cada píxel un valor que se corresponde con el resultado de la convolución del modelo centrado en esa zona de la imagen de partida. Un mayor valor en el pixel indica que la forma presente en esa zona se asemeja en mayor medida a la forma que presenta el modelo. Por tanto, para localizar este, basta con tomar el píxel que presenta el mayor valor. Sin embargo, se puede dar el caso en que el modelo no se encuentre en la imagen, y la elección del píxel con mayor valor no garantizará que realmente se haya encontrado la forma buscada.

Para resolver esto, se deberá determinar qué valor mínimo puede presentar el píxel que localiza el modelo en el caso de que este realmente se encuentre en la imagen. Tomando este valor como umbral, se determinará que la señal buscada se ha encontrado sólo si se supera tal valor.

Implementación y resultados

Se plantea disponer de cuatro señales diferentes que se podrían presentar en una intersección real. Estas son, STOP (*R-2*), y sentidos obligatorios recto, a la derecha y a la izquierda (*R-400 c*, *d* y *e*, respectivamente). En particular se eligen las de sentido de giro derecha e izquierda *(R-400 d* y *R-400 e*) en lugar de las *R-400 a* y *R-400 b* debido a que presentan una mayor variación espacial de la forma y será por tanto más sencillo distinguir entre ellas.

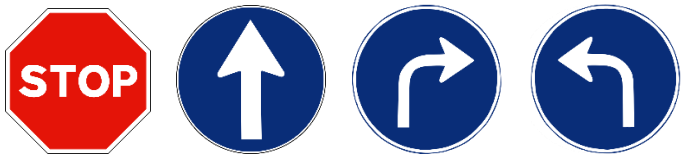


Figura 6.10. Señales de tráfico usadas

Para efectuar la técnica de comparación de modelos se necesita disponer de imágenes de un solo canal (escala de grises). Puesto que en este caso los modelos presentan formas simples con colores sólidos y contrastes bien definidos, resultará más eficiente capturar la información que representan mediante la obtención de sus bordes. Así pues, se usarán imágenes binarias (blanco y negro) obtenidas con el detector de bordes Canny.

Con tal de ajustar mejor los modelos a la realidad, estos se obtendrán directamente de las imágenes de las señales impresas en papel captadas por la cámara, en lugar de simplemente usar las imágenes normalizadas presentadas en la Figura 6.10.

El tamaño de las imágenes modelo se determinará en función de la distancia media esperada del vehículo respecto de las señales tras parar en frente de un cruce. Tras una serie de pruebas teniendo en cuenta este criterio, se obtienen los siguientes modelos.

Figura 6.11. Señales de tráfico con bordes

Con esto, queda cubrir la variación de tamaño que pueden presentar las señales en la imagen debido a la error de precisión asociado a las distintas distancias en las que se pueden situar las señales y el vehículo respecto del cruce. Para ello se efectuarán seis iteraciones de búsqueda por comparación de modelos en que se cambiará el tamaño del modelo de la señal aumentando la longitud de los lados

hasta un 20% y disminuyéndolos hasta un 30% respecto al determinado como tamaño medio. Puesto que el valor resultante de la convolución es proporcional al área o cantidad de píxeles del modelo, se dividirá el valor encontrado como máximo tras la convolución entre la relación de áreas del modelo escalado y el original (menor que 1 si se ha disminuido el tamaño y mayor si se ha aumentado), para obtener un resultado invariante a la escala. El tamaño del modelo que dé un mayor resultado entre las seis iteraciones será el que localice la señal en la imagen.

Tras esta operación, se determina, situando cada señal a diferentes distancias dentro del rango esperado, el valor mínimo que localiza correctamente el modelo en la imagen cuando la señal se encuentra realmente en frente de la cámara. Este valor (añadiendo un cierto margen inferior) será el umbral que discrimine la presencia o ausencia de la señal.

Cabe decir que, puesto que los modelos de imágenes binarias presentan áreas distintas, los valores obtenidos tras la convolución para una misma distancia diferirán ligeramente de un modelo a otro. Particularmente se encuentra que los valores obtenidos de las señales de sentido obligatorio son similares, mientras que los de la señal de STOP suelen ser mayores. Para corregir esta desviación se disminuyen en un 20% todos los resultados obtenidos con la señal de STOP.

Por último, para determinar correctamente qué señal se presenta en la imagen, se efectúan las seis iteraciones para cada uno de los cuatro modelos de señales y se escoge aquel que presente un mayor resultado. Si ninguno de ellos consigue superar el umbral, se determina que no hay ninguna señal.

Un aspecto importante que destacar en la implementación del programa es que, puesto que para realizar la tarea de detección de señales la cámara se situará paralela al suelo, dada la reducida altura de estas se puede prescindir de la mitad superior de la imagen y así eliminar tiempo de cálculo innecesario.

En la siguiente figura (Figura 6.12) se presentan los resultados para el caso de señal de sentido obligatorio a la derecha y para el de ausencia de señales. El valor que se muestra junto con el nombre de la señal en cada imagen representa el cociente entre el valor hallado y el umbral. Se observa que, en presencia de esta, ciertos modelos establecen su localización en el lugar de la señal, pero es la que proporciona mayor resultado (en este caso, la de sentido de giro a la derecha) la que determina qué señal se ha hallado.

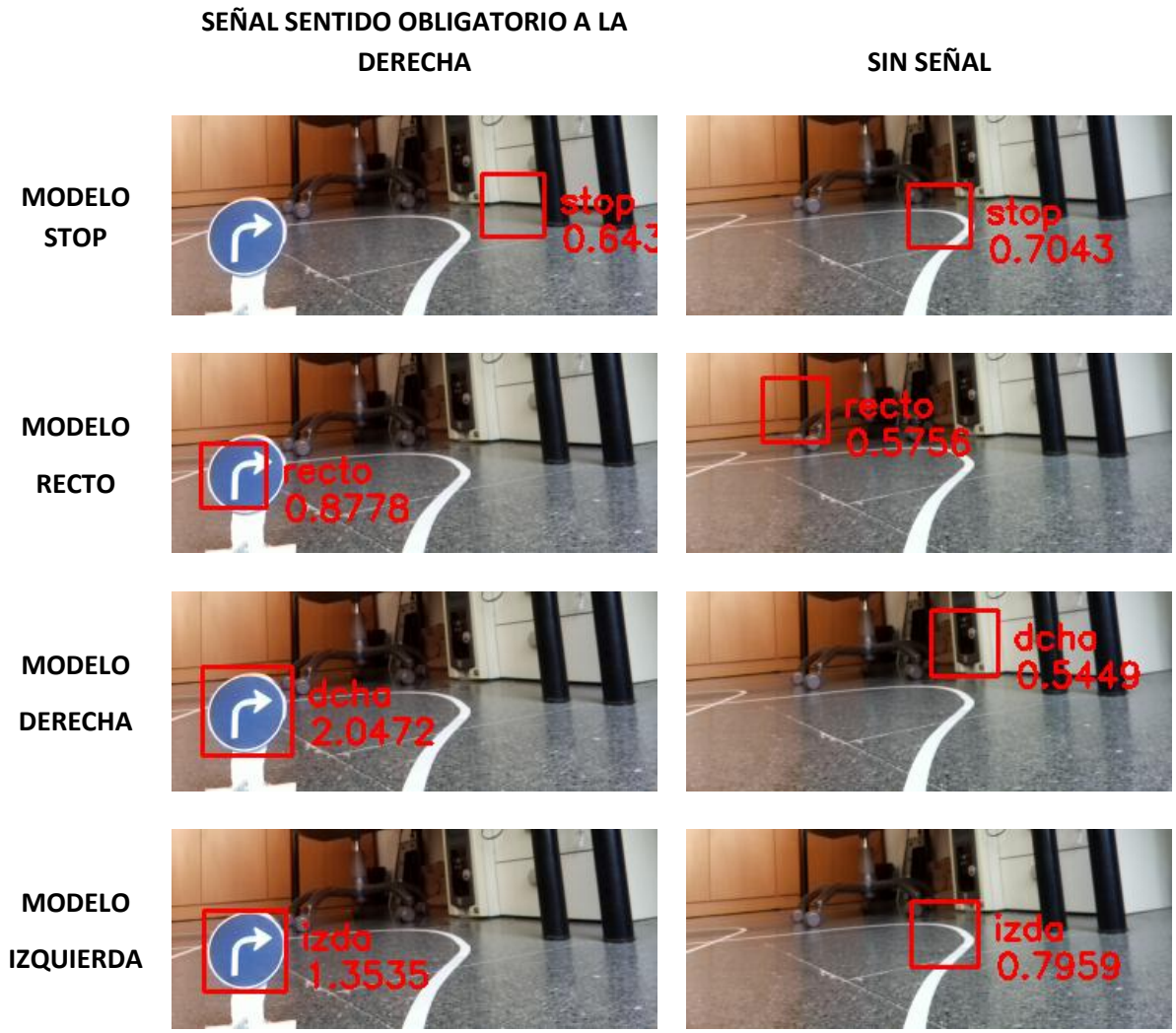


Figura 6.12. Resultados del algoritmo de detección y clasificación de señales de tráfico

Discusión

Pese a que este método de clasificación de señales ofrece generalmente buenos resultados, en condiciones adversas de iluminación puede presentar dificultades para obtener la imagen de bordes, y por tanto para efectuar la correcta localización del modelo de la señal. Esto, sin embargo, no supone un gran inconveniente, puesto que se puede elegir el lugar de la pista donde situar la señal en el momento de realizar las pruebas.

Otro asunto más importante se presenta cuando la señal se sitúa desplazada hacia los laterales de la imagen hasta el punto en que es ocultada parcialmente o incluso queda fuera del ángulo de visión de la cámara. Esto puede ocurrir si la señal no se encuentra bien situada en el cruce o si el vehículo lo ha abordado el con un cierto ángulo. En este caso el vehículo actuaría según la acción correspondiente al caso de ausencia de señales, y presentaría por tanto un fallo en el sistema causado por la visión. Generalmente este problema no se presentará en numerosas situaciones, pero en aquellas en las que suceda se tratará de solucionar en el capítulo de tolerancia a fallos

6.2 DEMOSTRADORES

Habiendo definido los algoritmos que permiten interpretar información más compleja del entorno, se podrán diseñar los distintos programas que harán uso de estos para constituir los demostradores. Cada uno de estos programas estará adaptado para dotar al vehículo de unas capacidades específicas en un determinado entorno. Con esto se conseguirá establecer un conjunto de sistemas de prueba que permitirán ensayar con distintos niveles de complejidad la tolerancia a fallos hardware del proyecto DINAMOS.

Todos estos programas tendrán en común tres funciones principales.

- La primera, y más esencial, será el **seguimiento del trayecto** del circuito mediante el uso del controlador del guiado descrito en el capítulo anterior.
- La segunda será la **parada de seguridad** mediante la detención del vehículo si cualquiera de los dos **sensores ultrasónicos** detecta la presencia de un objeto a una distancia inferior a un umbral determinado. Se establecerá que una distancia de 15 cm deja un margen de seguridad suficiente para que el vehículo reaccione a tiempo y no se produzca la colisión con el objeto.
- La tercera y última, será la **detención por ocupación de calzada** de un objeto detectado mediante **visión**. Se efectuará la detención para cualquier posición del objeto en la imagen. Ya que en algunas zonas del circuito bajo ciertas condiciones de iluminación ambiental se pueden dar falsos positivos en el reconocimiento de objetos, este método tan solo se usará en condiciones controladas favorables. En cualquier momento también se podrá modificar el programa para que no detenga el vehículo con este método. Esto se tratará también en el capítulo de tolerancia a fallos

6.2.1 PRIMER DEMOSTRADOR. CIRCUITO CERRADO, PARADA ANTE OBJETOS

Para este demostrador se usará un circuito cerrado sin intersecciones. Se modificará un tramo de la pista original para componer un trayecto cerrado aislado del resto, según se muestra en la Figura 6.13. Las capacidades que el robot presentará serán las básicas, esto es, guiado autónomo y detención ante objetos detectados por visión o sensores ultrasónicos.

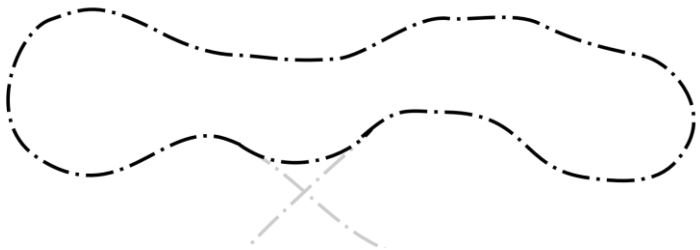


Figura 6.13. Circuito modificado para el primer demostrador.

El diagrama de la Figura 6.14 presenta la operación del programa. Se muestran los hilos de ejecución paralelos encargados de la toma de la imagen desde la cámara y las distancias desde los sensores ultrasónicos.

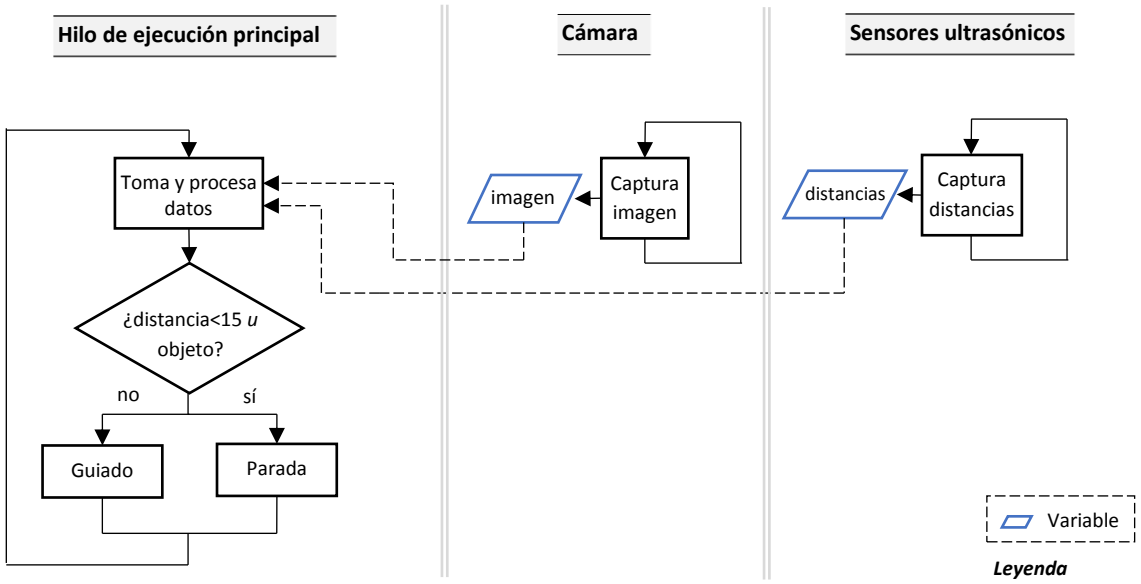


Figura 6.14. Demostrador 1, diagrama de flujo. Detección de objetos.

6.2.2 SEGUNDO DEMOSTRADOR. CIRCUITO CON CRUCES

En este caso se usará el circuito completo, que presentará tres intersecciones. El programa será capaz de identificar la presencia de cruces y, cuando se sitúe a una distancia concreta de estos, efectuará una acción preestablecida.

La distancia a la que se encuentra del cruce se determinará mediante una función sencilla que, aunque no garantizará una medida precisa, proporcionará un resultado aproximado y rápido que resultará útil en la práctica. Aprovechando la imagen binaria de bordes, se tomará la fila que mayor cantidad de píxeles con valor distinto de cero tenga. Esta se situará en la banda horizontal de la imagen que contiene los bordes del tramo de la pista que forma la intersección con la línea que está siguiendo el vehículo. Estableciendo una fila como umbral, el vehículo pasará a efectuar las acciones asociadas a los cruces cuando la fila calculada con este método se sitúe por debajo de dicho umbral. Se determina que la mitad de la imagen (fila 40 de 80), proporciona unos buenos resultados a la hora de efectuar las acciones que se describen a continuación.

Cabe decir que durante todo el tramo en que el vehículo identifica el cruce, se reducirá la velocidad para mejorar la respuesta del vehículo.

Las acciones que deberá llevar a cabo se definirán previamente en un fichero de texto que el programa leerá. Se establecerán unos códigos numéricos consecutivos de modo que cada uno represente la acción que se debe tomar cada vez que se detecta un nuevo cruce. Estos serán: 0, simbolizando una parada ante una señal de STOP; 1, para continuar recto; 2, para girar a la derecha, y 3 para efectuar un giro a la izquierda. Estos se ejecutarán de forma cíclica. A modo de ejemplo, si el fichero contiene los números 1, 2 y 3, en el primer cruce detectado el vehículo seguirá recto, en el segundo girará a la derecha y en el tercero a la izquierda. A partir del cuarto y posteriores repetirá la secuencia.

Los giros se llevarán a cabo de forma temporizada. Situando las ruedas delanteras en las posiciones angulares extremas izquierda o derecha y dejando transcurrir alrededor de 1,2 segundos a velocidad reducida, el vehículo se incorporará al carril correspondiente describiendo un giro de 90° aproximadamente.

En el caso de las paradas por STOP, el vehículo se detendrá y no reanudará la marcha hasta que se introduzca una orden desde el teclado.

Se presenta en la Figura 6.15 el diagrama que añade estas funcionalidades al programa. Por simplicidad se muestra tan solo el fragmento de programa que sufre cambios, omitiendo la toma y procesamiento de datos y los hilos paralelos.

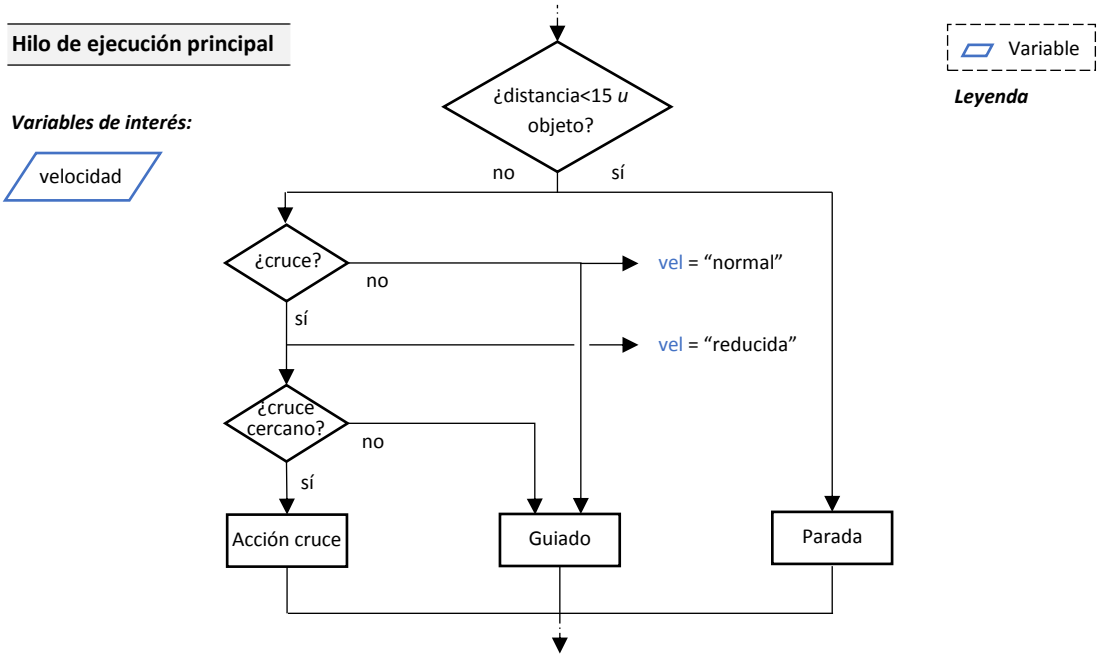


Figura 6.15. Demostrador 2, diagrama de flujo. Identificación de cruces.

6.2.3 TERCER DEMOSTRADOR. DETECCIÓN DE SEÑALES

Para este demostrador se usará también el circuito completo y se añadirán además señales de tráfico que el vehículo deberá interpretar en las intersecciones. Cuando se identifique el cruce y la distancia de este se sitúe por debajo del umbral definido en el apartado anterior, el vehículo se detendrá y pasará a efectuar la tarea de reconocimiento de señales, tras la cual obtendrá la información necesaria para llevar a cabo la acción asociada a tal cruce.

El programa se modificará para que, tras la detección del cruce a la distancia adecuada, se detenga el vehículo, se interrumpa el flujo de ejecución normal y se reconfiguren los elementos necesarios para llevar a cabo la detección de las señales de tráfico. Esta reconfiguración incluirá el cambio de resolución con el que la cámara deberá obtener las imágenes y el cambio de la posición de los servomotores que

determinan la orientación de la cámara. En concreto, se usará un tamaño de imagen de 640x480 px, y se moverá el servomotor que posicionará de forma paralela al suelo el eje normal al plano de la cámara. Tras el reconocimiento de las señales, se volverá a la configuración inicial y el programa continuará con su flujo normal.

En este caso, si no se detecta ninguna señal, el vehículo continuará recto.

Cabe señalar que, aunque la tarea de reconocimiento de señales indicará la acción que deberá efectuar el vehículo, esta solo se llevará a cabo si el vehículo sigue situado en el cruce. Esto permitirá que, a la hora de realizar pruebas, el vehículo pueda ser cambiado de posición durante el reconocimiento de señales y, si este es situado en un tramo de trayecto sin intersección, se asegurará que se lleve una acción de control acorde al tramo donde esté.

La nueva funcionalidad del programa se presenta a continuación en la Figura 6.16.

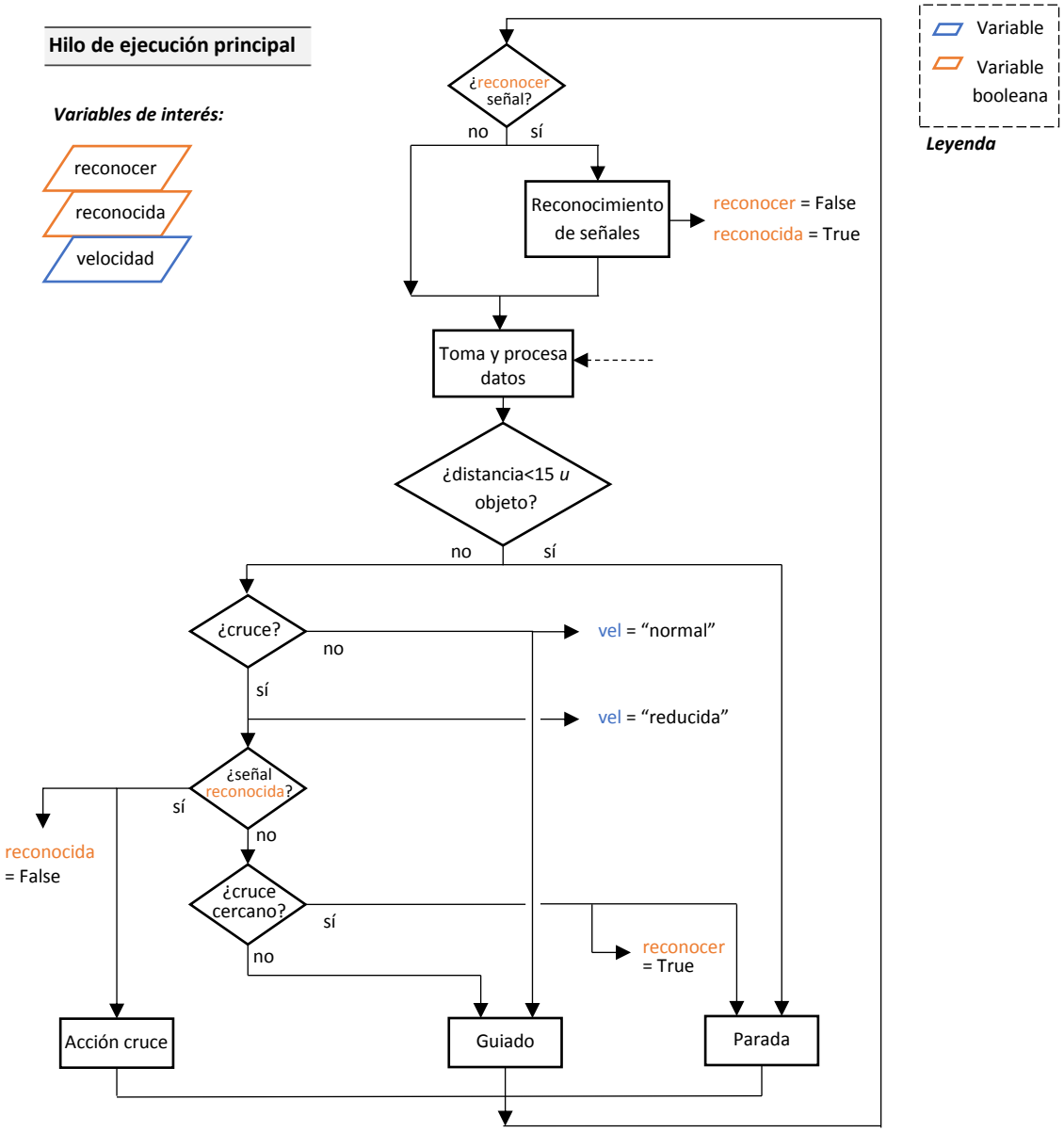


Figura 6.16. Demostrador 3, diagrama de flujo. Reconocimiento de señales.

CAPÍTULO 7

TOLERANCIA A FALLOS

Pese a que los demostradores presentados en los capítulos anteriores (con los distintos algoritmos de reconocimiento y entornos) proporcionan una infraestructura amplia y variada en la que se podrán llevar a cabo numerosos ensayos de tolerancia a fallos del proyecto DINAMOS, cabe plantearse si con los sistemas disponibles del vehículo se puede mejorar la calidad de esta, dadas las limitaciones que presentaban.

En este capítulo, tras presentar el tipo de diseño de sistemas tolerante a fallos y justificar su uso en la industria, se dotará sistema de guiado del vehículo de pruebas de unas capacidades básicas de tolerancia a fallos identificando aquellas tareas y subsistemas que se beneficiarán de su uso.

7.1 INTRODUCCIÓN A LA TOLERANCIA A FALLOS

A la hora de diseñar un sistema en cualquier campo de la ingeniería, se deben tener en cuenta una gran cantidad de aspectos. Si el sistema va a ser destinado a ofrecer un servicio, una de las principales cuestiones radicará en cómo deberá actuar dicho sistema en presencia de fallos en alguno de sus componentes o subsistemas. Si el servicio que ofrece involucra la integridad y seguridad de personas o requiere de una operación sin interrupciones, se hace indispensable el uso de diseños tolerantes a fallos. En efecto, este tipo de diseños garantiza que, si se produce un funcionamiento defectuoso de alguna parte del sistema, el conjunto podrá seguir ofreciendo un servicio que, aunque podrá presentar capacidades reducidas, garantizará la continuidad de la operación y la seguridad de los agentes que hacen uso de dicho servicio.

La tolerancia a fallos se consigue generalmente mediante la disposición de múltiples elementos integrados en el sistema que desempeñan una función idéntica o similar (Avizienis, Laprie, Randell, & Landwehr, 2004). Esto se conoce como redundancia espacial. Si el sistema que se diseña incluye el manejo de información variable en el tiempo (como es el caso de los sistemas informáticos), se cuenta además con redundancia temporal, que se basa en el reconocimiento de cierta información como válida a través de la repetición de esta en el tiempo (Algirdas Avizienis, 1976), (Avizienis, Laprie, Randell, & Landwehr, 2004).

En el ámbito de los automóviles, dada la gran cantidad de subsistemas de que disponen, se deberán efectuar diseños tolerantes a fallos en multitud de estos para garantizar la seguridad de los ocupantes en las situaciones más adversas. En el caso de los coches tradicionales, este tipo de diseño ha sido ampliamente estudiado y aplicado por la industria automovilística a lo largo de los años. Sin embargo, con el auge de los vehículos autónomos, dada la seguridad que estos deben ofrecer y sabiendo que los algoritmos de visión artificial pueden presentar fallos en múltiples situaciones, se hace necesario el

desarrollo de nuevos diseños tolerantes a fallos para los subsistemas encargados de la navegación autónoma del vehículo y percepción del entorno.

Para el caso concreto de este trabajo, como se ha expuesto en los primeros capítulos, se requerirá establecer un prototipo funcional que presente ciertas capacidades de tolerancia a fallos en el sistema de guiado autónomo, a partir del cual se desarrollarán y pondrán en práctica posteriormente otras técnicas y métodos de evaluación de tolerancia a fallos hardware con capacidad de adaptación, en el ámbito del proyecto DINAMOS.

El tipo de diseño tolerante a fallos que se implementará vendrá determinado por los componentes de que dispone el sistema y la relación que existe entre ellos. En este caso, contando con sensores de distancia ultrasónicos y una cámara, tras identificar en qué situaciones de percepción se obtendrá información relevante a través de estos canales simultáneamente, se podrá establecer un tipo de tolerancia a fallos basado en redundancia espacial. En concreto, se tratará de una redundancia hardware funcional, ya que una misma tarea de percepción se llevará a cabo por sensores de naturaleza distinta. Además, también se podrá hacer uso de la redundancia temporal comparando la información obtenida por los sensores en instantes de tiempo consecutivos.

7.2 APLICACIÓN AL VEHÍCULO DE PRUEBAS

Antes de identificar las tareas sobre las que se aplicará el diseño tolerante a fallos, será necesario señalar una serie de aclaraciones respecto a las capacidades de los sensores que se usarán para la redundancia que permitirán simplificar el desarrollo del sistema sin perder funcionalidad, al mismo tiempo que se simularán capacidades más avanzadas de un vehículo autónomo.

Disponiendo de dos tipos de sensores distintos (de distancia y óptico –o cámara), se podría plantear un diseño redundante cubriendo posibles fallos tanto de la cámara como los sensores ultrasónicos. Sin embargo, la fiabilidad de la información proporcionada por la cámara depende directamente de la capacidad del software dedicado a procesar e interpretar las imágenes obtenidas. Vista en el capítulo anterior la dificultad que supone un reconocimiento libre de errores de un entorno complejo y variable, resulta poco prudente ofrecer un servicio de guiado autónomo basado tan solo en visión. En el caso de un coche autónomo comercial, se dispondrá de software más avanzado para interpretar el entorno, pero de igual manera se presentarán también entornos más adversos (como por ejemplo, el caso de niebla en la carretera).

Respecto a los sensores de distancia, suponen una fuente de información más simple y directa, puesto que la señal obtenida no requiere un procesamiento adicional para extraer un significado útil. Además, aunque es cierto que las medidas de los sensores usados para este trabajo no están exentas de errores, estos se presentan con muy poca frecuencia y, en general, no impiden el correcto funcionamiento del sistema. Así, en este caso se asumirá que los sensores de distancia proporcionan siempre información fiable y se podrá hacer uso de ella para efectuar un diseño redundante del guiado. Esto equivaldrá a disponer en un vehículo autónomo comercial de sensores con mayores prestaciones e incluso de múltiples de estos para una misma zona de medida que garanticen la obtención de una medida fiable.

Por otra parte, puesto que el rango de estos sensores con relación al tamaño del vehículo es muy elevado, (4 metros frente a 15 cm, respectivamente), se podrán simular con ellos dos sistemas de medida de distancia de un vehículo autónomo real. Para distancias largas, representarán la disposición de un radar y, para cortas distancias, de sensores ultrasónicos.

Conocida la base sobre la que se parte para el diseño tolerante a fallos del sistema perceptivo, se estudiarán las tareas de reconocimiento que se beneficiarán de la redundancia hardware funcional y se plantearán las soluciones que garantizarán la calidad y fiabilidad de las acciones de guiado que efectuará el vehículo ante determinados escenarios.

Como se ha razonado, el primer paso será identificar cuáles de estas tareas podrán adquirir información de distintas fuentes, en este caso de la cámara y de los sensores ultrasónicos.

Partiendo del caso general real, en el propio guiado de la trayectoria de un vehículo autónomo pueden intervenir múltiples fuentes que proporcionan redundancia en la información, como son el posicionamiento GPS, el sistema de navegación inercial, la visión artificial, e incluso los sensores de distancia si el entorno dispone de objetos fijos en las inmediaciones de la vía. En el caso de este trabajo tan solo se cuenta con los dos últimos y, puesto que el trayecto del entorno no cuenta con barreras u objetos que marquen los límites de la vía, la única acción prudente que podría tomar el vehículo bajo condiciones de fallo de la cámara sería la detención.

Atendiendo a situaciones más específicas, se presentan las tareas de percepción definidas en el capítulo anterior.

- La **detección de objetos** admitirá evidentemente el uso de la visión y de los sensores de distancia. Puesto que el reconocimiento de objetos por visión presentaba claras limitaciones en ciertos casos, se justifica la necesidad de un sistema redundante.

- En la **identificación de cruces** podrán intervenir los sensores ultrasónicos si la intersección presenta señales de tráfico. Aunque en el caso de un vehículo real los sensores de distancia no podrían aportar la información necesaria para determinar la presencia de señales, este contaría, como se ha mencionado, con otros sistemas como la localización GPS o múltiples cámaras que tendrían la misma finalidad, esto es, identificar la presencia de una intersección. Así, aunque el método sea distinto al de un caso real, se aprovechará la redundancia que proporciona la detección de señales con los sensores de distancia. También cabrá esperar mejoras en esta tarea dado que en algunos casos el algoritmo de visión podía presentar resultados no deseados.

- El caso de la **detección de señales** de tráfico estará ligado al de la identificación de la intersección, ya que los sensores de distancia proporcionarán información sobre la localización de la señal en ambas situaciones. Como se verá, esta información podrá ser aprovechada si presenta incongruencias respecto a la obtenida con la visión.

Así pues, se decide aplicar el diseño tolerante a fallos a estos tres últimos casos específicos. A continuación se plantea la casuística de fallos que podrá presentar el sistema de visión, para proponer cómo el diseño tolerante a fallos los podrá mitigar, y de qué modos actuará y se recuperará el vehículo de estos. Finalmente se integrarán en el programa todas las funcionalidades diseñadas mediante un único módulo de Python.

7.2.1 TAREA DE RECONOCIMIENTO DE OBJETOS

Como se expuso en el capítulo anterior, la principal limitación del algoritmo de detección de objetos es la obtención de falsos negativos cuando los objetos presentan colores similares a los del modelo de

la pista. También se presentaban con menos frecuencia situaciones en que se detectaba por error la presencia de un objeto, en condiciones de iluminación desfavorables. En este último caso las consecuencias no serían tan negativas, ya que el vehículo actuaría del lado de la seguridad, reduciendo la velocidad. Sin embargo, en el caso de no detectar el objeto, los resultados en un vehículo real podrían ser catastróficos. Se hace evidente, pues, el uso de un sistema redundante que permita ofrecer unas capacidades mínimas de operación segura tras el fallo de la detección de un objeto por visión.

Cabría preguntarse, dados los casos múltiples en que la visión puede fallar al detectar un objeto y la certeza del funcionamiento correcto de los sensores de distancia, si esta tarea de reconocimiento se podría llevar a cabo solamente con estos últimos sensores. En el caso de un vehículo real esto se refuta rápidamente, ya que el sistema de visión proporciona información adicional útil, como la localización exacta en la carretera o el tipo de objeto de que se trata. En el caso del vehículo robótico de pruebas, aunque no se extraerá información relevante más allá de la posición del objeto en la imagen, resultará útil disponer de la detección por visión para ensayar la capacidad del sistema tolerante a fallos de actuar correctamente, y para proponer modos de actuación y recuperación ante los fallos de la visión que simulen un caso real.

Con esto, en primer lugar será necesario definir el rango de medida de los sensores ultrasónicos. Aunque estos tienen un alcance de hasta cuatro metros, no tendrá sentido usar distancias tan grandes para un entorno de pruebas reducido en tamaño. Además, puesto que el trayecto discurrirá cerca de muebles fijos en el entorno (como mesas o sillas), se deberá ajustar la distancia para evitar que se perciban erróneamente objetos situados fuera del circuito. Se escogerá una distancia máxima de medida que se encontrará dentro de la zona detectada por visión, que será especificada después de definir los requisitos de diseño redundante de las tareas de reconocimiento de cruces y señales.

Se establecerá también otra distancia inferior bajo la cual se define una zona de parada. En cualquier circunstancia, si un objeto es detectado en esta zona provocará la detención del vehículo.

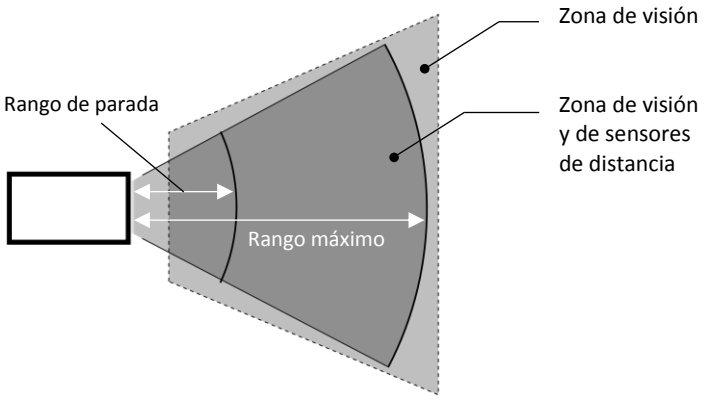


Figura 7.1. Rangos de medida del sensor ultrasónico

Bajo condiciones normales de operación en que el objeto es detectado tanto por visión como por algún sensor de distancia, el vehículo reducirá la velocidad hasta que el objeto quede en la zona de parada, en que el vehículo se detendrá. Si el objeto se retira de la pista o se desplaza hacia adelante saliendo de la zona de parado del sensor de distancia, el vehículo reanudará la marcha de forma normal.

En condiciones de fallo de la visión, se llevará a cabo un control de la velocidad del vehículo mediante redundancia temporal de la señal recibida por los sensores ultrasónicos. Con una primera medida de

distancia dentro del rango máximo definido (inclusive la zona de parado) sin confirmación visual del objeto, el vehículo reducirá la velocidad preventivamente. Esto permitirá al programa efectuar otra toma por la cámara para asegurarse de que el sistema de visión no está funcionando correctamente. Si efectivamente la cámara sigue sin proporcionar un reconocimiento válido del objeto, se podrán presentar dos casos distintos, configurables en el programa. Para un caso conservador, se detendrá el vehículo tras la obtención de dos medidas consecutivas (bien de un mismo sensor o alternando de uno al otro) dentro del rango máximo. En el otro caso, la detención se producirá cuando se hayan dado dos o más medidas consecutivas dentro del rango máximo y la última en particular sea inferior a la distancia que define la zona de parada. En este último caso, el vehículo operará con la velocidad reducida en todo momento hasta la detención.

En cualquier caso, si el vehículo se detiene sin haber reconocido el objeto por visión, este no volverá a reanudar la marcha hasta que se dé una señal a través del teclado del ordenador que estará conectado inalámbricamente a la Raspberry Pi. Esto simulará la detención del guiado autónomo y la cesión del control del vehículo al conductor en circunstancias de fallo de la visión.

7.2.2 TAREA DE RECONOCIMIENTO DE CRUCES Y SEÑALES DE TRÁFICO

Según lo expuesto en el capítulo anterior, aunque la identificación de cruces proporciona generalmente buenos resultados, podía presentar problemas frente a curvas muy pronunciadas y reflejos (dando lugar a falsos positivos) y, en menor medida, frente a desplazamientos elevados del vehículo respecto del centro de la pista (resultando en falsos negativos). En esta sección se tratará de solucionar el primer tipo de problemas haciendo uso de la redundancia espacial.

Para poder hacer uso de un diseño redundante en la identificación de intersecciones, como se ha mencionado, deberá presentarse una señal de tráfico en el cruce que aportará información adicional durante la tarea de reconocimiento. En este caso, a diferencia de el del objeto, la redundancia solo permitirá confirmar la presencia de un cruce detectado por la visión. En otras palabras, si tan solo se dispone de la información de los sensores ultrasónicos de la presencia de la señal sin la correspondiente identificación por visión del trazado de la intersección, no se podrá verificar que el vehículo se halla ante un cruce y este no podrá pasar a efectuar la tarea visual de clasificación de las señales.

Este tipo de redundancia permitirá descartar falsos positivos cuando se identifique por error un cruce con la visión. Sin embargo, también supondrá que en casos en que no haya señal en un cruce no se podrá garantizar la presencia de este, debido a la falta de la confirmación de los sensores de distancia. Por ello, se asumirá que las intersecciones sin señal dan preferencia de paso al carril del que viene el vehículo. Así, si un cruce es detectado por visión sin la confirmación de los sensores ultrasónicos, el vehículo tan solo reducirá la velocidad conforme a lo expuesto en el apartado de los demostradores del Capítulo 6.

Para el diseño del sistema redundante se partirá de la base definida en el caso de la detección redundante de objetos. En este caso, se podrá determinar con más exactitud la distancia máxima para la zona de medida de los sensores ultrasónicos. Para ello se coloca el vehículo a la máxima distancia del cruce a la que es capaz de identificarlo correctamente. En esta posición se toma la medida que proporcionan los sensores ultrasónicos. Se determina aproximadamente una distancia de 45 cm tras varios ensayos.

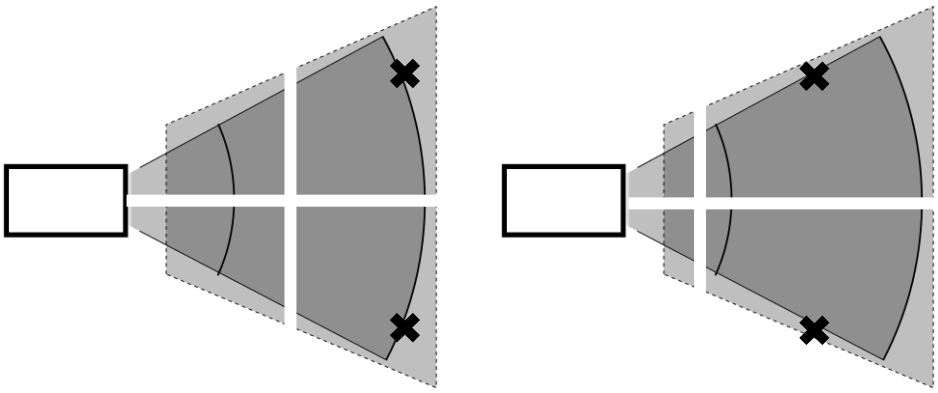


Figura 7.2. Señales de tráfico detectadas en el borde del rango máximo (izquierda), y no detectadas por estar situadas fuera del rango (derecha)

El modo de operación en este caso será el siguiente. Cuando se identifique un cruce por visión, el sistema encargado de la redundancia esperará la obtención de la medida dentro del rango definido de tan solo uno de los sensores ultrasónicos, que confirmará la presencia de una señal de tráfico a uno de los lados. Después de que esto ocurra, el vehículo seguirá avanzando hasta que se cumpla la condición de detención en cruces por cercanía expuesta en el capítulo anterior, tras lo cual, pasará a efectuar la tarea de reconocimiento de señales.

Es importante señalar que, aunque los sensores de distancia no podrán detectar las señales en todo el tramo en que la visión es capaz de identificar el cruce, como se muestra en la Figura 7.2, tan solo se requiere una medida de estos dentro de la zona máxima durante el reconocimiento visual de la intersección para que se pueda llevar a cabo la tarea de clasificación de señales de tráfico. Esto se conseguirá cambiando el estado de una variable booleana cuando el cruce es confirmado por uno de los sensores ultrasónicos, y dicha variable no volverá al estado original mientras la visión esté identificando el cruce, independientemente de que el sensor de distancia deje de detectar las señales.

Cabe también plantearse cómo deberá actuar el vehículo en el caso de que realmente se presente un objeto enfrente y este no sea identificado como tal sino como un cruce. En este caso, se diseña el programa para que, si se están tomando medidas dentro del rango definido con ambos sensores de distancia, la confirmación del reconocimiento del cruce no se llevará a cabo. En su lugar, se efectuará el procedimiento de disminución de velocidad preventiva y de parada basado en redundancia temporal explicado en el apartado del caso de los objetos.

Finalmente, si se reconoce y confirma el cruce correctamente y se inicia la tarea de clasificación de señales, se aplicará también un tipo de tolerancia a fallos simple. Tal y como se explicó en el capítulo anterior, la detección de señales podía no llevarse a cabo de forma correcta si el vehículo quedaba formando un cierto ángulo con el cruce, ya que estas quedaban fuera del campo de visión. Para solucionar esto, se puede aprovechar la información de la localización (izquierda o derecha) de la señal detectada con los sensores ultrasónicos. Así, según el lado en el que haya sido detectada la señal durante la identificación del cruce, se efectuará una rotación de la cámara para buscarla si no ha sido detectada en un primer intento. Se limita esta búsqueda a cuatro iteraciones de 15° de giro cada una, tras las cuales, si no se ha conseguido detectar, se considera que la señal u objeto que ha causado la confirmación de la presencia de un cruce queda fuera de rango, y el vehículo asumirá que debe seguir recto.

7.3 IMPLEMENTACIÓN

Tras definir el diseño que dotará de capacidades de tolerancia a fallos a cada una de las tareas, se propone un diagrama final de operación del sistema que integra todos los aspectos de redundancia y modos de actuación frente a fallos que se han tenido en cuenta en los apartados anteriores.

Este se implementará definiendo una clase de Python que tendrá como atributos los mostrados como "variables de interés" en la figura. El flujo de operación se llevará a cabo creando un método o función de la clase que tomará como parámetros las distancias obtenidas en la última iteración del bucle, así como también las variables booleanas *objeto* y *cruce* que indicarán el reconocimiento o no de estos en la última imagen obtenida. Las variables de interés booleanas son *previo*, que se encarga de la redundancia temporal; *confirm*, que indica si los sensores de distancia han detectado la señal mientras se está identificando el cruce por visión; *izq* y *der*, que determinan en qué lado ha sido detectada la señal mediante los sensores ultrasónicos, y *fallo*, que indica si se ha producido o no un fallo por visión.

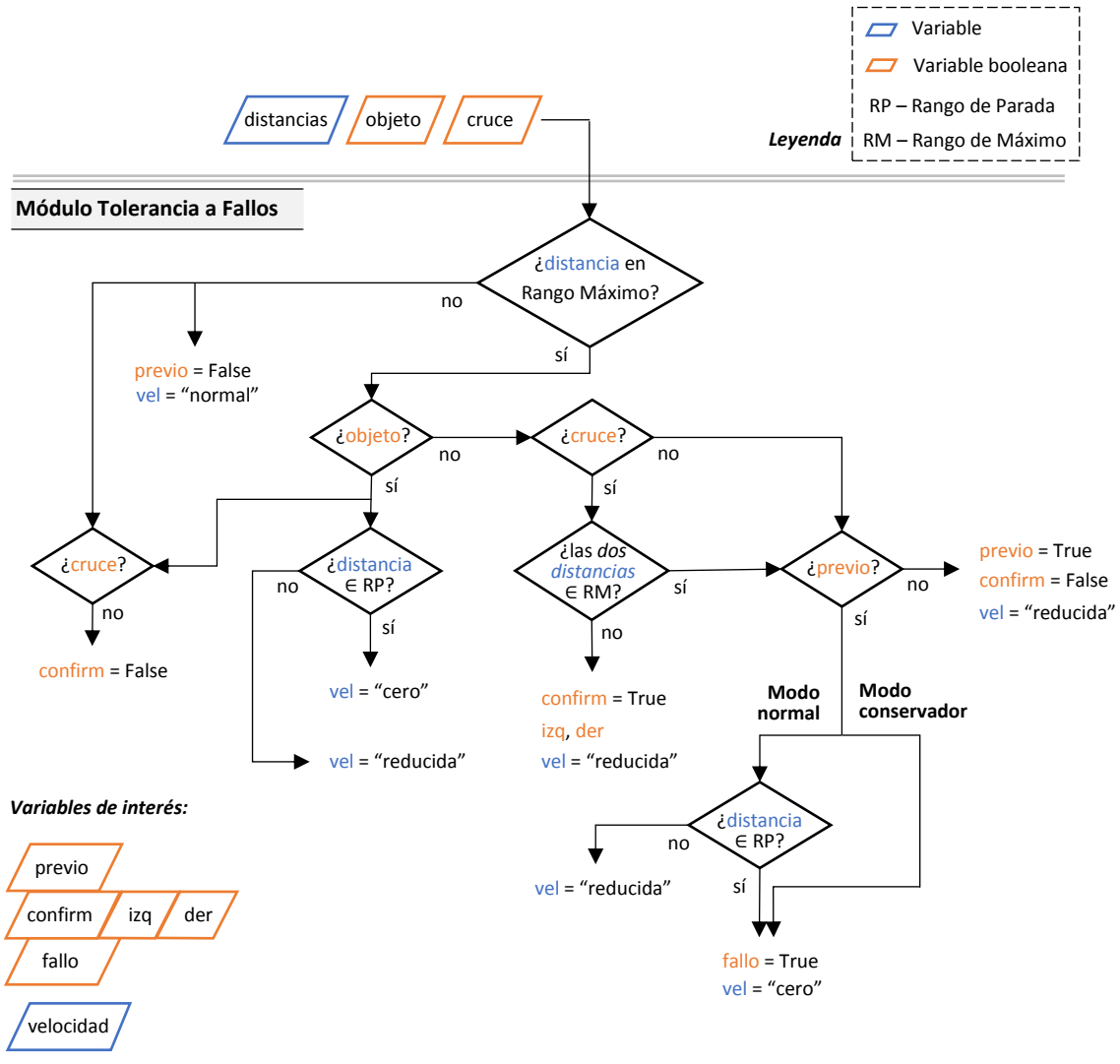


Figura 7.3. Módulo de tolerancia a fallos, flujo de operación

Integrando el módulo encargado de la tolerancia a fallos en el programa general y partiendo del diagrama del tercer demostrador, se tiene la siguiente figura que muestra el funcionamiento del hilo principal del programa.

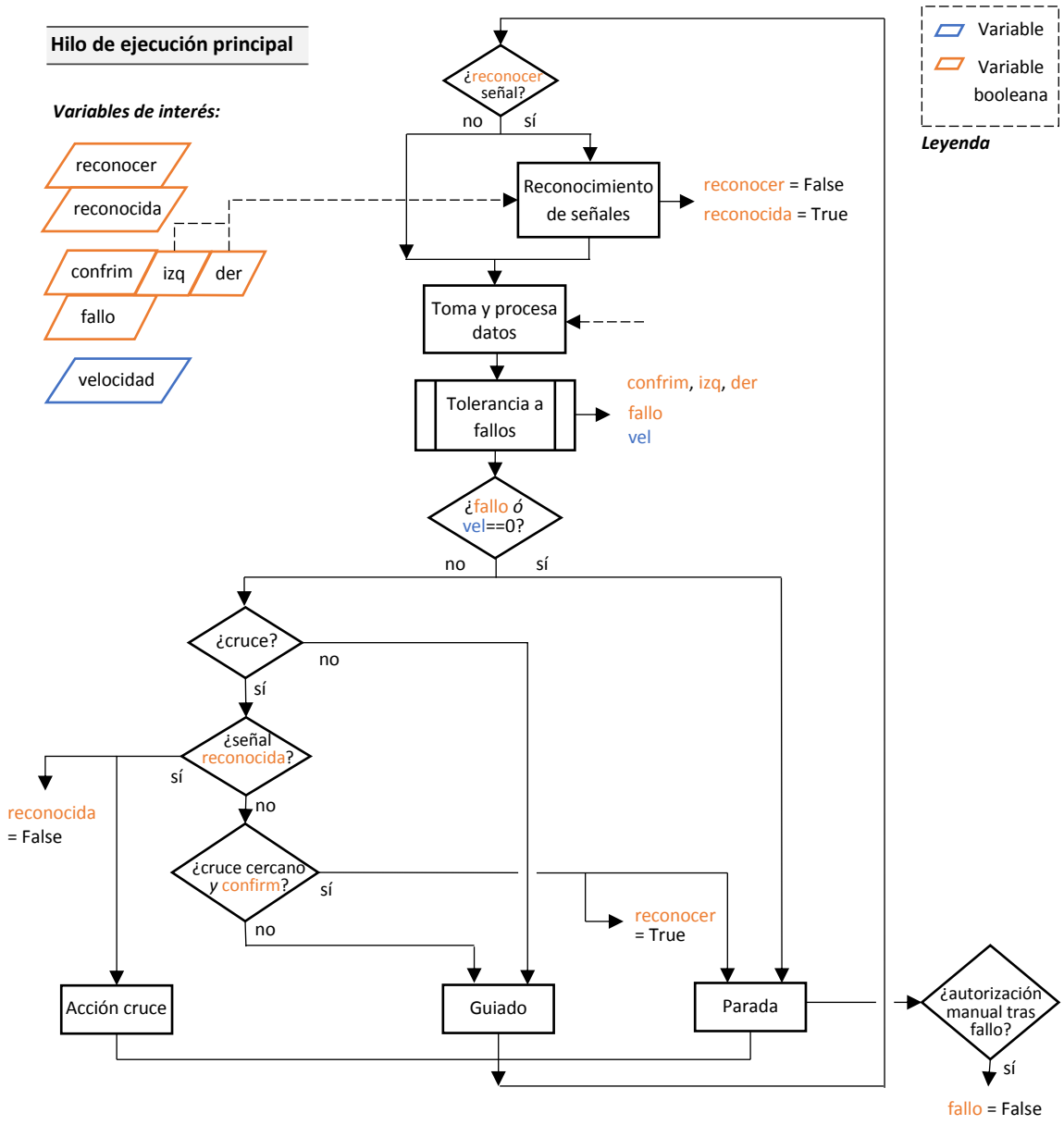


Figura 7.4. Demostrador 4, diagrama de flujo. Tolerancia a fallos

7.4 DISCUSIÓN

Los resultados obtenidos tras realizar múltiples pruebas en distintas situaciones se ajustan a los esperados durante el diseño de los sistemas redundantes. Por un lado, se ha podido comprobar que el vehículo se detiene y reanuda su marcha tras detectar correctamente con la visión un objeto que se desplaza enfrente, del mismo modo que se detiene hasta recibir la orden desde el teclado cuando no

ha sido capaz de obtener confirmación visual del objeto. Por otro lado, se han eliminado satisfactoriamente los casos de falsos positivos (aunque ya eran muy poco frecuentes) en la identificación de cruces bajo condiciones adversas de iluminación o en curvas cerradas. Finalmente, se ha asegurado la detección de las señales en los casos en que estas quedan fuera del campo visual de la cámara tras la detención del vehículo en la intersección.

Una de las principales limitaciones que presenta el sistema perceptivo es la incapacidad de identificar un objeto y un cruce simultáneamente de forma fiable, por la naturaleza distinta de los algoritmos de visión. Con el diseño actual, si un objeto se presenta en una intersección, el programa no será capaz de llevar a cabo la acción de reconocimiento de señales, y se comportará de la forma correspondiente al caso de un objeto. Esto es, avanzará tras él independientemente de si el trazado de la pista se corresponde o no con una intersección. Aunque esto limitará potencialmente la realización de cierto tipo de pruebas, a efectos prácticos no supondrá un gran inconveniente dada la gran variedad de casos de ensayo restantes.

Como se ha mencionado, los sensores ultrasónicos usados en este trabajo proporcionan, aunque con poca frecuencia, medidas falsas de corta distancia. Esto puede provocar la detención del vehículo en situaciones en las que no se presenta ningún objeto enfrente. Sin embargo, mediante la redundancia temporal implementada, se consigue reducir la frecuencia de aparición de este suceso a unos niveles despreciables a efectos prácticos ya que, en cualquier caso, se pueden identificar los momentos en que se presenta este problema.

CAPÍTULO 8

CONCLUSIONES Y TRABAJOS FUTUROS

8.1 CONCLUSIONES

En el presente trabajo se ha logrado alcanzar el objetivo principal del diseño y puesta en marcha de la plataforma de pruebas que permitirá realizar los primeros ensayos en el proyecto DINAMOS. Durante el desarrollo se han llevado a cabo diferentes tareas que han permitido cumplir con los objetivos específicos en mayor o menor medida, no sin presentar ciertas dificultades o limitaciones. Los principales aspectos a destacar se exponen a continuación.

La presencia general de reflejos en el suelo de pruebas ha condicionado considerablemente el tipo de algoritmos desarrollados en todas las fases del proyecto y, en general, ha dado lugar a soluciones con capacidades limitadas o conservadoras.

En el diseño del control de la dirección del guiado del robot, que se presenta como un proceso no lineal e inestable, se ha ajustado satisfactoriamente un controlador tipo PID haciendo uso del método empírico de Ziegler-Nichols. Se ha comprobado el carácter agresivo (dado por las oscilaciones del vehículo) de los controladores iniciales obtenidos con este método, especialmente para los tipos P y PD. Los resultados del controlador completo PID han sido los más cercanos al deseado y, tras ajustar manualmente mediante ensayos los parámetros de partida, ha sido posible mejorar la respuesta convenientemente.

También en el diseño del controlador, se ha determinado la importancia de disponer de un periodo de ejecución reducido, debido a falta de reactividad en el sistema que asociada a periodos elevados. Esto ha requerido limitar la complejidad y tiempo de procesamiento de los algoritmos de procesamiento de imágenes desarrollados para reconocer el entorno.

En cuanto a estos últimos, se ha adoptado una estrategia de división en las tareas de reconocimiento, en parte motivada por la limitación del campo de visión de la cámara que impide detectar señales y efectuar el guiado de forma simultánea. Abordando cada tarea de forma independiente, se consiguen optimizar los algoritmos para cada caso de reconocimiento aislado. Sin embargo, en casos en los que se requiere realizar dos tareas a la vez (como identificar una intersección y detectar un objeto), no se puede garantizar la fiabilidad de los resultados de estas.

Se ha mostrado la necesidad de emplear técnicas de análisis de imágenes más avanzadas para casos en que los patrones a identificar se presentan con mayor variabilidad de forma. Así, para la identificación de cruces se ha requerido un preprocesamiento más extensivo que para el detector de objetos y el clasificador de señales, en los que los modelos a reconocer presentaban características

más fácilmente diferenciables (como es el color distinto al del fondo de los objetos o la forma específica de la señal).

Por último, se ha demostrado la capacidad de un diseño tolerante a fallos básico, basado en redundancia hardware funcional, de corregir ciertas limitaciones de los algoritmos de reconocimiento, especialmente los falsos positivos para el caso de las intersecciones o falsos negativos para el caso de objetos y señales. Sin embargo, este diseño no ha permitido solucionar el problema de reconocimiento simultáneo de cruces y objetos. Por tanto se concluye que, para llevar a cabo esta tarea de forma satisfactoria, es necesario un cambio en el diseño del circuito o en el diseño base de los sistemas de percepción, ya sea en el hardware mediante la adición de nuevos sensores o en el software usando algoritmos más complejos.

8.2 TRABAJOS FUTUROS

Atendiendo a las limitaciones expuestas en este capítulo y a lo largo del trabajo, se plantean en este apartado una serie de mejoras de aplicación diversa en los distintos sistemas del robot móvil.

Para el tratamiento de reflejos, dada la gran variedad de formas en los que se pueden presentar en una imagen, no existe una sola función de OpenCV que pueda hacerse cargo de este problema. Sin embargo, la disminución del exceso de brillo en superficies especulares ha sido considerada ampliamente en la comunidad de visión por computador (Alessandro Artusi, 2011) (Falak Shah, 2016) y, aunque no existe un método universal, se propone para este caso el estudio de diferentes técnicas, la elección de aquellas que se ajustan mejor en este ámbito y su posterior implementación en el programa haciendo uso de distintas funciones de OpenCV.

Una vez eliminados o reducidos los reflejos, se podría obtener información sobre el trazado de la pista de forma más fiable. Esto, junto con el empleo de sensores adicionales como tacómetros o acelerómetros y giróscopos, permitiría efectuar un control predictivo del guiado del vehículo considerando la cinemática de este y las características de la pista. Además, esto abriría nuevas puertas en el diseño tolerante a fallos del sistema de guiado, ya que se podrían efectuar maniobras controladas de evacuación de la vía en casos en que la cámara dejase de capturar imágenes correctamente.

Dadas las limitaciones que supone el uso de una sola cámara parcialmente dirigida hacia el suelo, se proponen dos soluciones distintas. Por un lado, se considera el uso de dos cámaras, de modo que una se sitúe en una posición similar a la actual y cuya tarea principal sea el guiado y la identificación de intersecciones, y la otra se dirija hacia enfrente para captar la información de las señales. Por otro lado, empleando una cámara con un mayor campo de visión, se podría obtener una sola imagen que captara las dos zonas de interés. Esta última opción, aunque podría resultar más costosa en términos económicos, sería la más conveniente, puesto que la captura de video por dos canales podría disminuir notablemente el rendimiento de la Raspberry Pi.

Por último, se considera un sistema de reconocimiento de imágenes que unifique distintas tareas. Una propuesta se basa en el uso de una cámara con mayor ángulo de visión que permita identificar la presencia de una intersección no solo por la forma del trazado de la pista sino también por la detección simultánea de señales de tráfico con visión. En este caso, si se presentase un objeto enfrente ocultando parte del cruce, se podría determinar independientemente de la información de los sensores de distancia la presencia del objeto y de la intersección.

CAPÍTULO 9

BIBLIOGRAFÍA

Alessandro Artusi, F. B. (2011). A Survey of Specularity Removal Methods. *Computer Graphics Forum*.

Algirdas Avizienis. (1976). Fault-Tolerant Systems. *IEEE Transactions on Computers*, 1307-1308.

Astrom, K. J. (2009). *Control PID Avanzado.* Prentice Hall.

Avizienis, A., Laprie, J., Randell, B., & Landwehr, C. (2004). Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Transactions on Dependable and Secure Computing*, 11-33.

Falak Shah, P. S. (2016). Specularity removal for robust road detection. *IEEE Region 10 Conference (TENCON)*.

HC-SR04 Raspberri Pi. (s.f.). Obtenido de https://electrosome.com/hc-sr04-ultrasonic-sensor-raspberry-pi/

Rosebrock, A. (2015). *How to install OpenCV 3 on Raspbian Jessie*. Obtenido de http://www.pyimagesearch.com/2015/10/26/how-to-install-opencv-3-on-raspbian-jessie/

Rosebrock, A. (2016). *Unifying picamera and cv2.VideoCapture into a single class with OpenCV*. Obtenido de http://www.pyimagesearch.com/2016/01/04/unifying-picamera-and-cv2-videocapture-into-a-single-class-with-opencv/

Rosebrock, A. (s.f.). *pyimagesearch*. Obtenido de http://www.pyimagesearch.com

Salcedo, J. V. (2016). Apuntes de Tecnología Automática. Valencia: UPV.

SunFounder. (2016). *Código fuente robot móvil*. Obtenido de https://github.com/sunfounder/Sunfounder_Smart_Video_Car_Kit_for_RaspberryPi

SunFounder. (2016). *Instrucciones de montaje del robot móvil.* Obtenido de https://www.sunfounder.com/learn/download/U21hcnRfVmlkZW9fQ2FyX2Zvcl9SYXNwYmVycnlfUGlfMTYxMDI3LnBkZg==/dispi

SunFounder. (s.f.). *Kit del robot móvil*. Obtenido de https://www.SunFounder.com/learn/Smart-Video-Car-for-Raspberry-Pi/36-40-smart-video-car.html

Yannick Benezeth, P.-M. J. (2012). Comparative study of background subtraction. *Inria*, 3-26.

Ziegler–Nichols method. (s.f.). Obtenido de https://en.wikipedia.org/wiki/Ziegler%E2%80%93Nichols_method

Parte II
Presupuesto

PRECIOS DE MANO DE OBRA

El trabajo ha sido realizad por un Ingeniero Graduado en Tecnologías Industriales.

Código			Precio (€/h)
001	h	Ingeniero Graduado en Tecnologías Industriales	30,00

PRECIOS DE MATERIALES

Robot móvil – Estado inicial

Pese a que el robot cuenta con distintos componentes que pueden ser adquiridos de forma separada, se ha tenido en cuenta solamente el precio total de este, que incluye tanto la estructura como los componentes electrónicos básicos. Entre estos últimos se encuentran los actuadores (servomotores, motores de corriente continua) controladores, y alimentación.

Código			Precio unidad (€)
002	u	Robot móvil con componentes iniciales	87,11

Robot móvil – Ampliación

Estas son las partes adquiridas de forma independiente al robot.

Código			Precio unidad (€)
003	u	Raspberry Pi 2	39,95
004	u	Cámara Raspberry Pi	34,95
005	u	Sensor ultrasónico	2,85
006	u	Resistencia de 1000 Ω	0,18
007	u	Regleta de conexión de 12 terminales	1,50
008	u	Cable Arduino macho-hembra*	0,072
009	u	Cable Arduino hembra-hembra*	0,072

**Un juego de 40 unidades cuesta 2,89 €.*

Entorno del robot

En este ámbito tan solo se tiene en cuenta la cinta aislante del circuito, ya que el cartón y el papel dedicados a las señales de tráfico ya se disponían con anterioridad.

Código			Precio (€/m)
010	m	Cinta aislante blanca**	0,045

***Un rollo de 20 metros de cinta aislante cuesta 90 céntimos.*

Infraestructura de comunicación y operación remota

Estos elementos han sido proporcionados por el laboratorio de Tolerancia a Fallos, y su precio no se incluye en el presupuesto final.

Código			Precio unitario (€)
011	u	Módem	0,00
012	u	Ordenador con Windows 7	0,00

PRECIOS UNITARIOS

Nº Actividad		Descripción de las unidades de obra	Medición	Precio	Importe
01		**DISEÑO DE PARTIDA DE LA PLATAFORMA DE TRABAJO**			
01.01	u	Montaje del hardware	1,00	165,02	445,52
01.02	u	Instalación de software e infraestructura de comunicación	1,00	459,00	459,00
01.03	u	Diseño y trazado del circuito	1,00	153,45	153,45
01.04	u	Prueba de funcionamiento y calibrado inicial	1,00	612,00	612,00
			Precio Total Unidad de Obra 01		**1669,97 €**
03		**DISEÑO Y DESARROLLO DE LA PLATAFORMA DE PRUEBAS**			
03.01	u	Diseño del controlador del guiado del vehículo	1,00	1224,00	1224,00
03.02	u	Diseño e implementación de algoritmos de visión	1,00	1836,00	1836,00
03.03	u	Diseño del sistema de guiado tolerante a fallos	1,00	1224,00	1224,00
			Precio Total Unidad de Obra 03		**4284,00 €**

PRECIOS DESCOMPUESTOS

Nº Actividad	Código		Descripción	Rendimiento	Precio	Importe
01			**DISEÑO DE PARTIDA DE LA PLATAFORMA DE TRABAJO**			
01.01	**U01.01**	u	**MONTAJE DEL HARDWARE**. Incluye conexionado de las distintas partes a la Raspberry Pi y montaje de los divisores de tensión para los sensores ultrasónicos. También incluye el estudio previo de los componentes electrónicos del robot.			
	001	h	Ingeniero Graduado en Tecnologías Industriales	10,00	30,00	300,00
	002	u	Robot móvil con componentes iniciales	1,00	87,11	87,11
	003	u	Raspberry Pi 2	1,00	39,95	39,95
	004	u	Sensor ultrasónico	2,00	2,85	5,70
	005	u	Resistencia de 1000 Ω	6,00	0,18	1,08
	006	u	Regleta de conexión de 12 terminales	1,00	1,50	1,50
	007	u	Cable Arduino macho-hembra	10,00	0,072	0,72
	008	u	Cable Arduino hembra-hembra	10,00	0,072	0,72
				Coste total de mano de obra y material		436,78
		%	Costes directos complementarios	0,02	436,78	8,74
				Coste total de Unidad de Obra		**445,52 €**
01.02	**U01.02**	u	**INSTALACIÓN DE SOFTWARE E INFRAESTRUCTURA DE COMUNICACIÓN**. Incluye instalación del sistema operativo para la Raspberry Pi y las librerías OpenCV, NumPy, de control de los motores y de acceso a la cámara. También la instalación del módem y la conexión remota VNC del Ordenador a la Raspberry Pi			
	001	h	Ingeniero Graduado en Tecnologías Industriales	15,00	30,00	450,00
	011	u	Módem	1,00	0,00	0,00
	012	u	Ordenador con Windows 7	1,00	0,00	0,00
				Coste total de mano de obra y material		450,00
		%	Costes directos complementarios	0,02	450,00	9,00
				Coste total de Unidad de Obra		**459,00 €**

Nº Actividad	Código		Descripción	Rendimiento	Precio	Importe
01.03	**U01.03**	u	**DISEÑO Y TRAZADO DEL CIRCUITO**. Incluye diseño del circuito, trazado de este con cinta aislante, selección de señales de tráfico y montaje de estas.			
	001	h	Ingeniero Graduado en Tecnologías Industriales	5,00	30,00	150,00
	010	m	Cinta aislante blanca	10,00	0,045	0,45
			Coste total de mano de obra y material			150,45
		%	Costes directos complementarios	0,02	150,45	3,00
			Coste total de Unidad de Obra			**153,45 €**
01.04	**U01.04**	u	**PRUEBA DE FUNCIONAMIENTO Y CALIBRADO INICIAL**. Incluye modificación de los programas de control de los actuadores y de los sensores ultrasónicos, así como las pruebas de funcionamiento de todos estos y el calibrado inicial.			
	001	h	Ingeniero Graduado en Tecnologías Industriales	20,00	30,00	600,00
			Coste total de mano de obra y material			600,00
		%	Costes directos complementarios	0,02	600,00	12,00
			Coste total de Unidad de Obra			**612,00 €**
02			**DISEÑO DE PARTIDA DE LA PLATAFORMA DE PRUEBAS**			
02.01	**U03.01**	u	**DISEÑO DEL CONTROLADOR DEL GUIADO DEL VEHÍCULO**. Incluye estudio inicial de las alternativas, selección, análisis de la alternativa escogida, diseños preliminares y diseño final del controlador PID			
	001	h	Ingeniero Graduado en Tecnologías Industriales	40,00	30,00	1200,00
			Coste total de mano de obra y material			1200,00
		%	Costes directos complementarios	0,02	1200,00	24,00
			Coste total de Unidad de Obra			**1224,00 €**

Nº Actividad	Código		Descripción	Rendimiento	Precio	Importe
02.02	**U03.02**	u	**DISEÑO E IMPLEMENTACIÓN DE ALGORITMOS DE VISIÓN**. Incluye identificación y análisis de las tareas de reconocimiento visual que debe realizar el vehículo, propuesta y diseños preliminares de algoritmos y desarrollo y verificación experimental de estos.			
	001	h	Ingeniero Graduado en Tecnologías Industriales	60,00	30,00	1800,00
				Coste total de mano de obra y material		1800,00
		%	Costes directos complementarios	0,02	1800,00	36,00
				Coste total de Unidad de Obra		**1836,00 €**
02.03	**U03.03**	u	**DISEÑO DEL SISTEMA DE GUIADO TOLERANTE A FALLOS**. Incluye estudio de la aplicabilidad de un diseño redundante de percepción usando cámara y sensores de distancia, identificación de las tareas que lo admiten, desarrollo del sistema y verificación experimental.			
	001	h	Ingeniero Graduado en Tecnologías Industriales	40,00	30,00	1200,00
				Coste total de mano de obra y material		1200,00
		%	Costes directos complementarios	0,02	1200,00	24,00
				Coste total de Unidad de Obra		**1224,00 €**

PRESUPUESTO

Nº Actividad		Importe
01	Diseño de partida de la plataforma de trabajo	1669,97
02	Diseño y desarrollo de la plataforma de pruebas	4284,00

Presupuesto de Ejecución Material	5953,97 €
13% Gastos Generales	774,02 €
6% Beneficio Industrial	357,24 €
Presupuesto de Ejecución por Contrata	7085,22 €
I.V.A. 21%	1487,90 €
Presupuesto Base de Licitación	8573,12 €

Asciende el presente presupuesto a la expresada cantidad de

OCHO MIL QUINIENTOS SETENTA Y TRES EUROS CON DOCE CÉNTIMOS