



Universidad
de Cádiz

ESCUELA SUPERIOR DE INGENIERÍA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

OpenTSR:
Vehículo reconocedor de señales de tráfico

Manuel López Urbina

11 de septiembre de 2012



ESCUELA SUPERIOR DE INGENIERÍA

INGENIERO TÉCNICO EN INFORMÁTICA DE SISTEMAS

OpenTSR:
Vehículo reconocedor de señales de tráfico

- Departamento: Ingeniería en Automática, Electrónica, Arquitectura y Redes de Computadores
- Director del proyecto: Arturo Morgado Estévez
- Autor del proyecto: Manuel López Urbina

Cádiz, 11 de septiembre de 2012

Fdo: Manuel López Urbina

Agradecimientos

Este proyecto significa la culminación de mi carrera, por lo que me gustaría dedicárselo a todas las personas que me han ayudado a conseguir acabarla.

En primer lugar me gustaría agradecerle a mi familia el apoyarme y ayudarme durante estos años, y el esfuerzo que han hecho para que yo haya podido llegar a este momento.

Agradecimientos a D. Arturo Morgado por su ayuda y dedicación durante la realización y dirección de este proyecto, así como su carácter amable y servicial que han hecho más ameno el trabajo realizado.

También me gustaría agradecérselo a mis compañeros, con los que tantos ratos inolvidables he pasado, y que tanto me han ayudado, entre ellos, agradecimientos a Noelia Sales Montes por la elaboración del logotipo del proyecto y a Andrés Francisco Aparicio por proporcionar un nombre al mismo. A Moisés Gautier Gómez por ser un excelente compañero tras numerosas horas de trabajo en el aula de robótica.

Por último quiero dedicarle este proyecto a todos los estudiantes de informática, en especial a todos los amantes del fascinante mundo de la robótica, a los que espero que mi trabajo les sea de utilidad.

Licencia

Este documento ha sido liberado bajo Licencia GFDL 1.3 (GNU Free Documentation License). Se incluyen los términos de la licencia en inglés al final del mismo.

Copyright (c) 2012 Manuel López Urbina.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Índice general

Índice general	v
Índice de figuras	ix
Índice de tablas	xiii
1. Introducción	1
1.1. Introducción y antecedentes	1
1.2. Objetivos	2
1.3. Acerca de este documento	3
2. Conceptos básicos	5
2.1. Vehículo Autónomo	5
2.2. Inteligencia artificial	5
2.3. Visión artificial	5
2.4. Reconocimiento de patrones	6
2.4.1. Aplicaciones	7
2.5. Modelos de color	8
2.5.1. Modelo RGB	8
2.5.2. Modelo HSV	9
2.5.3. Uso	9
2.5.4. Características	10
2.5.5. Modelo CYMK	11
3. Herramientas utilizadas	13
3.1. Herramientas software	13
3.1.1. Licencia	15
3.2. Herramientas hardware	15
3.2.1. Vehículo SRV-1	16
3.2.2. Router	19
3.2.3. Cámara, receptor inalámbrico y capturadora de vídeo	20
3.2.4. Gamepad	22
3.2.5. Ordenador	22
4. Organización temporal	23

5. Montaje hardware	29
5.1. Colocación de la cámara en el vehículo	29
5.2. Conexión cámara - PC	30
5.3. Comunicaciones vehículo - PC	31
6. Desarrollo software	33
6.1. Metodología de desarrollo	33
6.2. Recolección de requisitos	34
6.2.1. Requisitos funcionales	34
6.2.2. Diagramas de casos de uso	35
6.2.3. Especificación de los casos de uso	36
6.2.4. Desarrollo	42
7. Software de reconocimiento	43
7.1. Primer prototipo	44
7.1.1. Obtención de la imagen	44
7.1.2. Extracción de características	46
7.1.3. Template matching	54
7.1.4. Problemas detectados	57
7.2. Segundo prototipo	58
7.2.1. Extracción de características	58
7.2.2. Clasificación	69
7.2.3. Problemas detectados	71
7.3. Tercer y último prototipo	72
7.3.1. Extracción de características	72
7.3.2. Clasificación	74
8. Software de control	77
8.1. Control por parte del usuario	77
8.1.1. Control desde teclado	78
8.1.2. Control mediante gamepad	80
8.2. Control automático	80
9. Interfaz gráfica	87
9.1. Elementos de la interfaz gráfica	87
10. Guía de usuario	91
10.1. Introducción	91
10.1.1. Objetivo de esta guía	91
10.1.2. Dirigido a	92
10.2. Obtener OpenTSR	92
10.3. Ingreso al sistema	92
10.4. Convenciones y estándares a utilizar	92
10.4.1. Convenciones del uso del teclado	93
10.5. Operaciones de la interfaz	96
10.6. Operaciones automatizadas	99
10.7. Montaje del sistema de comunicaciones	101

10.8. Configuración del router	104
10.9. Configuración del vehículo SRV-1	105
10.9.1. Configuración del MatchPort	105
10.9.2. Carga de batería del vehículo SRV-1	107
10.10 Instalación de la capturadora	108
11. Comentarios finales	111
11.1. Presupuesto	111
11.2. Conclusiones	112
11.3. Mejoras futuras	114
Anexos	117
.1. Instalación de OpenCV	117
.2. Instalación Qt y Qt Creator	120
.3. Instalación de SDL	123
Bibliografía	125
GNU Free Documentation License	127
1. APPLICABILITY AND DEFINITIONS	127
2. VERBATIM COPYING	129
3. COPYING IN QUANTITY	129
4. MODIFICATIONS	130
5. COMBINING DOCUMENTS	131
6. COLLECTIONS OF DOCUMENTS	132
7. AGGREGATION WITH INDEPENDENT WORKS	132
8. TRANSLATION	132
9. TERMINATION	133
10. FUTURE REVISIONS OF THIS LICENSE	133
11. RELICENSING	133
ADDENDUM: How to use this License for your documents	134

Índice de figuras

1.1. Logo OpenTSR ¹	2
2.1. Componentes básicos de un sistema de reconocimiento de patrones.	7
2.2. Modelo de color RGB.	9
2.3. Espacio de color HSV.	9
2.4. Modelo de color CMYK.	11
3.1. Logotipo de OpenCV	14
3.2. Logotipo de Qt	14
3.3. Logotipo de SDL	15
3.4. Logotipo de Qt Creator	15
3.5. Imagen de los diferentes elementos hardware empleados.	16
3.6. Imagen del vehículo SRV-1	17
3.7. Imagen del router empleado.	20
3.8. Imagen de la capturadora o conversor analógico/digital.	21
3.9. Imagen de la cámara y el receptor inalámbricos.	22
3.10.Imagen del gamepad USB utilizado.	22
4.1. Descomposición de las tareas implicadas en el desarrollo del proyecto. .	24
4.2. Diagrama de Gantt 1. Desarrollo del proyecto.	25
4.3. Diagrama de Gantt 2. Desarrollo del proyecto.	26
4.4. Diagrama de Gantt 3. Desarrollo del proyecto.	27
5.1. Vehículo SRV-1 tras el montaje de la cámara.	29
5.2. Esquema del sistema de recepción de imágenes desde la cámara al ordenador.	30
5.3. Visión del sistema de recepción de imágenes montado a falta de la conexión a corriente del receptor.	31
5.4. Comunicaciones vía Wi-Fi entre el ordenador y el vehículo robótico mediante un router.	32
5.5. Vista del sistema de comunicaciones entre el vehículo y el ordenador. .	32
6.1. Subsistemas existentes en el proyecto junto con el modelo de ciclo de vida utilizado para su desarrollo.	34
6.2. Diagrama de casos de uso.	36
7.1. Conjunto de señales de tráfico detectables por el sistema.	43
7.2. Interpretación de una imagen por parte de una cámara	44

7.3.	Possible imagen de entrada al sistema.	46
7.4.	Programa elaborado para la obtención de los valores del píxel pulsado.	47
7.5.	Resultado de aplicar una selección por color sobre una imagen.	51
7.6.	Imagen con una región definida de interés.	52
7.7.	Copia de una imagen mediante el uso de máscaras.	53
7.8.	Matrices de entrada y de salida para la técnica de comparación de plantillas.	55
7.9.	El valor de cada píxel de la posición (x, y) es procesado para obtener su característica de similitud entre la plantilla y el rectángulo resultante de tomar (x, y) como píxel situado en la esquina superior izquierda.	55
7.10.	Vista de la señal detectada en la imagen.	57
7.11.	Programa elaborado para la obtención de los valores del píxel pulsado.	59
7.12.	Fórmula para el cálculo del ángulo entre dos vectores.	63
7.13.	Vector de características de un elemento desconocido.	70
7.14.	Fórmula para el cálculo de la distancia euclídea, definida como la raíz cuadrada de las sumas de las diferencias de cada una de las componentes del vector al cuadrado.	70
7.15.	Proceso seguido para la extracción de los objetos interiores de una señal.	73
7.16.	Representación del número cinco en una matriz.	73
7.17.	Resultado de la transformación de la matriz V_1 en vector columna añadiendo las diferentes columnas de V_1 sucesivamente una bajo la otra en V_2 . El vector resultante posee en la primera posición el identificador de la clase, en el caso mostrado el carácter “5”.	74
7.18.	Identificación de los elementos interiores de la señal, el valor “5” y el valor “0”, determinándose el objeto como una señal de 50 km/h.	75
8.1.	Visualización del chasis del robot y su sistema de desplazamiento tipo tanque.	78
8.2.	Representación del funcionamiento del buffer creado para la captura de eventos múltiples.	79
8.3.	Fotografía de un circuito realizable por el vehículo	81
9.1.	Vista de la ventana principal.	88
9.2.	Vista de la ventana de información.	89
10.1.	Ventana principal de la aplicación.	92
10.2.	Ventana donde se visualiza la información relacionada con OpenTSR.	99
10.3.	Vista de los elementos conectados.	102
10.4.	Interruptor para búsqueda del canal de la cámara.	102
10.5.	Vista de OpenTSR en funcionamiento.	103
10.6.	Localización del interruptor con tres modos de funcionamiento del robot SRV-1.	103
10.7.	Imagen de la placa de configuración.	105
10.8.	Conexión de la placa al SRV-1.	106
10.9.	Menú de configuración del Matchport.	106
10.10.	Adaptador de corriente para realizar cargas de batería en el robot SRV-1. .	108

11.1.	Pasos para la elaboración de la matriz modelos.	112
11.2.	Comunicaciones vía WiFi entre el ordenador y el vehículo robótico mediante un router.	113
11.3.	Elementos implicados en la transmisión de imágenes desde la cámara hasta el ordenador.	114
4.	Chequeo de la configuración de OpenCV.	118
5.	Salida producida tras la ejecución del programa de ejemplo.	120
6.	Ventana de bienvenida del instalador.	121
7.	Solicitud de directorio para la instalación.	121
8.	Ventana para la visualización de los términos de licencia.	122
9.	Ventana de comienzo de instalación.	122
10.	Proceso de instalación de Qt SDK.	123
11.	Proceso de instalación finalizado.	123

Índice de tablas

6.1. Descripción de caso de uso: Conexión coche.	37
6.2. Descripción del caso de uso: Desconexión coche.	37
6.3. Descripción del caso de uso: Pilotaje tradicional.	38
6.4. Descripción del caso de uso: Pilotaje autónomo.	39
6.5. Descripción del caso de uso: Medir distancias.	40
6.6. Descripción del caso de uso: Activar láseres.	41
6.7. Descripción del caso de uso: Desactivar láseres.	42
7.1. Tabla con los datos extraídos para algunas de las clases (señales) a identificar.	69
10.1.Uso del ratón.	93
10.2.Controles del teclado para teclas simples.	93
10.3.Controles del teclado para las combinaciones de teclas.	94
10.4.Controles del gamepad.	96
10.5.Señales reconocidas por el vehículo junto con la acción realizada.	101

Capítulo 1

Introducción

1.1. Introducción y antecedentes

En la actualidad, los robots comerciales e industriales son ampliamente utilizados y realizan tareas de forma más exacta o más barata que los humanos. También se les utiliza en trabajos demasiado sucios, peligrosos o tediosos. Los robots son muy utilizados en plantas de fabricación, montaje y embalaje, en transporte, en exploraciones en la Tierra y en el espacio, cirugía, armamento, investigación en laboratorios y en la producción en masa de bienes industriales o de consumo. Otras aplicaciones incluyen la limpieza de residuos tóxicos, minería, búsqueda y rescate de personas y localización de minas terrestres, en definitiva, la robótica está presente en prácticamente cualquier ámbito de la actualidad.

Por otra parte, ninguno de los sistemas robóticos actuales podrían ser funcionales sin un software adecuado para automatización de tareas, siendo esencial una correcta sincronización entre los diferentes elementos hardware y software implicados con la finalidad de garantizar un correcto funcionamiento del conjunto robótico.

Centrándonos en lo anterior y aplicándolo en el campo de la automoción, ocasiona que en la actualidad se empleen importantes esfuerzos por parte de grupos investigadores y fabricantes de automóviles para dotar sus vehículos de los elementos necesarios para proporcionar a sus vehículos de un sistema de conducción autónoma. Campo en la actualidad en pleno desarrollo y presentando multitud problemas aún por resolver.

Entre los problemas existentes en los sistemas de conducción autónoma destaca su escasa adaptabilidad en carreteras reales donde existen multitud de imperfecciones, como pueden ser el desgaste o la falta de las señales viales, falta de iluminación o existencia de señalización provisional no fija como las indicadoras de obras en la vía. Estas circunstancias especiales impiden el correcto funcionamiento de los vehículos de dentro de unos niveles de seguridad mínimos aceptables para su implantación en carreteras transitables junto con vehículos conducidos de modo tradicional.

La problemática actual presentada, junto con que robótica y automoción son mis dos pasiones, hizo que me lanzara a la elaboración de este proyecto que unifica ambos cam-

pos anteriormente citados.

Así surgió *OpenTSR*.

Se trata de un vehículo controlado vía WiFi dotado de una cámara de vídeo la cual visualiza su entorno en busca de aquellas señales de tráfico presentes y actuar en consecuencia.



Figura 1.1: Logo OpenTSR ¹.

OpenTSR (que será el nombre del sistema resultante) será una combinación de un elemento hardware (vehículo entre otros) y software (consola de control) surgido como una aproximación a la solución de los citados problemas.

1.2. Objetivos

El objetivo principal del proyecto es elaborar un software de detección de señales de tráfico mediante el análisis de las imágenes obtenidas a partir de una cámara. Dicho software irá asociado a un vehículo dotado de una cámara para permitir la visualización del terreno además de ser capaz de reconocer las señales de tráfico mostrando en el ordenador la última señal reconocida. Por otra parte, permitirá al vehículo actuar en consecuencia a la señal detectada. En definitiva se desea dotar al vehículo de un sistema de conducción autónoma ² a partir de señales de tráfico.

El sistema a desarrollar, por tanto, dispondrá de dos modos de funcionamiento, el primero de ellos proporciona una conducción a gusto del usuario, el cual podrá controlar a su antojo el vehículo por el entorno que le rodea mediante un joystick o haciendo uso de las teclas del teclado de un ordenador. En el segundo modo de funcionamiento, el vehículo actuará automáticamente según las señales de tráfico detectadas por la cámara, siendo capaz de ajustar velocidad o realizar giros y paradas de manera automática según los elementos visionados a través de la cámara.

¹Logotipo elaborado por Noelia Sales Montes para el proyecto OpenTSR.

²En la sección 2.1 referente a conceptos básicos se incluye la definición de vehículo autónomo.

1.3. Acerca de este documento

El documento se ha sido elaborado en un lenguaje sencillo y claro para permitir que un estudiante universitario de cualquier Ingeniería Informática pueda comprender los contenidos sin apenas dificultad añadida.

Este documento se organiza en los siguientes capítulos:

- En el capítulo 1, Introducción, se comentan las razones que han motivado la creación de este proyecto, así como el propósito del mismo.
- En el capítulo 2, Conceptos básicos, se incluyen definiciones de aquellos conceptos considerados de interés para la correcta comprensión del contenido de la presente memoria.
- En el capítulo 3, Herramientas utilizadas, se realiza una descripción de las diferentes elementos hardware y software empleados durante el desarrollo del proyecto y necesarios para la utilización del mismo.
- En el capítulo 4, Organización temporal, se recoge todo lo que concierne a la distribución y duración de cada una de las tareas llevadas a cabo durante el desarrollo del proyecto que el presente documento describe.
- En el capítulo 5, Configuración y montaje de los dispositivos hardware, se explica el proceso seguido para la correcta integración de los dispositivos hardware empleados describiendo la interconexión entre ellos así como su configuración.
- En el capítulo 6, Desarrollo software, se realiza un análisis sobre la metodología empleada para el desarrollo software, describiendo los modelos de ciclo de vida utilizados, la descripción de los requisitos funcionales junto con el diagrama de casos de uso.
- En el capítulo 7, Software de reconocimiento, se hace una descripción explicando los diferentes aspectos y elementos de cada uno de los prototipos desarrollados junto con los problemas encontrados y soluciones adoptadas.
- En el capítulo 8, Software de control, se describe cómo se ha llevado a cabo la comunicación ordenador-vehículo a nivel software.
- En el capítulo 9, Interfaz gráfica, se recogen aquellos aspectos técnicos de interés referentes a la elaboración de la interfaz gráfica.
- En el capítulo 10, Guía de usuario, se describen los diferentes aspectos necesarios para la correcta utilización del conjunto software y hardware de los que se compone el presente proyecto.
- En el capítulo 11, Conclusiones, se hace mención de las conclusiones obtenidas tras la realización del proyecto además de las posibles mejoras aplicables.

- En el capítulo Anexos [11.3](#), aparecen los manuales de instalación del software que ha sido necesario para la realización del proyecto.

Capítulo 2

Conceptos básicos

2.1. Vehículo Autónomo

Un vehículo autónomo, es un vehículo capaz de cumplir con las capacidades de transporte llevadas a cabo por los humanos en un vehículo tradicional por sí mismo. Dicho vehículo es capaz de detectar los diferentes elementos de su entorno permitiendo un desplazamiento por sí mismo. Un ser humano puede elegir un destino, siendo innecesario realizar ninguna otra operación mecánica en el vehículo.

Los Vehículos autónomos perciben el mundo que le rodea gracias a la utilización de multitud de sensores tales como radares, GPS y visión por computador, siendo esta última la principal temática de este proyecto. Posteriormente un sistema avanzados de control interpretará la información para identificar las rutas adecuadas de navegación, así como los obstáculos y la señalización correspondiente. Dichos vehículos suelen actualizar sus mapas basados en la información sensorial, de forma que permiten la navegación a través de entornos desconocidos.

2.2. Inteligencia artificial

La inteligencia artificial (IA) se define como la capacidad de un dispositivo para llevar a cabo funciones que normalmente se asocian con la inteligencia humana, tales como el razonamiento y la optimización de la experiencia. La IA es la rama de la informática que trata de aproximar los resultados del razonamiento humano mediante la organización y manipulando conocimiento de los hechos y heurísticos. Las áreas de actividad de AI incluyen los sistemas expertos, la comprensión del lenguaje natural, reconocimiento de voz, la visión artificial y la robótica.

2.3. Visión artificial

La visión artificial, también conocida como visión por computador, es un subcampo perteneciente a la inteligencia artificial¹. El propósito de la visión artificial es programar

¹Definición de *inteligencia artificial* incluida en la sección 2.2.

un computador para que *entienda* una escena o las características de una imagen.

Los objetivos típicos de la visión artificial incluyen:

- La detección, segmentación, localización y reconocimiento de objetos en imágenes (por ejemplo, caras humanas).
- La evaluación de resultados (por ejemplo, segmentación, registro).
- Registro de diferentes imágenes de una misma escena u objeto, es decir, hacer concordar un mismo objeto en diversas imágenes.
- Seguimiento de un objeto en una secuencia de imágenes.
- Mapeo de una escena para generar un modelo tridimensional de la escena; este modelo podría ser usado por un robot para navegar por la escena.
- Búsqueda de imágenes digitales por su contenido.

Estos objetivos se consiguen por medio de reconocimiento de patrones², aprendizaje estadístico, geometría de proyección, procesamiento de imágenes, teoría de grafos y otros campos. En este proyecto nos centraremos en técnicas propias del reconocimiento de patrones.

2.4. Reconocimiento de patrones

El reconocimiento de patrones, también llamado lectura de patrones, identificación de figuras y reconocimiento de formas, consiste en el reconocimiento de patrones de señales. Los patrones son obtenidos a partir de procesos de segmentación, extracción de características y descripción para que cada objeto quede representado de manera lo más representativa posible por una colección de descriptores. El sistema de reconocimiento debe asignar a cada objeto su categoría o clase (conjunto de entidades que comparten alguna característica que las hacen diferentes al resto). Para poder reconocer los patrones se siguen tres procesos principales:

- Adquisición de datos.
- Extracción de características.
- Toma de decisiones.

La esencia del reconocimiento de patrones es la clasificación, un ejemplo práctico de clasificación puede ser la clasificación de imágenes digitales de letras en las clases de la A a la Z dependiendo de sus píxeles u otras situaciones que podamos extraer.

²Definición de *reconocimiento de patrones* disponible en la sección [2.4](#).

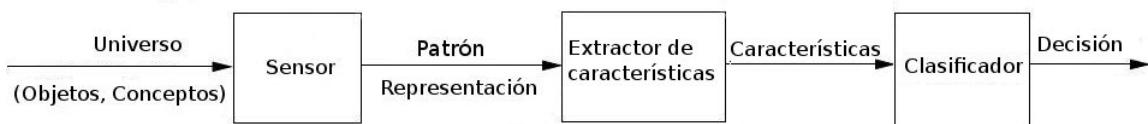


Figura 2.1: Componentes básicos de un sistema de reconocimiento de patrones.

2.4.1. Aplicaciones

Los sistemas de reconocimiento de patrones tienen multitud de aplicaciones. Algunas de las más relevantes y utilizadas actualmente son:

- Previsión meteorológica: poder clasificar datos meteorológicos según diversos patrones, y con el conocimiento a priori que tenemos de las diferentes situaciones que pueden aparecer nos permite crear mapas de predicción automática.
- Reconocimiento de caracteres escritos a mano o a máquina: es una de las utilidades más populares de los sistemas de reconocimiento de patrones ya que los símbolos de escritura son fácilmente identificables.
- Reconocimiento de voz: el análisis de la señal de voz se utiliza actualmente en muchas aplicaciones, un ejemplo claro son los teleoperadores informáticos.
- Aplicaciones en medicina: análisis de biorritmos, detección de irregularidades en imágenes de rayos-x, detección de células infectadas, marcas en la piel...
- Reconocimiento de huellas dactilares: utilizado y conocido por la gran mayoría, mediante las huellas dactilares todos somos identificables y con programas que detectan y clasifican las coincidencias, resulta sencillo encontrar correspondencias.
- Reconocimiento de caras: utilizado para contar asistentes en una manifestación o simplemente para detectar una sonrisa, ya hay diferentes cámaras en el mercado con esta opción disponible. Interpretación de fotografías aéreas y de satélite: gran utilidad para propuestas militares o civiles, como la agricultura, geología, geografía, planificación urbana...
- Predicción de magnitudes máximas de terremotos.
- Reconocimiento de objetos: con importantes aplicaciones para personas con discapacidad visual.
- Reconocimiento de música: identificar el tipo de música o la canción concreta que suena.

2.5. Modelos de color

El color, como cualquier otro recurso, también tiene su técnica y se encuentra sometido a ciertas leyes, y según la aplicación que se desea, se trabaja con distintos modelos de color. Los modelos de color describen los colores que se ven en las imágenes digitales e impresas y el trabajo con ellos.

Los modelos de color permiten, además de establecer un espacio único común a todos los equipos que realizan operaciones de adquisición y reproducción de color, permiten simular cómo lucirá la imagen y su color en otro dispositivo; así por ejemplo, podemos ver en la pantalla del computador cómo saldrá la imagen impresa en el papel, después de que haya pasado por las tintas que se usan normalmente en prensa y con diferentes papeles, permitiendo un trabajo de edición de imagen y color mucho más sencilla y fiel a los resultados finales.

Cada modelo de color como, por ejemplo, RGB, CMYK o HSV representa un método diferente (y por lo general, numérico) de descripción de los colores.

A continuación se describen los aquellos modelos de mayor importancia:

2.5.1. Modelo RGB

La descripción RGB, del inglés Red, Green, Blue, de un color hace referencia a la composición del color en términos de la intensidad de los colores primarios con que se forma: el rojo, el verde y el azul. Este modelo de color está basado en la síntesis aditiva, permitiendo representar un color mediante la mezcla por adición de los tres colores de luz primarios de los que se compone. El modelo de color RGB no define por sí mismo lo que significa exactamente rojo, verde o azul, por lo que los mismos valores RGB pueden mostrar colores notablemente diferentes en diferentes dispositivos que usen este modelo de color. Aunque utilicen un mismo modelo de color, sus espacios de color pueden variar considerablemente. Con el fin de indicar con qué proporción se mezcla cada color, se asigna un valor numérico a cada uno de los colores primarios, de manera que, por ejemplo, el valor 0 significa que ese color no interviene en la mezcla y, a medida que ese valor aumenta, se entiende que aporta más intensidad al conjunto. Aunque el intervalo de valores podría ser cualquiera (valores reales entre 0 y 1, valores enteros entre 0 y 255, etc.), aunque en la mayoría de ocasiones cada color primario es codificado con un byte (8 bits). Así, de manera usual, la intensidad de cada una de las componentes se mide con una escala del 0 al 255.

Por lo tanto, el rojo es obtenido con (255,0,0), el verde con (0,255,0) y el azul con (0,0,255), siendo para los tres casos un color monocromático. La ausencia de color, lo que identificamos como el color negro, se obtiene cuando las tres componentes son 0, (0,0,0).

La combinación de dos colores a nivel 255 con un tercero en nivel 0 da lugar a la obtención de los tres colores intermedios. De esta forma el amarillo es (255,255,0), el cian

(0,255,255) y el magenta (255,0,255).

Finalmente, el color blanco se forma con los tres colores primarios a su máximo nivel (255,255,255).

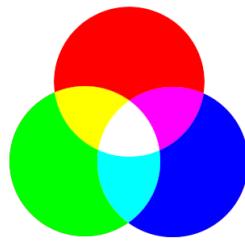


Figura 2.2: Modelo de color RGB.

2.5.2. Modelo HSV

El modelo HSV, del inglés Hue, Saturation, Value que traducidos son, Matiz, Saturación, Valor, se define como un modelo de color según sus componentes. Dicho modelo aplica una transformación no lineal sobre el espacio de color RGB, y se puede usar en progresiones de color.

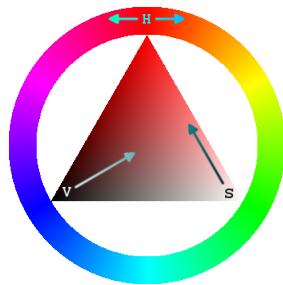


Figura 2.3: Espacio de color HSV.

2.5.3. Uso

Es común que deseemos elegir un color adecuado para alguna de nuestras aplicaciones, cuando es así resulta muy útil usar la ruleta de color HSV. En ella el matiz se representa por una región circular; una región triangular separada, puede ser usada para representar la saturación y el valor del color. Normalmente, el eje horizontal del triángulo denota la saturación, mientras que el eje vertical corresponde al valor del color. De este modo, un color puede ser elegido al tomar primero el matiz de una región circular, y después seleccionar la saturación y el valor del color deseados de la región triangular.

2.5.4. Características

Constituyentes en coordenadas cilíndricas:

Matiz

Se representa como un grado de ángulo cuyos valores posibles van de 0 a 360° (aunque para algunas aplicaciones se normalizan del 0 al 100 %). Cada valor corresponde a un color. Ejemplos: 0 es rojo, 60 es amarillo y 120 es verde. De forma intuitiva se puede realizar la siguiente transformación para conocer los valores básicos RGB: Disponemos de 360 grados dónde se dividen los 3 colores RGB, eso da un total de 120° por color, sabiendo esto podemos recordar que el 0 es rojo RGB(1, 0, 0), 120 es verde RGB(0, 1, 0) y 240 es azul RGB(0, 0, 1). Para colores mixtos se utilizan los grados intermedios, el amarillo, RGB(1, 1, 0) está entre rojo y verde, por lo tanto 60° . Se puede observar como se sigue la secuencia de sumar 60 grados y añadir un 1 o quitar el anterior:

- $0^\circ = \text{RGB}(1, 0, 0)$
- $60^\circ = \text{RGB}(1, 1, 0)$
- $120^\circ = \text{RGB}(0, 1, 0)$
- $180^\circ = \text{RGB}(0, 1, 1)$
- $240^\circ = \text{RGB}(0, 0, 1)$
- $300^\circ = \text{RGB}(1, 0, 1)$
- $360^\circ = 0^\circ$

Esta transformación permite saber los tonos de matices que contienen colores puros que contienen toda la cantidad (o ninguna) de los colores R, G y B. Para el color blanco puede poner cualquier matiz y establecer una saturación de 0. Por contra, para el color negro se puede poner cualquier color y saturación, siempre que se ponga un valor de 0.

Saturación

Se representa como la distancia al eje de brillo negro-blanco. Los valores posibles van del 0 al 100 %. Dicho parámetro también es conocido como “pureza”. Cuanto menor sea la saturación de un color, mayor tonalidad grisácea tendrá y más decolorado estará.

Valor

El Valor es representativo de la altura en el eje blanco-negro. Los valores posibles varian del 0 al 100 %. 0 se corresponde con el negro. Dependiendo de la saturación, 100 podría ser blanco o un color más o menos saturado.

2.5.5. Modelo CYMK

El modelo CMYK, siendo acrónimo de Cyan, Magenta, Yellow y Key, es un modelo de color ampliamente utilizado en la impresión de colores. Este modelo permite la representación de una amplia gama de colores y tiene una mejor adaptación a los medios industriales.

Este modelo se basa en la mezcla de pigmentos de los siguientes colores para crear otros más:

- C = Cyan (Cian).
- M = Magenta (Magenta).
- Y = Yellow (Amarillo).
- K = Black o Key (Negro).

La mezcla de colores CMY ideales en un fondo blanco da como resultado el color negro aunque por diversas razones, el negro generado no es ideal. Por esta razón los dispositivos de impresión a cuatro tintas incorpora la de color negro.

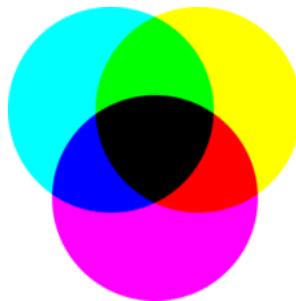


Figura 2.4: Modelo de color CMYK.

Capítulo 3

Herramientas utilizadas

El presente capítulo recoge información acerca de las diferentes herramientas, tanto hardware como software, utilizadas durante el desarrollo del proyecto y para su posterior utilización.

3.1. Herramientas software

El proyecto ha sido realizado haciendo uso del lenguaje de programación C y C++ junto con el uso de diversas bibliotecas. A continuación se realiza una breve descripción de cada una de ellas:

OpenCV Es una biblioteca libre de visión artificial originalmente desarrollada por Intel. Desde la salida de su primera versión alfa en el mes de enero de 1999, se ha utilizado en infinidad de aplicaciones. Desde sistemas de seguridad con detección de movimiento, hasta aplicaciones para de control de procesos donde es requerido el reconocimiento de objetos. Esto es debido se publica bajo licencia BSD, permitiendo su utilización para propósitos comerciales y de investigación con las condiciones en ella expresadas.

OpenCV es multiplataforma, existiendo versiones para GNU/Linux, Mac OS X y Windows. Contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos (reconocimiento facial), calibración de cámaras, visión estereo y visión robótica.

El proyecto pretende proporcionar un entorno de desarrollo fácil de utilizar y altamente eficiente. Esto se ha logrado, realizando su programación en código C y C++ optimizados, aprovechando además las capacidades que proveen los procesadores multinúcleo, que como se verá en la presente memoria, su alta eficiencia, ha sido uno de los motivos principales para la elección de esta biblioteca.



Figura 3.1: Logotipo de OpenCV

Qt es una biblioteca multiplataforma ampliamente utilizada para el desarrollo de aplicaciones con una interfaz gráfica de usuario, también es usado para el desarrollo de programas sin interfaz gráfica como herramientas para la línea de comandos y consolas para servidores.

Qt utiliza el estándar C++, pero hace un amplio uso de un generador de código especial (llamado compilador Meta Object, o MOC), junto con varias macros para enriquecer el lenguaje. Qt también puede ser usado con otros lenguajes de programación a través de enlaces al lenguaje. Permite su ejecución en las principales plataformas de escritorio y algunas plataformas móviles. También es usado en sistemas informáticos empotrados para automoción, aeronavegación y aparatos domésticos como frigoríficos.

Distribuido bajo los términos de la licencia GNU (entre otras), Qt es software libre y de código abierto. Todas las ediciones soportan el compilador GCC y la suite de Visual Studio entre otros.



Figura 3.2: Logotipo de Qt

Surveyor Robot Software software desarrollado por John Cummins junto con los agentes de laboratorio de la Universidad de Brooklyn con la asistencia de M.P. Azhar, y la supervisión del profesor Sklar empleando el lenguaje de programación C++ para el control del vehículo SRV-1 Surveyor. El código está liberado bajo Copyleft.

Simple DirectMedia Layer (SDL) es un conjunto de bibliotecas desarrolladas en el lenguaje de programación C. Proporciona las funciones básicas para la realización de operaciones de dibujo en dos dimensiones, gestión de efectos de sonido y música, además de carga y gestión de imágenes. Fueron desarrolladas inicialmente por Sam Lantinga, un desarrollador de videojuegos para la plataforma GNU/Linux.

Pese a estar programado en C, tiene wrappers a otros lenguajes de programación como C++, Ada, C#, BASIC, Erlang, Lua, Java, Python, etc. También proporciona

herramientas para el desarrollo de videojuegos y aplicaciones multimedia. Una de sus grandes virtudes es el tratarse de una biblioteca multiplataforma, siendo compatible oficialmente con los sistemas Microsoft Windows, GNU/Linux, Mac OS y QNX, además de otras arquitecturas y sistemas como Sega Dreamcast, GP32, GP2X, etc. La biblioteca se distribuye bajo la licencia GPL, que es la que ha provocado el gran avance y evolución de SDL.



Figura 3.3: Logotipo de SDL

Qt Creator es un IDE creado por Trolltech para el desarrollo de aplicaciones con las bibliotecas Qt.



Figura 3.4: Logotipo de Qt Creator

3.1.1. Licencia

La aplicación OpenTSR es software libre: usted puede redistribuirlo y / o modificar bajo los términos de la Licencia Pública General de GNU según lo publicado por la Free Software Foundation, ya sea la versión 3 de la Licencia, o (a su elección) cualquier versión posterior.

Este programa se distribuye con la esperanza de que sea útil, pero SIN NINGUNA GARANTÍA, incluso sin la garantía implícita de COMERCIALIZACIÓN o IDONEIDAD PARA UN PROPÓSITO PARTICULAR. Ver el GNU General Public License para más detalles. Debería haber recibido una copia de la Licencia Pública General de GNU junto con este documento. Si no es así, consulte [GNU Licenses](#).

3.2. Herramientas hardware

La figura 3.5 nos proporciona una visión general de los diferentes elementos hardware empleados en el desarrollo del proyecto. En los sucesivos puntos se enumeran las ca-

racterísticas de cada uno de ellos.



Figura 3.5: Imagen de los diferentes elementos hardware empleados.

3.2.1. Vehículo SRV-1

El elemento hardware principal utilizado es el vehículo *Surveyor SRV-1 WiFi webcam robot*. Dicho vehículo ha sido diseñado para la investigación, la educación, y la explotación, Surveyor SRV-1 es robot que cabe en la palma de la mano, puede desplazarse por superficies lisas y rugosas gracias a sus ruedas tipo tanque fabricadas con caucho.

El robot SRV-1 incorpora un potente procesador a 500 MHz denominado Blackfin BF537 con 32 MB de RAM y 4 MB de memoria Flash. Equipado con una cámara a color de resolución 1280x1024 (1,3 megapíxeles) y dos sensores láser que lo dotan de la capacidad de evitar obstáculos. La batería recargable de 7,2 V permite una autonomía de hasta 4 horas de duración y una placa de comunicación WLAN 802.11b/g para su seguimiento y control.



Figura 3.6: Imagen del vehículo SRV-1

Características principales

- Diseño Open Source con acceso completo al código fuente (GPL) y esquemáticos.
- El robot es totalmente programable para su funcionamiento autónomo.
- Amplio soporte de software a través de aplicaciones de terceros.
- Posibilidad de conducir o teleoperar con el robot por medio del software proporcionado (consola java) o de forma remota a través de navegador web.
- El software del host es procesado en el servidor web junto con almacenamiento de vídeo en streaming.
- El robot puede ejecutar programas escritos en C y almacenados en su memoria flash de a bordo.
- Control remoto inalámbrico con alcance de 100 metros en interiores y de hasta 1000m en aire libre (visión directa).
- El robot puede ser controlado desde un terminal o consola para las pruebas de fácil Linux 2.6 apoyo, así como "bare metal" de programación con GNU BFIN-elf-gcc

Características hardware

- **Procesador:** de 500 MHz y 1000 MIPS de nombre Blackfin BF537 con 32MB SDRAM y 4MB de memoria Flash.
- **Cámara:** 1,3 megapíxeles Omnipix OV9655 160x128 resolución de 1280x1024.
- **Placa de control:** Lantronix MatchPort 802.11b/g Wi-Fi

- **Alcance:** 100 metros en interiores, 1000m en situaciones de visión directa.
- **Sensores:** 2 punteros láser para mediciones y soporte para hasta 4 módulos Maxbotics ultrasónicos y varios sensores I2C.
- **Conjunto:** estilo tanque con tracción diferencial a través de cuatro motores reductores de precisión (reducción 100:1).
- **Velocidad:** 20 cm - 40 cm por segundo (aproximadamente 1 pie/seg o 0.5 millas/hora).
- **Chasis:** Aluminio mecanizado.
- **Dimensiones:** 120 mm largo x 100mm ancho x 80mm de altura (5"x 4"x 3").
- **Peso:** 350gm (12 oz).
- **Potencia:** 7,2 V 2AH Li-poli batería - 4 + horas de autonomía por carga.
- **Cargador:** 100-240VAC 50/60Hz (enchufe de EE.UU.). 1.1.3.

Debido a que el cargador posee el enchufe siguiendo el estándar americano ha sido necesario un conversor a formato europeo para realizar las cargas de la batería que incorpora el vehículo.

Características software

- **Firmware del Robot:** fáciles de actualizar, escrito en lenguaje C bajo GPL Open Source, compilado con GNU BFIN-elf-gcc y cadenas de herramientas BFIN-gcc-uClinix.
- **Programación:** el vehículo dispone de un intérprete de lenguaje C con funciones específicas.
- **Comandos de control:** el robot proporciona una serie de comandos para la ejecución de movimientos programables por el usuario desde la memoria Flash de a bordo.
- **Herramientas de desarrollo:** a través de cadenas de herramientas GNU <http://blackfin.uclinux.org>.
- **Consola del programa:** aplicación basada en Java, funciona en Windows, Mac, Linux.
- **WebcamSat:** módulo de servidor web integrado en el software de la consola, permite que varios espectadores simultáneos remotos a través de Internet.
- **Protocolo de control de robot:** fácil de usar desde otras aplicaciones.

En cuanto a software disponible proporcionado por terceros:

- **RoboRealm:** SRV-1 puede ser controlado directamente desde RoboRealm, un software basado en Windows con paquete específico para visión artificial en robots. Las extensiones para Robo-Realm SRV-1 permite la creación de guiones que combinan el procesamiento de imágenes de vídeo en vivo desde el robot, por ejemplo, filtrado de color, seguimiento, detección de bordes, extracción de características, y posterior procesamiento de la decisión y movimiento del robot, facilitando la creación de conductas tales como la localización de objetos y el rastreo, evasión de obstáculos, la detección de movimiento, notificación, etc, con una red interfaz, el control puede ser escrito en C / C++, Python, Java, C, Lisp, Visual Basic, WScript y COM a través de la API RoboRealm.
- **Microsoft Robotics Studio:** Drivers para el SRV-1 de Microsoft Robotics Studio están disponibles. MSRS es un entorno basado en Windows para desarrolladores académicos, aficionados y comerciales para crear aplicaciones de robótica a través de una amplia variedad de hardware.
- **Webots:** Soporte SRV-1 incluido en Webots, es un software de simulación robótica móvil, proporciona un entorno de prototipado rápido para el modelado, la programación y simulación de robots móviles con sistema operativo Windows, Mac / X y Linux. El modelado en 3D y la física son excepcionales.

Para más información acerca del vehículo robótico acceda a la web del [sitio oficial¹](#).

3.2.2. Router

Ha sido necesario disponer de un router para establecer una comunicación inalámbrica entre el ordenador y el vehículo SRV-1. Dicho vehículo posee la característica de realización de conexiones punto a punto o ad-hoc con el inconveniente de ser redes mucho más inestables.

Para solucionar el problema de la inestabilidad se ha optado por la creación de una infraestructura de red con la ayuda de un router proporcionando una mayor robustez a la conexión además de permitir incorporar más dispositivos a nuestra red en caso de que sea necesario.

¹Web del sitio oficial: http://www.surveyor.com/SRV_info.html.



Figura 3.7: Imagen del router empleado.

3.2.3. Cámara, receptor inalámbrico y capturadora de vídeo

Para la transmisión de vídeo desde el vehículo al ordenador, debido al estado defectuoso de la cámara incorporada en el vehículo SRV-1, ha sido necesario disponer de una cámara inalámbrica, un receptor inalámbrico para recoger las imágenes de la cámara y una capturadora de vídeo para digitalizarlas y visualizarla en el ordenador.

Capturadora de vídeo

El conversor analógico/digital o capturadora empleada es una König modelo CMP-USBVG. Sus características son las siguientes:

- Marca: König.
- Modelo: CMP-USBVG5.
- Resolución máxima: 720x480 Píxeles.
- Sistemas operativos compatibles: Windows 2000/XP/Vista, drivers no oficiales para Linux.
- Requisitos de energía: 5V.
- Interfaz: USB 2.0.
- Velocidad de captura en vídeo digital: 30 fps.
- Entrada de S-Video.
- Entrada compuesta de vídeo.

Consulte la sección [10.10](#) correspondiente a la guía de usuario para acceder a las instrucciones para la instalación de los drivers de la capturadora.



Figura 3.8: Imagen de la capturadora o conversor analógico/digital.

Cámara y receptor inalámbrico

La minicámara utilizada está formada por un sensor 1/4" OmniVision CMOS que ofrece una imagen a color, con una resolución de 380 líneas en TV. Posee un micrófono incorporado que hace que el audio se grabe junto al vídeo. La minicámara es capaz de captar imágenes donde la iluminación es baja siempre y cuando sea superior a 3.0 Lux lo que la hacen adecuada en situaciones de escasa luminosidad. Permite alimentación conectándola directamente a la red eléctrica o bien usando una pila, gracias al clip de alimentación para pilas de 9V que incorpora.

La señal de vídeo y audio se transmite hasta el receptor atravesando paredes y muros. Dispone de una distancia máxima de utilización de 30 metros en situaciones de interior con obstáculos o de hasta 100 metros en exteriores.

El receptor inalámbrico tiene la posibilidad de recoger la señal de 4 minicámaras gracias a sus cuatro canales, los cuales están capacitados para recoger audio y vídeo por cada canal. Los canales se pueden seleccionar a través de un switch que se encuentra en la parte trasera del receptor. La salida del receptor permite su conexión hacia un video-grabador, tarjeta capturadora o monitor mediante un cable RCA. Los canales tienen dos modalidades de visualización y/o grabación, bien mostrando de forma permanente un mismo canal o bien mostrando los cuatro canales de manera secuencial. La alimentación del receptor se hace mediante un alimentador de DC 8V/180 mA.



Figura 3.9: Imagen de la cámara y el receptor inalámbricos.

3.2.4. Gamepad

Para dotar al vehículo sistema de un control más ergonómico y funcional se ha utilizado un mando con conexión USB como el que muestra la figura 3.10.



Figura 3.10: Imagen del gamepad USB utilizado.

3.2.5. Ordenador

Por último ha sido necesario un ordenador portátil con un entorno GNU/Linux para la ejecución del software de control, el cual se encargará del procesamiento de las señales de vídeo y la transmisión de las respuestas al vehículo. La aplicación desarrollada también proporciona un medio para el manejo del vehículo por parte del usuario. Imprescindible que el ordenador incorpore de tarjeta WiFi inalámbrica.

Capítulo 4

Organización temporal

Para el desarrollo de OpenTSR ha sido necesario emplear varias herramientas, utilidades y bibliotecas. Algunas de ellas, ya habían sido utilizadas en ciertas ocasiones a lo largo de la carrera. Sin embargo, otras han requerido un periodo de formación previo, en el que se han adquirido los conocimientos necesarios para poder desarrollar el proyecto.

La mayor parte del proceso de investigación fue dedicado al estudio de las diferentes tecnologías existentes para el procesamiento de imágenes. Se comenzó realizando diferentes prototipos funcionales con Matlab, que además de ser un software privativo, no ofrecía la posibilidad de procesamiento de vídeo en tiempo real debido a su escasa velocidad de cálculo.

Posteriormente, tras la necesidad de solventar el problema de la velocidad, se optó por elaborar código con la alternativa libre a Matlab, Octave, implementando las funciones críticas en el lenguaje orientado a objetos C++. Resultando también insuficiente.

Continuando con las labores de investigación descubrí una biblioteca de uso muy extendido en robótica llamada OpenCV disponible en C, C++ y Python, resultando la disponible en C especialmente rápida en sus cálculos y proporcionando las características deseadas a las necesidades del proyecto, sobre todo la de dotar al sistema de la capacidad cálculo en tiempo real.

Todo ello ha implicado un tiempo bastante considerable en el uso, aprendizaje e investigación de las diferentes tecnologías existentes y comprobar su potencial.

Una vez decidida la herramienta para el procesamiento de imágenes, se consideró necesaria la idea de implementar una interfaz acorde con el sistema que permita el visionado de resultados al usuario y el entorno por el que se desplaza el vehículo. Para ello se utilizó la biblioteca Qt escrita en C++. Por tanto fue necesario tiempo para adquirir las nociones básicas de la biblioteca con la ayuda del entorno de desarrollo QtCreator.

Por otro lado, existen multitud de algoritmos y técnicas propias del reconocimiento de patrones que no son estudiadas durante la Ingeniería Técnica en Informática y que sí

son vistas en la Ingeniería Informática añadiendo la necesidad de estudios de los conceptos propios del reconocimiento de patrones. Todo ello me ha supuesto un esfuerzo de investigación y aprendizaje bastante considerable.

Una vez determinadas las diferentes herramientas a utilizar se comenzó con la implementación de los diferentes prototipos funcionales para el software de reconocimiento de imágenes. Finalizado el software de reconocimiento se elaboró el sistema de comunicaciones con el vehículo y la interfaz gráfica.

La figura 4.1 muestra una visión de las diferentes tareas desarrolladas para la elaboración del proyecto junto con la descomposición de cada una de ellas:

WBS	Nombre	Trabajo
1	SRV Traffic Console	132d
1.1	Investigación - aprendizaje previos	45d
1.1.1	Matlab pruebas	15d
1.1.2	Octave - C++ pruebas	15d
1.1.3	OpenCV	15d
1.1.3.1	OpenCV aprendizaje	10d
1.1.3.2	OpenCV pruebas	5d
1.2	Implementación algoritmo reconocedor OpenCV	35d
1.2.1	Prototipo 1 Reconocimiento	10d
1.2.2	Prototipo 2 Reconocimiento	10d
1.2.3	Versión final	15d
1.3	SRV-1 Puesta en funcionamiento	9d
1.3.1	Configuración vehículo	3d
1.3.2	Comunicación - control vehículo	3d
1.3.3	Implementación pad	2d
1.3.4	Implementación eventos teclado	1d
1.4	Desarrollo Interfaz gráfica	28d
1.4.1	QT aprendizaje	8d
1.4.2	Desarrollo	20d
1.5	Documentación	15d

Figura 4.1: Descomposición de las tareas implicadas en el desarrollo del proyecto.

CAPÍTULO 4. ORGANIZACIÓN TEMPORAL

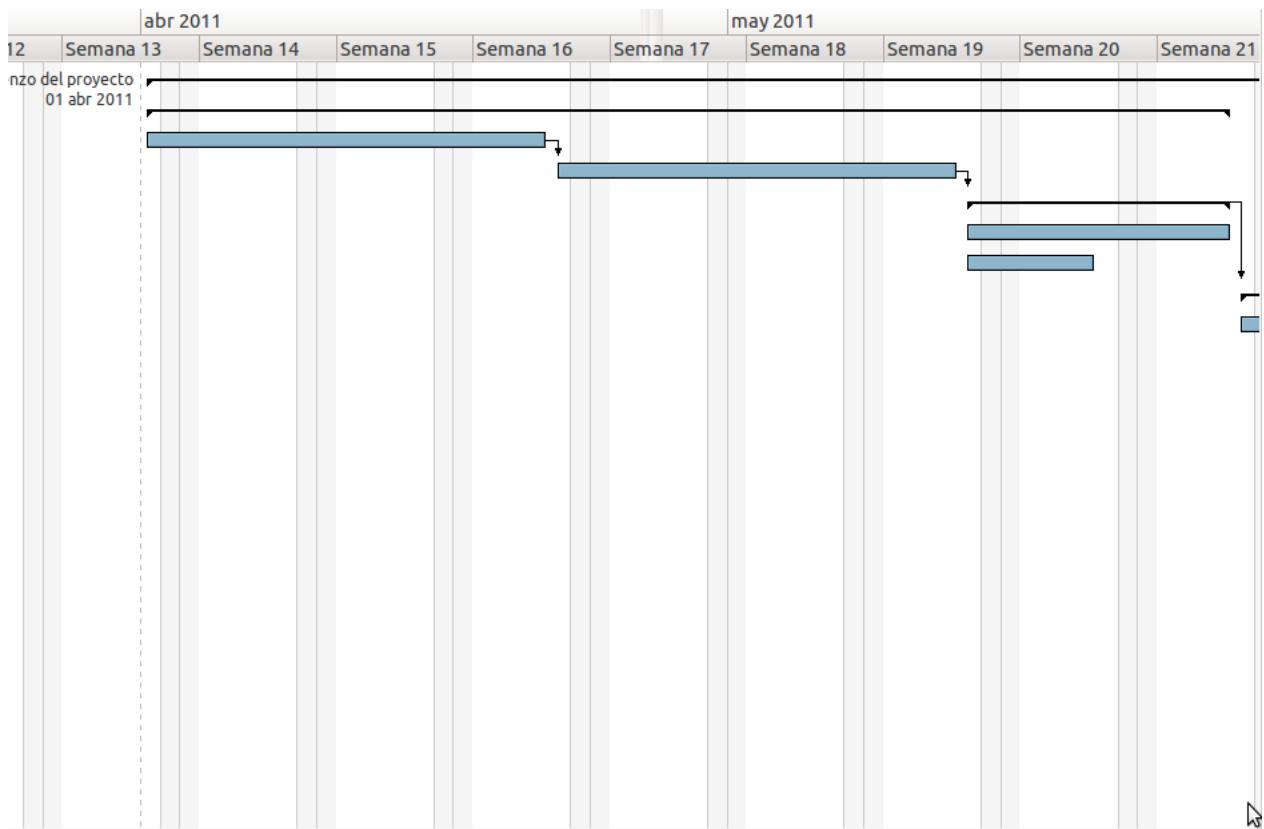


Figura 4.2: Diagrama de Gantt 1. Desarrollo del proyecto.

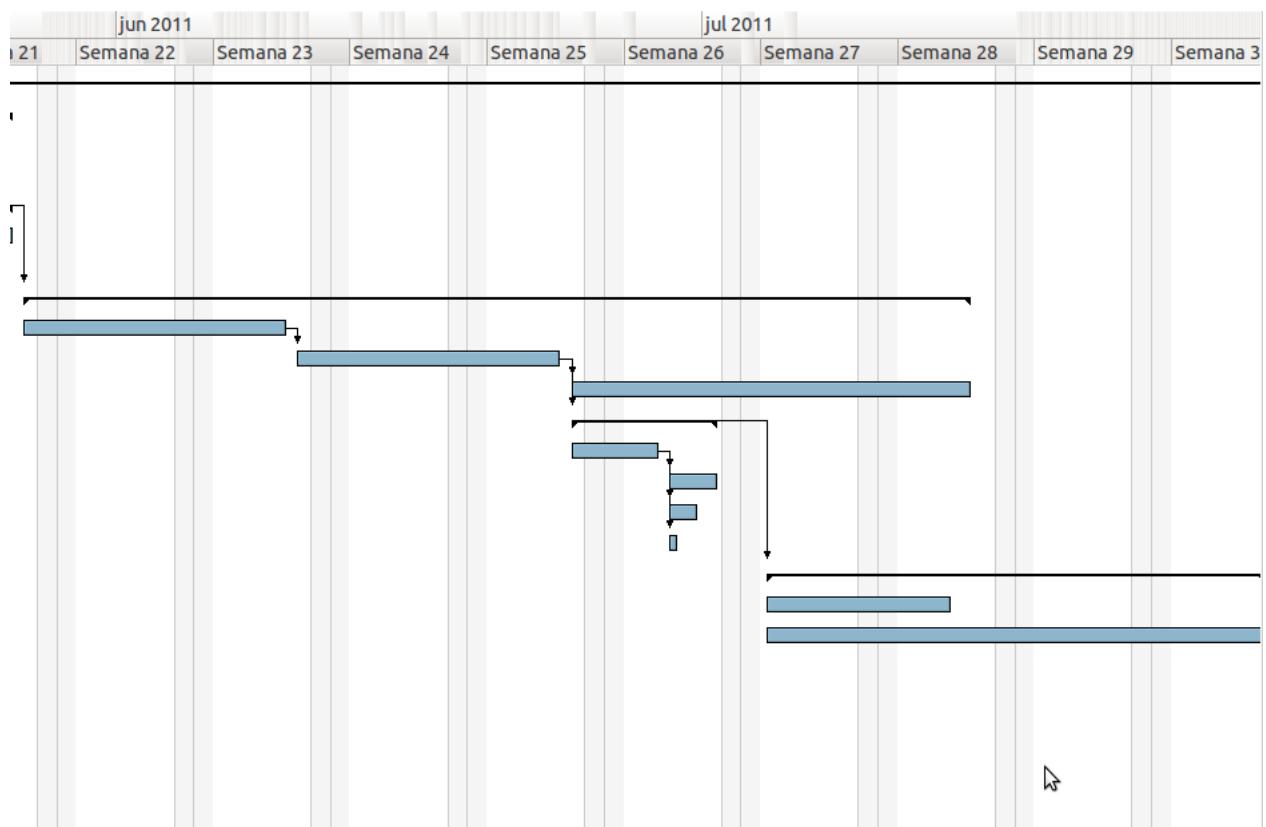


Figura 4.3: Diagrama de Gantt 2. Desarrollo del proyecto.

CAPÍTULO 4. ORGANIZACIÓN TEMPORAL

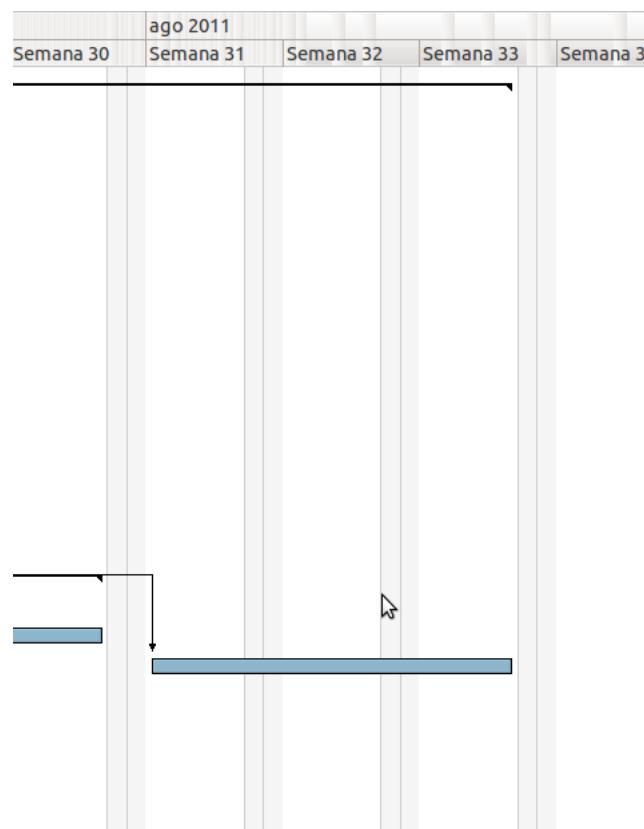


Figura 4.4: Diagrama de Gantt 3. Desarrollo del proyecto.

CAPÍTULO 4. ORGANIZACIÓN TEMPORAL

Capítulo 5

Montaje hardware

5.1. Colocación de la cámara en el vehículo

Para la obtención de la imágenes se ha elegido una pequeña cámara inalámbrica con el fin de no limitar la capacidad de movimiento del vehículo.

La cámara ha sido situada en la parte delantera del vehículo en el soporte que dispone para la colocación de la misma. Destacar que se ha empleado el sistema de cámara independiente debido al estado defectuoso de la cámara original del vehículo.

Para la alimentación de la cámara se ha añadido una pila de 9 voltios en la parte trasera del vehículo. En la imagen 5.1 se puede visualizar el conjunto vehículo, cámara y pila de alimentación montados.

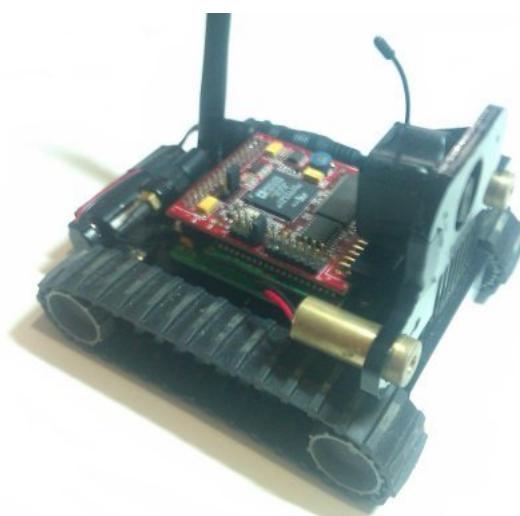


Figura 5.1: Vehículo SRV-1 tras el montaje de la cámara.

5.2. Conexión cámara - PC

Para conectar la cámara al ordenador, al tratarse de una cámara inalámbrica, resulta necesario utilizar un receptor. Las imágenes son trasmisidas mediante radiofrecuencia desde la cámara al sistema receptor que se encuentra conectado al PC mediante una capturadora de vídeo USB.

El sistema receptor de vídeo consiste en un capturador de imágenes analógico que recibe imágenes analógicas a través de radiofrecuencia. El receptor, posteriormente, a través de conectores RCA envía la imagen analógica al conversor analógico/digital convirtiendo la imagen analógica en imagen digital con una resolución de 720x576 píxeles, y que a su vez se encuentra conectado al PC mediante un cable USB, éste último dispositivo es también conocido como capturadora de vídeo.

El ordenador detecta la cámara como si de una webcam se tratara. El funcionamiento del sistema receptor se esquematiza en la figura 5.2. Indicar que el receptor de imagen analógico se alimenta con un adaptador de corriente.

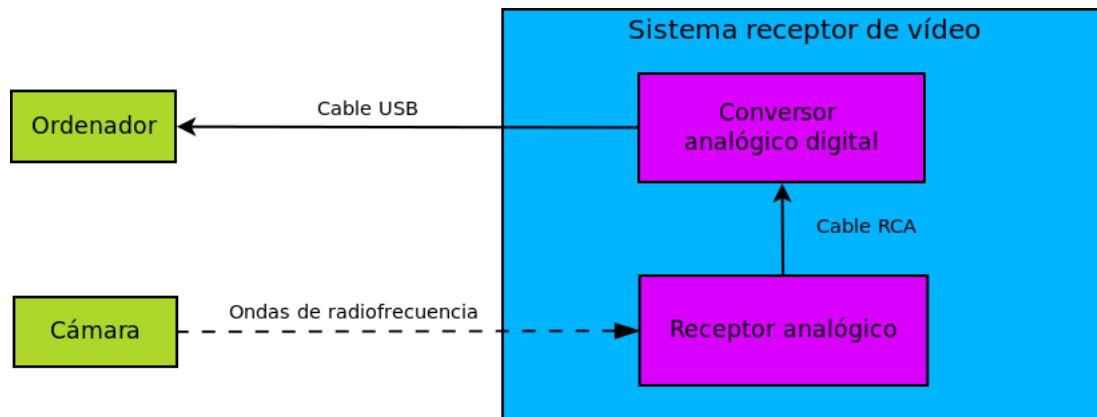


Figura 5.2: Esquema del sistema de recepción de imágenes desde la cámara al ordenador.

En la figura 5.5 se muestra una fotografía del conjunto correctamente conectado:



Figura 5.3: Visión del sistema de recepción de imágenes montado a falta de la conexión a corriente del receptor.

5.3. Comunicaciones vehículo - PC

Para establecer las comunicaciones entre el vehículo robótico y el ordenador ha sido necesario disponer de un router Wi-Fi para la creación de una infraestructura de red, de tal manera que, el vehículo SRV-1 realice automáticamente una conexión al router tras activarse el interruptor de encendido. Por otra parte, el ordenador deberá de conectarse al router como de si una red convencional se tratara. Se ha optado por este sistema debido a que presenta una mayor estabilidad que realizar una conexión directa ad hoc entre el ordenador y el vehículo.

En la sección [10.8](#) del manual de usuario se detalla el proceso de configuración del router Xavi 7868r utilizado.

El vehículo debe ser configurado según los valores previamente fijados en el router de tal manera que cuando el vehículo sea puesto en marcha realice una conexión de manera automática e inalámbrica con el router integrándose en la red. En el manual de usuario, sección [10.9](#), se detalla los pasos y parámetros necesarios para su correcta configuración.

La figura [11.2](#) de a continuación muestra un esquema de las comunicaciones entre el vehículo y el ordenador mediante el uso de un router.

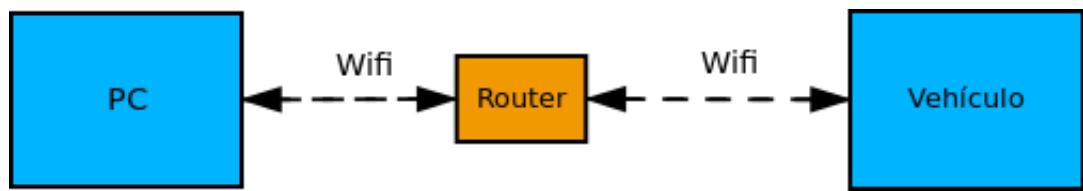


Figura 5.4: Comunicaciones vía Wi-Fi entre el ordenador y el vehículo robótico mediante un router.



Figura 5.5: Vista del sistema de comunicaciones entre el vehículo y el ordenador.

Capítulo 6

Desarrollo software

6.1. Metodología de desarrollo

Este proyecto ha sido obtenido empleando una metodología de desarrollo basada en el modelo de construcción de prototipos para la parte software referente a la visión artificial y una metodología de desarrollo en cascada para el desarrollo de la interfaz gráfica y software de control del vehículo.

El modelo de construcción de prototipos proporciona una serie de características que lo hacen idóneo para este proyecto. Dicho modelo resulta especialmente útil en las siguientes situaciones:

- El cliente define los objetivos generales del software sin entrar en detalles pormenorizados de los requisitos de entrada, procesamiento o salida a obtener.
- El responsable del desarrollo del software desconoce demasiados aspectos del desarrollo tales como, eficacia de un determinado algoritmo, capacidad de adaptación de un sistema operativo, definición de la interacción hombre-máquina entre otros.

Por otro lado, el modelo de desarrollo en cascada resulta adecuado para las siguientes situaciones:

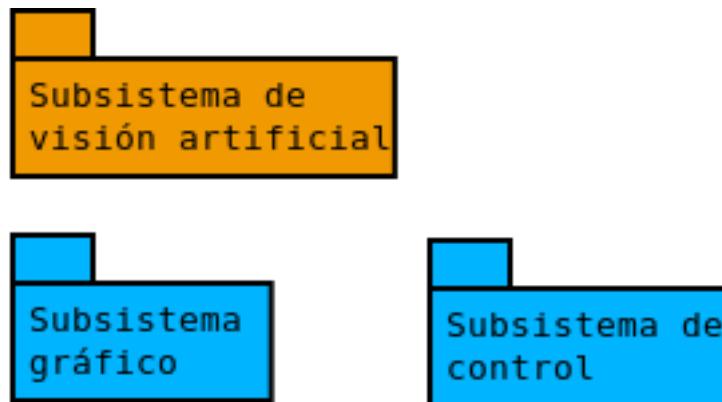
- Se dispone de unos requisitos claros y precisos.
- El sistema a desarrollar es de pequeña envergadura.
- Las tecnologías utilizadas son conocidas por los desarrolladores.

Centrándonos nuevamente en el desarrollo del proyecto, los motivos que llevaron a cabo la elección del modelo de desarrollo por prototipos para la obtención del software de visión artificial fueron el amplio desconocimiento existente en cuanto a los algoritmos existentes y su eficacia e idoneidad al producto deseado.

En cambio, para el desarrollo de la interfaz gráfica y software de control del vehículo, se optó por el seguimiento del modelo de desarrollo en cascada al tratarse de un desarrollo con una menor dificultad técnica y la disponibilidad de los requisitos necesarios.

Por tanto el proyecto queda distribuido en tres subsistemas:

- Subsistema de visión artificial.
- Subsistema de control.
- Subsistema gráfico.



Modelo de ciclo de vida empleado:

- Desarrollo en cascada.
- Desarrollo por prototipos.

Figura 6.1: Subsistemas existentes en el proyecto junto con el modelo de ciclo de vida utilizado para su desarrollo.

6.2. Recolección de requisitos

6.2.1. Requisitos funcionales

Para la elaboración de este proyecto se ha utilizado la metodología Métrica V3 así como el estándar ISO/IEC 12207. Métrica es una metodología de planificación, desarrollo y mantenimiento de los sistemas de información desarrollada por el Ministerio de Administraciones Públicas del Gobierno de España. Tiene como objetivo proporcionar una guía para la sistematización de actividades del ciclo de vida de los proyectos software en el ámbito de las administraciones públicas.

Métrica V3[2] está basada en el modelo de procesos del ciclo de vida de desarrollo ISO/IEC 12207 (Information Technology - Software Life Cycle Processes) así como en la norma ISO/IEC 15504 SPICE (Software Process Improvement And Assurance Standards Capability Determination).

En la página oficial de Métrica V3, sus desarrolladores indican que puede ser utilizada libremente con la única restricción de citar la fuente de su propiedad intelectual, la del Ministerio de Administraciones Públicas.

Un caso de uso es una descripción de los pasos o actividades que deberán realizarse para llevar a cabo algún proceso. Los objetivos de los casos de uso son los siguientes:

- Obtener los requisitos funcionales del sistema y expresarlos desde un punto de vista más cercano al usuario.
- Proporcionar una guía de todo el proceso de desarrollo del sistema de información. Los casos de uso proporcionan, por tanto, un modo claro y preciso de comunicación entre lo que desea el cliente y el desarrollador proporcionando desde el punto de vista del cliente una visión de “caja negra” del sistema eliminando los detalles de su construcción o desarrollo. Para los desarrolladores es utilizado como punto de partida y el eje sobre el que se apoya todo el desarrollo del sistema en los procesos de análisis y diseño.

Los requisitos funcionales que se han obtenido después del proceso de obtención de requisitos son los siguientes:

- Conectar con vehículo.
- Desconectar con vehículo.
- Pilotaje tradicional.
- Pilotaje autónomo

6.2.2. Diagramas de casos de uso

Los diagramas de caso de uso muestra una representación del comportamiento ofrecido por el sistema de información desde el punto de vista del usuario. El comportamiento del sistema es representado como un conjunto de transacciones ejecutadas entre el sistema y los actores junto con la descripción sobre las de las relaciones de comunicación existentes entre un actor y el sistema.

En la figura 6.2 podemos observar el diagrama de casos de uso resultante tras la recolección de requisitos:

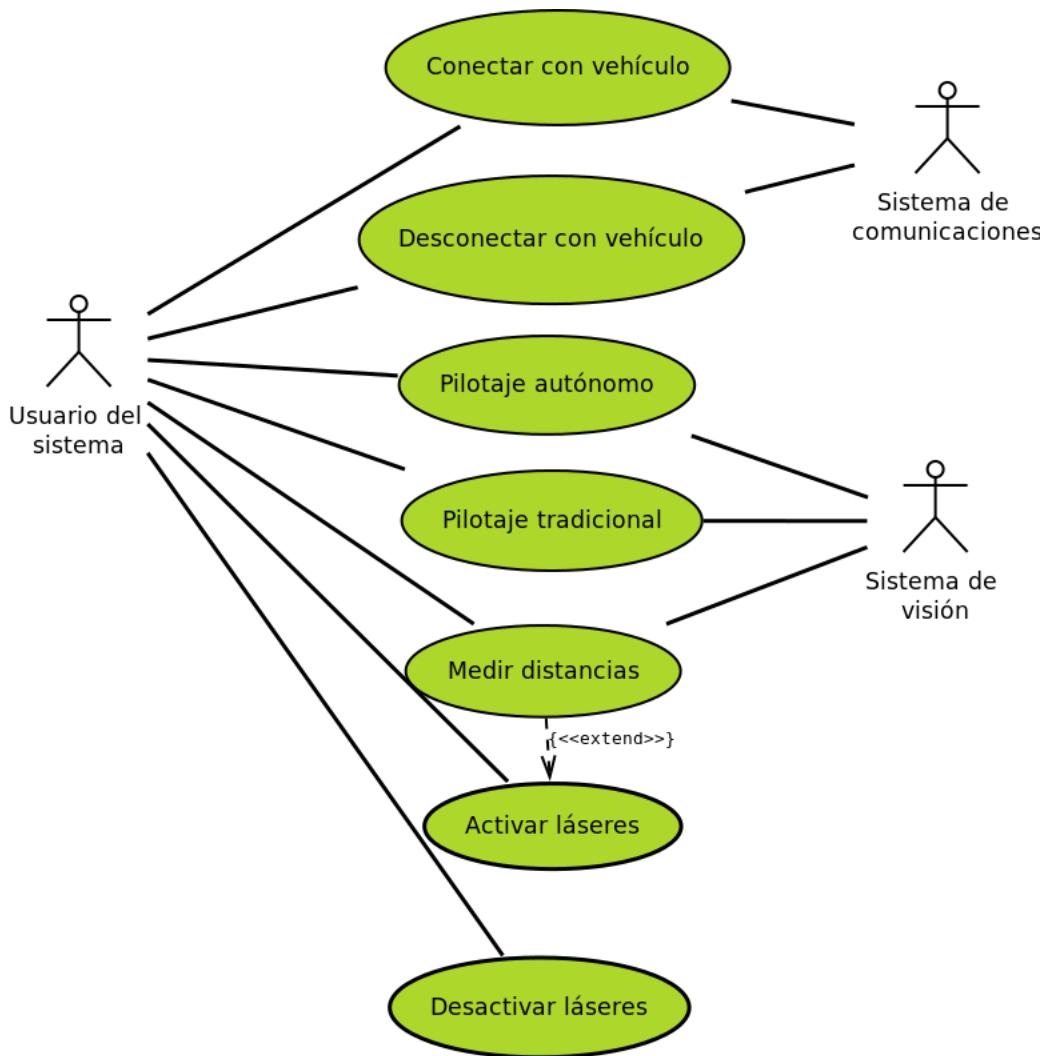


Figura 6.2: Diagrama de casos de uso.

6.2.3. Especificación de los casos de uso

A continuación se proporciona la especificaciones de cada uno de los casos de uso de la sección 6.2.2.

Caso de uso:	Conexión coche.
Descripción:	El sistema deberá conectarse como describe el presente caso de uso cuando el operador del sistema desee conectar con el vehículo. <i>continua en la siguiente página ...</i>

continua de la página anterior...

Actor principal:	Operador del sistema.
Actor secundario:	Sistema de comunicaciones.
Precondiciones:	El sistema de comunicaciones deberá estar libre.
Flujo principal:	<ol style="list-style-type: none"> 1. El actor operador del sistema solicita establecer conexión con el coche. 2. El sistema comprueba que se puede establecer comunicación. 3. El sistema establece una conexión entre el ordenador personal y el coche mostrando por pantalla que la conexión se ha realizado exitosamente.
Postcondición:	Se establece una conexión entre el ordenador personal y el coche.
Excepciones:	Si no se encuentra el vehículo el sistema indicará error.

Tabla 6.1: Descripción de caso de uso: Conexión coche.

Caso de uso:	Desconexión del vehículo.
Descripción:	El sistema deberá actuar como describe en este caso de uso cuando el operador del sistema solicita la desconexión con el vehículo.
Actor principal:	Operador del sistema.
Actor secundario:	sistema de comunicaciones.
Precondiciones:	El sistema de comunicaciones deberá estar ocupado con una conexión activa.
Flujo principal:	<ol style="list-style-type: none"> 1. El actor operador del sistema solicita la desconexión con el vehículo. 2. El sistema comprueba que se encuentra establecida una conexión entre el ordenador y el coche. 3. El sistema finaliza la conexión entre el ordenador y el vehículo e indica por pantalla que la desconexión se ha realizado con éxito.
Postcondición	Se finaliza la conexión entre el ordenador y el vehículo.
Excepciones:	No existe conexión entre el equipo y el vehículo.

Tabla 6.2: Descripción del caso de uso: Desconexión coche.

Caso de uso:	Pilotaje tradicional.
Descripción:	El sistema deberá comportarse como describe el presente caso de uso cuando el operador del sistema decide comenzar con el pilotaje tradicional del vehículo.
Actor principal:	Operador del sistema.
Actor secundario:	Sistema de visión artificial
Precondiciones:	Se encuentra establecida una conexión entre el ordenador y el vehículo y se están procesando imágenes.
Flujo principal:	<ol style="list-style-type: none"> 1. El actor operador del sistema solicita el pilotaje tradicional del vehículo. 2. El sistema comprueba que existe una conexión activa entre el ordenador y el vehículo, y también que se están procesando imágenes. 3. El sistema muestra las señales de tráfico detectadas en un momento determinado.
Postcondición:	Se comienza a pilotar el vehículo visualizándose las señales de tráfico del entorno.
Excepciones:	Si no se encuentra establecida la conexión entre el vehículo y el ordenador informa del error.

Tabla 6.3: Descripción del caso de uso: Pilotaje tradicional.

Caso de uso:	Pilotaje autónomo.
Descripción:	El sistema deberá comportarse como describe el presente caso de uso cuando el operador del sistema decide comenzar con el pilotaje autónomo del vehículo.
Actor principal:	Operador del sistema.
Actor secundario:	Sistema de visión artificial
Precondiciones:	Se encuentra establecida una conexión entre el ordenador y el vehículo y se están procesando imágenes.

continua en la siguiente página ...

continua de la página anterior...

Flujo principal:	<ol style="list-style-type: none"> 1. El actor operador del sistema solicita el pilotaje autónomo del vehículo. 2. El sistema comprueba que existe una conexión activa entre el ordenador y el vehículo, y también que se están procesando imágenes. 3. El sistema muestra las señales de tráfico detectadas en un momento determinado. 4. El vehículo realiza la operación correspondiente a la señal detectada.
Postcondición:	Se comienza a pilotar el vehículo visualizándose las señales de tráfico del entorno.
Excepciones:	Si no se encuentra establecida la conexión entre el vehículo y el ordenador informa del error.

Tabla 6.4: Descripción del caso de uso: Pilotaje autónomo.

Caso de uso:	Medir distancias.
Descripción:	El sistema deberá mostrar la distancia a la que se encuentra un obstáculo del vehículo utilizando sus punteros láser.
Actor principal:	Operador del sistema.
Actor secundario:	Sistema de visión artificial
Precondiciones:	Se encuentra establecida una conexión entre el ordenador y el vehículo y se están procesando imágenes.

continua en la siguiente página ...

continua de la página anterior...

Flujo principal:	<ol style="list-style-type: none"> 1. El actor operador del sistema solicita la medición de distancias. 2. El sistema comprueba que existe una conexión activa entre el ordenador y el vehículo, y también que se están procesando imágenes. 3. El sistema comprueba que se encuentran los láseres activados. 4. El sistema calcula la distancia a partir de los puntos del láser reflejados en la imagen captada. 5. Los punteros láser no se encuentran activados: extend activar láseres. 6. El sistema muestra la distancia real en centímetros entre el vehículo y el obstáculo.
Postcondición:	El sistema muestra la distancia real en centímetros entre el vehículo y el obstáculo frontal.
Excepciones:	Si no se encuentra establecida la conexión entre el vehículo y el ordenador informa del error.

Tabla 6.5: Descripción del caso de uso: Medir distancias.

Caso de uso:	Activar láseres.
Descripción:	El vehículo deberá activar los punteros láser.
Actor principal:	Operador del sistema.
Actor secundario:	
Precondiciones:	Se encuentra establecida una conexión entre el ordenador.
Flujo principal:	<ol style="list-style-type: none"> 1. El actor operador del sistema solicita la activación de los punteros láser. 2. El sistema comprueba que existe una conexión activa entre el ordenador y el vehículo. 3. El sistema comprueba que se encuentran los láseres desactivados. 4. El sistema activa los láseres.
Postcondición:	El vehículo activa los punteros láser.
Excepciones:	Si no se encuentra establecida la conexión entre el vehículo y el ordenador se informa del error.

Tabla 6.6: Descripción del caso de uso: Activar láseres.

Caso de uso:	Desactivar láseres.
Descripción:	El vehículo deberá desactivar los punteros láser.
Actor principal:	Operador del sistema.
Actor secundario:	
Precondiciones:	Se encuentra establecida una conexión entre el ordenador.
Flujo principal:	<ol style="list-style-type: none"> 1. El actor operador del sistema solicita la desactivación de los punteros láser. 2. El sistema comprueba que existe una conexión activa entre el ordenador y el vehículo. 3. El sistema comprueba que se encuentran los láseres activados. 4. El sistema desactiva los láseres.

continua en la siguiente página ...

continua de la página anterior...

Postcondición:	El vehículo desactiva los punteros láser.
Excepciones:	Si no se encuentra establecida la conexión entre el vehículo y el ordenador se informa del error.

Tabla 6.7: Descripción del caso de uso: Desactivar láseres.

6.2.4. Desarrollo

La elaboración de los diferentes prototipos del software de visión artificial junto con la obtención del software de comunicación y la interfaz gráfica queda recogido al detalle en los sucesivos temas.

La distribución es la siguiente:

- El capítulo 7 se describen cada uno de los prototipos desarrollados para el software de visión artificial.
- El capítulo 8 se centra en el desarrollo de la comunicación con el vehículo.
- El capítulo 9 abarca el desarrollo de la interfaz gráfica.

Capítulo 7

Software de reconocimiento

Una vez descritos los diferentes elementos hardware utilizados y la metodología de desarrollo empleada, en el presente capítulo sucesivos se realizará un profundo análisis de las técnicas y problemas detectados durante el desarrollo, comenzando en el presente tema por el software de reconocimiento. Este subsistema ha sido elaborado siguiendo el modelo de construcción por prototipos.

A modo introductorio, destacar que el software de reconocimiento ha sido elaborado haciendo uso de la biblioteca OpenCV para el lenguaje C. Esta biblioteca incluye los elementos necesarios para el tratamiento de imágenes para, como se ha descrito en la sección 1.2 de objetivos, proporcionar al vehículo de un sistema de reconocimiento de señales de tráfico.

Las señales de tráfico detectables por el sistema serán las siguientes:

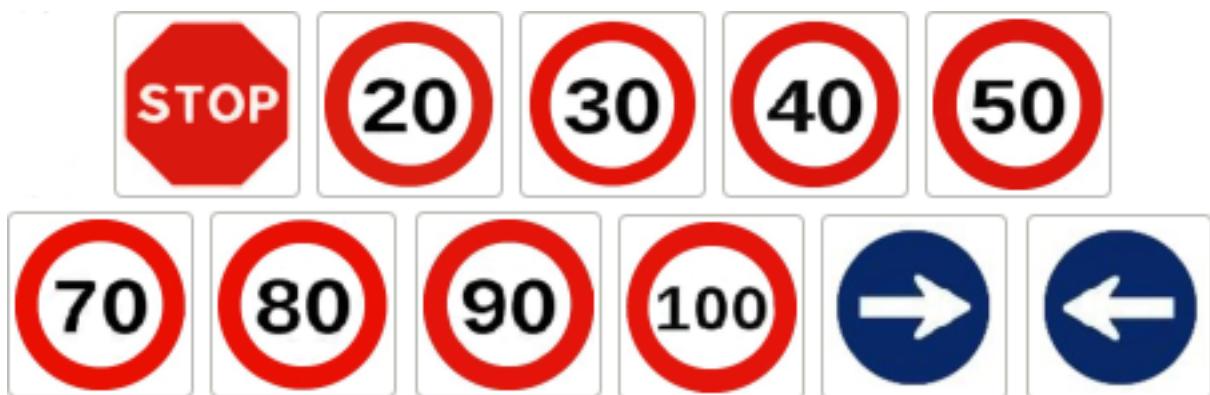


Figura 7.1: Conjunto de señales de tráfico detectables por el sistema.

Entre las señales incluidas disponemos las diferentes señales indicadoras de velocidad máxima, existiendo desde la de 20 km/h hasta la de 100 km/h con el objetivo de poder ajustar automáticamente la velocidad del vehículo en función de la señal detectada.

Por otro lado existen las señales indicadoras de dirección obligatoria, con el fin de efectuar giros y la señal de stop con el propósito de efectuar paradas de manera automática.

Para la elaboración del sistema ha sido necesario el desarrollo de tres prototipos. En las sucesivas secciones del tema se describirán cada uno de los ellos junto con los problemas detectados y motivos por los que finalmente se acabaron desechando junto con las soluciones adoptadas.

7.1. Primer prototipo

Para el desarrollo del primer prototipo se han empleado las siguientes técnicas:

1. Segmentación por color de la imagen de entrada.
2. Eliminación de objetos pequeños resultantes con área inferior a un determinado valor.
3. Extracción de la imagen de objetos existentes.
4. Comparación de cada uno de los objetos con plantillas para su identificación.

7.1.1. Obtención de la imagen

El paso previo de todo sistema de reconocimiento de objetos es la extracción de características a partir de unos datos de entrada para su posterior análisis. La información de entrada es proporcionada al sistema mediante una imagen, la cual es obtenida a través de la cámara que dota el vehículo.

Sin embargo, lo único que se recibe es una matriz de números desde la cámara.

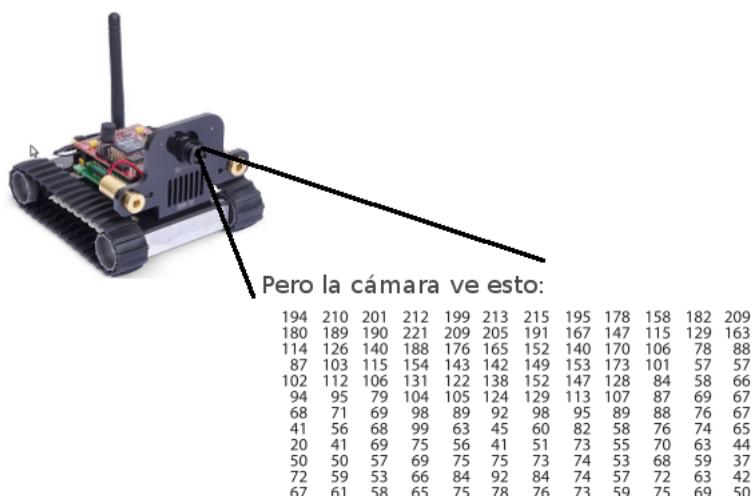


Figura 7.2: Interpretación de una imagen por parte de una cámara

Dicha matriz de números es recibida por el ordenador para su posterior interpretación y tratamiento¹.

A continuación se muestra el código realizado en OpenCV para la obtención de las imágenes a partir de una cámara:

Algoritmo de obtención de imágenes

```
#include <stdio.h>
#include <stdlib.h>
#include "opencv/cv.h"
#include "opencv/highgui.h"

int main(int argc, char *argv[])
{
    IplImage *img=NULL;

    char c=NULL;

    // Crea una ventana llamada Imagen
    // con tamaño predeterminado.
    cvNamedWindow("Imagen", CV_WINDOW_AUTOSIZE);

    // Crea la conexión con la Webcam.
    CvCapture* captura = cvCaptureFromCAM(-1);

    // Si la tecla pulsada es ESC se sale del bucle.
    while(c != 27) {

        // Añade el frame capturado dentro de la imagen img.
        img = cvQueryFrame(captura);

        // PROCESAMIENTO DE LA IMAGEN.

        // Se espera la pulsación de alguna tecla
        // durante 10 milésimas de segundo.
        // De ser alguna tecla pulsada,
        // se almacena en la variable c.
        c = cvWaitKey(10);
    }

    // Liberación de la cámara.
    cvReleaseCapture(&captura);
}
```

¹El procedimiento empleado para la transmisión de la imagen desde el vehículo al computador de manera física queda reflejado en el tema 5 correspondiente al montaje de los dispositivos hardware.

```
// Eliminamos todas las estructuras ventana existentes.  
cvDestroyAllWindows();  
  
return 0;  
}
```

Una vez obtenida la imagen, ésta es almacenada en la variable, *img* del código anterior. A partir de ese instante se dispone de la base para aplicar la extracción de características y el procesamiento de la imagen que se desee aplicar añadiendo el código necesario en la parte interna del bucle del código mostrado. Una vez finalizado el procesamiento aplicado a la imagen capturada se obtiene un nuevo frame para continuar con el procesamiento de una nueva imagen mientras la tecla ESC no sea presionada.

Un ejemplo de una imagen de entrada capturada por la cámara y visionada en la pantalla del ordenador utilizando el código anteriormente mostrado puede ser el siguiente:



Figura 7.3: Posible imagen de entrada al sistema.

7.1.2. Extracción de características

Puesto que el fin es la detección de un objeto, la segmentación de la imagen es empleada para aislar en ella a todos los conjuntos de píxeles candidatos a ser el objeto a seguir o detectar. Como hemos visto en la imagen 7.3 las señales a detectar presentan colores llamativos, visualizándose en su contorno más externo los colores rojos y azules, siendo

éstos los más representativos y predominantes. Por tanto resulta de especial utilidad la aplicación de una segmentación por color. Existen numerosos modelos de color utilizables para su realización como se ha descrito en la sección de [2.5](#).

Para la primera etapa del desarrollo del primer prototipo se ha empleado una técnica de segmentación por color siguiendo el modelo RGB. La finalidad de la segmentación es la de realizar una correcta selección de aquellos píxeles que según una determinada tonalidad resultan útiles para el posterior procesamiento e identificación de objetos.

Segmentación por color siguiendo el modelo RGB

Dado que los colores predominantes de las señales a detectadas son los rojos y azules intensos se ha optado por la selección de ambas tonalidades, centrándonos, para este primer prototipo, en el color rojo exclusivamente.

Una vez decidido el color a identificar, resulta necesario conocer las tonalidades exactas y cómo ésta es representada por el ordenador. La solución fue crear una herramienta con la que al pinchar sobre una imagen, obtuviera de forma automática el color sobre el que se ha pulsado mostrando por pantalla el código del color en los tres canales: rojo, verde y azul. El programa se escribió en lenguaje C haciendo uso de la biblioteca OpenCV.

En la figura [7.4](#) se puede observar el funcionamiento del programa mostrando el color sobre el que se ha hecho click, apreciándose como el último color seleccionado posee los valores 84, 70 y 202. Destacar que OpenCV invierte el orden de las componentes empleando BGR en lugar de RGB. Es decir, azul, verde y rojo, presentando, por tanto, un valor numérico mayor en la tercera componente. Esto se debe a que el píxel pulsado en la imagen se corresponde con el contorno exterior de la señal de 50 km/h cuya tonalidad es roja y que predomina sobre las otras dos.



Figura 7.4: Programa elaborado para la obtención de los valores del píxel pulsado.

El código del programa elaborado se muestra a continuación:

Programa elaborado para la obtención del valor de un píxel en una imagen

```
#include "cv.h"
#include "highgui.h"
#include <stdio.h>
#include <stdlib.h>

void mouseHandler(int event, int x, int y, int flags, void* param)
{
    // Declaración de variables.
    IplImage* img0, * img1;
    CvFont      font;
    uchar*      ptr;
    char        label[20];

    img0 = (IplImage*) param;
    img1 = cvCloneImage(img0);

    cvInitFont(&font, CV_FONT_HERSHEY_PLAIN, .8, .8, 0, 1, 8);

    if (event == CV_EVENT_LBUTTONDOWN)
    {
        // Se realiza la lectura del píxel pulsado.
        ptr = cvPtr2D(img1, y, x, NULL);

        // Se muestra el valor RGB.
        sprintf(label, "(%d, %d, %d)", ptr[0], ptr[1], ptr[2]);

        // Se crea un rectángulo de color en la imagen...
        cvRectangle(
            img1,
            cvPoint(x, y - 12),
            cvPoint(x + 100, y + 4),
            CV_RGB(255, 0, 0),
            CV_FILLED,
            8, 0
        );
        // Y se dibuja los valores numéricos en su interior.
        cvPutText(
            img1,
            label,
            cvPoint(x, y),
            &font,
```

```

        CV_RGB(255, 255, 0)
    );

    // Finalmente se muestra la imagen.
    cvShowImage("img", img1);
}
}

int main(int argc, char *argv[]){
    IplImage* img,*hsv_image;

    // Se comprueba que se ha introducido un argumento.
    if (argc != 2) {
        printf("Usage: %s <image>\n", argv[0]);
        return 1;
    }

    // Se realiza la carga de la imagen a partir de la ruta pasada
    // como argumento de entrada.
    img = cvLoadImage(argv[1], 1);

    //Copia de la imagen de entrada en una nueva imagen.
    hsv_image = cvCloneImage(img);

    //Transformacion a HSV
    cvCvtColor(img, hsv_image, CV_BGR2HSV);

    // Se crea una ventana con los eventos del ratón.
    cvNamedWindow("img", 1);
    cvSetMouseCallback("img", mouseHandler, (void*)hsv_image);

    cvShowImage("img", hsv_image);
    cvWaitKey(0);

    // Eliminación de estructuras.
    cvDestroyAllWindows();
    cvReleaseImage(&img);
    cvReleaseImage(&hsv_image);

    return 0;
}

```

Una vez conocidos los umbrales sobre los que trabajar, siempre en torno a un pequeño margen, se procede a una criba de píxeles que entran dentro del citado rango.

El código mostrado a continuación recibe una imagen de entrada realizando su recorrido y activando en una nueva imagen binaria, originalmente inicializada a cero, con el valor 1 aquellos píxeles situados dentro del umbral de color indicado. El recorrido de la imagen se ha realizado utilizando la aritmética de punteros en lugar de las funciones consultoras propias de OpenCV con el fin de ganar en eficiencia.

Segmentación de una imagen siguiendo el modelo de color RGB

```
IplImage* segmentacion_RGB(IplImage* img){

    // Declaración de variables.
    int altura,anchura,anchura_fila,canales;
    uchar *data;
    int i,j;

    // Recopilación de los datos de la imagen de entrada.
    altura = img->height;
    anchura = img->width;
    anchura_fila = img->widthStep;
    canales = img->nChannels;
    data = (uchar *)img->imageData;

    // Declaración e inicialización a cero de la imagen de salida.
    IplImage* img_selec_color;
    img_selec_color = cvCreateImage(cvGetSize(img), IPL_DEPTH_8U,1);
    cvZero(img_selec_color);

    // Recorrido de la imagen de entrada.
    for(i=0;i<altura;i++){
        for(j=0;j<anchura;j++){

            // Selección de tonalidades rojas.
            if ((data[i*anchura_fila+j*canales + 2] > 80) &&
                !((data[i*anchura_fila+j*canales + 0] >
                    data[i*anchura_fila+j*canales + 2]/2) ||
                (data[i*anchura_fila+j*canales + 1] >
                    data[i*anchura_fila+j*canales + 2]/2))) {

                //Activación de los píxeles que cumplen la condición de color fijada.
                img_selec_color->imageData[ img_selec_color->widthStep * i + j * 1]=1;
            }
        }
    }

    // Erosión para la eliminación de píxeles sueltos.
    cvErode(img_selec_color, img_selec_color,NULL,1);
}
```

```

// Dilatación para expandir los objetos de la imagen.
cvDilate(img_selec_color,img_selec_color,NULL,3);

// Se devuelve la imagen resultante.
return img_selec_color;
}

```

Una posible imagen de salida obtenida tras la aplicación del algoritmo de segmentación por color anterior se puede apreciar en la figura 7.5 de a continuación:



Imagen de entrada (a).

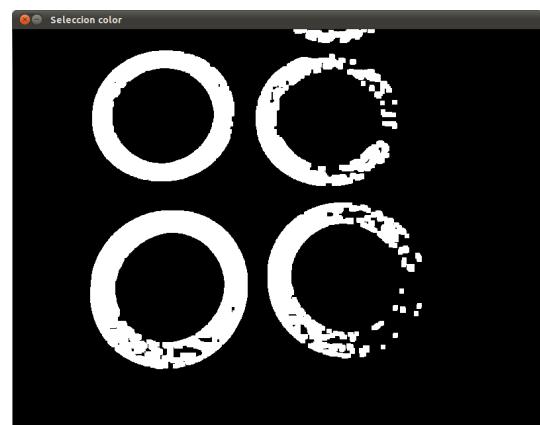


Imagen resultante (b).

Figura 7.5: Resultado de aplicar una selección por color sobre una imagen.

En la imagen 7.5 de salida se puede apreciar un conjunto de objetos resultantes a los que se les debe de aplicar algún mecanismo que permita trabajar con cada uno de los diferentes elementos por separado con la finalidad de poder clasificar cada uno de ellos de manera independiente.

Extracción de objetos

Una vez obtenida las agrupaciones de píxeles de tonalidad roja, figura 7.5, se procede a la identificación de cada una de esas agrupaciones por separado. Para ello utilizamos un método basado en la detección de contornos propio de OpenCV que está basado en el algoritmo de Canny. La aplicación de esta técnica nos permite averiguar los contornos de la imagen o, dicho de otro modo, los cambios bruscos de intensidades lumínicas.

Del paso anterior se obtiene una estructura tipo lista donde se encuentran almacenados todos los objetos detectados. El primer paso aplicado es descartar aquellos objetos cuyos valores de área son demasiado pequeños como para ser algunos de los elementos buscados, evitando así continuar con su análisis.

Los objetos restantes son, cada uno de ellos, encerrados dentro de una estructura rectangular de área mínima conociendo así su tamaño y coordenadas de localización.

Una vez conocidas las coordenadas que encierran al objeto, se establece una región de interés (ROI) en la imagen de entrada y en la obtenida tras la selección por color. Una región de interés es un área rectangular en una imagen, obteniendo el objeto segmentado para su posterior procesamiento. La figura 7.6 muestra un claro ejemplo de una delimitación de una región de interés en una imagen.

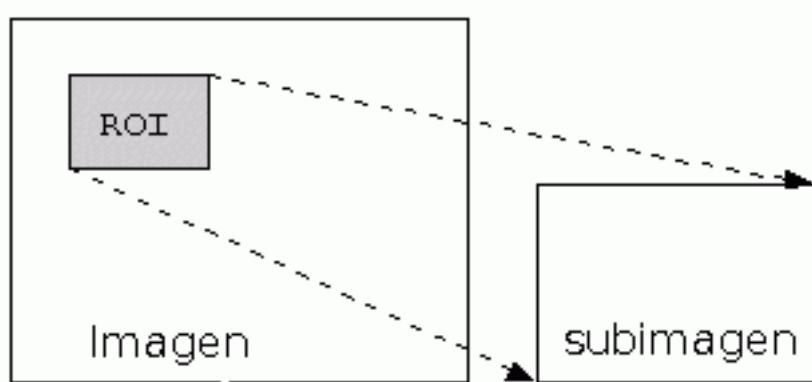


Figura 7.6: Imagen con una región definida de interés.

Cuando es asignada una región de interés en una imagen, todas las modificaciones realizadas sobre dicha imagen serán aplicadas sobre la zona seleccionada, una vez terminado el proceso de identificación del objeto encerrado en la región, ésta se libera y se procede a trabajar con una nueva región de interés en caso de existir más objetos candidatos a análisis en la imagen de entrada.

Una vez definidas las regiones de interés sobre ambas imágenes de la figura 7.5, se desea obtener la señal de forma independiente sin el fondo, para ello se emplea la técnica de copia usando una máscara, que no es más que copiar aquellos píxeles en una nueva imagen que se encuentren marcados en la imagen máscara. La imagen máscara es obtenida rellenando los huecos interiores de la imagen obtenida tras la aplicación de la selección por color. La figura 7.7 muestra un ejemplo ilustrativo:

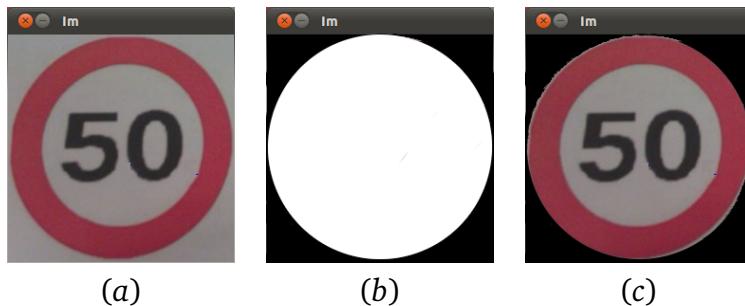


Figura 7.7: Copia de una imagen mediante el uso de máscaras.

El código mostrado realiza las acciones de extracción de cada uno de los objetos por separado una vez aplicada la segmentación por color:

Código para la extracción de los objetos de una imagen

```
// Segmentación por color
img_selec_color = segmentacion_RGB(img);

dst = cvCreateImage( cvGetSize(img), IPL_DEPTH_8U, 1 );
cvZero(dst);

// Búsqueda de contornos en la imagen obtenida tras la selección por color.
cvFindContours(img_selec_color, storage, &contour, sizeof(CvContour),
CV_RETR_EXTERNAL, CV_CHAIN_APPROX_NONE);

senial = cvCreateImage(cvGetSize(img), IPL_DEPTH_8U, 3);
cvZero(senial);

// Para objeto encontrado:
for(; contour != 0; contour = contour->h_next ){

    // Se calcula su área.
    area=fabs(cvContourArea(contour,CV_WHOLE_SEQ));

    // Si el área supera un determinado valor de área.
    if(area>=300){

        // Se dibuja el objeto en una imagen.
        cvDrawContours(dst, contour, cvScalar(1),cvScalar(1), -1, CV_FILLED, 8);

        // Se calcula el área mínima que encierra al objeto.
        rect = cvBoundingRect(contour,0);

        // Selección de las regiones de interés.
    }
}
```

```

cvSetImageROI(dst,rect);
cvSetImageROI(senial,rect);
cvSetImageROI(img,rect);

// Se realiza la copia mediante la
// utilización de máscaras para eliminar el fondo del objeto.
cvCopy(img,senial,dst);

// Se eliminan las regiones de interés establecidas previamente.
cvResetImageROI(dst);
cvResetImageROI(img);

// Se redimensiona la imagen.
identificar = cvCreateImage( cvSize(100,100), IPL_DEPTH_8U,3 );
cvResize(senial,identificar,CV_INTER_LINEAR);

// Y ya disponemos la imagen con el elemento a identificar.
cvShowImage( "Elemento",identificar);

// ALGORITMO IDENTIFICADOR

} // Fin if.
} // Fin for.

```

Una vez obtenida una imagen independiente con el objeto a identificar en color, imagen *c* de la figura 7.7 o la variable *identificar* del código anterior, debe ser redimensionada para que todos los objetos a análisis sean de igual tamaño para proceder a la aplicación de la técnica de clasificación denominada “Template matching” o comparación de plantillas.

7.1.3. Template matching

Para la identificación de los objetos en el primer prototipo se ha utilizado una técnica denominada Template Matching o comparación de plantillas.

Esta técnica consiste en la búsqueda de la región de una imagen que mayor semejanza guarda con la imagen dada como plantilla.

En la comparación de plantillas, todas las posibles localizaciones que pueden ser comparadas con la plantilla son almacenadas en una matriz *R*, en la cual se almacenan los coeficientes resultantes de las comparaciones. El tamaño de la imagen de entrada es de *W x H* donde *W* y *H* representan el ancho y el alto de la imagen respectivamente y *w x h* se corresponde con el producto del ancho por el alto de la imagen plantilla. El tamaño de la matriz viene dado por $(W - w + 1) \times (H - h + 1)$.

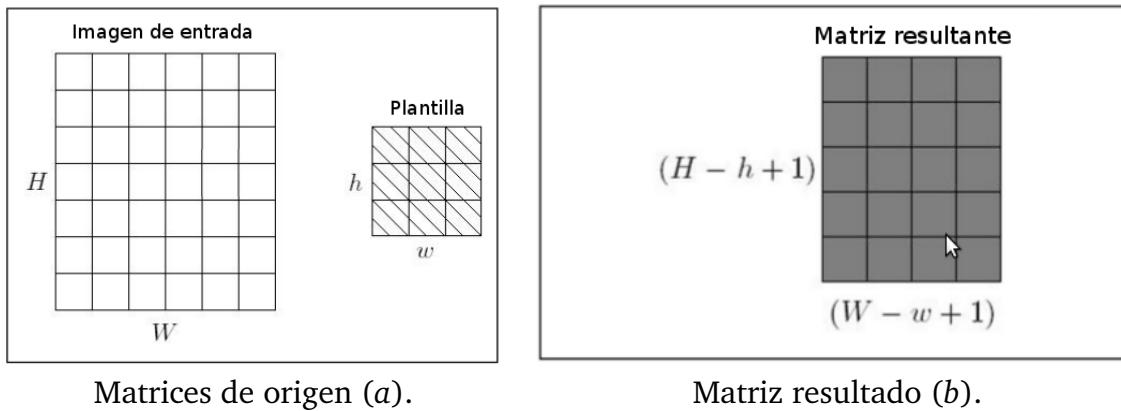


Figura 7.8: Matrices de entrada y de salida para la técnica de comparación de plantillas.

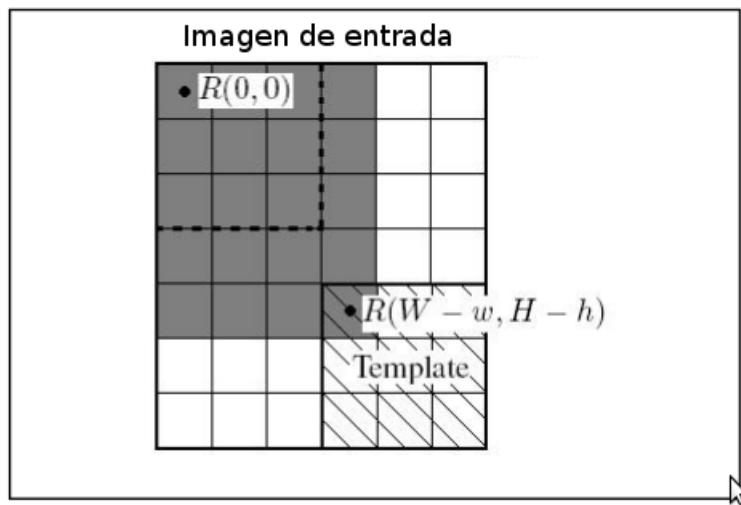


Figura 7.9: El valor de cada píxel de la posición (x, y) es procesado para obtener su característica de similitud entre la plantilla y el rectángulo resultante de tomar (x, y) como píxel situado en la esquina superior izquierda.

Las referencias del tamaño y la representación del proceso de comparación de plantillas se muestra en las figuras 7.1.3 (a) y 7.9, respectivamente.

La figura 7.1.3 (b), muestra la matriz final obtenida tras el proceso de búsqueda de una plantilla en una imagen.

Existen varios métodos para la comparación de plantillas, algunos de los más conocidos son la suma de las diferencias al cuadrado, correlación cruzada y el método del Coeficiente de Correlación. Dependiendo del algoritmo, el resultado correspondiente puede ser ligeramente diferente. Se considera T como la imagen de la plantilla e I como la imagen de entrada. En el presente prototipo el método de las diferencias al cuadrado es el que ha resultado más eficaz. A continuación se muestra su fundamento matemáti-

co:

$$R(x, y) = \frac{\sum_{x',y'}(T(x', y') - I(x + x', y + y'))^2}{\sqrt{\sum_{x',y'} T(x', y')^2 \cdot \sum_{x',y'} I(x + x', y + y')}} \quad (7.1)$$

Los valores obtenidos tras aplicar 7.1 en cada píxel de la imagen de entrada son almacenados en una matriz de salida R anteriormente descrita. Dicha matriz finalmente se recorre localizando aquellos valores que superen un umbral fijado con el fin de dar por localizado el objeto que se encontraba en la imagen que ha sido empleada como plantilla.

A continuación se muestra el código realizado en OpenCV para la comparación de plantillas:

Código para la detección de un objeto

```
// ALGORITMO DE IDENTIFICACIÓN

// Carga de imagen referencia o plantilla.
tpl = cvLoadImage( directorio, CV_LOAD_IMAGE_COLOR );

// Cálculo de ancho y alto de la imagen resultado.
width = identificar->width - tpl->width + 1;
height = identificar->height - tpl->height + 1;

// Se crea la imagen donde se almacenará el resultado de la comparación.
res = cvCreateImage( cvSize( width, height ), IPL_DEPTH_32F, 1 );

// Se realiza la comparación:
cvMatchTemplate(identificar, tpl, res, CV_TM_SQDIFF_NORMED);

for( i = 0 ; i < height ; i++ ) {
    for( j = 0 ; j < width ; j++ ) {

        // sacamos el elemento:
        s = cvGet2D( res, i, j );

        // Si el valor se localiza entre el rango establecido...
        if( s.val[0] <= umbral ) {
            // Dibujamos el contorno al objeto identificado ENCONTRADO!
            fprintf( stderr, "¡50 m/h!" );

            pt1.x = rect.x;
            pt1.y = rect.y;
            pt2.x = rect.x + rect.width;
            pt2.y = rect.y + rect.height;
        }
    }
}
```

```

    pt2.y = rect.y + rect.height;

    // Se dibuja el rectángulo que encierra al objeto.
    cvRectangle(img,pt1,pt2,CV_RGB(0,255,0),3,8,0);
}
}
}

```

La figura 7.10 muestra una imagen donde se aprecia cómo el objeto utilizado como plantilla ha sido detectado:

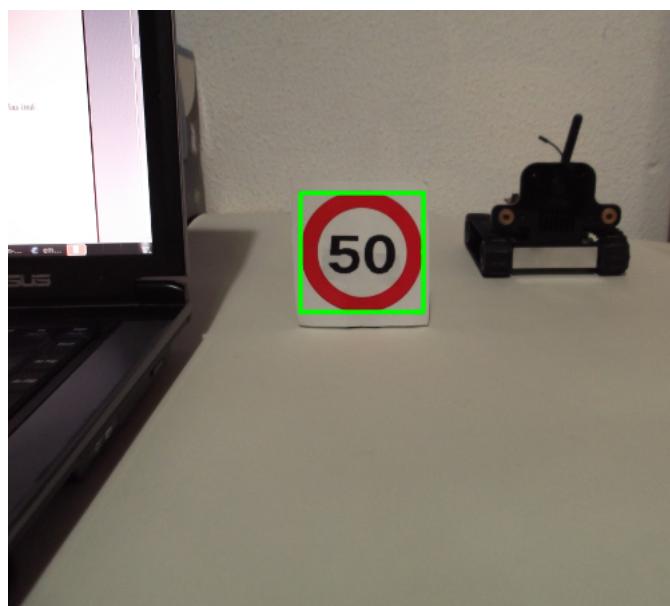


Figura 7.10: Vista de la señal detectada en la imagen.

La técnica de comparación de plantillas, como se ha descrito, presenta la particularidad de encontrar un determinado objeto dentro de una imagen, por tanto, a priori el preprocesado previo realizado para aplicar una segmentación por color y extracción de los objetos por separado antes de emplear este algoritmo de clasificación no resulta necesario para su funcionamiento. El preprocesado previo se ha realizado para evitar las innecesarias comparaciones en zonas de la imagen donde sabemos que el objeto buscado no se va a encontrar, ganando por tanto en eficiencia.

7.1.4. Problemas detectados

Los problemas que han llevado a cabo el rechazo de este primer prototipo han sido los siguientes:

- El problema de la utilización de una segmentación por color siguiendo el modelo RGB es que sufre una extrema sensibilidad de variación de los valores de un determinado color a las condiciones lumínicas del entorno. Esto ocasiona que el

prototipo desarrollado solo sea funcional en un único entorno, por ejemplo, la sala donde se ha desarrollado y probado, dejando de ser funcional en otras situaciones y lugares siendo necesario idear una nueva metodología de segmentación para el siguiente prototipo.

- El problema del Template Matching es que ha de comparar muchas características (para él, un píxel es una característica), y si tenemos en cuenta que en la base de datos encontramos N señales, observamos que este método el resultado no ofrece garantías de funcionamiento en tiempo real, condición indispensable del proyecto.

7.2. Segundo prototipo

Este segundo prototipo difiere del anterior por la utilización para la segmentación por color del modelo de color HSV. Este modelo de color es mucho menos sensible a las condiciones lumínicas del entorno, permitiendo la detección de las tonalidades deseadas en situaciones donde el primer prototipo presentaba problemas.

Por otro lado, para la clasificación se ha optado por la utilización del algoritmo de los K-vecinos en lugar de la comparación de plantillas o Template Matching.

Destacar que para la función del algoritmo de los k-vecinos resulta necesaria una recolección previa de los datos de los elementos a detectar, los denominados clases, con el fin de realizar una serie de comparaciones entre el objeto desconocido y los datos previamente extraídos con la finalidad de determinar a qué clase corresponde. Por tanto se describirá el algoritmo de entrenamiento (extracción de datos) y el algoritmo de detección (clasificación) utilizados.

7.2.1. Extracción de características

Para poder clasificar satisfactoriamente los elementos, es necesario un proceso de aprendizaje previo en el cual el sistema crea un modelo de cada una de las clases a partir de una secuencia de entrenamiento o conjunto de vectores de características de cada una de las clases.

Debido a esta razón, debemos previamente, antes de realizar cualquier tarea de clasificación, elaborar un conjunto de vectores de características o también denominado matriz de modelos con el fin de cotejar las características del elemento a identificar con los datos de la matriz modelos. A la fase de obtención de la matriz de modelos se le denomina etapa de entrenamiento en un sistema de reconocimiento de patrones. Dicha matriz ha sido obtenida en una serie de etapas que se describen en los sucesivos puntos de esta sección.

Segmentación por color siguiendo el modelo HSV

En la primera etapa del entrenamiento se ha optado por continuar con una técnica de segmentación por color, pero en esta ocasión siguiendo el modelo HSV. Cabe recordar que la finalidad de la segmentación es la de realizar una correcta selección de aquellos píxeles que según una determinada tonalidad resultan útiles para el posterior procesamiento e identificación de objetos.

En esta ocasión se presenta el mismo problema que con el primer prototipo, se desconocen los valores correspondiente al color a identificar, siendo, de igual modo, necesario conocer las tonalidades exactas y cómo la susodicha tonalidad es representada por el ordenador. La solución en este caso fue adaptar la herramienta empleada en el primer prototipo. Dicha herramienta mostraba el valor de un determinado píxel al pulsar sobre el mismo, solo que en esta ocasión en formato HSV. El código del programa es idéntico al mostrado en el primer prototipo pero realizando previamente una transformación de la imagen del formato RGB al formato HSV.

En la figura 7.11 se puede observar el funcionamiento del programa mostrando el color sobre el que se ha hecho click, apreciándose como el último color seleccionado posee los valores en sus tres canales: Hue, Saturation, Value, en castellano, tono, saturación y valor. Los valores obtenidos son (175,52,220) fijándonos en los valores correspondientes al tono y la saturación. El programa se escribió en lenguaje C haciendo uso de la biblioteca OpenCV.



Figura 7.11: Programa elaborado para la obtención de los valores del píxel pulsado.

La aplicación de este modelo resulta mucho más ventajosa con respecto al modelo RGB debido a que HSV sólo precisa de un solo valor numérico para detectar el color, desde tonos relativamente ligeros hasta los tonos más oscuros. La cantidad de color y el brillo del color son manejados por la *saturación* y los parámetros *valor*, respectivamente.

A continuación se muestra el código realizado para realizar la segmentación por color de una imagen de entrada, en la cual, al igual que la función desarrollada para la segmentación siguiendo el modelo RGB, se han fijado unos umbrales y se ha recorrido la imagen identificando aquellos píxeles que cumplen las condiciones dejando el píxel activado en una imagen de salida.

Segmentación de una imagen siguiendo el modelo de color HSV

```
IplImage* seleccion_colorHSV(IplImage* img ){

    // Declaración de variables
    IplImage *hsv_Image, *mono_Image;
    int height , width , step , channels;
    int step_mono , channels_mono;
    uchar *data_hsv , *data_mono;

    // Declaración de los valores umbral.
    int hupper=175,hlower=110,Saturation=60,Brightness=90,Erode=1,Dilate=1;

    int hupper2=35,hlower2=0,Saturation2=60,Brightness2=60,Erode2=0,Dilate2=3;
    //

    // Se crean dos imágenes del tamaño de la imagen fuente.
    hsv_Image = cvCreateImage( cvGetSize(img), 8, 3 );
    mono_Image = cvCreateImage( cvGetSize(img), 8, 1 );
    // Se inicializan a cero.
    cvZero(hsv_Image);
    cvZero(mono_Image);

    // Obtenemos atributos de la imagen HSV.
    height      = hsv_Image->height;
    width       = hsv_Image->width;
    step        = hsv_Image->widthStep/sizeof(uchar);
    channels    = hsv_Image->nChannels;
    step_mono   = mono_Image->widthStep/sizeof(uchar);
    channels_mono = mono_Image->nChannels;

    // Convertimos imagen de entrada de RGB a HSV.
    cvCvtColor(img,hsv_Image,CV_RGB2HSV);

    // Obtenemos los valores RGB de la Imagen.
    data_hsv = (uchar *)hsv_Image->imageData;
    data_mono = (uchar *)mono_Image->imageData;

    //Recorremos la imagen.
```

```

for(int i = 0; i < height; i++ ) {
    for(int j = 0; j < width; j++ ) {

        // Segmentamos la imagen mediante los valores umbrales
        // correspondientes al color rojo y azul.
        if (((data_hsv[i*step+j*channels])>= hlower) &&
            ((data_hsv[i*step+j*channels]) <= hupper) &&
            (data_hsv[i*step+j*channels+1]>= Saturation ) &&
            (data_hsv[i*step+j*channels+2]>= Brightness) ||
            ((data_hsv[i*step+j*channels])>= hlower2) &&
            ((data_hsv[i*step+j*channels]) <= hupper2) &&
            (data_hsv[i*step+j*channels+1]>= Saturation2 ) &&
            (data_hsv[i*step+j*channels+2]>= Brightness2)){


            // Coloreamos el píxel deseado en la
            // imagen de salida.
            data_mono[i*step_mono+j*channels_mono] = 1;
        }
    }
}

// Se aplica una erosión y una dilatación,
// para la eliminacion de píxeles sueltos.
cvErode(mono_Image,mono_Image,0,Erode);
cvDilate( mono_Image,mono_Image,0,Dilate);

// Eliminación de estructuras innecesarias.
cvReleaseImage(&hsv_Image);

// Devolución de la imagen de salida.
return mono_Image;
}

```

Extracción de objetos

Una vez obtenida la imagen con los píxeles seleccionados, el proceso de extracción de cada uno de los objetos y su análisis por separado se ha realizado de modo idéntico al utilizado en el primer prototipo. Ver sección 7.1.2.

Obtención de datos representativos

Una vez aplicada la segmentación por color siguiendo el modelo de color HSV, disponemos una nueva imagen la cual presenta unas agrupaciones de píxeles que han resultado seleccionados de la etapa anterior. Esas agrupaciones de píxeles son los objetos que pasaremos a extraer sus características, que en nuestro caso son objetos que conocemos a que clase pertenecen y que deberemos de aplicar un procesamiento con el fin de extraer

los datos y almacenarlos en la matriz modelos.

En esta ocasión, a diferencia del primer prototipo, en vez de aplicar la comparación de plantillas, se utilizará la matriz de modelos con aquellos datos que resulten identificativos de las clases. Dicha matriz tiene como objetivo proporcionar información de las diferentes señales de tráfico siendo un medio para cotejar los datos de un elemento desconocido y llevar a cabo su posterior clasificación.

Entre los datos extraídos disponemos:

- Forma de la figura:
 - ◊ Círculo.
 - ◊ Octógono.
- Perímetro del objeto/s interior/es.
- Área del objeto/s interior/es.
- Número de piezas interiores.
- Número de vértices de las figuras interiores.

A continuación se describe la metodología seguida para el cálculo de los datos anteriormente enumerados:

Forma de la figura: Se consideró necesario la realización de un algoritmo para la identificación de las formas externas de los objetos a ser analizados con el fin de poder distinguir correctamente entre los objetos circulares y octogonales, que son las que reconocerá el sistema. Aunque el algoritmo desarrollado también se ha desarrollado para la detección de formas triangulares y cuadradas.

Para la detección de circunferencias se ha empleado un algoritmo denominado la Transformada de Hough, el cual es incorporado en la biblioteca OpenCV. Este algoritmo es ampliamente utilizado en reconocimiento de patrones en imágenes ya que permite encontrar ciertas formas dentro de una imagen, como líneas, círculos, etc.

El algoritmo es aplicable a una imagen binarizada obtenida tras la realización de una detección de bordes devolviendo el número de círculos y su secuencia de puntos de cada uno de ellos.

Se consulta el radio de cada círculo y nos quedamos con el mayor, si el radio es de una tamaño considerable, se acepta la figura como un círculo.

Para la detección de las figuras octogonales se ha procedido a calcular mediante las funciones propias de la biblioteca OpenCV, obteniendo el polígono aproximado de tamaño mínimo que encierra la figura y consultando su número de vértices. Si dicho número se

corresponde con ocho se procede a calcular el ángulo de cada uno de esos vértices.

Para ello se ha desarrollado la siguiente función:

Cálculo del ángulo entre dos vectores

```
double angulo( CvPoint* pt1, CvPoint* pt2, CvPoint* pt0 )
{
    double dx1 = pt1->x - pt0->x;
    double dy1 = pt1->y - pt0->y;
    double dx2 = pt2->x - pt0->x;
    double dy2 = pt2->y - pt0->y;

    return ((dx1*dx2 + dy1*dy2)/sqrt((dx1*dx1 + dy1*dy1)*
                                         (dx2*dx2 + dy2*dy2) + 1e-10));
}
```

La función calcula el ángulo de dos vectores obtenidos a partir de las coordenadas de entrada. La fórmula para calcular el ángulo entre dos vectores es la siguiente:

$$\cos \alpha = \frac{u_1 \cdot v_1 + u_2 \cdot v_2}{\sqrt{u_1^2 + u_2^2} \cdot \sqrt{v_1^2 + v_2^2}}$$

Figura 7.12: Fórmula para el cálculo del ángulo entre dos vectores.

Si los ángulos obtenidos, valor de α , para cada vértice se sitúan en torno a los 45° se determina el objeto como un octógono.

A continuación se añade el código para la detección de las formas de las figuras, la cual realiza la llamada a la función ángulo antes mencionada.

Código para el cálculo del ángulo entre dos rectas dadas tres coordenadas

```
//Localiza la forma de la figura a partir de un polígono.

int tipo_figura(CvSeq* result, IplImage* plantilla){

    double PI = 3.1415;

    double A,B,C,D,E,F,G,H;

    // CUADRADO, cuatro vértices, área superior a 1000
```

```

// y estructura convexa. En un conjunto convexo se puede ir
// de cualquier punto a cualquier otro en vía recta, sin salir del mismo.
// Formalmente: para todo (A,B) pertenece a C -> [A,B] pertenece a C.
if(result->total==4 && fabs(cvContourArea(result,CV_WHOLE_SEQ))
> 1000 && cvCheckContourConvexity(result))
{
    CvPoint *pt[4];

    // Extraemos los puntos de los vértices.
    for(int i=0;i<4;i++)
        pt[i] = (CvPoint*)cvGetSeqElem(result, i);

    // Procedemos a calcular el ángulo de los 4 vértices
    // (condición: 90 grados cada uno).
    A = (acos(angulo(pt[3],pt[1],pt[0])) * 180)/ PI;
    B = (acos(angulo(pt[0],pt[2],pt[1])) * 180)/ PI;
    C = (acos(angulo(pt[3],pt[1],pt[2])) * 180)/ PI;
    D = (acos(angulo(pt[0],pt[2],pt[3])) * 180)/ PI;

    if( A < 92 && A > 88 && B < 92 && B > 88 && C < 92
        && C > 88 && D < 92 && D > 88 ){
        // Cuadrado detectado.
        return 4;
    }
}

// OCTÓGONO
if(result->total==8 && fabs(cvContourArea(result,CV_WHOLE_SEQ))
> 1000 && cvCheckContourConvexity(result))
{
    // Extraemos los puntos de los vértices
    CvPoint *pt[8];
    for(int i=0;i<8;i++)
        pt[i] = (CvPoint*)cvGetSeqElem(result, i);

    A = (acos(angulo(pt[7],pt[1],pt[0])) * 180)/ PI;
    B = (acos(angulo(pt[0],pt[2],pt[1])) * 180)/ PI;
    C = (acos(angulo(pt[3],pt[1],pt[2])) * 180)/ PI;
    D = (acos(angulo(pt[4],pt[2],pt[3])) * 180)/ PI;
    E = (acos(angulo(pt[5],pt[3],pt[4])) * 180)/ PI;
    F = (acos(angulo(pt[6],pt[4],pt[5])) * 180)/ PI;
    G = (acos(angulo(pt[7],pt[5],pt[6])) * 180)/ PI;
    H = (acos(angulo(pt[0],pt[6],pt[7])) * 180)/ PI;

    if( A < 47 && A > 43 && B < 47 && B > 43

```

```

    && C < 47 && C > 43 && D < 47 &&
    D > 43 && E < 47 && E > 43 && F < 47
    && F > 43 && G < 7 && G > 43 && H < 47
    && H > 43){

        // Octógono detectado.
        return 8;
    }

}

// TRIANGULO
if(result->total==3)
{
    // Extraemos los puntos de los vértices
    CvPoint *pt[3];
    for(int i=0;i<3;i++)
        pt[i] = (CvPoint*)cvGetSeqElem(result, i);

    A = (acos(angulo(pt[2],pt[1],pt[0])) * 180)/ PI;
    B = (acos(angulo(pt[0],pt[2],pt[1])) * 180)/ PI;
    C = (acos(angulo(pt[0],pt[1],pt[2])) * 180)/ PI;

    if( A < 126 && A > 115 && B < 126 && B > 115 &&
        C < 126 && C > 115 ){
        // Triángulo detectado.
        return 3;
    }

}

// Forma no identificada.
return 0;
}

```

Perímetro, área y número de objetos y vértices interiores:

Los datos restantes se han obtenido con haciendo uso de las funciones propias de OpenCV. A continuación se muestra el código realizado para la extracción de características:

Código para la extracción de características de un objeto.

```
CvMat extraeDatosIMG(IplImage* candidato){
```

```
CvMat *DatosExtraidos = cvCreateMat(1,5,CV_32FC1);

// Se almacenan las características de la imagen en un vector:

// Columna 0, contiene su forma, 3 -> triángulo,
// 4 -> cuadrado, 8 -> octógono, 9 -> círculo.

// Columna 1, contiene el perímetro del objeto/objetos
// situado en interior de la señal.

// Columna 2, contiene el área del objeto/objetos situado
// en interior de la señal.

// Columna 3, contiene el número de vértices del objeto/objetos
// situado en interior de la señal.

// Columna 4, contiene el número de objetos situados en
// interior de la señal.

// Declaraciones
CvSeq* circles, *result,*contour,*contour2,*contour3;

IplImage *color_seleccionado,*dibujo_interior,*gray,*binaria,*basura1,
*basura2;

double area=0, perimetro=0;
int piezas=0, vertices=0;
CvSeq *result2;

CvMemStorage* storage=cvCreateMemStorage(0),*storage2=cvCreateMemStorage(0),
*storage3=cvCreateMemStorage(0),*storage4=cvCreateMemStorage(0);

// Paso a escala de grises de la imagen original.
gray = cvCreateImage( cvGetSize(candidato), IPL_DEPTH_8U, 1 );
cvCvtColor(candidato, gray, CV_RGB2GRAY );

// Binarización de la imagen anterior.
binaria = cvCreateImage( cvGetSize(candidato), IPL_DEPTH_8U, 1 );
cvThreshold(gray,binaria,100,255, CV_THRESH_BINARY );

// Se busca contorno exterior (CV_RETR_EXTERNAL) de la imagen.
piezas = cvFindContours(binaria, storage, &contour, sizeof(CvContour),
CV_RETR_EXTERNAL,CV_CHAIN_APPROX_SIMPLE);

int forma_figura =0;
```

```

// Si el contorno es único en la imagen
// (solo un contorno externo) se analiza.
if( piezas == 1 ){

    cvDrawContours(binaria, contour, cvRealScalar(255), cvRealScalar(255),
    0, CV_FILLED,8);

    // Detectamos si se trata de un círculo
    circles = cvHoughCircles(binaria, storage4, CV_HOUGH_GRADIENT,
    2,600,10,11,90,300);

    if(circles->total == 1){ // Si es un círculo...
        forma_figura=9;
        cvSetReal2D(DatosExtraidos,0,0,9);
    }
    else{ // Si no, se busca que otra figura puede ser.

        // Detección del polinomio APROXIMADO que encierra la figura.
        result = cvApproxPoly(contour, sizeof(CvContour), storage,
        CV_POLY_APPROX_DP,15, 0);

        forma_figura = tipo_figura(result,candidato);
        cvSetReal2D(DatosExtraidos,0,0,forma_figura);
    }

    // Si la forma de la figura es alguna de las buscadas...
    if (forma_figura != 0){

        dibujo_interior = cvCreateImage(cvGetSize(candidato), IPL_DEPTH_8U,1);
        basura1 = cvCreateImage(cvGetSize(candidato), IPL_DEPTH_8U,1);
        basura2 = cvCreateImage(cvGetSize(candidato), IPL_DEPTH_8U,1);

        cvZero(dibujo_interior);
        cvZero(basura1);
        cvZero(basura2);

        // Se localizan los contornos interiores
        cvFindContours(binaria, storage2, &contour2, sizeof(CvContour),
        CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE);

        // Se dibujan:

        // Dibuja contorno externo.
        cvDrawContours(basura1, contour2, CV_RGB(255,255,255),
        CV_RGB(255,255,255),1,1,8);
    }
}

```

```

// Dibuja todos los contornos.
cvDrawContours(basura2, contour2, CV_RGB(255,255,255),
CV_RGB(255,255,255),10,1,8);

// Se realiza la diferencia de ambas imágenes para quedarnos
// con los contornos internos.
cvSub(basura2,basura1,dibujo_interior,NULL);

// Extracción de características del dibujo interior.
piezas = cvFindContours(dibujo_interior, storage3,
&contour3, sizeof(CvContour),CV_RETR_TREE,CV_CHAIN_APPROX_SIMPLE);

if(piezas > 0){ // Si existen piezas interiores
    CvSeq *contour_aux = contour3;
    for(; contour3 != 0; contour3 = contour3->h_next ){
        // Se calcula el área del objeto.
        area = area + fabs(cvContourArea(contour3));

        // Se calcula el perímetro del objeto.
        perimetro = perimetro + cvContourPerimeter(contour3);

        //Número de vértices.
        result2 = cvApproxPoly(contour3, sizeof(CvContour),
storage3, CV_POLY_APPROX_DP,1.5, 0);

        vertices = vertices + result2->total;
    }

    if(contour_aux!=NULL){
        cvClearSeq(contour_aux);
    }

    // Eliminación de estructuras.
    cvReleaseImage( &basura2 );
    cvReleaseImage( &basura1 );
    cvReleaseImage( &candidato );
    cvReleaseImage( &color_seleccionado );
    cvReleaseImage(&dibujo_interior);
    cvReleaseImage(&gray);
    cvReleaseImage(&binaria);

}
}

cvShowImage( "Plantilla",candidato );

```

```

    }

// Se devuelven los datos extraídos de la imagen.
return *DatosExtraidos;
}

```

Todos estos datos obtenidos durante la etapa de entrenamiento conforman un patrón, en la tabla 7.1 se puede apreciar los diferentes datos obtenidos para algunas de las clases del sistema:

Señal:						
Identificador:	1	2	3	4	5	6
Figura externa:	8	9	9	9	9	9
Perímetro:	506.877	1025.25	1565.18	1971.43	2510.69	2979.51
Área:	8384.5	12774.5	17323.5	21771	26734	30611.5
Número de vértices:	39	77	125	156	197	236
Piezas interiores:	4	2	2	2	2	2

Tabla 7.1: Tabla con los datos extraídos para algunas de las clases (señales) a identificar.

Por tanto la matriz de modelos obtenida queda de la siguiente manera:

$$M = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & \dots \\ 8 & 9 & 9 & 9 & 9 & 9 & 9 & 9 & \dots \\ 506,877 & 1025,25 & 1565,18 & 1971,43 & 2510,69 & 2979,51 & 3406,49 & 3828,48 & \dots \\ 40018384,5 & 12774,5 & 17323,5 & 21771 & 26734 & 30611,5 & 35896,5 & 40857 & \dots \\ 39 & 77 & 125 & 156 & 197 & 236 & 264 & 299 & \dots \\ 4 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & \dots \end{pmatrix}$$

7.2.2. Clasificación

Para la clasificación se ha empleado un algoritmo denominado *el vecino más cercano* haciendo uso de los datos extraídos del objeto desconocido y de la matriz modelos obtenida durante la etapa de entrenamiento.

Para proceder a la extracción de los datos del objeto desconocido se ha realizado los mismos pasos que durante la etapa de entrenamiento solo que aplicándose a una imagen de entrada, en lugar de aplicarse con las imágenes reservadas para entrenar el sistema.

Una vez obtenido el vector de característica del elemento desconocido, se realiza la llamada al algoritmo del vecino más cercano recibiendo la matriz de modelos previamente obtenida de la etapa de entrenamiento junto con el vector de característica del

objeto a identificar. La figura 7.13 muestra una imagen representativa de un vector característica del que cuya clase de pertenencia es desconocida:

$$V_{desconocido} = \begin{pmatrix} 8 \\ 502,237 \\ 4022 \\ 34 \\ 4 \end{pmatrix}$$

Figura 7.13: Vector de características de un elemento desconocido.

Como podemos apreciar, el objeto desconocido (del que no se conoce su clase) es representado por una matriz columna. A continuación se calcula la distancia euclídea entre cada uno de los vectores columnas integrantes de la matriz modelos y el vector desconocido, seleccionándose el ejemplo más cercano. El nuevo ejemplo es clasificado con la clase que pertenece el vector seleccionado dentro de la matriz modelos.

Este método supone que el vecino más cercano nos dé la mejor clasificación y esto se hace utilizando todos los atributos; el problema de dicha suposición es que es posible que se tengan muchos atributos irrelevantes que dominen sobre la clasificación: dos atributos relevantes perderían peso entre otros irrelevantes.

A continuación se muestra la fórmula para el cálculo de la distancia euclídea, definida como la raíz cuadrada de las sumas de las diferencias de las componentes al cuadrado:

$$d(x_i, y_j) = \sqrt{\sum_{r=1}^p (x_{ir} - x_{jr})^2}$$

Figura 7.14: Fórmula para el cálculo de la distancia euclídea, definida como la raíz cuadrada de las sumas de las diferencias de cada una de las componentes del vector al cuadrado.

A continuación se muestra el código realizado para la clasificación utilizando el algoritmo del vecino más cercano:

Aplicación del vecino más cercano

```
int Clasificador(CvMat DatosEntrenamiento, CvMat DatosCandidato){
    // Aplicación del algoritmo KNN.
```

```

const int K = 1; // Se fija valor de K. (K=1, vecino más cercano).
int solucion; // Solución

// Se crea e inicializa el vector de clases, cada elemento columna de
// DatosEntrenamiento es una clase

int clases[elementos];
CvMat Clases = cvMat(elementos, 1, CV_32FC1, clases);

for (int a=0;a<DatosEntrenamiento.rows;a++){
    for (int b=0;b<Clases.cols;b++){
        // Se añaden los identificadores.
        cvSetReal2D(&Clases,a,b,a+1);
    }
}

// Se crea el clasificador.
CvKNearest knn( &DatosEntrenamiento, &Clases, 0, false, K );
CvMat* nearests = cvCreateMat( 1, K, CV_32FC1 );

// Se estima el vecino más cercano.
solucion = knn.find_nearest(&DatosCandidato,K,0,0,nearests,0);

// Se devuelve el resultado.
return solucion;
}

```

7.2.3. Problemas detectados

Los problemas que han llevado a cabo el rechazo de este segundo prototipo han sido los siguientes:

- Los datos obtenidos durante la extracción de características tales como área, perímetro, número de vértices, número de objetos interiores de la señal, entre otros, no resultaban determinantes y excluyentes a la hora de realizar la clasificación proporcionando un alto número de errores cuando el objetivo es elaborar un sistema lo más fiable y eficiente posible.
- Como punto positivo, se ha comprobado que el algoritmo de los k-vecinos presenta un funcionamiento más adecuado a las características exigibles del proyecto trabajando con los datos de entrada adecuados ya que proporciona una buena relación entre tiempo computacional y efectividad. Por otro lado, el algoritmo presenta la posibilidad de identificación de más de una clase ya que como entrada es capaz de recibir datos de distintas clases a identificar como no ocurría con la comparación de plantillas . Por estas razones se ha considerado adecuado continuar el desarrollo del siguiente prototipo haciendo uso del algoritmo K-vecinos

más cercanos cambiando los datos de entrada por otro conjunto de datos que hagan mejorar su funcionamiento.

7.3. Tercer y último prototipo

Dado que el prototipo anterior presentaba una baja relación de aciertos, y observando la idoneidad de continuar con el algoritmo del k-vecinos más cercano con valores de K = 1. Se ha optado por continuar con la misma técnica de clasificación cambiando los datos integrantes de la matriz de modelos por unos más representativos.

7.3.1. Extracción de características

Para la extracción de las características necesarias para la creación de la matriz de modelos por parte del algoritmo de entrenamiento han sido necesarias una serie de etapas que se describen en las sucesivas secciones.

Segmentación por color

Como etapa inicial, se ha seguido con la metodología empleada en el segundo prototipo utilizando, del igual modo, una segmentación por color aplicando el modelo de color HSV. La única salvedad radica en que en esta ocasión se han añadido los valores necesarios para filtrar las tonalidades azules. En la sección [7.2.1](#), referente al segundo prototipo, se describe al detalle la metodología seguida.

Obtención de datos representativos

En esta ocasión se ha procedido extraer de forma separada los elementos interiores de una señal de tráfico, para ello a cada objeto identificado tras la selección de color se le aplica un relleno y se extrae mediante máscaras el objeto de la imagen original siguiendo la misma metodología que en el primer prototipo. Ver sección [7.1.2](#) para consultar el proceso detalladamente.

Posteriormente, la imagen obtenida con el objeto candidato es pasada a escala de grises y se identificaban los contornos. Cada uno de los contornos detectados se corresponden con un elemento interno de la posible señal de tráfico. Los contornos son dibujados rellenando sus huecos vacíos con el fin de eliminar los contornos interiores.

Cada elemento es a su vez extraído en una nueva imagen donde es redimensionado a un tamaño fijado y dispuesto en un vector columna, dicho vector columna es utilizado para ser agregado a la matriz modelos. La figura [7.15](#) muestra una representación del proceso seguido.

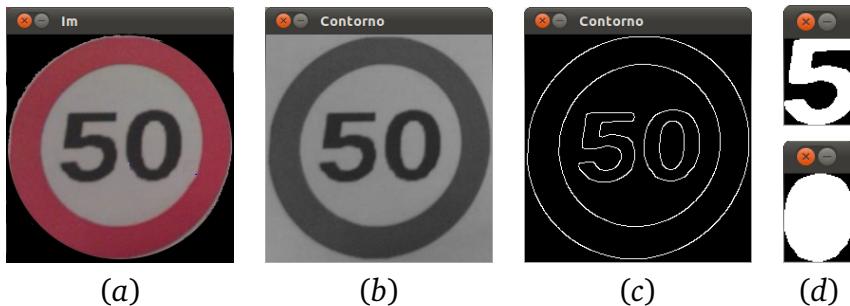


Figura 7.15: Proceso seguido para la extracción de los objetos interiores de una señal.

En la imagen (d) de la figura 7.15 se visualiza el elemento 0 y el 5 por separado. Ambas matrices representativas de las imágenes, junto con las del resto de elementos a identificar, son dispuestas en vectores columnas y agregadas a la matriz de modelos junto con su identificador, se ha empleado como identificador un carácter numérico para el caso de los números, caracteres alfabéticos para las letras implicadas como las existentes en la señal de Stop y otros caracteres para el resto de elementos como las flechas indicadoras de dirección. Ver figuras 7.16 y 7.17.

$$V_1 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Figura 7.16: Representación del número cinco en una matriz.

Figura 7.17: Resultado de la transformación de la matriz V_1 en vector columna añadiendo las diferentes columnas de V_1 sucesivamente una bajo la otra en V_2 . El vector resultante posee en la primera posición el identificador de la clase, en el caso mostrado el carácter “5”.

7.3.2. Clasificación

Para proceder con la clasificación, previamente la matriz modelos es cargada de un fichero donde se encuentra almacenada.

Antes de realizar la clasificación hay que procesar la imagen de entrada del mismo modo que se ha realizado durante la etapa de entrenamiento. Cada elemento existente dentro de una figura circular roja es extraído y dispuesto en un vector columna para ser clasificado mediante el algoritmo del vecino más cercano. Supongamos la señal de 50km/h. Sus elementos interiores son el carácter “5” y el carácter “0”. Por tanto, si se aplica el algoritmo del vecino más cercano nos producirá como salida en la primera llamada un “5” y en la segunda un “0”. Dado que ambos elementos se encuentran dentro de una circunferencia de color rojo, se determina que el objeto procesado se trata de la señal de 50km/h. Ver figura 7.18.



Figura 7.18: Identificación de los elementos interiores de la señal, el valor “5” y el valor “0”, determinándose el objeto como una señal de 50 km/h.

El algoritmo de clasificación empleado ha sido el mismo que el utilizado en el prototipo anterior con la salvedad de que en esta ocasión se rechazan aquellos valores de salida cuya distancia supera un determinado umbral. Con esta modificación los elementos serán clasificados cuando su grado de parentesco es elevado y no con la clase con la que más se le parece aunque sea un objeto totalmente diferente.

A continuación se muestra el código modificado:

Algoritmo del vecino más cercano modificado para la obtención de las distancias

```

int Clasificador(CvMat DatosEntrenamiento, CvMat DatosCandidato){

    // Aplicación del algoritmo KNN.

    const int K = 1; // Se fija valor de K. (K=1, vecino más cercano).
    int solucion;    // Solución
    int umbral_minimo = 10000; // Grado de semejanza exigido.

    // Vector donde se almacenarán las distancias.
    CvMat* distancias = cvCreateMat(1,K,CV_32FC1);

    // Se crea e inicializa el vector de clases, cada elemento columna de
    // DatosEntrenamiento es una clase.

    int clases[elementos];
    CvMat Clases = cvMat(elementos,1, CV_32FC1,clases);

    for (int a=0;a<DatosEntrenamiento.rows;a++){
        for (int b=0;b<Clases.cols;b++){
            // Se añaden los identificadores.
            cvSetReal2D(&Clases,a,b,a+1);
        }
    }
}

```

```
// Se crea el clasificador.  
CvKNearest knn( &DatosEntrenamiento, &Clases, 0, false, K );  
CvMat* nearests = cvCreateMat( 1, K, CV_32FC1 );  
  
// Se estima el vecino más cercano.  
solucion = knn.find_nearest(elemento_analisis,K,0,0,nearests,distancias);  
  
//Si hay bastante semejanza, aceptamos como válido.  
if (cvGetReal2D(distancias,0,0) < umbral_minimo){  
    return solucion;  
}  
// Si no se devuelve cero.  
return 0;  
}
```

Capítulo 8

Software de control

En presente capítulo contiene información relativa a las herramientas software utilizadas y desarrolladas para la realización de la comunicación con el vehículo ordenando acciones como movimientos, modificación de velocidad, entre otros. Todo ello desde el ordenador por parte del usuario.

Cabe recordar que la aplicación consta de dos modos de funcionamiento; un modo automático donde los movimientos son realizados a partir de los elementos visionados por la cámara, y un segundo modo donde el vehículo es manejable por el usuario. En este tema de desarrollan ambos aspectos.

Para el control del vehículo se ha utilizado un software denominado *Surveyor Robot Software*¹, el cual ha sido desarrollado por John Cummins junto con los agentes de laboratorio de la Universidad de Brooklyn con la asistencia de M.P. Azhar, y la supervisión del profesor Sklar empleando el lenguaje de programación C++. El código está liberado bajo Copyleft.

El código *Surveyor robot software* ha sido ligeramente modificado para adaptarlo a las necesidades del proyecto, la principal modificación añadida ha sido la de añadir un método para permitir la conexión y desconexión con el vehículo. Permitiendo al usuario desconectar y reconectar a su antojo.

8.1. Control por parte del usuario

Lo interesante era proporcionar una serie de mecanismos que permitiera mantener un control del vehículo de una manera cómoda y eficaz para el usuario. Los elementos proporcionados para realizar el control del vehículo son el uso del teclado del ordenador y un gamepad.

¹Para la descarga de código fuente visite el siguiente enlace <http://agents.sci.brooklyn.cuny.edu/robotics.edu/bcsoftware.php>.

8.1.1. Control desde teclado

Los movimientos clásicos del vehículo, avance, retroceso y giros, han sido ordenados mediante el uso de las teclas de dirección. Para las capturas de los eventos desde el teclado se han utilizado las herramientas proporcionadas por la biblioteca Qt.

De este modo, el vehículo, al presionar la tecla de avance, comenzaba a andar, pero no se detenía hasta nueva orden haciendo incómodo su manejo. Como solución al problema se optó por capturar el evento de “soltado” de tecla para, en ese instante, mandar una orden de detención de tal manera que el vehículo avance sólo y exclusivamente mientras se encuentra la tecla pulsada.

Por otro lado, el robot SRV-1, al tratarse de un vehículo tipo tanque, presenta la particularidad de poder controlar las velocidades de movimiento de los motores de un mismo lateral por igual permitiendo la realización de giros. Como consecuencia de lo anterior, si se aplica una velocidad levemente inferior en un lateral que en el otro el vehículo realizará un giro muy abierto. De modo contrario, si detenemos por completo los motores de un lado, el giro producido será de pequeña amplitud. Esta circunstancia permite que desde las flechas de teclado no podamos controlar tal efecto, puesto que los giros serán siempre iguales, que para el caso de las teclas de dirección será únicamente moviendo los motores de un lateral.



Figura 8.1: Visualización del chasis del robot y su sistema de desplazamiento tipo tanque.

Para permitir realizar ambos tipos de giros se pensó en capturar combinaciones de teclas, de tal modo que si se presiona la tecla “derecha”, el vehículo realice un giro cerrado y si se presiona la tecla “arriba” más “derecha”, el giro será de mayor amplitud. Los diferentes movimientos existentes en el vehículo quedan descritos en la sección [10.4.1](#) del manual.

Para su realización, Qt no dispone de los elementos necesarios para capturar combinaciones de teclas, solamente es posible con las teclas “Shift”, “Ctrl”, “Alt” y “Meta”.

Como solución al problema de la captura de eventos múltiples se implementó un buffer donde se almacenaba un identificador de las dos últimas teclas pulsadas, eliminándose un identificador del buffer en el momento en que la tecla sea soltada. De ese modo tras la captura de un evento, se analizaba el estado del buffer y se determinaba la orden de movimiento a realizar. Obteniendo los movimientos descritos en la sección [10.4.1](#) del manual.

En la siguiente imagen podemos ver una representación gráfica del comportamiento del buffer de control del vehículo:



Figura 8.2: Representación del funcionamiento del buffer creado para la captura de eventos múltiples.

8.1.2. Control mediante gamepad

Para hacer más cómodo, aún si cabe, el manejo del vehículo, se decidió incorporar un mando o gamepad, para ello ha sido necesaria la utilización de la biblioteca SDL con el fin de detectar el dispositivo y capturar los eventos correspondientes no presentando dificultades destacables. Para ver los controles fijados vea sección [10.4.1](#) de la guía de usuario.

8.2. Control automático

Para dotar al vehículo de un sistema de conducción autónoma se han diseñado una serie de respuestas automáticas ordenadas cuando una determinada señal es detectada. En esta sección se describen cada una de ellas.

Entre el conjunto de señales de tráfico reconocibles por el sistema disponemos de las indicadoras de velocidad máxima. Estas señales detectables varían desde los 20 hasta los 100 km/h. Cuando una determinada velocidad es detectada, ésta debe ser asignada al vehículo con la finalidad de realizar los movimientos de una manera proporcional a la velocidad indicadora de la señal reconocida.

La biblioteca *Surveyor Robot Software* permite ajustar las velocidades de movimiento del vehículo en una escala desde el 0 hasta el 100. Por tanto, si la señal de tráfico detectada se corresponde con la de 50km/h, se asigna al vehículo un valor de velocidad de 50.

En el momento de la detección de la señal de Stop, se ordena su detención manteniéndose intacto el valor de velocidad que se encuentra en ese momento fijado.

Para las señales correspondientes a las indicadoras de dirección obligatoria (flechas), se efectúa un giro de manera automática. En la sección [10.6](#) del manual se muestra al detalle los diferentes movimientos realizados para cada señal de tráfico detectada.

Por otro lado, cuando una señal es detectada, la orden automatizada es enviada al vehículo una sola vez de tal modo que si en el entorno apareciese más adelante la misma señal reconocida anteriormente, ésta no se volverá a realizar. En definitiva, no se reconoce la misma señal de tráfico dos veces seguidas.



Figura 8.3: Fotografía de un circuito realizable por el vehículo .

A continuación se muestra el código realizado para la realización de los movimientos autónomos. En ella se analiza un vector para comprobar si contiene los identificadores de las clases de aquellos objetos que se correspondan con una señal. Por ejemplo, si el vector contiene los identificadores “2” y “0” determinamos que el elemento detectado es una señal de 20 km/h.

Código para el control automatizado del vehículo

```
int MainWindow::accion_SVR(std::vector<char> &v){

    // 100 km/h
    if ((std::find(v.begin(), v.end(), '1') != v.end()) &&
        (std::find(v.begin(), v.end(), '0') != v.end())){
        // Se elimina un '0'.
        std::vector<char>::iterator pos = std::find(v.begin(), v.end(), '0');
        v.erase(pos);

        // Si existe otro cero...
        if (std::find(v.begin(), v.end(), '0') != v.end()){

    }
}
```

```

// 100 km/h
if(conectado && ui->checkBox_2->isChecked() && ultima_senial != 10){
    ultima_senial = 10;
    robot.move(STOP);
    this->velocidad = 100;
    robot.drive(this->velocidad,this->velocidad);
}
//Mostrar señal detectada
mostrar_senial_detectada("10.jpg");

return 0;
}

// 20 km/h
if ((std::find(v.begin(), v.end(), '2') != v.end()) &&
    (std::find(v.begin(), v.end(), '0') != v.end())){
    if(conectado && ui->checkBox_2->isChecked() && ultima_senial != 2){
        ultima_senial = 2;
        robot.move(STOP);
        this->velocidad = 20;
        robot.drive(this->velocidad,this->velocidad);
    }
    //Mostrar señal detectada
    mostrar_senial_detectada("2.jpg");
    return 0;
}

// 30 km/h
if ((std::find(v.begin(), v.end(), '3') != v.end()) &&
    (std::find(v.begin(), v.end(), '0') != v.end())){
    if(conectado && ui->checkBox_2->isChecked() && ultima_senial != 3){
        ultima_senial = 3;

        robot.move(STOP);
        this->velocidad = 30;
        robot.drive(this->velocidad,this->velocidad);
    }
    //Mostrar señal detectada
    mostrar_senial_detectada("3.jpg");
    return 0;
}

```

```

// 40 km/h
if ((std::find(v.begin(), v.end(), '4') != v.end()) &&
    (std::find(v.begin(), v.end(), '0') != v.end())){
    if(conectado && ui->checkBox_2->isChecked() && ultima_senial != 4){
        ultima_senial = 4;
        robot.move(STOP);
        this->velocidad = 40;
        robot.drive(this->velocidad,this->velocidad);
    }
    //Mostrar señal detectada
    mostrar_senial_detectada("4.jpg");
    return 0;
}

// 50 km/h
if ((std::find(v.begin(), v.end(), '5')!=v.end()) &&
    (std::find(v.begin(), v.end(), '0') != v.end())){
    if(conectado && ui->checkBox_2->isChecked() && ultima_senial != 5){
        ultima_senial = 5;
        robot.move(STOP);
        this->velocidad = 50;
        robot.drive(this->velocidad,this->velocidad);
    }
    //Mostrar señal detectada
    mostrar_senial_detectada("5.jpg");
    return 0;
}

// 60 km/h
if ((std::find(v.begin(), v.end(), '6')!= v.end()) &&
    (std::find(v.begin(), v.end(), '0') != v.end())){
    if(conectado && ui->checkBox_2->isChecked() && ultima_senial != 6){
        ultima_senial = 6;
        robot.move(STOP);
        this->velocidad = 60;
        robot.drive(this->velocidad,this->velocidad);
    }
    //Mostrar señal detectada
    mostrar_senial_detectada("6.jpg");
    return 0;
}

```

```

// 70 km/h
if ((std::find(v.begin(), v.end(), '7') != v.end()) &&
    (std::find(v.begin(), v.end(), '0') != v.end())){
    if(conectado && ui->checkBox_2->isChecked() && ultima_senial != 7){
        ultima_senial = 7;
        robot.move(STOP);
        this->velocidad = 70;
        robot.drive(this->velocidad,this->velocidad);
    }

    mostrar_senial_detectada("7.jpg");
    return 0;
}

// 80 km/h
if ((std::find(v.begin(), v.end(), '8') != v.end()) &&
    (std::find(v.begin(), v.end(), '0') != v.end())){
    if(conectado && ui->checkBox_2->isChecked() && ultima_senial != 8){
        ultima_senial = 8;
        robot.move(STOP);
        this->velocidad = 80;
        robot.drive(this->velocidad,this->velocidad);
    }

    //Mostrar señal detectada
    mostrar_senial_detectada("8.jpg");
    return 0;
}

// 90 km/h
if ((std::find(v.begin(), v.end(), '9') != v.end()) &&
    (std::find(v.begin(), v.end(), '0') != v.end())){
    if(conectado && ui->checkBox_2->isChecked() && ultima_senial != 9){
        ultima_senial = 9;

        robot.move(STOP);
        this->velocidad = 90;
        robot.drive(this->velocidad,this->velocidad);
    }

    //Mostrar señal detectada
}

```

```

    mostrar_senial_detectada("9.jpg");
    return 0;
}

// Stop
if ((std::find(v.begin(), v.end(), 't') != v.end()) &&
    (std::find(v.begin(), v.end(), 'p') != v.end())){
    if(conectado && ui->checkBox_2->isChecked() && ultima_senial != 1){
        ultima_senial = 1;
        robot.move(STOP);
        robot.drive(this->velocidad,this->velocidad);
    }
}

//Mostrar señal detectada
mostrar_senial_detectada("1.jpg");
return 0;
}

// Derecha
if (std::find(v.begin(), v.end(), 'd') != v.end() && v.size() == 1){

    if(conectado && ui->checkBox_2->isChecked() && ultima_senial != 12){
        ultima_senial = 12;

        robot.move(STOP);
        robot.drive(0,this->velocidad,10);
        robot.drive(this->velocidad,this->velocidad);
    }
}

//Mostrar señal detectada
mostrar_senial_detectada("12.jpg");

return 0;
}

// Izquierda
if (std::find(v.begin(), v.end(), 'i') != v.end() && v.size() == 1){

    if(conectado && ui->checkBox_2->isChecked() && ultima_senial != 13){
        ultima_senial = 13;
    }
}

```

```
    robot.move(STOP);
    robot.drive(this->velocidad,0,10);
    robot.drive(this->velocidad,this->velocidad);
}

//Mostrar señal detectada
mostrar_senial_detectada("13.jpg");

return 0;

}

return 1;
}
```

Capítulo 9

Interfaz gráfica

9.1. Elementos de la interfaz gráfica

Se consideró antes de comenzar el desarrollo de la interfaz gráfica, una serie de elementos mínimos imprescindibles que dicha interfaz debería incorporar.

Dada las características del proyecto, era necesario proporcionar a la aplicación de una interfaz gráfica sencilla pero a la vez funcional teniendo como principal objetivo ofrecer al usuario, tras una visión rápida, la localización de los diferentes elementos para control del vehículo y deducir su funcionamiento.

En este punto se realiza una descripción de los diferentes elementos presentes en la interfaz gráfica, la cual se compone de dos ventanas, una ventana principal donde se visualiza el estado del vehículo permitiendo su control y una segunda ventana para proporcionar información o ayuda al usuario.

Los elementos incorporados en la ventana principal son los siguientes:

- Panel para el visionado de las imágenes captadas por la cámara.
- Conjunto de botones para manejo del vehículo, con imágenes representativas de su acción.
- Panel para el visionado de resultados donde se mostrarán aquellas señales de tráfico detectadas.
- Información acerca del detector de distancias.

La figura 10.1 muestra una imagen de la ventana principal de la aplicación:



Figura 9.1: Vista de la ventana principal.

En el manual de usuario se encuentra disponible toda la información acerca del modo de utilización de la interfaz gráfica. Ver sección [10](#).

La ventana secundaria proporcionara información al usuario acerca de los controles del teclado, del gamepad, y otras informaciones de uso como dirección ip del vehículo. Esta ventana ha sido desarrollada con la intención, en mejoras futuras, permitir que dichos parámetros sean configurables.

La figura [9.2](#) muestra una imagen de la ventana de información:

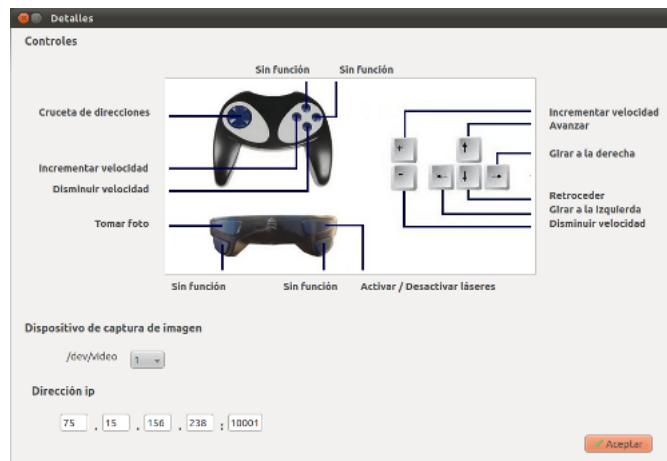


Figura 9.2: Vista de la ventana de información.

Capítulo 10

Guía de usuario

10.1. Introducción

Este documento describirá los objetivos e información de cómo utilizar la aplicación junto con sus diferentes componentes hardware de: OpenTSR.

La aplicación OpenTSR ha sido creada por Manuel López Urbina para el proyecto fin de carrera de la titulación de ingeniería técnica informática de la universidad de Cádiz.

Resulta de vital importancia consultar esta guía antes y/o durante la utilización de los diferentes elementos tanto hardware como software del proyecto OpenTSR ya que le proporcionará una guía paso a paso en el manejo de sus funciones. La resolución recomendada para la aplicación debe ser superior o igual a 1024x768 (Estándar XGA).

Con el fin de facilitar la comprensión de la guía, se incorporan gráficos explicativos. Los diferentes iconos existentes en la aplicación han sido obtenidos de la página web <http://www.iconfinder.com>.

10.1.1. Objetivo de esta guía

Esta guía tiene como objetivo la de proporcionar al usuario un soporte de ayuda e iniciación a la utilización de los diferentes elementos hardware y software integrantes del proyecto OpenTSR. Esta sección comprende:

- Guía de acceso a la aplicación.
- Guía de uso de la aplicación.
- Guía para la puesta en marcha de los dispositivos hardware.
- Guía para la configuración de los dispositivos hardware.

10.1.2. Dirigido a

Esta guía esta dirigida al usuario final del proyecto OpenTSR que utilicen como medio de control del vehículo *Surveyor SRV-1 WiFi webcam robot* para proporcionarle de un sistema de visión artificial para el reconocimiento de señales de tráfico y de conducción autónoma.

10.2. Obtener OpenTSR

La aplicación se encuentra disponible en la forja [rediris\[1\]](#) o usando la herramienta [Subversion\[10\]](#), escribiendo en la consola el siguiente comando:

```
1 svn checkout https://forja.rediris.es/svn/opentsr
```

10.3. Ingreso al sistema

Una vez descargada la aplicación hacemos doble click sobre el ícono para ejecutar la aplicación. Inmediatamente se mostrará la ventana principal mostrando el siguiente aspecto, ver Figura 10.1.



Figura 10.1: Ventana principal de la aplicación.

10.4. Convenciones y estándares a utilizar

Las convenciones y estándares utilizados son los siguientes:

Convenciones del uso del ratón

Término	Elemento aplicación	Significado
	Sobre cualquier botón.	Apertura del menú o acción designada para tal botón.

Tabla 10.1: Uso del ratón.

10.4.1. Convenciones del uso del teclado

Para teclas simples:

Tecla	Significado
	Avanzar. Orden de movimiento frontal, el vehículo se detiene al soltar la tecla.
	Retroceder. Orden de movimiento en retroceso, el vehículo se detiene al soltar la tecla.
	Giro cerrado a la derecha. Orden de giro a la derecha, el vehículo se detiene al soltar la tecla.
	Giro cerrado a la izquierda. Orden de giro a la izquierda, el vehículo se detiene al soltar la tecla.
	Incremento de velocidad. Se incrementa en una unidad la velocidad del vehículo.
	Disminución de velocidad. Se disminuye en una unidad la velocidad del vehículo.

Tabla 10.2: Controles del teclado para teclas simples.

Para combinaciones de teclas:

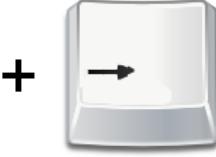
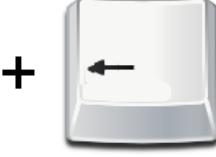
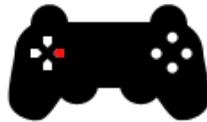
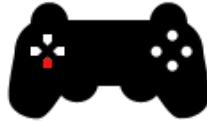
Tecla	Significado
 + 	Giro abierto a la derecha. Orden de movimiento para la realización de un giro más abierto hacia la derecha. Los motores del vehículo mueven ambas cadenas aplicando a la derecha una menor velocidad. El vehículo se detiene al soltar ambas teclas o continua con el movimiento de la tecla que se sigue manteniendo pulsada.
 + 	Giro abierto a la izquierda. Orden de movimiento para la realización de un giro más abierto hacia la izquierda. Los motores del vehículo mueven ambas cadenas aplicando a la izquierda una menor velocidad. El vehículo se detiene al soltar ambas teclas o continua con el movimiento de la tecla que se sigue manteniendo pulsada.
 + 	Giro abierto a la derecha retrocediendo. Orden de movimiento para la realización de un giro más abierto hacia la izquierda marcha atrás. Los motores del vehículo mueven ambas cadenas aplicando a la izquierda una menor velocidad. El vehículo se detiene al soltar ambas teclas o continua con el movimiento de la tecla que se sigue manteniendo pulsada.
 + 	Giro abierto a la izquierda retrocediendo. Orden de movimiento para la realización de un giro más abierto hacia la derecha marcha atrás. Los motores del vehículo mueven ambas cadenas aplicando a la derecha una menor velocidad. El vehículo se detiene al soltar ambas teclas o continua con el movimiento de la tecla que se sigue manteniendo pulsada.

Tabla 10.3: Controles del teclado para las combinaciones de teclas.

Convenciones del uso del gamepad

Botón	Significado
	Giro cerrado a la derecha. Orden de giro a la derecha, el vehículo se detiene al soltar la cruceta.
	Giro cerrado a la izquierda. Orden de giro a la izquierda, el vehículo se detiene al soltar la cruceta.
	Retroceder. Orden de movimiento en retroceso, el vehículo se detiene al soltar la cruceta.
	Avanzar. Orden de movimiento hacia delante, el vehículo se detiene al soltar la cruceta.
	Sin función.
	Disminuir velocidad. La velocidad de movimiento del vehículo disminuye.

	Incrementar velocidad. La velocidad de movimiento del vehículo aumenta.
	Sin función.
	Tomar fotografía. Realiza una captura de la imagen visualizada en pantalla.
	Sin función.
	Activación/desactivación de láseres. Los láseres son desactivados si se encuentran en estado encendido o activados si se encuentran en estado apagado.
	Sin función.

Tabla 10.4: Controles del gamepad.

10.5. Operaciones de la interfaz

En la ventana principal de la aplicación 10.1, se visualizan una serie de elementos, a continuación se describen cada uno de ellos.

- **Conecitar:**

Icono representativo:



Acción: establece conexión con el vehículo SRV-1. Una vez accionado el botón cambia su estado pasando a **Desconectar**.

- **Desconectar:**

Icono representativo:



Acción: elimina la conexión con el vehículo SRV-1. Una vez accionado el botón cambia su estado pasando a **Conecitar**.

- **Activar láser:**

Icono representativo:



Acción: activa los láseres del vehículo SRV-1. Una vez accionado el botón cambia su estado pasando a **Desactivar láser**.

- **Desactivar láser:**

Icono representativo:



Acción: desactiva los láseres del vehículo SRV-1. Una vez accionado el botón cambia su estado pasando a **Activar láser**.

- **Tomar fotografía:**

Icono representativo:



Acción: toma una fotografía de la imagen visionada por la cámara.

- **Velocidad alta:**

Icono representativo:



Acción: asigna velocidad alta de desplazamiento al vehículo SRV-1. Una vez accionado el botón cambia su estado pasando a **Velocidad baja**.

- **Velocidad baja:**

Icono representativo:



Acción: asigna velocidad baja de desplazamiento al vehículo SRV-1. Una vez accionado el botón cambia su estado pasando a **Velocidad alta**.

- **Avanzar:**

Icono representativo:



Acción: ordena avanzar al vehículo, pulsar icono *Stop* para detener o indicar nueva orden.

- **Retroceder:**

Icono representativo:



Acción: ordena retroceder al vehículo, pulsar icono *Stop* para detener o indicar nueva orden.

- **Girar a la derecha:**

Icono representativo:



Acción: ordena realizar giro a la derecha al vehículo, pulsar icono *Stop* para detener o indicar nueva orden.

- **Girar a la izquierda:**

Icono representativo: 

Acción: ordena realizar giro a la izquierda al vehículo, pulsar ícono Stop para detener o indicar nueva orden.

- **Parada:**

Icono representativo: 

Acción: ordena la parada del vehículo si se encontraba en movimiento.

- **Información controles:**

Icono representativo: 

Acción: abre una nueva ventana con información sobre los controles de la aplicación.

- **Ayuda:**

Icono representativo: 

Acción: realiza la apertura de la guía de usuario de la aplicación.

Dentro de este documento se encuentran tres posibles opciones de consulta para el usuario.

- ◊ Guía de uso de la aplicación.
- ◊ Guía hardware.
- ◊ Sobre OpenTSR.

La guía de uso servirá para solventar las dudas de uso de la aplicación. En ella se hará un recorrido por todas las funcionalidades, así como los posibles usos que tiene cada componente de la misma.

En la guía hardware se describe paso a paso, la metodología a seguir para la configuración de los elementos hardware presentes en OpenTSR junto con pasos previos a su interconexión y puesta en funcionamiento de cada uno de ellos.

- **Información:**

Icono representativo: 

Acción: realiza la apertura de la ventana “sobre OpenTSR” que hace mención sobre el creador del proyecto, el nombre del mismo y para la universidad que ha sido desarrollada la aplicación.



Figura 10.2: Ventana donde se visualiza la información relacionada con OpenTSR.

- **Iniciar:**

Icono representativo:

Acción: Se inicia la conexión con la cámara mostrándose las imágenes en la interfaz. Una vez accionado se produce un cambio de estado del botón pasando a **Detener**.

- **Detener:**

Icono representativo:

Acción: Se cierra la conexión con la cámara. Una vez accionado se produce un cambio de estado del botón pasando a **Iniciar**.

- **Piloto automático** checkbox

Acción: Se inicia el modo de pilotaje automático en el vehículo a partir de las señales de tráfico detectadas por el sistema, ver sección 10.6 para consultar la relación entre señales detectables y acción automática realizada.

10.6. Operaciones automatizadas

Al activar el botón “piloto automático” en la interfaz gráfica se produce una conducción autónoma del vehículo a partir de las señales de tráfico visionadas por la cámara.

La correspondencia entre las señales detectables y su acción ejecutada se muestra en la siguiente tabla:

Señal	Acción
	Orden automática de parada.
	Velocidad fijada a 20*.
	Velocidad fijada a 30*.
	Velocidad fijada a 40*.
	Velocidad fijada a 50*.
	Velocidad fijada a 60*.
	Velocidad fijada a 70*.
	Velocidad fijada a 80*.
	Velocidad fijada a 90*.

	Velocidad fijada a 100*.
	Efectúa un giro cerrado a la derecha.
	Efectúa un giro cerrado a la izquierda.

Tabla 10.5: Señales reconocidas por el vehículo junto con la acción realizada.

Nota. Las velocidades son asignadas al vehículo dentro de una escala de 0 a 100, siendo 0 la velocidad nula y 100 su velocidad máxima.

10.7. Montaje del sistema de comunicaciones

En esta sección se detallada el montaje del sistema de comunicaciones entre el coche y el ordenador personal para su utilización.

Primeramente se realiza la conexión del router inalámbrico enchufando el adaptador de corriente a una toma de 220V y presionando el botón de encendido. Tras la espera de unos segundos escanearemos con un ordenador las conexiones WiFi existentes para proceder a conectarnos a la red de nuestro router, en este caso SRV-1. Si el router no ha sido configurado previamente visite la sección [10.8](#).

El siguiente paso consiste en conectar el sistema de recepción de imágenes al ordenador personal. Para ello, se enchufa el adaptador de corriente del dispositivo sintonizador a la toma de 220V y éste, a su vez con la capturadora de vídeo mediante los cables RCA. Finalmente se conecta el cable USB de la capturadora de vídeo al ordenador personal tal y como muestra la figura [10.3](#). Si no se han instalado los drivers de la capturadora de vídeo visite la sección [10.10](#).



Figura 10.3: Vista de los elementos conectados.

Una vez realizadas las conexiones, en el siguiente paso se debe sintonizar el sistema de recepción de imágenes para la captura de las imágenes de la cámara inalámbrica. Para ello desplazamos el interruptor hacia cada una de sus posiciones hasta obtener una imagen nítida y legible como la que se aprecia en la figura 10.5.



Figura 10.4: Interruptor para búsqueda del canal de la cámara.



Figura 10.5: Vista de OpenTSR en funcionamiento.

Por último, se debe realizar el encendido del vehículo robótico SRV-1 Surveyor accionando el interruptor situado en la parte trasera del mismo. Ver figura 10.6. Si el robot no ha sido configurado para localizar el router y formar parte de la infraestructura de red visite la sección 10.9.



Figura 10.6: Localización del interruptor con tres modos de funcionamiento del robot SRV-1.

Finalmente pulse sobre el botón conectar de la aplicación (🔗) para establecer conexión con el vehículo y proceder a su utilización.

A partir de este momento el sistema hardware se encuentra conectado y listo para ser usado por sistema software, del que se detalla los controles y su funcionamiento en el

punto 10.5 y 10.4.

10.8. Configuración del router

El router actúa de intermediario entre el ordenador y el vehículo robótico, por tanto, dicho router debe ser configurado de tal manera de que disponga de la misma dirección ip que ha sido introducida como puerta de enlace en la configuración del SRV-1 entre otros factores.

Para proceder con la configuración del router, primeramente, debemos conectar un extremo de un cable Ethernet a nuestro equipo y el otro a una de las interfaces Ethernet que dispone el router en su parte trasera.

Una vez conectado, para acceder al programa de configuración, basta con teclear en la barra de direcciones de nuestro navegador la dirección IP del router, debemos tener en cuenta que la dirección será 192.168.1.1 si el router posee su configuración por defecto, si hemos modificado la dirección ip, habrá que teclear la que corresponda.

Para consultar la dirección ip del router introducimos en terminal el siguiente comando:

```
1 ifconfig
```

Localizamos nuestra tarjeta y visualizamos la ip correspondiente a la puerta de enlace, la cual se corresponderá con la dirección de nuestro router.

La configuración por defecto del router tiene los siguientes valores:

- Usuario: 1234
- Password: 1234
- Dirección IP: 192.168.1.1
- Red inalámbrica: deshabilitada
- Servidor DHCP: habilitado, dirección IP inicial 192.168.1.33 , dirección IP final 192.168.1.254

Una vez hemos accedido al menú de configuración del router debemos configurar los siguientes parámetros:

- Habilitar la red inalámbrica.
- Asignar como SSID del router SRV-1.
- Establecer como dirección ip del router la 75.15.156.1.
- Establecer como máscara de subred la ip 255.255.255.0.

- Habilitación de contraseña de red inalámbrica según preferencia.

Nota. es importante recordar los parámetros configurados en el router para establecer la configuración del vehículo SRV-1 compatible con la del router.

10.9. Configuración del vehículo SRV-1

10.9.1. Configuración del MatchPort

El vehículo SRV-1 debe ser configurado para que, tras su encendido, localice automáticamente el router y establezca conexión con el mismo formando parte de una infraestructura de red.

Para acceder al menú de configuración MatchPort es necesario disponer de una interfaz serial de 3,3V como la mostrada en la imagen [10.7](#)

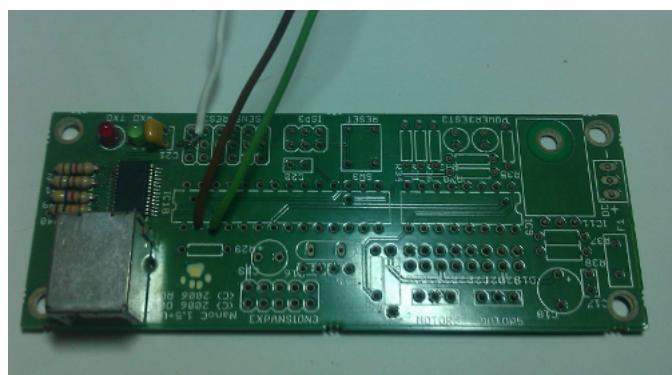


Figura 10.7: Imagen de la placa de configuración.

Se procede a la retirada la tarjeta Blackfin situada en la parte superior del SRV-1 para realizar la conexión con el puerto de expansión de 32 pines tal y como se muestra en la imagen [10.8](#).



Figura 10.8: Conexión de la placa al SRV-1.

A continuación conecte la placa mediante el uso de un cable USB al ordenador. Los controladores del controlador CP2102 están integrados en Linux así que no es necesario instalar drivers adicionales.

Para configurar el MatchPort, inicie un programa terminal para que establezca conexión con la placa USB, para ello debemos ajustar la velocidad en 9600 baudios, encienda el robot y rápidamente (en unos pocos segundos) teclee 3 veces el carácter 'x', tras este paso se obtendrá en pantalla el menú de configuración para la MatchPort. Ver figura 10.9.

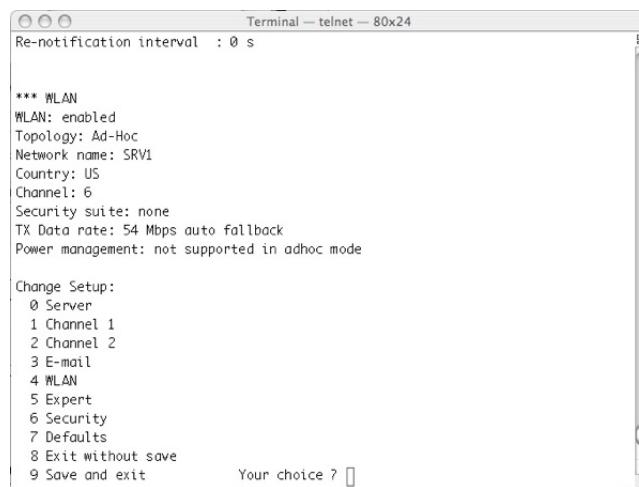


Figura 10.9: Menú de configuración del Matchport.

Al restaurar el MatchPort a 2500kbps, los cambios a realizar son:

1. Experto (5)

- Rendimiento de la CPU, se introduce el valor FF
- Valores de clk, se introduce el valor 81.
- Tamaño de MTU, pasamos de 1400 a 1024.
- Se omiten el resto de opciones.

2. WLAN (4)

- Establecemos un ssid, SRV-1
- Seleccionamos red de infraestructura.
- Canal 1 de serie (1).
- Baudrate, escriba -1.
- Divisor, introduzca 2.
- Control de flujo, introduzca 2.
- FlushMode, introduzca 80.
- para el paquete de Cntrl, introduzca C0
- Hora InterCh, introduzca 3
- Se omiten el resto de opciones.

3. Network (0)

- Se establece la dirección IP, Se asigna la 75.15.156.238, máscara: 255.255.255.0.

4. Guardar y salir (9)

Nota. Para usuarios Windows el programa Hyperterminal presenta problemas. Se recomienda utilizar en Windows el programa Br@y Terminal. Para usuarios Linux se recomienda la utilización de PuTTY.

Nota. El módulo WLAN MatchPort es un dispositivo de 3.3V. Si tratas el MatchPort con una interfaz serial RS232 (+/-12V niveles), es probable que dañe la placa MatchPort y quede inutilizada.

10.9.2. Carga de batería del vehículo SRV-1

Para la carga de batería se debe conectar el adaptador jack al vehículo y el enchufe a una toma de 220V, es necesario utilizar un conversor del sistema Americano al Europeo. Ver figura 10.10.



Figura 10.10: Adaptador de corriente para realizar cargas de batería en el robot SRV-1.

Posteriormente encender el vehículo SRV-1 en modo carga, *charge*, mostrando el adaptador de corriente el led en color rojo. El periodo de carga es de una hora de duración. Ver figura 10.6.

10.10. Instalación de la capturadora

La capturadora de vídeo es el dispositivo conversor de la imagen de analógico a digital, para, posteriormente ser captada por el ordenador. En esta guía se detalla el proceso de instalación de los drivers en una distribución Ubuntu.

Los drivers son incluidos en la versión Ubuntu 12.04 LTS siendo innecesario realizar los pasos indicados en esta guía. Si dispones de una versión de Ubuntu anterior, proceda a la realización de los pasos aquí indicados:

1. Primeramente conecte el dispositivo a un puerto USB de su equipo e introduzca en una terminal el siguiente comando:

```
1 lsusb
```

Entre los dispositivos detectados debe existir la siguiente línea:

```
1      USB ID 05e1:0408
2      Bus 002 Device 017: ID 05e1:0408 Syntek Semiconductor Co., Ltd
```

2. Una vez comprobado que el dispositivo se corresponde con el del driver desconéctelo de su puerto USB. Para comenzar con la instalación se debe previamente descargar el driver, para ello introduzca en una terminal:

```
1      wget
          http://sourceforge.net/projects/easycapdc60/files/easycap\_dc60.0.9.tar.gz/download
=> `easycap\_dc60.0.9.tar.gz`
```

3. Instalamos los paquetes necesarios:

```
1      \$ sudo apt-get install linux-headers-\$(uname -r) build-essential
2      \end{lstlisting}
3
4 \item Para descomprimir el driver:
5   \begin{lstlisting}[style=consola]
6     mkdir ~/EASYCAP
7     cd ~/EASYCAP
8     cp -p ~/Desktop/easycap\_dc60.x.y.tar.gz .
9     tar zxvf easycap\_dc60.0.9.tar.gz
10    cd easycap\_dc60.0.9.tar.gz
```

4. En el directorio donde se extrajo el archivo .tar, introduzca el siguiente comando para instalar el driver:

```
1      sudo ./install.sh
```

5. Después de que la instalación haya finalizado, compruebe que el módulo ha sido cargado satisfactoriamente en el kernel mediante el uso del siguiente comando:

```
1      lsmod | grep easycap
```

6. Ahora conecte en el puerto USB el dispositivo y compruebe que ha sido detectado, mediante el uso del siguiente comando:

```
1  ls /dev/easy*
```

Si todo funcionó correctamente, se debería visualizar algo como lo siguiente:

```
1      /dev/easycap0
2      /dev/easysnd1/
```


Capítulo 11

Comentarios finales

11.1. Presupuesto

Descripción	Unidades	Precio €	Total €
Surveyor SRV-1 Blackfin Robot	1	379,8	379,8
Router WIFI Xavi 7968	1	35	35
Capturadora de vídeo	1	15	15
Cámara y receptor inalámbrico	1	54	54
Gamepad	1	10	10
Horas de programación	350	45/h	15750

Total bruto: 16.264,8 €

I.V.A. %: 21 %

Total presupuesto: 19.676,00 €

11.2. Conclusiones

La elaboración de este proyecto ha resultado muy gratificante a nivel personal. Uno de los motivos principales ha sido la necesidad de trabajar en numerosas áreas de conocimiento entre las que encontramos la programación en los lenguajes C y C++, la robótica y la visión artificial. Algunas de las plataformas mencionadas eran desconocidas al inicio del desarrollo de proyecto y han sido adquiridas tras una amplia labor de investigación y pruebas de desarrollo con diferentes tecnologías como Matlab y Octave sin éxito. Otro de los motivos ha sido la necesidad de continuar con el aprendizaje de conceptos propios de la visión artificial y del reconocimiento de patrones adquiridos mediante la asistencia a asignaturas correspondientes a Ingeniería Informática de segundo ciclo.

Entre los elementos desarrollados se destaca:

- **Software de procesamiento de imágenes:** escrito en el lenguaje de programación C, utilizando la biblioteca de visión por computador OpenCV. Se han estudiado los principios básicos del procesamiento de imágenes y reconocimiento de patrones aplicados a la visión por computador para construir un sistema software inteligente capaz de controlar un vehículo de manera inalámbrica desplazándose por el entorno actuando según las señales de tráfico existentes. Todo ello en tiempo real.

Ha sido la parte del proyecto en la que más tiempo se ha invertido, el software de reconocimiento de señales de tráfico ha sido obtenido siguiendo una metodología de desarrollo por prototipos habiéndose desarrollado tres prototipos. El algoritmo de clasificación utilizado ha sido el del vecino más cercano. Como datos de entrada se han utilizado las diferentes imágenes de cada uno de los elementos internos de las señales de tráfico dispuestos en columnas y añadidos a la matriz modelos.

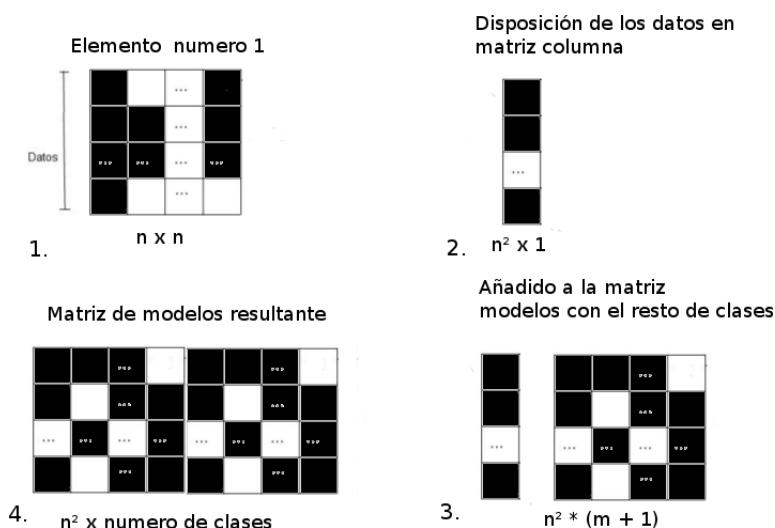


Figura 11.1: Pasos para la elaboración de la matriz modelos.

Para la clasificación se ha realizado la comparación de la matriz columna obtenida a partir del elemento desconocido con cada una de las columnas existentes en la matriz de modelos. Cada una de ellas corresponde a una clase. La comparación se realiza mediante el cálculo de todas las distancias euclídeas y quedándonos con la clase que menor distancia ha obtenido, es decir, la más cercana o la que guarda más semejanza. Una vez clasificados todos los objetos internos de una señal de tráfico se evalúa el conjunto y se determina la señal de tráfico resultante. Los elementos internos son analizados siempre que se encuentren dentro de un contorno rojo o azul y sus formas sean las propias de las señales de tráfico, redondas, octogonales, triangulares o cuadradas.

- **Software control del vehículo:** se ha empleado el software denominado *Surveyor Robot Software* desarrollado por la universidad de Brooklyn escrito en el lenguaje de programación C++. El software ha sido adaptado a las necesidades del proyecto. Se ha dotado al vehículo de dos modos de funcionamiento, uno de pilotaje tradicional mostrando las señales detectadas por el software de reconocimiento y otro de pilotaje automático realizado a partir de las señales detectadas.
- **Interfaz gráfica:** se ha dotado al conjunto de una interfaz gráfica con la finalidad de proporcionar una experiencia de control al usuario mejorada empleando la biblioteca Qt en C++.
- **Sistema de comunicaciones:** el proyecto se compone de dos sistemas de comunicaciones, utilizando cada uno de ellos tecnologías distintas:
 - ◊ **Sistema de comunicaciones WiFi:** hace posible el envío de las órdenes desde el ordenador hasta el coche para su control de forma remota. La comunicación se realiza a través de un router donde se conectan ambos dispositivos, el router y el ordenador, formando una infraestructura de red.

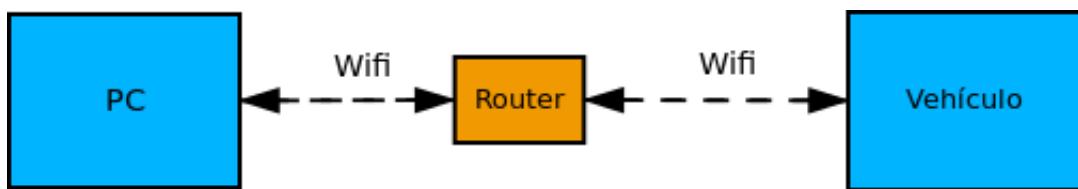


Figura 11.2: Comunicaciones vía WiFi entre el ordenador y el vehículo robótico mediante un router.

- ◊ **Sistema de comunicaciones por radiofrecuencia:** hace posible la recepción de imágenes de vídeo desde la cámara al ordenador de forma inalámbrica. Para hacer esto posible se ha dividido el sistema en tres pasos, primariamente se ha comenzando por la recepción inalámbrica de la imagen emitida por la cámara mediante una receptora de imágenes analógicas. A continuación se realiza una conversión de la imagen de vídeo analógica a digital y finalmente se realiza el envío de la imagen digital al ordenador.

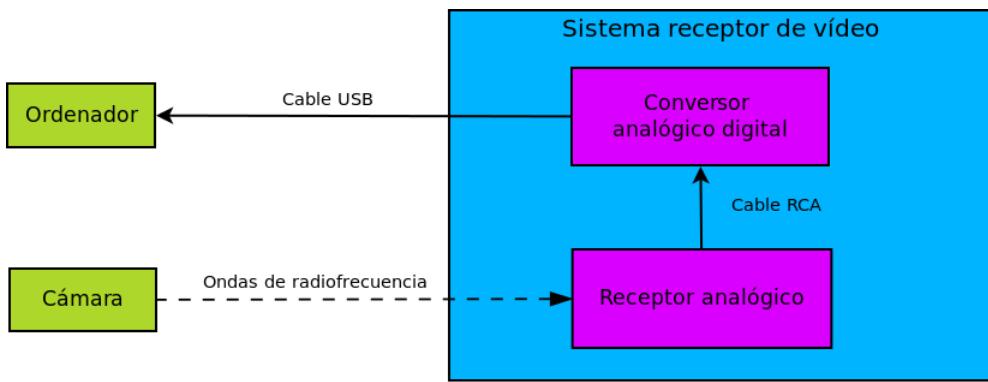


Figura 11.3: Elementos implicados en la transmisión de imágenes desde la cámara hasta el ordenador.

Pienso que el resultado final del proyecto es ideal para aquellas personas aficionadas a la robótica y programación proporcionando una herramienta sea utilizable por la gran comunidad poseedora del vehículo SRV-1 Surveyor en todo el mundo haciéndoles pasar unos momentos divertidos utilizando el vehículo.

Una vez presentado podré continuar añadiendo mejoras y muchas cosas que tengo pensadas y que, posiblemente, se realicen para el proyecto de la Ingeniería Informática que me encuentro realizando en la actualidad.

11.3. Mejoras futuras

La aplicación puede mejorarse en diversos aspectos. A continuación, se citan algunas de las mejoras que pueden llevarse a cabo:

- Incorporación de advertencias acústicas tras la detección de una señal de tráfico.
- Mejora del diseño de la interfaz gráfica.
- Permitir la redefinición de las teclas de control del teclado y mando.
- Permitir la redefinición de la dirección ip en la aplicación en caso de que el vehículo tenga configurada otra dirección ip diferente a la prefijada.
- Permitir la introducción de nuevas señales detectables al gusto del usuario.
- Permitir la programación de respuestas al vehículo a señales de tráfico incorporadas por parte del usuario.
- Mejora del algoritmo de reconocimiento.
- Mejorar el funcionamiento del algoritmo en situaciones de elevada oscuridad.

- Adaptar la aplicación al funcionamiento de la cámara que incorpora el vehículo sin la utilización de una cámara independiente empleada debido al estado defec-tuoso de la original.

Anexos

.1. Instalación de OpenCV

El siguiente anexo indica la instalación de OpenCV 2.4.1 en Ubuntu 12.04 LTS.

La presente guía de instalación detalla los pasos para instalación de OpenCV con la nueva interfaz Qt highgui, que ofrece muchas más funcionalidades que la sencilla interfaz highgui. Además, se procederá a la instalación de OpenCV con soporte para OpenGL, así como vídeos de lectura y escritura, el acceso a una cámara web, Python, C y C++ interfaz, e Intel Threading Building Blocks (TBB).

Previamente hay que asegurarse de que todo en el sistema está actualizado:

```
1 sudo apt-get update  
2 sudo apt-get upgrade
```

A continuación se deben instalar muchas dependencias y secos como soporte para leer y escribir archivos de imagen, dibujar en la pantalla, algunas herramientas necesarias, etc. Para ello introducimos en la terminal terminal:

```
1 sudo apt-get install build-essential libgtk2.0-dev libjpeg-dev libtiff4-dev  
libjasper-dev libopenexr-dev cmake python-dev python-numpy python-tk libtbb-dev  
libeigen2-dev yasm libfaac-dev libopencore-amrnb-dev libopencore-amrwb-dev  
libtheora-dev libvorbis-dev libxvidcore-dev libx264-dev libqt4-dev  
libqt4-opengl-dev sphinx-common texlive-latex-extra libv4l-dev libdc1394-22-dev  
libavcodec-dev libavformat-dev libswscale-dev !/bin/bash
```

A continuación descargamos el código fuente de OpenCV 2.4.1:

```
1 cd ~  
2 wget http://downloads.sourceforge.net/project/opencvlibrary/opencv-unix/  
2.4.1/OpenCV-2.4.1.tar.bz2  
3 tar -xvf OpenCV-2.4.1.tar.bz2  
4 cd OpenCV-2.4.1
```

Ahora tenemos que generar el Makefile usando cmake. De aquí podemos definir qué partes de OpenCV queremos compilar. Como queremos usar Python, TBB, OpenGL, Qt, el trabajo con vídeos, etc, aquí es donde tenemos que establecer eso. Simplemente ejecute la ayuda en línea en la terminal para crear el makefile apropiado. Tenga en cuenta

que hay dos puntos al final de la línea, es un argumento para el programa cmake y significa el directorio padre (porque estamos dentro del directorio de construcción, y queremos hacer referencia al directorio de OpenCV, que es su padre).

```
1 mkdir build
2 cd build
3 cmake -D WITH_TBB=ON -D BUILD\_NEW\_PYTHON\_SUPPORT=ON -D WITH_V4L=ON -D
    INSTALL\_C\_EXAMPLES=ON -D INSTALL\_PYTHON\_EXAMPLES=ON -D BUILD\_EXAMPLES=ON -D
    WITH\_QT=ON -D WITH\_OPENGL=ON ..
```

Compruebe que el comando anterior produce ningún error y que en particular aparece FFMPEG como S. Si no está activado no será capaz de visualizar videos. Asimismo, compruebe que Python, TBB, OpenGL, V4L, OpenGL y Qt son detectados. Ver figura 4.

```
samontab@virtubuntu: ~/OpenCV-2.4.1/build
-- OpenGL support: YES (/usr/lib/i386-linux-gnu/libGLU.so /usr/
r/lib/i386-linux-gnu/libICE.so /usr/lib/i386-linux-gnu/libX11.so /usr/lib/i386-l
-- Media I/O:
--   ZLib:           /usr/lib/i386-linux-gnu/libz.so (ver 1.2.3.4)
--   JPEG:          /usr/lib/i386-linux-gnu/libjpeg.so (ver )
--   PNG:           /usr/lib/i386-linux-gnu/libpng.so (ver 1.2.4)
--   TIFF:          /usr/lib/i386-linux-gnu/libtiff.so (ver 42)
--   JPEG 2000:     /usr/lib/i386-linux-gnu/libjasper.so (ver 1.
--   OpenEXR:       /usr/lib/libImath.so /usr/lib/libIImImf.so /
ver 1.6.1)
-- 
-- Video I/O:
--   DC1394 1.x:    NO
--   DC1394 2.x:    YES (ver 2.2.0)
--   FFMPEG:
--     codec:        YES (ver 54.23.100)
--     format:       YES (ver 54.6.100)
--     util:         YES (ver 51.54.100)
--     swscale:      YES (ver 2.1.100)
--     gentoo-style: YES
--   GStreamer:     NO
--   OpenNI:        NO
--   OpenNI PrimeSensor Modules: NO
--   PvAPI:         NO
--   UniCap:        NO
--   UniCap ucil:  NO
--   V4L/V4L2:      Using libv4l (ver 0.8.6)
--   Xine:          NO
-- 
-- Other third-party libraries:
--   Use IPP:       NO
--   Use TBB:       YES (ver 4.0 interface 6000)
--   Use Cuda:      NO
--   Use Eigen:     YES (ver 2.0.17)
-- 
-- Python:
--   Interpreter:   /usr/bin/python (ver 2.7.3)
--   Libraries:    /usr/lib/libpython2.7.so
```

Figura 4: Chequeo de la configuración de OpenCV.

Si algún elemento anterior aparece como desactivado, revise los pasos anteriores y corrja los errores mediante la instalación de paquetes adicionales. Revise de nuevo la configuración tras la ejecución nuevamente de cmake.

Pasado este punto, todo está listo para compilar e instalar OpenCV 2.4.1:

```
1 make  
2 sudo make install
```

Ahora tienes que configurar OpenCV. En primer lugar, abrir el archivo opencv.conf con la siguiente orden:

```
1 sudo gedit /etc/ld.so.conf.d/opencv.conf
```

Agregue la siguiente línea al final del archivo y guarde los cambios:

```
1 /usr/local/lib
```

Ejecute el siguiente comando para configurar la biblioteca:

```
1 sudo ldconfig
```

Ahora se debe editar otro archivo:

```
1 sudo gedit /etc/bash.bashrc
```

Se añaden estas dos líneas al final del archivo. Guarde los cambios:

```
1 PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/usr/local/lib/pkgconfig  
2 export PKG_CONFIG_PATH
```

Por último, reinicie el equipo o cierre la sesión e inicie nuevamente. OpenCV no funcionará correctamente hasta que lo haga.

Llegado a este punto OpenCV 2.4.1 está instalado en su ordenador con Python, TBB, OpenGL, video y soporte Qt.

El siguiente paso es opcional aunque se recomienda para comprobar que todo se ha instalado correctamente. Para ello se va a compilar algunos de los ejemplos proporcionados por OpenCV:

```
1 cd ~/OpenCV-2.4.1/samples/c  
2 chmod +x build_all.sh  
3 ./build_all.sh
```

Una vez compilados se procede a ejecutar los ejemplos:

```
1 ./facedetect --cascade="/usr/local/share/OpenCV/haarcascades  
2 /haarcascade\_frontalface\_\_alt.xml" --nested-cascade="/usr/local/share  
3 /OpenCV/haarcascades/haarcascade\_\_eye.xml" --scale=1.5 lena.jpg
```

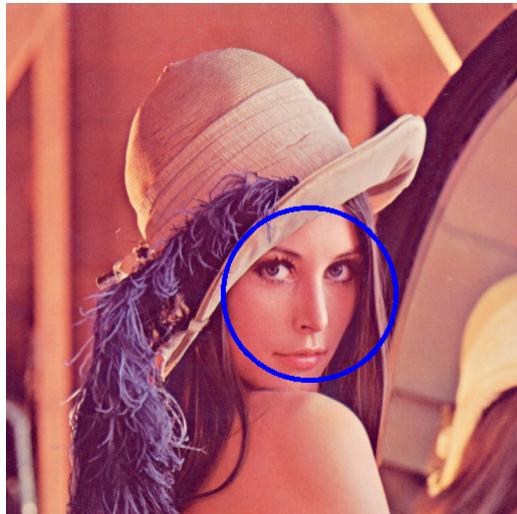


Figura 5: Salida producida tras la ejecución del programa de ejemplo.

.2. Instalación Qt y Qt Creator

El SDK de Qt incluye las herramientas que necesarias para la elaboración de interfaces gráficas compatibles con multitud de dispositivos como ordenadores de escritorio y móviles entre otros. Todas las herramientas necesarias se encuentran incorporadas en este instalador.

Como paso previo a la instalación se debe descargar el instalador del siguiente enlace <http://qt.nokia.com/downloads/>. Se recomienda la descarga del instalador offline, en el caso de esta guía se explica el proceso de instalación de la versión para Linux 64 bits. El resto de las versiones no difiere demasiado.

Una vez descargado el instalador, se procede con la instalación.

En sistemas Linux/Unix, es necesario asignar permisos para la ejecución del instalador. Para asignar permisos de ejecución previamente identificado como administrador y en una terminal, escriba:

```
1 chmod u + x Qt\_\_SDK\_\_Lin64\_\_offline\_\_v1\_\_2\_\_en.run
```

Ahora debería ser capaz de ejecutar el archivo. Para su ejecución, desde la línea de comandos, escriba:

```
1 ./Qt\_SDK\_Lin64\_offline\_v1\_2\_en.run
```

A continuación se muestra la ventana de bienvenida del instalador, pulsamos en *Next*.

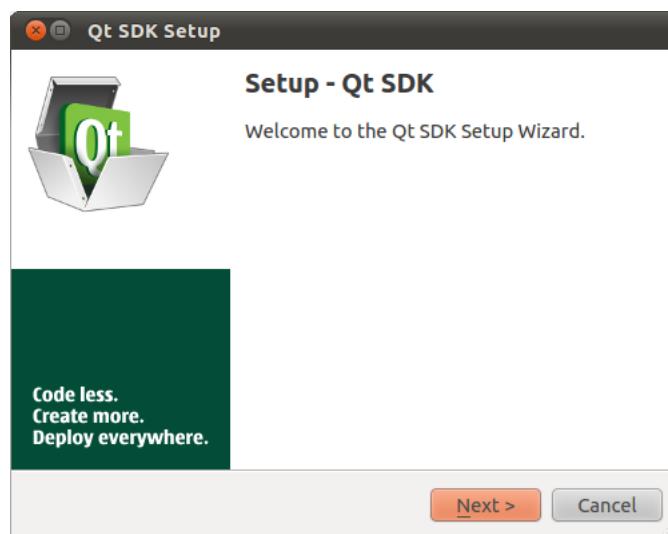


Figura 6: Ventana de bienvenida del instalador.

La siguiente ventana solicita el establecimiento de la rutas de instalación, se dejan valores por defecto.

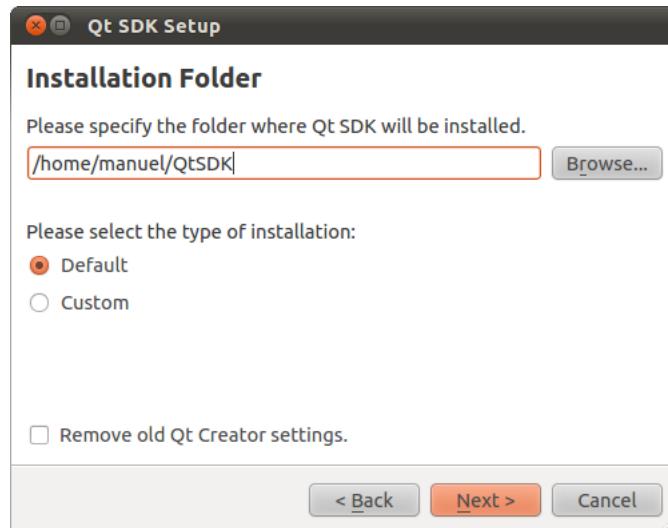


Figura 7: Solicitud de directorio para la instalación.

En la nueva ventana aceptamos los términos de licencia y pulsamos sobre *Next*.

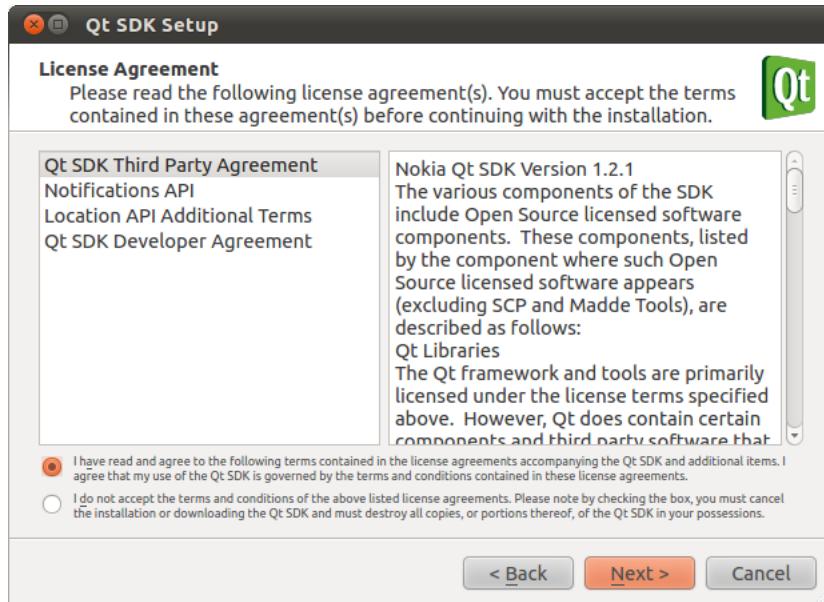


Figura 8: Ventana para la visualización de los términos de licencia.

Finalmente, ya introducida toda la configuración, pulsamos sobre el botón *Install* para proceder con la instalación.

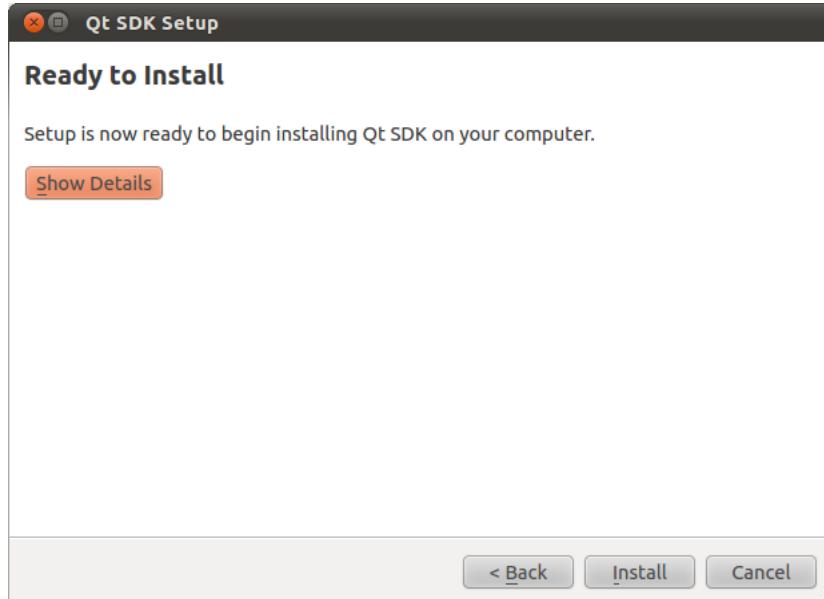


Figura 9: Ventana de comienzo de instalación.

Se espera durante unos minutos a que se complete el proceso de instalación.

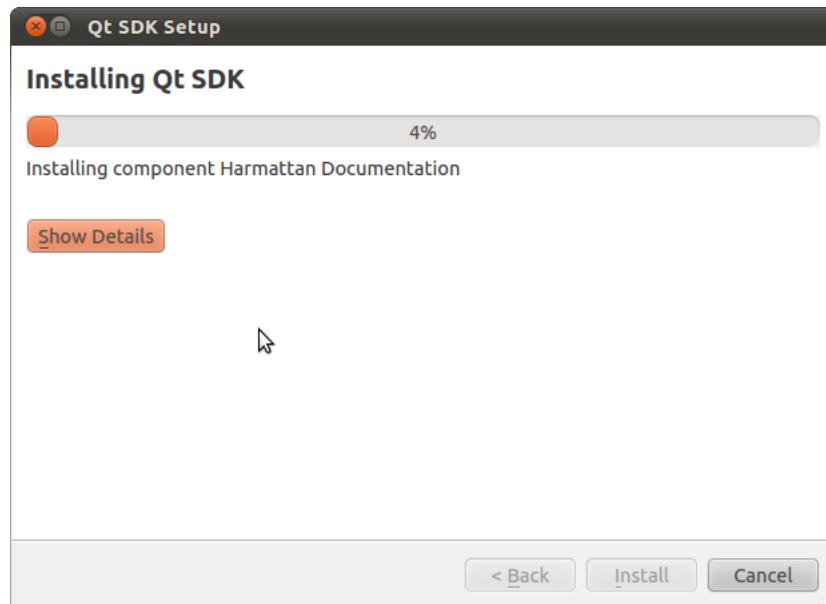


Figura 10: Proceso de instalación de Qt SDK.

Llegados a este punto disponemos de la aplicación Qt SDK instalada. Pulsamos en *Finish*.

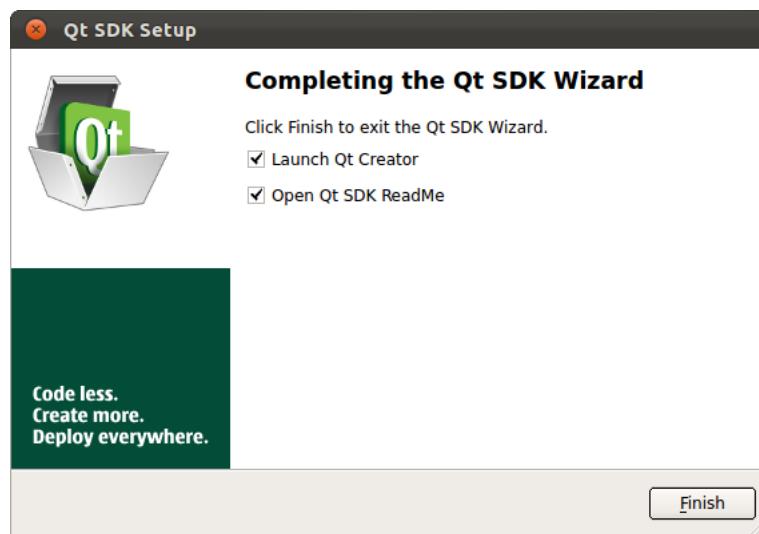


Figura 11: Proceso de instalación finalizado.

.3. Instalación de SDL

Los paquetes necesarios se encuentran disponible en el repositorio universe, por tanto es recomendable proceder a su instalación desde terminal. Los paquetes necesarios son los siguientes:

- libsdl1.2debian.
- libsdl1.2-dev.
- libsdl-image1.2.
- libsdl-image1.2-dev.
- libsdl-mixer1.2.
- libsdl-mixer1.2-dev.
- libsdl-ttf1.2.
- libsdl-ttf1.2-dev.

Para la instalación de los citados paquetes, introducimos en una terminal:

```
1 sudo apt-get install libsdl1.2debian libsdl1.2-dev libsdl-image1.2 libsdl-image1.2-dev  
libsdl-mixer1.2 libsdl-mixer1.2-dev libsdl-ttf1.2 libsdl-ttf1.2-dev
```

Una vez completado el proceso de instalación dispondremos de los diferentes elementos que incorpora esta biblioteca.

Bibliografía

- [1] Forja rediris. <https://forja.rediris.es/>.
- [2] Métrica v3. <http://www.csi.map.es/csi/metrica3>.
- [3] Qt Reference Documentation. <http://doc.qt.nokia.com/4.7-snapshot/designer-manual.html>.
- [4] Reference of OpenCV Language Library. <http://docs.opencv.org/>.
- [5] Surveyor Corporation. http://www.surveyor.com/SRV_info.html.
- [6] John Cummins. M.P. Azhar.Sklar. Surveyor Robot Software 1. <http://agents.sci.brooklyn.cuny.edu/robotics.edu/bcsoftware.php>.
- [7] Francisco Palomo Lozano. Inmaculada Medina Bulo. Gerardo Aburruzaga García. *Fundamentos de C++*. Servicio de Publicaciones de la Universidad de Cádiz, 2006.
- [8] Gerardo Aburruzaga García. Make. Un programa para controlar la recompilación. <http://www.uca.es/softwarelibre/publicaciones/make.pdf>.
- [9] Gary Bradski. Adrian Kaehler. *Learning OpenCV Computer Vision with the OpenCV Library*. O'Reilly Media, 2008.
- [10] Ben Collins-Sussman. Brian W. Fitzpatrick. C. Michael Pilato. *Control de versiones con Subversion*. O'Reilly Media, 2006.
- [11] Wikibooks. The Book of LaTeX. <http://en.wikibooks.org/wiki/LaTeX>.

GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.

<<http://fsf.org/>>

Everyone is permitted to copy and distribute verbatim copies of this license document,
but changing it is not allowed.

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “**Document**”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “**you**”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “**Modified Version**” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “**Secondary Section**” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “**Invariant Sections**” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none. The “**Cover Texts**” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “**Transparent**” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “**Opaque**”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “**Title Page**” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “**publisher**” means any person or entity that distributes copies of the Document to the public.

A section “**Entitled XYZ**” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “**Acknowledgements**”, “**Dedications**”, “**Endorsements**”, or “**History**”.) To “**Preserve**

the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or non-commercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the

“History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

- K. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements”.

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects. You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and

disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

“Massive Multiauthor Collaboration Site” (or “MMC Site”) means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for

anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A “Massive Multiauthor Collaboration” (or “MMC”) contained in the site means any set of copyrightable works thus published on the MMC site.

“CC-BY-SA” means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

“Incorporate” means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is “eligible for relicensing” if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with ... Texts.” line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.