CrossDetect: Detección de Pasos de Peatones Usando Algoritmos de Visión por Computador para la Mejora del Posicionamiento de Vehículos

PROYECTO DE FIN DE CARRERA

Adrián T. Marín Cabeza

Tutor

Basam Musleh Lancis

Departamento de Ingeniería de Sistemas y Automática
Escuela Politécnica
Universidad Carlos III de Madrid

Junio 2014

Documento maquetado con TEXIS v.1.0.

Este documento está preparado para ser imprimido a doble cara.

CrossDetect: Detección de Pasos de Peatones Usando Algoritmos de Visión por Computador para la Mejora del Posicionamiento de Vehículos

Memoria que presenta para optar al título de Ingeniero Industrial

Adrián T. Marín Cabeza

Tutor

Basam Musleh Lancis

Revisada por

Basam Musleh Lancis

Departamento de Ingeniería de Sistemas y Automática
Escuela Politécnica
Universidad Carlos III de Madrid

Junio 2014

If I have seen further than others, it
is by standing upon the shoulders
of giants.
Isaac Newton

Agradecimientos

Podría parecer que ésta es una de las partes más insustanciales de un proyecto, algo poco valorado, aunque quizás lo más leído y lo que más interesa a todos aquellos que se acercan al trabajo por el simple hecho de conocer al autor del mismo.

Yo pienso que es una parte muy personal que ayuda a conocer, y quizás a comprender un poquito mejor, a la persona detrás de las letras y los números, al ser humano que se oculta entre las ecuaciones, los cálculos y la prosa impersonal que caracteriza un documento científico-técnico como es el proyecto de fin de carrera de una titulación de ingeniería. Una parte, sobre todo, que ayuda a darse cuenta de la cantidad de gente que rodea a un trabajo de este estilo y que, si bien no interviene directamente en su redacción, se merece el reconocimiento derivado de haber ayudado al autor a llegar a terminar estas páginas; todo el mérito que en su creación tienen.

En mi caso, en primer lugar, como creo que no podría ser de otra manera, quiero que las primeras palabras de agradecimiento sean para mi familia, toda ella, para mi hermana, mis abuelos y mis abuelas y, especialmente, para mis padres. Ellos son las personas que desde antes de que la razón aterrizara sobre mi cabeza han estado ahí apoyándome, siempre. Han aportado todo lo que fuera posible para ayudarme, darme una palabra de ánimo y mantener en algunos momentos más esperanzas de las que yo mismo he sido capaz de conservar. Llegar hasta este punto, que pone el broche final a un camino tan largo, es tanto mi ilusión como la suya y es tanto su orgullo el que haya sido capaz de lograr llegar hasta el día de hoy, como el mío el poder agradecerles en este momento todo lo que han hecho, de poder reconocer ante cualquiera que lea este trabajo que ellos son mis padres, que me han educado, guiado, enseñado y, sobre todo, que se han, y todavía a día de hoy se desviven, por tratar de hacerme feliz y darme su cariño. Para vosotros es este trabajo. Gracias.

También quiero tener unas palabras de agradecimiento a María porque, a pesar de su carácter, siempre tiene un buen consejo, un abrazo, un beso, una

forma de liberarme cuando los agobios me han tenido acorralado... Quiero agradecerle su capacidad inmensa de escuchar, de hacerme reír, de hacerme soñar, de poner mis pies en la tierra y de aguantar miles de historias acerca de este proyecto que a ambos se nos ha hecho tan largo y sobre otras tantas cosas con las que la torturo. En definitiva, gracias por ser mi inspiración. No en vano la he dicho muchas veces que es mi musa y que todo fue a mejor desde que entró en mi vida. Gracias por estar a mi lado.

La amistad también es algo muy importante, y yo he tenido la suerte de encontrar a muchas personas maravillosas a lo largo de esta carrera que me han ayudado a superar los momentos más duros de estos años. Esas personas que han estado ahí, ya fuera para dedicarnos mutuamente unas sonrisas, aliviar las cargas del trabajo, celebrar las victorias, llorar las derrotas o tener una clave para superar aquella asignatura que se atragantaba: David, gracias por estar siempre ahí, Carlos, Isabel y Verónica, por demostrar que también se encuentran grandes amigos en la veintena, Lidia y Cynthia, por ser mis infatigables colegas de especialidad, Juanjo, Luis, San Román, Anita, Álvaro, Alvarito, Dubois, Laurita, Estalrich, Guille, Marta, Cesar, José, todo los compañeros de especialidad y tantos otros. Gracias por allanar el camino hasta esta meta.

A mis amigos del barrio, que aunque nunca entendieron muy bien de lo que hablaba, también se alegraron siempre de mis triunfos: Raquel, Alba, Cesar y Sara. Gracias por darme esperanzas. Gracias también a los que no están por cuando estaban.

Por último, y no por ello menos importante, quería tener unas palabras de agradecimiento para la persona que ha ayudado a concebir y dar forma a este trabajo y que, a pesar de que no haber tenido su mejor año, siempre ha estado disponible cuando le he necesitado, ya fuera para resolverme una duda, templar mis nervios o sacarme de un atolladero: a mi tutor, Basam, que es en gran parte el padre de este proyecto. Muchas gracias por tu ayuda y millones de gracias por tu tiempo.

Resumen

El objetivo de este proyecto es utilizar pasos de peatones, de los denominados pasos de cebra, como puntos de referencia dentro del entorno, de tal forma que estos sirvan para corregir el error de posicionamiento que cometen sistemas como la Odometría Visual o el GPS, a la hora de localizar un vehículo en el plano. El trabajo desarrollado en este proyecto se podría dividir en dos partes.

La primera de ellas, es el desarrollo de un algoritmo de visión por computador que permite reconocer los pasos de peatones mediante un sistema de cámaras estereoscópicas instalado en un coche. Para lograr este objetivo, en primer lugar, las imágenes obtenidas por una de las cámaras son procesadas con la combinación de un filtro de percentil 43 aplicado a nivel local más una umbralización (Sebsadji et al., 2010). Este proceso se realiza con el fin de extraer aquellos píxeles de la imagen original que pertenecen potencialmente a marcas pintadas en la carretera. Si un píxel de la captura inicial tiene un nivel de gris mayor que el de su correspondiente de la imagen filtrada más el umbral seleccionado, ese píxel es considerado perteneciente a una marca vial, lo que en definitiva, quiere decir que éste tiene un nivel de gris mucho más alto que el de aquellos píxeles próximos a él. El resultado de este proceso, por tanto, es una imagen binaria que representa en blanco todos aquellos píxeles que pueden pertenecer a una marca vial.

Aparte de las marcas viales, fuera de la calzada hay muchos píxeles que cumplen esta condición. Para eliminar de la imagen binaria obtenida todos aquellos que no están sobre la carretera, se utiliza un algoritmo, perteneciente a otro proyecto de la universidad, que realiza este proceso utilizando el mapa de disparidad y el *u-disparity* de la escena. Es posible obtener estas estructuras gracias al sistema estereoscópico de cámaras.

Tras estos dos procesos se obtiene una imagen binaria que representa únicamente las marcas viales. Si además la parte inferior de la imagen es muy oscura, previamente a estas operaciones se le realiza una mejora de contraste, que realza la zona de la misma correspondiente a la calzada. También se

realiza una erosión para eliminar errores de segmentación.

Una vez extraídas las marcas viales de la imagen original, se distinguen las que son pasos de peatones del resto mediante un análisis de las frecuencias de la transformada rápida de Fourier (*Fast Fourier Transform*, FFT). Para obtener ésta se usa la proyección horizontal, la cual se obtiene como resultado de la suma de los píxeles de cada columna de la imagen binaria obtenida. Cuando se detecte un paso de peatones, se calculará su posición en la imagen y su distancia con respecto al vehículo. Para determinar la posición, se utiliza la filosofía anterior de la proyección horizontal, aplicada en esta ocasión a las filas de la imagen binaria (proyección vertical). Una vez calculada ésta, se hace uso del perfil de la calzada y los parámetros ópticos del sistema para calcular su distancia. Para aproximar el perfil de la calzada se utilizará el *v-disparity* y la Transformada de Hough para rectas.

En la segunda parte del trabajo, una vez detectado un paso de peatones y conocida su posición relativa respecto al vehículo, se pretende usar esta información para corregir el error de posicionamiento que introducen otros sistemas de localización. Usando como dato la cartografía de OpenStreetMap (OSM), si un sistema de localización como el GPS, o la Odometría Visual, predice que el coche se aproxima a un paso de peatones, el algoritmo de visión entraría en funcionamiento. Si el paso de peatones fuera detectado por el algoritmo de visión antes o después de lo previsto por el sistema de posicionamiento tradicional, el error que éste cómete podría ser estimado y corregido, reposicionando el vehículo en su localización correcta dentro del mapa. Esta parte sólo es desarrollada de manera teórica en el presente proyecto, de tal manera que se propone su implementación para futuros trabajos.

Abstract

The objective of this project is to use zebra crossings as reference points within the environment, in order to correct the positioning error of systems like visual odometry or GPS, when locatinh a vehicle in a map. The work developed in this project can be divided into two parts.

The first one, consists in the development of a computer vision algorithm that can recognize zebra crossings using a stereoscopic camera system installed in a car. To achieve this goal, images obtained by the cameras are processed by a mix of a local 43rd percentile filter plus a thresholding (Sebsadji et al., 2010). The final purpose of this is to extract from an image taken by one of the cameras those pixels which potentially belong to markings painted on the road. If the pixel in the original image had a gray level higher than its corresponding in the filtered image plus the threshold, that pixel would be considered to belong-to a road marking and therefore that pixel contains a gray level much higher than the pixels next to it. The result of this process is a binary image which represents in white colour those pixels that belong to a road marking.

In addition to the road markings, outside the road there are many pixels that meet this condition. An algorithm belonging to another college project is used to remove all those pixels that are not on the road from the binary image. This algorithm performs this process using the disparity map and the *u disparity*. This is possible thanks of the stereoscopic camera system.

After these two processes take place, a binary image which represents only road markings is obtained. A contrast enhancement of this part of the image is done if the bottom part of the picture is too dark. An erode operation is also performed to remove segmentation errors.

When road markings in the image have been extracted from it, zebra crossings are distinguished of the rest of road markings using a frequency analysis of the Fast Fourier Transform (FFT); this transform is obtained from the histogram that results of the adding of the pixels of each of the

colums from the filtered image (horizontal projection). When the zebra crossing is detected its position is computed using the vertical projection and its distance is obtained by using the *v disparity* and the Hough Transform.

In the second part of this paper, once a zebra crossing has been detected and its position relative to the vehicle is known, this information will be used to correct the positioning error that another location systems, like GPS or Visual Odometry, have. Using the OpenStreetMap mapping as an input, if a system like GPS predicted the car is close to a zebra crossing, the algorithm would be activated. If the zebra crossing was detected by our algorithm before or later than the traditional positioning system expects, the error could be inferred and corrected, repositioning the vehicle in its correct location on the map. This part is only theoretically developed, however its implementation is intended in future researches.

Índice

Índice de figuras

Índice de Tablas

Capítulo 1

Introducción

El mundo actual representa un entorno altamente dinámico, en el que los cambios se suceden de manera rápida y continua, afectando a todos los aspectos de la vida diaria de las personas. Las nuevas tecnologías, los avances en las redes de comunicación y de información, la globalización, etc. son sólo algunos de los elementos que impulsan estos procesos. Cosas que en un momento dado eran habituales en la cotidianidad de las personas, desaparecen rápidamente, muchas veces sin dejar apenas huella de su paso. También ocurre que cosas nuevas irrumpen con fuerza en las sociedades, volviéndose imprescindibles en muy poco tiempo, o mueren y desaparecen con la velocidad con que llegaron. En muchos otras ocasiones, simplemente aparecen nuevos usos a cosas que ya existían o éstas evolucionan dando lugar a algo en lo que cuesta ver la esencia de lo que fue en un principio.

La industria de la automoción no es una excepción a esta máxima que domina el mundo de hoy, destacándose como una de las principales potencias en su cumplimiento. Desde su aparición en el siglo XVIII, el concepto de automóvil no ha parado de experimentar cambios destinados a aumentar su eficiencia, comodidad y seguridad, incluyendo para ello en su diseño numerosos elementos pertenecientes a prácticamente todas las ramas de la ingeniería. En este sentido, los avances en computación y electrónica de las últimas décadas están entrando a formar parte de los nuevos modelos creados por las empresas de automoción y centros de investigación, dando lugar a distintos sistemas que pretenden cumplir con estos objetivos. La tendencia actual en esta línea de acción, es dotar a los vehículos de una gama cada vez más amplia de componentes que pretenden lograr una automatización progresiva del transporte particular por carretera. Surgen de esta manera dos tipos de sistemas: los Sistemas Avanzados de Asistencia al conductor (ADAS, de sus siglas en inglés *Advanced Driver Assistance Systems*) y los Vehículos Inteligentes (IVs, *Intelligent Vehicles*).

Por otro lado, los llamados sistemas globales de navegación por satélite (GNSS, *Global Navigation Satellite System*), entre los cuales destaca el GPS (*Global Positioning System*), han impuesto su uso durante los últimos años, convirtiéndose en imprescindibles en multitud de aplicaciones: aviación, agricultura, topografía, etc. Estos sistemas, a través de los cuáles se puede posicionar un objeto en cualquier punto de la tierra, han generalizado su uso también en el mundo de la automoción. Su aplicación más comúnmente conocida es la de los navegadores GPS, que utilizan ya millones de usuarios en todo el mundo, pero los GNSS y su perfeccionamiento resultan también claves para lograr el objetivo de la navegación autónoma en los vehículos.

A continuación, se analizan con más detalle estos aspectos que sirven de marco de referencia y motivación de este proyecto.

1.1. Contexto del proyecto: Vehículos Inteligentes

Los accidentes de tráfico son una de las principales causas de muerte en el mundo, ocupando el noveno puesto en la lista más reciente elaborada por la Organización Mundial de la Salud (Organización Mundial de la Salud (2013) y World Health Organisation (2013)), según la cual en 2011 casi 1.3 millones de personas murieron en el mundo debido a esta causa. Sólo en la Unión Europea, en ese mismo año, 300000 personas fallecieron en las carreteras, produciéndose en España el 7 % de esas muertes (Jiménez (2013)). Más allá de la tragedia que supone la pérdida de tal cantidad de vidas humanas, económicamente los accidentes de tráfico y las consecuencias derivadas de ellos suponen para los gobiernos un coste económico muy importante. En España los costes por cada persona fallecida en un accidente de tráfico se estiman en 1.4 millones de euros, mientras que los costes por cada herido grave ascienden a 219000 euros (RACC, 2013).

La preocupación acerca estos problemas de gran interés socio-económico, hace que las empresas de automoción inviertan cada vez más recursos con el objetivo de desarrollar e integrar en sus vehículos comerciales elementos destinados a incrementar la seguridad de sus ocupantes. En esta línea de acción, la incorporación en los automóviles de elementos de los denominados de *seguridad activa*, así como la investigación en sistemas de este tipo, se ha incrementado notablemente en los últimos años. Este tipo de elementos, frente a los llamados de *seguridad pasiva*, como por ejemplo el cinturón de seguridad o el *airbag*, cuyo objetivo es paliar las consecuencias de un accidente; tratan de evitar que éste se produzca. Ejemplos habituales serían los conocidos sistemas ABS o ESP.

Un estudio sobre las principales causas de los accidentes de circulación

(Muñoz-Repiso, 2000) revela además que el factor humano provoca entre el 71 % y el 93 % de los accidentes de tráfico que se producen, entendiendo por factor humano elementos como distracciones, fatiga, realización de maniobras imprudentes, etc. Debido a este hecho, la inversión de las empresas de automoción y centros de investigación está cada vez más dirigida al desarrollo de sistemas que asistan al conductor durante la maniobra del vehículo. Esta clase de elementos de seguridad activa son los denominados *Advanced Driver Assistance Systems (ADAS)*.

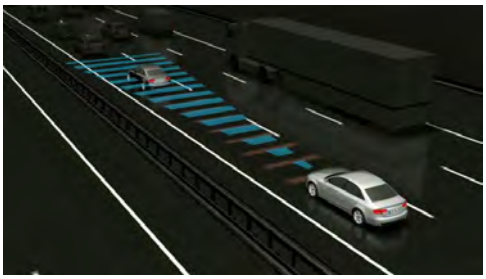
Los ADAS, se basan en la tecnología de sensores, tales como cámaras, radares, infrarrojos y ultrasonidos, para monitorizar de manera continua el entorno del vehículo, con el fin de facilitar y hacer más cómoda la conducción o de detectar situaciones peligrosas en una fase temprana, advirtiendo al conductor al respecto, dándole soporte o, en casos extremos, interviniendo en la maniobra del automóvil para evitar un accidente. Algunos de los ejemplos más comunes de este tipo de sistemas serían:

- Control de crucero adaptativo (*Adaptative Cruise Control, ACC*). El sistema ACC (ver figura 1.1a) es una evolución del control de crucero que, además de mantener una velocidad predefinida por el usuario como hace esta tecnología, puede adaptar ésta a las condiciones del tráfico, acelerando o frenando automáticamente.

  Un ejemplo de este tipo de sistemas es el creado por Bosch, que utiliza la información que genera un sensor de radar para registrar la situación presente delante del vehículo. Si el conductor se aproxima al vehículo de delante, el sistema ACC frena para garantizar que se mantenga la distancia mínima. Cuando el carril vuelve a estar despejado, el sistema ACC vuelve a acelerar hasta llegar a la velocidad seleccionada anteriormente por el conductor.
  El sistema está activo a velocidades de entre 30 km/h y 200 km/h aproximadamente (fuente: Bosch (2014)).

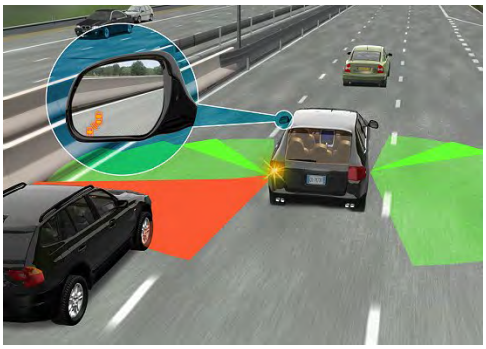
- Sistema de detección de peatones (ver figura 1.1b). Los peatones son el elemento más débil y desprotegido de entre todos los que circulan por las calles y carreteras. El sistema de detección debe indicar al conductor la presencia de posibles colisiones. Para ello el sistema no sólo debe detectar a todos los peatones que rodean al vehículo, sino analizar su actividad y movimiento para determinar aquellos con los que pueda existir un conflicto. Para ello el sistema utiliza también cámaras y algoritmos de visión adecuados. El vehículo IvvI (Laboratorio de sistemas inteligentes UC3M) cuenta con uno de estos sistemas.

- Detección del punto ciego (*Blind Spot Detection, BSD*). En una autopista, es fácil ver por los espejos retrovisores un vehículo que circula



(a) Adaptative Cruise Control

(b) Sistema de detección de peatones



(c) Blind Spot Detection

(d) Lane Departure Warning

Figura 1.1: Distintos tipos de sistemas ADAS

por detrás a cierta distancia. Sin embargo, si el vehículo se aproxima, hay una zona donde no es posible verlo con ninguno de los espejos, ni con el interior ni con los exteriores. Normalmente, esto se produce cuando el vehículo está justo detrás y a un lado del automóvil que está adelantando. Es un error muy común cambiar de carril cuando hay un vehículo en lo que se conoce como "punto ciego", una maniobra que provoca muchos accidentes en las autopistas europeas (ver figura 1.1c).

Diversos fabricantes (Audi, Mazda, etc.) han desarrollado sistemas que controlan el punto ciego y ayudan al conductor a cambiar de carril con seguridad. Hay sistemas que cuentan con cámaras de vídeo y otros que basan su funcionamiento en un radar. En cualquier caso, estos sistemas controlan la zona de la esquina trasera del vehículo y avisan al conductor de la aproximación de un vehículo que, con los retrovisores, habría sido imposible detectar (fuente: EURO NCAP (2014)).

- Sistemas de aviso de cambio de carril (*Lane Departure Warning, LDW*)

y de mantenimiento de carril (*Lane Keeping Assist, LKA*). El sistema de aviso al cambio de carril es un mecanismo diseñado para advertir al conductor cuando el vehículo comienza a abandonar de forma involuntaria el carril por el que circula, debido a distracciones por parte del usuario (ver figura 1.1d). Los sistemas de mantenimiento de carril van un paso más allá interviniendo en la maniobra del vehículo y devolviendo a éste al carril en caso necesario. Algunos modelos de coche, como la última versión del Ford Focus o del Opel Zafira, ya cuentan con este tipo de sistemas.

- Sistema eCall. La atención temprana de los accidentados en un siniestro de tráfico, en menos de una hora desde que se produce, reduce el número de fallecidos entre un 5 % y un 15 %. En esta línea de acción el sistema eCall, que será obligatorio en Europa para aquellos vehículos homologados a partir de 1 de Octubre de 2015, realiza una llamada a los servicios de emergencia automáticamente, sin la intervención del conductor, cuando detecta que se ha producido un accidente grave, enviando un mensaje de emergencia pregrabado y la posición exacta del coche, mediante coordenadas GPS, con la dirección, la hora y el número de ocupantes del vehículo (fuente: motorpasión futuro (2014)).

La integración de los ADAS en los vehículos podría considerarse, sin embargo, únicamente como un primer paso en el avance hacia la obtención de un **vehículo de conducción totalmente autónoma**, que eliminaría por completo el elemento humano presente en los accidentes de tráfico. Este concepto, el cual debe superar aún importantes barreras técnicas, económicas, psicológicas y legales, antes de poder ser implantado en el mercado (Collado et al., 2003), da lugar a la idea del **Vehículo Inteligente (VI o *IV*, del inglés *Intelligent Vehicle*)**, enmarcado en los denominados Sistemas de Transporte Inteligentes (STI o *ITS*, de sus siglas en inglés *Intelligent Transportation Systems*).

Los ITS buscan dotar a los vehículos e infraestructuras de funcionalidades que permitan, además de reducir la siniestralidad en las carreteras, solucionar otros problemas derivados del uso de los vehículos, como la creciente congestión del tráfico, que reduce la eficiencia de la infraestructura de transporte, incrementando el tiempo de viaje y, consecuentemente, el consumo de combustible y la contaminación. Según declaraciones del año 2010 realizadas por el entonces delegado de Seguridad y Movilidad del ayuntamiento de Madrid, Pedro Calvo (20 minutos.es, (2010)), el tiempo perdido en los atascos en Madrid equivale a siete día de vacaciones al año, lo que supone un total de 329000 horas al día y un coste de 3.4 millones de euros diarios.

El IV, como elemento que forma parte de los ITS, combina el uso de varios ADAS para tratar de solucionar los problemas citados, mediante la integración en el vehículo de elementos sensores y algoritmos inteligentes que perciben el entorno del vehículo y responden ante las circunstancias exteriores con distintos grados de automatización, que van desde la simple transmisión de información al conductor, hasta la maniobra totalmente autónoma del vehículo. Para ello, los IV cuentan además con sistemas de procesamiento que permiten tratar la información recibida de sus distintos componentes, para así tomar decisiones sobre las acciones a llevar a cabo, que sean lo suficientemente fiables para el grado de independencia del vehículo y la criticidad de la operación. Estos sistemas de computación son sistemas de tiempo real en los que la prioridad es obtener los resultados en un tiempo inferior a un cierto tiempo crítico dependiente de la aplicación. Para cumplir este requisito, los IV cuentan con sistemas empotrados, que son sistemas de computación de recursos limitados en los que prima el cumplimiento de los tiempos (Guindel Gómez, 2012).

Dentro de las investigaciones y proyectos en el ámbito de los IV se pueden citar como algunos ejemplos:

- **Google Self Driving Car**
  Desde hace ya varios años, el gigante de internet, Google, trabaja en el proyecto de crear vehículos de conducción autónoma a través de su rama de investigación y desarrollo Google X. Hasta mayo de 2014, la compañía contaba con 6 Toyota Prius, un Audi TT y un Lexus RX450h, modificados para labores de investigación y pruebas, que hasta la fecha habían recorrido 1126000km (Sánchez J.M. (2014)) de forma autónoma, con un conductor a bordo por si fuese necesaria la intervención. Este tipo de vehículos cuentan con los siguientes componentes para hacer posible su navegación independiente (Markoff, J. (2010)):

  - *LIDAR* (*Laser Imaging Detection and Ranging*), situado sobre el techo del vehículo. Es un elemento giratorio que emite pulsos de luz láser y calcula la distancia a los objetos en función del tiempo en que la luz reflejada por estos tarda en ser devuelta al sistema. Tiene una radio de actuación de 70m con una precisión del orden de centímetros.
  - Cuatro radares estándar de automoción que complementan al LIDAR, tres delante y uno detrás, con un mayor rango de acción pero menor resolución, para calcular la distancia a objetos lejanos.
  - Una cámara instalada cerca del espejo retrovisor, capaz de detectar objetos delante del vehículo tales como peatones, señales de tráfico o los gestos realizados por las manos de los ciclistas.
  - Receptor GPS, apoyado por un sistema de navegación inercial.

Figura 1.2: Modelo de vehículo autónomo fabricado por Google

Usando estos elementos, el sistema es capaz de maniobrar el vehículo sin intervención humana, usando la técnica *SLAM (Simultaneous Localization and Mapping)* (ver 1.2.1), que construye y actualiza un mapa de los alrededores del vehículo, a la vez que mantiene éste posicionado en el mapa.

El 27 de mayo de 2014, Google presentó un nuevo **vehículo de fabricación propia**(ver figura 1.2), basado en las técnicas de navegación autónoma de los anteriores, cuya principal característica frente a estos es la ausencia de volante y pedales, su tamaño compacto y el uso exclusivo de motor eléctrico en su funcionamiento. Google asegura que durante los próximos años fabricará 100 unidades de estos vehículos, capaces de circular de forma totalmente independiente a 40km/h en entornos urbanos (Jiménez de Luis, Á. (2014)), aunque se espera que aún pase bastante tiempo hasta que puedan ser comercializados.

- **IvvI 2.0**
  Del acrónimo en inglés Intelligent Vehicle based on Visual Information, el IvvI 2.0 es la última versión del proyecto de investigación IvvI desarrollado por la universidad Carlos III de Madrid. El IvvI 2.0 es un vehículo inteligente que integra varios sistemas ADAS, utilizando principalmente la información captada por cámaras instaladas en el coche, para alertar al conductor acerca de distintas situaciones de peligro.

  Para ello, el sistema montado sobre un vehículo comercial (Nissan No-

Figura 1.3: IvvI 2.0 de la universidad Carlos III de Madrid

te) cuenta con seis cámaras, cuatro de ellas en el espectro visibles y las otras dos en el infrarrojo. Según el Laboratorio de sistemas inteligentes UC3M, de las primeras, una está supervisando al conductor, vigilando que esté prestando atención a lo que ocurre delante del vehículo y atento a los primeros síntomas de sueño para avisarle. Otras dos cámaras están dedicadas a la detección de peatones y otros vehículos para evitar colisiones. La última cámara captura en color y detecta las señales de tráfico para comprobar que se están cumpliendo las normas y el vehículo no realiza maniobras peligrosas. Las dos cámaras de infrarrojo se encargan de la detección de peatones en condiciones de baja visibilidad. Está previsto además dotarle de un sensor láser que permita la detección de posibles obstáculos a grandes distancias.

La información proveniente de los Sistemas de Ayuda a la Conducción solo se le comunica al conductor cuando el vehículo se encuentra en una situación real de peligro. Se evita así distraer al conductor con información inútil o redundante que solo sirva para distraerle de su tarea y suponga un peligro o un estorbo, más que una ayuda. Existen dos formas de hacer llegar esa información: a través de una señal sonora se le avisa del peligro y de la maniobra que debe realizar y a través de una pantalla que se encuentra integrada en el salpicadero se le muestra

la información captada y analizada por los sensores.

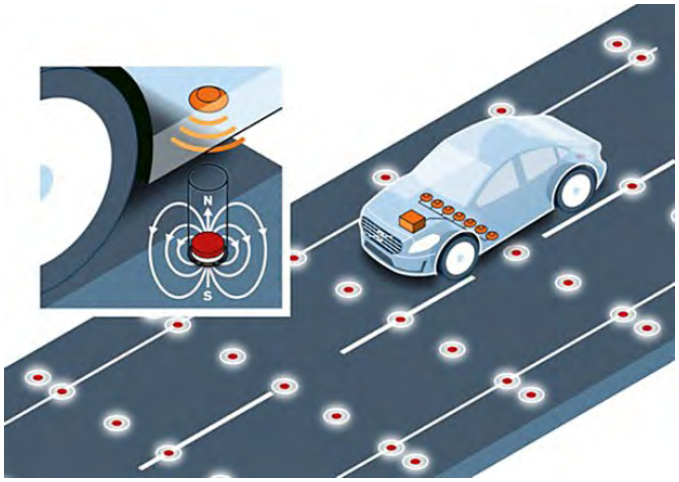
Para determinar el estado del vehículo se dispone de una sonda CAN-Bus que obtiene información del funcionamiento del vehículo así como de un sistema GPS-IMU que da la información de la posición y velocidad del vehículo. Tres ordenadores situados en el maletero analizan la información sensorial. Al estar conectados en red pueden intercambiar información de sus respectivos módulos.

- **Drive me**
  La compañía sueca VOLVO es una de las más activas en la búsqueda del vehículo de conducción autónoma. En este sentido, el proyecto *Drive Me* de la compañía VOLVO (ver figura 1.4a), que cuenta con la colaboración de la Administración para el Transporte y la Agencia de Transporte en Suecia, así como del gobierno de Gotemburgo y la institución de investigación Lindholmen Science Park, tratarán de tener en 2017 100 vehículos conducidos por consumidores reales en las calles de Gotemburgo capaces de lidiar con el tráfico real de la ciudad a lo largo de un circuito de calles de más de 50km en los que se probará su tecnología *Autopilot*. De momento la compañía está haciendo pequeñas pruebas con tráfico real, en días de diario. Los coches autónomos de VOLVO aún son prototipos y su sistema necesita algunos ajustes que se irán realizando a medida que se sepa más del rendimiento de los vehículos. El sistema de conducción autónoma está diseñado para adaptar la velocidad y maniobrar entre el tráfico sin necesidad de intervención humana. Para ello cuenta con un radar y una cámara que monitorizan el entorno constantemente, lo que le permite realizar seguimiento de líneas, controlar la velocidad, detectar otros vehículos y realizar incorporaciones al tráfico.

  Además de este proyecto, la compañía ha realizado otras pruebas con vehículos en carreteras magnéticas (ver figura 1.4b) para proporcionar una conducción autónoma, en un proyecto que aúna los vehículos inteligentes con las infraestructuras inteligentes. Para ello los investigadores colocaron dos filas de sensores en una carretera, con una formación orientada a delimitar el carril por el que se mueven los vehículos. El coche, por su parte, contenía sensores que medían con cuánta precisión y fiabilidad podía conocerse la posición del mismo. Una de las cuestiones que tuvieron que resolver los ingenieros fue el desarrollo de sensores capaces de recibir datos a grandes velocidades.

  En una primera prueba el coche tenía que recorrer 100 metros sembrados con 100 sensores magnéticos, que recogían la información proce-

(a) Coche del proyecto Drive Me de Volvo



(b) Investigación de VOLVO sobre el uso de carreteras magnéticas

Figura 1.4: Investigaciones de VOLVO sobre vehículos inteligentes

dente del vehículo. Rodando por debajo de los 72 km/h los dispositivos eran capaces de determinar la posición del coche con un margen de error de 10 cm. Posteriormente, las carreteras magnéticas lograron conocer la posición del automóvil con el mismo margen pero a velocidades que superaban los 145 km/h.

La ventaja de las carreteras magnéticas es que su rendimiento no empeora con las malas condiciones climáticas, como le ocurre a los sistemas electrónicos. Sin embargo, el coste es considerablemente superior (fuentes: G. Bejarano, P. (2014) (b) y G. Bejarano, P. (2014) (a)).

- **Renault Next Two**

En febrero de 2014, la marca francesa Renault presentó su primer prototipo de vehículo inteligente, el Renault Next Two (ver figura 1.5), que pretenden sea una realidad comercial para el año 2020. Desarrollar un coche autónomo capaz de circular solo de forma totalmente segura y en todas circunstancias aún necesitará unos años de investigación por parte de los constructores, y en este sentido el Renault Next Two asocia la delegación de la conducción y la conductividad para que los conductores dispongan de tiempo a la vez que mejoran su seguridad mediante una cámara, un radar y un cinturón de ultrasonidos, que crean una burbuja protectora alrededor del vehículo, garantizando así la seguridad de sus ocupantes.

Gracias al desarrollo de éstas nuevas tecnologías, el Next Two es capaz de liberar al conductor de la conducción en condiciones muy concretas: en caso de atasco hasta 30km/h en vía rápida y con una función de *aparca coches* que se encarga de aparcar el vehículo de manera totalmente autónoma en parkings adaptados para este tipo de dispositivos.

En atasco, el radar se utiliza para detectar el vehículo que circula delante y calcula su distancia y su velocidad. La cámara sirve para detectar las marcas laterales en el suelo con el fin de situar correctamente al vehículo en su carril.

Los datos procedentes del radar y de la cámara se transmiten a un supervisor que comunica con los calculadores de la dirección asistida eléctrica, del motor eléctrico y del frenado con pedal desacoplado.

En cuanto a la función de *aparca coches*, el vehículo es capaz de aparcar solo o de ir a buscar a su conductor al pie del edificio donde trabaja o a casa. Esta función se activa mediante un smartphone a través de una aplicación específica.

El Next Two dispone también de un sistema de conectividad capaz de captar todas las redes disponibles (3G, 4G, Wifi, etc.) y es compatible con el 100 % de sistemas para que los ocupantes puedan encontrarse como en casa o en la oficina. Estos elementos de conectividad, permiten además al coche valorar las condiciones de tráfico ya que los datos se procesan en conjunto con información GPS que el vehículo obtiene gracias a su naturaleza conectada. De esta forma, es capaz de ofrecer rutas alternativas a los conductores para evitar atascos y para ahorrar tiempo (fuentes: Bravo Corrales, D. (2014) y Smart Life (2014)).

(a)

(b)

Figura 1.5: Prototipo Next Two de Renault

Además de estos ejemplos, otras muchas compañías se han embarcado en el reto de la conducción autónoma, como la **japonesa Nissan** que es uno de los más firmes defensoras de esta nueva tecnología, estableciendo el 2020 como fecha límite para tener un vehículo totalmente autónomo en las calles, como una extensión a sus tecnología Safety Shield (Nissan (2014)), que cuenta con varios ADAS y sistemas de seguridad activa ya implantados en sus vehículos.

Los vehículos inteligentes son ya una realidad y los vehículos de navegación semi-autónoma no tardarán en implantarse en las carreteras de todo el mundo, sin embargo, aún quedan años hasta que la conducción sea una tarea totalmente en manos de una máquina y en este sentido la investigación y el desarrollo de sistemas capaces de cumplir con esta tarea sigue aún muy

abierto.

1.2. Motivación, objetivos y alcance del proyecto

1.2.1. Motivación: Precisión de los sistemas de posicionamiento

Como se ha visto en el apartado anterior, la consecución de vehículos inteligentes con un alto grado de independencia requiere aún años de desarrollo, aunque los esfuerzos de las empresas y centros de investigación están aproximando al presente esta realidad.

Uno de los elementos clave de cara a la consecución de un vehículo totalmente autónomo, y para el funcionamiento de muchos sistemas ADAS, es lograr sistemas de posicionamiento robustos y precisos, que permitan conocer con gran exactitud la localización del vehículo en cualquier momento y circunstancia.

Los sistemas de posicionamiento más ampliamente utilizados y conocidos son los Sistemas de Navegación Global por Satélite (*Global Navigation Satelite System, GNSS*), entre los que destaca el *Global Positioning System*, o ***GPS***. El GPS es un servicio propiedad de los EEUU, pero disponible para usuarios de todo el planeta desde el año 2009, que proporciona información sobre posicionamiento, navegación y cronometría. Este sistema está constituido por tres segmentos:

- **Segmento espacial.** Esta formado por una constelación de 24 satélites operativos, distribuidos en 6 órbitas a unos 20000km de la superficie terrestre y que tardan alrededor de 12 horas en completar una vuelta a la Tierra, de tal manera que se cumple que desde cualquier punto del planeta siempre hay más de 4 satélites visibles, que es el número mínimo necesario para que un receptor GPS en la tierra logre establecer una ubicación en longitud, latitud y altura. Cuantos más satélites son visibles a la vez, mayor precisión tendrá el posicionamiento, siendo este número habitualmente de 8. Para lograr el objetivo de posicionamiento los satélites están equipados con un transmisor de señales de radio de alta frecuencia (1575.42 MHz para uso civil), un sistema de computación y un reloj atómico de cesio de muy alta precisión.

- **Segmento de control.** Está formado por una red de estaciones de seguimiento y control en tierra distribuidas por todo el mundo a fin de mantener los satélites en la órbita apropiada, mediante maniobras de mando, y ajustar los relojes satelitales. También realizan el seguimiento de los satélites, cargan información de navegación actualizada

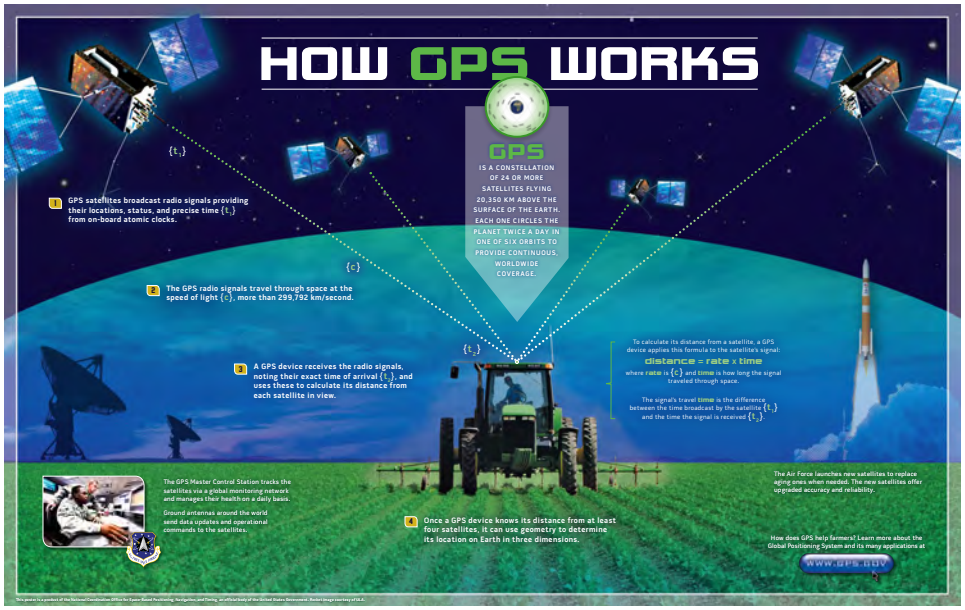


Figura 1.6: Funcionamiento del sistema GPS(Gobierno de Estados Unidos (2014))

y garantizan el correcto funcionamiento de la constelación.

- **Segmento de usuario.** Consiste en el receptor de GPS que recibe las señales de los satélites y las procesa para calcular la posición tridimensional y la hora precisa.

Con estos elementos, a grandes rasgos, el GPS funciona de la siguiente manera:

1. Los satélites GPS transmiten señales de radio con su localización, estado y hora precisa (t_1) obtenida de sus relojes atómicos.

2. Estas señales cruzan el espacio a la velocidad de la luz.

3. Un receptor GPS en la tierra recibe las señales de radio y anota la hora exacta de recepción (t_2). A partir de la diferencia de tiempos y conocida la velocidad de la luz, se puede calcular la distancia entre el receptor GPS y el satélite en el momento de emisión de la señal.

4. Una vez el receptor GPS sabe su distancia con respecto a al menos cuatro satélites, puede determinar su posición en un espacio tridimensional (longitud, latitud y altura). Tres de las señales de satélite son necesarias como mínimo para determinar la posición geométricamente por triangulación y una más para resolver la sincronización temporal.

Figura 1.7: Visualización del efecto multitrayectoria

De esta manera, el sistema GPS es capaz de proporcionar una localización en el espacio para cualquier punto de la tierra, suficientemente exacta para aplicaciones generalistas. La precisión exacta de posicionamiento del GPS, depende sin embargo de diversas fuentes de error (Blogspot De Topografía (2012)):

- **Retrasos atmosféricos.** Al pasar la señal de satélite por la atmósfera, su velocidad disminuye. Esta disminución de la velocidad de la señal con respecto a la velocidad de la luz en el vacío induce un error en el cálculo de la distancia al satélite. Este efecto no es constante y se produce principalmente en la ionosfera y en la troposfera.

- **Errores en el reloj del satélite o del receptor.** Siendo más importantes los segundos debido a la imprecisión del receptor. Los primeros son debidos principalmente a efectos relativistas, siendo corregidos constantemente desde el segmento de control.

- **Efecto multitrayectoria** (ver figura 1.7). Cuando el receptor GPS se encuentra cerca de objetos cuyas superficies reflejan las señales de radio de los satélites, tales como edificios, algunas de las señales de radio que procesa el receptor no viajan directamente a su antena, sino que rebotan antes en los objetos alrededor, produciendo mediciones falsas.

- **Dilución de la precisión (DOP)**. Son coeficientes que caracterizan la influencia de la geometría de la constelación GPS observada por un receptor. Estos coeficientes varían con el tiempo, al variar la posición de los satélites observados o al cambiar algunos de ellos. Cuando los satélites están bien distribuidos, la posición se determina en un área menor y el margen de error posible es mínimo. Cuando los satélites están muy cerca unos de otros, el área de encuentro también aumenta, de manera que se incrementa la incertidumbre de la posición.

Diferentes técnicas utilizadas para compensar estos errores de medida en el receptor, hacen que la precisión del GPS sea, en general, del orden de 10m. Sin embargo, en entornos urbanos éste puede aumentar hasta los 100m debido principalmente al efecto multitrayectoria. Esto, unido a la pérdida de señal de satélites en estos entornos debido a oclusiones de las señales, producidas por árboles, edificios, túneles, etc., hace que el posicionamiento GPS no sea lo suficientemente robusto, en entornos urbanos como para asegurar un correcto posicionamiento en aplicaciones críticas, como es la navegación de un vehículo autónomo.

Para asegurar un posicionamiento más preciso del sistema GPS o tratar de sustituir por completo a los sistemas GNSS, diferentes métodos para la localización han sido desarrollados, entre los que se pueden citar (ver también 2.2 para un análisis bibliográfico sobre aquellos sistemas basados en información visual):

- **DGPS o GPS diferencial.** Es un sistema que proporciona a los receptores de GPS correcciones de los datos recibidos de los satélites GPS, con el fin de proporcionar una mayor precisión en la posición calculada. El fundamento radica en el hecho de que los errores producidos por el sistema GPS afectan por igual (o de forma muy similar) a los receptores situados próximos entre sí. Los errores están fuertemente correlacionados en los receptores próximos. Siguiendo esta idea, Un receptor GPS fijo en tierra (referencia) que conoce exactamente su posición basándose en otras técnicas, recibe la posición dada por el sistema GPS, y puede calcular los errores producidos por el sistema GPS, comparándola con la suya, conocida de antemano. Este receptor transmite la corrección de errores a los receptores próximos a él, y así estos pueden, a su vez, corregir también los errores producidos por el sistema dentro del área de cobertura de transmisión de señales del equipo GPS de referencia (máximo 200km). La precisión de este tipo de sistemas puede ser del orden de centímetros, pero el inconveniente es que es necesario el uso de estaciones de referencia, con el coste derivado en infraestructuras (fuente: Wikipedia (Sistema de posicionamiento global)).

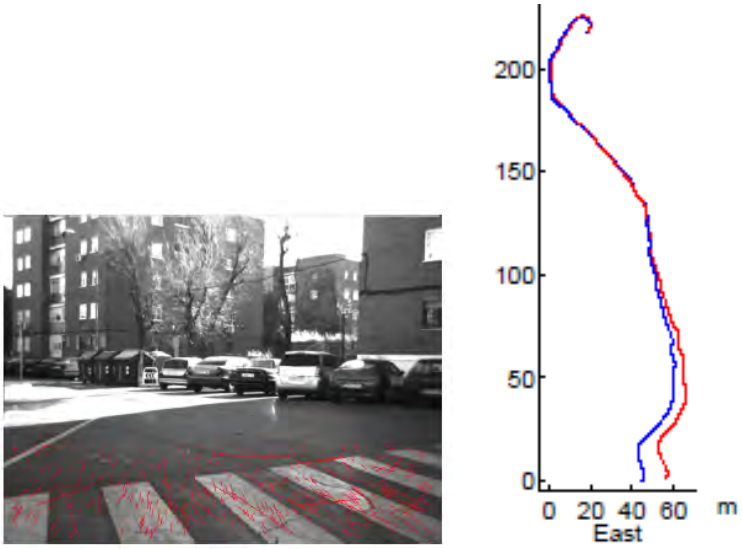
- **Odometría.** Combina la información del GPS con la de sensores instalados en el vehículo (**Unidad de medición inercial o *Inertial Measurement Unit, IMU***), como giróscopos y acelerómetros, a partir de los cuales se estima la trayectoria seguida por el vehículo incrementando de esta manera la precisión del posicionamiento o siendo capaz de proporcionar información sobre la localización del vehículo en ausencia de señal GPS. El proceso a través del cual se realiza el cálculo de la trayectoria se denomina odometría, y su funcionamiento se basa en conocer el diámetro de las ruedas del vehículo, el número de vueltas realizadas por éstas, la orientación de la dirección del vehículo, obtenida a partir de los datos de los giróscopos, y la aceleración del vehículo, dada por los acelerómetros.

  Una de las grandes desventajas de usar IMU para la navegación es que estas normalmente son afectadas por un error acumulativo. Debido a que el sistema de guía está continuamente agregando los cambios detectados a las posiciones previamente calculadas, cualquier error en la medición, no importa lo pequeño que sea, se va acumulando de punto a punto. Esto lleva a una "deriva", o diferencia, que aumenta siempre entre donde el sistema piensa que se encuentra localizado y la posición real (Wikipedia, Unidad de medición inercial). Debido a este motivo, este tipo de sistemas pueden servir de apoyo al GPS, pero no funcionar de forma autónoma durante largas distancias si la señal de GPS no está disponible.

- **Odometría Visual (*Visual Odometry, VO*).** Esta técnica, denominada así por analogía con la odometría tradicional, consiste en determinar la posición y la orientación de una cámara, o de un sistema de cámaras, mediante el análisis de una secuencia de imágenes adquiridas sin ningún conocimiento previo del entorno. De esta manera, determinando las posiciones sucesivas de la cámara anclada al vehículo usando principios de óptica (ver seccón 3.3), es posible determinar la trayectoria de éste (García y Vázquez, 2007). Para calcular las posiciones sucesivas de la cámara, los algoritmos de odometría visual procesan las imágenes recibidas para detectar una serie de puntos pertenecientes a objetos estáticos, e ir siguiendo su movimiento en sucesivas capturas (Guindel Gómez, 2012).

  Al igual que la odometría tradicional, los sistemas de odometría visual presentan problemas de deriva, que hacen que los datos dejen de ser validos tras pocos kilómetros de recorrido.

- **Localización y Mapeado Simultáneo (*Simultaneous Localization and Mapping (SLAM)*)**. Estas técnicas son heredadas del cam-



(a) Extracción de puntos característicos de las imágenes (b) Trayectoria calculada

Figura 1.8: Proceso de odometría visual

po de la robótica, en donde los robots deben moverse por entornos desconocidos en los que en muchas ocasiones no estas disponibles los sistemas de localización satelitales (interior de edificios, otros planetas, etc.). Las técnicas SLAM, como se deduce de su nombre, se usan para crear un mapa tridimensional del entorno por el que se mueve el robot, al mismo tiempo que se estima la posición de éste dentro de dicho mapa. Para implementar este tipo de sistemas tradicionalmente se han usado sensores de ultrasonidos, radares o sensores láser (LIDAR), pero en los últimos tiempos ha crecido el número de trabajos en el contexto del SLAM que utilizan cámaras para la resolución del problema, por las ventajas que estos sistemas ofrecen en cuanto a coste y facilidad de adquisición y uso. Esto ha dado lugar a lo que se denomina *visual SLAM* (*vSLAM*). El funcionamiento de vSLAM es similar al de la VO, con la diferencia de que en el primero, los puntos 3D obtenidos a partir de las imágenes son utilizados como puntos de referencia y sus posiciones son optimizadas junto con la posición de la cámara (Floros et al., 2013). Al igual que los sistemas de VO tienen el problema de de la acumulación de la deriva. Para corregir éste, los vehículos están obligados a conducir en bucles u otro vehículo debe haber mapeado la calle antes, de manera que el algoritmo pueda reconocer los mismos puntos de referencia y por lo tanto compensar los errores en la localización. En el primer caso, la conducción en bucles limita en gran medida la libertad de movimiento del vehículo, y en el segundo, un

mapa pre-calculado de puntos de referencia requiere gran cantidad de datos almacenados. A pesar de este hecho, esta segunda técnica es la utilizada por los vehículos autónomos de Google, junto con un sensor LIDAR.

- Algoritmos de *map-matching* (Quddus et al., 2007). Son algoritmos usados en conjunto con otros sistemas de posicionamiento, a partir de los cuales se obtienen datos de localización, que son integrados a continuación con datos espaciales de la red de carreteras, ya sea imágenes por satélite, o Sistemas de Información Geográfica (GIS) tales como Google Maps. De esta manera pueden identificar el segmento correcto por el que circula un vehículo y determinar la posición del mismo en ese segmento. Usando este tipo de algoritmos en conjunto con los sistemas de GPS o los sistemas de odometría, se pueden corregir los errores de posicionamiento de los mismos, minimizando además la deriva de los segundos.

  Un buen número de algoritmos de map-matching han sido desarrollados por los investigadores usando diferentes técnicas: análisis topológico de datos de la red de carreteras, teorías probabilísticas, filtros de Kalman, lógica difusa y propagación de la creencia (*belief propagation*). En la sección 2.2, se analizan más en detalle algunos de los ejemplos de estos algoritmos, presentes en la bibliografía, que se utilizan en conjunto con información visual.

A pesar de las alternativas expuestas, el desarrollo de un sistema de posicionamiento lo suficientemente preciso como para permitir asistir a tareas críticas, como la navegación autónoma de un vehículo, sigue siendo un problema abierto a la investigación, el cual supone la **motivación de este proyecto.**

1.2.2. Objetivos

Derivado de la motivación del presente trabajo, expuesta en el apartado anterior, y en el contexto de los vehículos inteligentes (ver apartado 1.1), el objetivo de este proyecto es la **concepción** de un método de apoyo a sistemas de posicionamiento como el GPS, o la odometría visual (GPS/VO), desarrollado para mejorar la navegación de un vehículo, especialmente en entornos urbanos.

Para ello, se pretende utilizar un algoritmo de visión por computador que, a partir de imágenes obtenidas por un sistema de cámaras estéreo, sea capaz de reconocer, posicionar y medir la distancia a pasos de peatones pintados en la calzada, utilizando la información obtenida sobre la localización

de estas marcas viales como referencia dentro del entorno. De esta manera, la idea es comparar el dato de distancia al paso de peatones calculado por el algoritmo, con el dato de distancia con respecto a ese mismo paso de peatones que arroja el sistema GPS/VO, corrigiendo éste último valor si el resultado de la comparación es lo suficientemente grande.

Para determinar cuándo debe actuar el algoritmo y qué paso de peatones del mundo real se corresponde con el detectado en las imágenes, se pretende utilizar la información aportada por un GIS. El GIS seleccionado a este fin es la cartografía de OpenStreetMaps (OSM) (ver sección 3.11), ya que dispone en su base de datos de la localización de los pasos de peatones y es además libre y editable por cualquier usuario, por lo que se podrá añadir esta información si no estuviera disponible en algún caso.

El objetivo de este proyecto no es la implementación del desarrollo en un sistema real, sino únicamente el planteamiento de la idea y la comprobación acerca de la factibilidad de ésta, especialmente en lo referido a la posibilidad de crear un algoritmo de reconocimiento de pasos de peatones lo suficientemente robusto como para cumplir con el objetivo global planteado.

1.2.3. Alcance

En este apartado se pretende establecer los aspectos que serán desarrollados en el presente trabajo con el fin de cumplir con los objetivos planteados. A tal fin, el desarrollo de este proyecto cubre los siguientes aspectos:

- **Creación de un algoritmo** capaz de detectar, posicionar y medir la distancia a un paso de peatones que aparezca en un par de imágenes estéreo. Para ello se utilizará el software de **MATLAB** (ver sección 3.10), con el fin de crear un conjunto de funciones que puedan llevar a cabo esta tarea.

- **Planteamiento teórico** acerca de la manera de abordar la coordinación entre el algoritmo creado y el sistema GPS/VO. En este aspecto, se analizarán la forma de extraer la información acerca de la posición de un paso de peatones, a partir de la cartografía de OSM, y la lógica planteada para el sistema completo en caso de que éste fuera implementado. Puesto que los objetivos del proyecto no cubren la implementación del concepto expuesto en un sistema real, el desarrollo del sistema completo que cubra esta parte se deja abierto a trabajos futuros.

1.3. Estructura del documento

En este primer capítulo se ha introducido el contexto en que este proyecto de investigación queda enmarcado y se ha expuesto el interés que suscita la creación del mismo, así como los objetivos que pretenden lograrse durante su desarrollo y el alcance de las acciones llevadas a cabo para la consecución de los mismos.

A continuación, en el capítulo 2, se incluye un estado del arte relativo principalmente a los dos bloques clave de este trabajo: por un lado, el reconocimiento de pasos de peatones mediante algoritmos de visión por computador y, por otro, la utilización de información visual del entorno para mejorar la precisión de sistemas de posicionamiento. A tal fin, en esta sección se analiza una extensa bibliografía acerca de estos dos temas, que en su momento sirvió como punto de partida para la elaboración del proyecto.

Por su parte, en el siguiente capítulo (capítulo 3), se incluye una introducción teórica, la cual da algunas ideas básicas sobre conceptos que se consideran importantes en el entendimiento de las explicaciones que aparecen a lo largo de este documento: una aproximación a la transformada de Fourier, el concepto de mapa de disparidad, una mención a las herramientas usadas en el proyecto y la estructura básica de un algoritmo de visión por computador, así como fundamentos acerca de imágenes digitales y algunas de las operaciones básicas usadas en su procesamiento; son los apartados incluidos en este punto.

Por su parte, en los siguientes tres capítulos, se desarrolla el contenido principal del trabajo: el capítulo 4 explica con detalle los pasos dados, problemas encontrados y soluciones aportadas durante el transcurso de la investigación. En el capítulo 5 se analizan los resultados obtenidos y su adecuación a los objetivos planteados al comienzo de la misma, mientras que en el capítulo 6 se elabora una conclusión a partir de estos resultados y se proponen algunas alternativas de diseño y opciones que podrían ser tenidas en cuenta en desarrollos futuros que tengan como base este trabajo.

Tras estos tres capítulos, se añade un capítulo final (capítulo 7) en que se analizan los costes imputados a la elaboración del presente proyecto.

A continuación de estos capítulos se incluyen también una serie de apéndices que complementan la información aportada en los capítulos principales: el apéndice A muestra todas las funciones escritas en MATLAB para implementar el algoritmo resultante del proyecto y otras funciones que han sido útiles durante su desarrollo. Por su parte, en el apéndice B se ofrece un análi-

sis de las librerías utilizadas para tratar con la cartografía de OpenStreetMap y el enlace de descarga donde se pueden encontrar las mismas.

Capítulo 2

Estado del arte

Una vez marcados los objetivos del proyecto, en este capítulo se realiza un estudio sobre las investigaciones existentes en la literatura que se pueden relacionar con los mismos: por un lado, **el reconocimiento visual de pasos de peatones** y, por otro, **la utilización de información visual como apoyo a los sistemas de posicionamiento**. A partir de las conclusiones obtenidas por otros autores, se pretende extraer una idea de las soluciones planteadas a los dos problemas expuestos, para así tomar éstas como base del desarrollo posterior. No es por tanto el objetivo de este punto hacer un análisis exhaustivo de toda la bibliografía relacionada con los dos temas citados, sino dar una visión general de los trabajos existentes afines a los mismos.

2.1. Reconocimiento visual de pasos de peatones

Los pasos de peatones, también conocidos como pasos de cebra, son un tipo de marca vial que suele verse pintada en la calzada de la mayoría de las ciudades del mundo. Están formados por un número variable de franjas, generalmente de color blanco, dibujadas en el suelo de la carretera y separadas entre sí por una distancia constante, lo que da lugar a un patrón muy característico de bandas de color alterno, paralelas entre sí y dispuestas transversalmente al área de circulación de los vehículos.

Los pasos de peatones están destinados al cruce de los viandantes de un lado a otro de la carretera a través de la zona delimitada por esta señal, la cual les otorga preferencia ante el paso de los vehículos. Los automóviles están obligados a detenerse si a través del paso de cebra está cruzando un peatón o éste se dispone a hacerlo. Sin embargo, el mal estado de la señalización, la falta de visibilidad, distracciones de los conductores, etc. provoca que estas zonas de cruce se conviertan en ocasiones en zonas de peligro para los transeúntes. Según un estudio elaborado por el RACC (RACC, 2008),

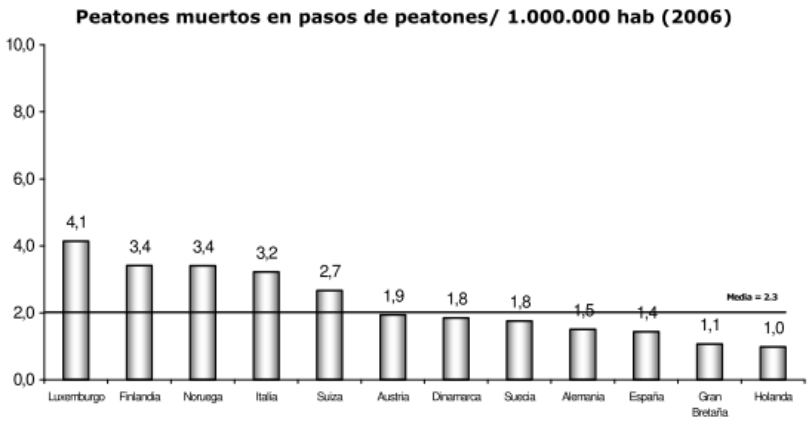


Figura 2.1: Estadística de peatones muertos en pasos de peatones en 2006

cada año 8000 peatones pierden la vida en Europa en accidentes de tráfico, de los cuales uno de cada cuatro se produce como consecuencia de atropellos en pasos de peatones.

Por otro lado, **este tipo de señalización supone un inconveniente importante para aquellas personas con algún tipo de discapacidad visual**. Para los peatones con este tipo de minusvalía es un verdadero reto enfrentarse cruces de este tipo, puesto que los sistemas que utilizan tradicionalmente para orientarse, tales como bastones o perros guía, resultan ineficientes a la hora de cruzar adecuadamente una calle atravesando un paso de peatones (de manera perpendicular a las franjas del mismo y sin abandonar las marcas viales). Además, sistemas más modernos relacionados con el uso de sensores, como radáres, no reconocen este tipo de señalizaciones, y otros dispositivos desplegados recientemente en muchas ciudades, como es la señalización sonora del cruce, no está implantada en todos ellos y sólo se encuentra en algunos de los regulados por semáforo. Ocurre lo mismo con otros sistemas como el pavimento táctil.

El reconocimiento visual de pasos de peatones es una aplicación de las técnicas de visión por computador bastante recurrente en la literatura. Ésta consiste en la detección, mediante un algoritmo adecuado y el uso de una o más cámaras, del patrón geométrico que compone este tipo de señales. De los dos problemas derivados del uso de los pasos de peatones citados anteriormente, es de donde surge la motivación para la mayor parte de los trabajos encontrados en la bibliografía con respecto a este tema.

Sin embargo, aunque la solución a ambos inconvenientes tiene como pun-

to de partida el reconocimiento del patrón geométrico que representa el paso de peatones en las imágenes, la forma de enfrentarse a cada caso tiene sutiles diferencias en función de la aplicación para la que esté diseñado el algoritmo de visión. Esto se debe principalmente al hecho de que la perspectiva desde la que el paso de peatones debe ser detectado por el mismo y la colocación de la cámara, varían según la aplicación para la que éste sea ideado. En un algoritmo cuyo uso está dirigido a personas con discapacidad visual, el paso de peatones deberá ser detectado preferentemente desde el punto de vista en que las franjas son perpendiculares al sentido de la marcha de la persona y la cámara estará instalada en algún dispositivo móvil portado por ésta. Debido a esto, el entorno de actuación del sistema no está limitado únicamente a la carretera como en el caso de un vehículo. Por otro lado, en el supuesto de una aplicación diseñada para ser instalada en un automóvil, las bandas del paso de peatones estarán colocadas paralelamente a la dirección de circulación del mismo, estando la cámara anclada de una manera más o menos firme en el coche. Este segundo caso es por tanto un problema con muchas más restricciones y por tanto más fácil de afrontar. Este paradigma es más similar al que será abordado en este trabajo.

Dadas estas diferencias a la hora de abordar el reconocimiento visual del paso de peatones en función de la aplicación, se analizarán a continuación diferentes trabajos presentes en la bibliografía en relación con este problema, desde el punto de vista de los dos usos distintos mencionados.

2.1.1. El reconocimiento de pasos de peatones en la ayuda a personas con discapacidad visual

Por un lado, los sistemas de ayuda a personas con algún tipo de discapacidad visual que aparecen en la bibliografía, tratan de detectar el paso de peatones en las imágenes con el objetivo final de guiar a las personas hasta el mismo y asistirles a la hora de cruzarlos. En este sentido, una de las características del paso de peatones más utilizada en su detección, es el hecho de que en la realidad este patrón geométrico está compuesto por franjas paralelas. Sin embargo, cuando las bandas son proyectadas en el plano de la imagen, sólo siguen siendo paralelas aquellas que lo son además a la llamada *línea del horizonte*. En caso contrario, las líneas que componen la representación del paso de peatones en la captura se encuentra en el horizonte en lo que se denomina *punto de fuga*.

Basándose en esta idea, Se y Brady (2003) propone un método de reconocimiento a través del cual busca en las imágenes rectas mediante la ***Transformada de Hough***. De las rectas obtenidas, obtiene grupos de ellas aproximadamente paralelas y comprueba su concurrencia en un punto de

fuga. Para ello utiliza *RANSAC* (Fischler y Bolles, 1981). Previamente a esto, aplica un *detector de bordes de Canny* (Canny, 1986). Luego, busca otras rectas que apoyen esta hipótesis, para así determinar la mejor de ellas. Con esto obtiene una estructura compuesta por líneas paralelas, que son potenciales bordes de las bandas del paso de peatones. Para distinguir si esta estructura es realmente o no un paso de peatones introduce una restricción adicional: las rectas obtenidas pueden ser divididas en dos grupos de líneas paralelas equiespaciadas. Estos grupos son formados en función de los cambios de intensidad a un lado y a otro de las rectas (claro a oscuro u oscuro a claro). El hecho de imponer esta restricción se debe a otra característica importante de un paso de peatones; está compuesto por franjas oscuras y claras alternativas. Por este motivo, un borde detectado perteneciente a un paso de peatones tendrá a un lado un nivel de gris alto y al otro un nivel de gris bajo mientras que en el siguiente será al contrario, repitiéndose esta alternancia a lo largo de toda la señal. Además, en este trabajo se estudia la posibilidad de que un paso de peatones pueda ser confundido por el algoritmo con escaleras que aparezcan en las imágenes y aporta tres métodos distintos para distinguir ambas estructuras.

En Wang y Tian (2012) se utiliza una aproximación al problema muy similar a la anterior, pero hace uso de una cámara RGBD a la hora de distinguir los pasos de peatones de escaleras, utilizando la profundidad de los objetos aportada por la capa D de la imagen como elemento de clasificación a través de *Support Vector Machines(SVM)*.

En la práctica, el método implementado en Se y Brady (2003) es muy costoso computacionalmente. Debido a este motivo, Uddin y Shioyama (2005) desarrollan otra aproximación más eficiente al problema, utilizando la idea de que el paso de peatones está compuesto por franjas que alternan niveles de intensidades extremas. Debido a esta característica, el histograma de una zona de la imagen en la que aparezcan fragmentos del paso de peatones va a ser bimodal o bipolar, es decir, va a tener dos zonas bien diferenciadas dentro del mismo. Se usa por tanto un **método de segmentación local que detecta zonas con una bipolaridad alta, uniendo a continuación regiones vecinas con una distribución de intensidades similar**. Una vez extraída de la imagen la zona candidata a ser un paso de peatones, ésta se binariza usando el valor medio. Además, debido a la intención de aplicar este algoritmo en la ayuda a personas con algún tipo de problema de visión, sólo interesan aquellos pasos de peatones que estén cerca de la persona y vistos desde una perspectiva frontal al mismo. Para lograr esto, por un lado las regiones obtenidas son descartadas si no se encuentran centradas en la imagen y, por otro, se aplica un umbral Gaussiano sobre la *Transformada Rápida de Fourier (FFT) bidimensional* de la región (ver sección 4.1.5). Si

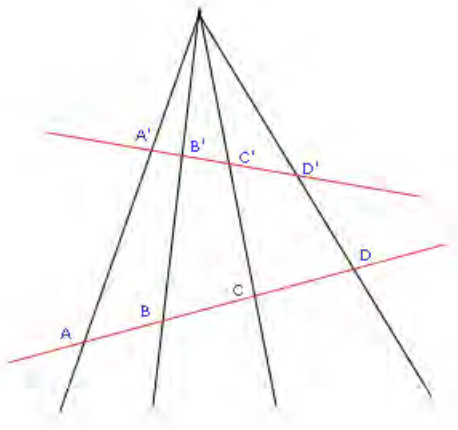


Figura 2.2: Definición del cross-ratio. Fuente: Wikipedia (Cross ratio)

algún pico dentro de la FFT bidimensional supera este umbral, se considera que no existe un paso de peatones en esa imagen, puesto que un paso de peatones visto frontalmente sólo produciría un pico en el espectro de frecuencias. Adicionalmente, como paso previo al proceso de segmentación, se ecualiza el histograma de la imagen para mejorar el contraste si la media de intensidad de ésta es menor que un valor. Una idea similar será usada posteriormente en este trabajo. Un filtro de mediana de 5x5 elimina también el ruido impulsional.

Una vez obtenida la binarización de la zona candidata de la imagen, **se evalúa si ésta representa un paso de peatones a través de un invariante proyectivo denominado *cross-ratio***. Según esta propiedad, si un conjunto de cuatro rectas concurrentes es intersectado por otra recta, los segmentos que quedan definidos entre ellas mantienen una relación que es constante independientemente de la recta de cruce, cumpliendo la expresión 2.1 (ver figura 2.2):

$$I = \frac{AC \times BD}{BC \times AD} = \frac{A'C' \times B'D'}{B'C' \times A'D'} \tag{2.1}$$

Teniendo en cuenta esto, el método anterior traza una recta por el centro de la zona candidata y calcula el corte con el borde de las bandas, obteniendo los puntos equivalentes a ABCD. Si el cross-ratio calculado se encuentra dentro de unos márgenes, se considera que en la zona candidata hay un paso de peatones.

Este algoritmo es mucho más eficiente que el aplicado en Se y Brady (2003), pero presenta el problema de que falla ante oclusiones parciales del paso de peatones, producidas por ejemplo por el paso de un vehículo por encima del mismo. Ante este hecho, surgen varios trabajos que tratan de solucionar este inconveniente a partir del método denominado ***figure-ground segmentation***(Yu y Shi, 2003). Ejemplos de estos trabajos son los presentados en Coughlan y Shen (2006) y Ivanchenko et al. (2008).

En 3.1 se explicaron los dos posibles enfoques desde el que puede abordarse un algoritmo de visión por computador y los principales inconvenientes de cada uno de estos enfoques. Los trabajos basados en *figure-ground segmentation*, presentados en los trabajos mencionados, aportan un nuevo enfoque que trata de aunar las fortalezas de ambos métodos de creación de algoritmos de visión.

En su aplicación al problema del reconocimiento de un paso de peatones, en Coughlan y Shen (2006) y Ivanchenko et al. (2008) se propone un método que extrae primero segmentos de recta de la imagen (enfoque bottom-up) y a continuación los relaciona a través de condiciones como paralelismo, cross-ratio, etc. Para asignar cada segmento a figura (paso de peatones) o fondo se considera cada uno como un nodo, distribuyendo luego estos en un modelo gráfico que es un campo aleatorio de Markov (He et al., 2004), el cual expresa las relaciones existentes entre dos o más nodos en función de las relaciones establecidas previamente entre ellos. El estado más probable de cada nodo en el grafo es inferido utilizando *factor Belief Propagation (factor BP)* (Kschischang et al., 2001) (enfoque top-down). Si tras estos procesos suficientes nodos han sido clasificados como figura, se concluye que en la captura hay un paso de peatones.

Para realizar la primera etapa de extracción de características, en Ivanchenko et al. (2008) se filtra la imagen y a continuación se utiliza un detector de bordes basado en el gradiente. A continuación, de los bordes extraídos se queda con aquellos que son aproximadamente horizontales, pues sólo desea obtener pasos de peatones frontales. Por último trata de unir segmentos co-lineales cercanos que no hayan quedado conectados durante los procesos anteriores.

En cuanto a las relaciones de dependencia que utiliza entre los nodos para construir el campo aleatorio de Markov, se pueden clasificar en función del número de nodos que conectan como sigue:

- Las de relación 1, que sólo describen al propio nodo, son la longitud y la magnitud del gradiente. Un segmento perteneciente a un paso de peatones tiende a tener una mayor longitud y un valor del gradiente

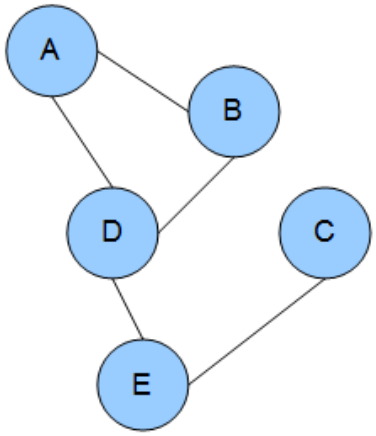


Figura 2.3: Ejemplo de un grafo de un campo aleatorio de Markov. Cada línea representa las relaciones de dependencia entre nodos. En este caso A depende de B y D. B depende A y D. D de depende de A, B y E. E depende de D y C. C depende de E. Fuente Wikipedia: `http://en.wikipedia.org/wiki/Markov_random_field`

más alto (a cada lado del segmento hay niveles de intensidad muy dispares) que el de aquellos que pertenecen a otros elementos dentro de la fotografía.

- Las relaciones que utiliza para asociar dos nodos son el paralelismo, solapamiento horizontal y dos características relativas a la consistencia del color. El solapamiento horizontal se refiere al hecho de que dos segmentos de línea perteneciente a un paso de peatones frontal, van a coincidir en su coordenada x dentro de la imagen. Por otro lado, en cuanto a las de consistencia del color, la primera de ellas analiza que el nivel de gris entre dos segmentos consecutivos sea aproximadamente igual para todos los píxeles, mientras que la segunda estudia el signo de sus gradientes, que deberían ser opuestos en el caso de un paso de peatones (cambio de "oscuro" a "claro" en uno y de "claro" a "oscuro" en el siguiente).

- La dependencia de orden tres utilizada se refiere al carácter monótono decreciente del ancho de las franjas del paso de peatones, conforme éstas están más alejadas de la cámara debido a la perspectiva.

- El cross-ratio, ya citada anteriormente, es la relación de dependencia de orden cuatro utilizada.

Además, el trabajo desarrollado en Ivanchenko et al. (2008) funciona en

tiempo real, implementado en un teléfono móvil Nokia N95, emitiendo un pitido de aviso cuando el paso de peatones es localizado por la cámara del dispositivo.

En la línea de implementación del sistema de reconocimiento del paso de peatones en teléfonos móviles, el trabajo desarrollado en Ahmetovic et al. (2011) desarrolla una aplicación llamada *ZebraLocalizer* para dispositivos iOS, que no sólo detecta el paso de peatones en la imagen, sino que además guía a la persona hasta el mismo a través de comandos de voz y señales acústicas. Para el reconocimiento del paso de peatones en las imágenes utiliza una librería de software llamada *ZebraRecognizer*.

La novedad frente a trabajos anteriores del método desarrollado por *ZebraRecognizer* para encontrar pasos de peatones en las imágenes, es el uso de los acelerómetros que los *smartphones* modernos llevan integrados. Utilizando estos sensores, el primer paso llevado a cabo por el algoritmo es calcular la línea del horizonte de la imagen de entrada. Esta información es útil porque reduce la zona de la imagen en que hay que buscar a la mitad (sólo debajo del horizonte). Además, este paso ayuda a calcular rectas paralelas y a estimar la orientación de la persona con respecto al paso de peatones. En el siguiente paso, *ZebraRecognizer* extrae características de la parte de la imagen por debajo de la línea del horizonte, que son segmentos de línea aproximadamente horizontales, utilizando una adaptación del algoritmo Line Segment Detector (LSD) (Grompone et al., 2010). Una vez obtenidos estos segmentos que representan potenciales bordes del paso de peatones, son agrupados entre ellos usando características similares a las utilizadas en los procesos de figure-ground segmentation, pero con la diferencia de que las restricciones son aplicadas de manera consecutiva, descartando un grupo en el momento en que tiene menos de 4 elementos (se considera que un paso de peatones debe estar compuesto por al menos dos bandas, lo que implica cuatro segmentos de recta consecutivos). La secuencia de operaciones es:

1. Se agrupan líneas que tienen aproximadamente la misma pendiente, esto es segmentos paralelos.

2. Para cada grupo se comprueba que todos los segmentos tienen una distancia al centro del conjunto aproximadamente igual. Si no es así, se divide adecuadamente el conjunto.

3. De los grupos restantes, se comprueba que el signo del gradiente de dos segmentos consecutivos tiene signos opuestos. En caso contrario, se divide ese grupo consecuentemente.

4. En los grupos formados, se evalúan la consistencia del color dentro de una banda (dos segmentos consecutivos) eliminando aquellas que no la

cumplen.

5. Por último se evalúa el cross-ratio. Los grupos restantes que contengan más de 4 segmentos de línea, se consideran pasos de peatones.

En el trabajo anterior, queda probada la utilidad de los smartphones y de los sensores que estos llevan incorporados en el reconocimiento de pasos de peatones, a la hora de ayudar a personas con algún tipo de discapacidad visual a cruzarlos. Sin embargo, tanto en este trabajo como en Ivanchenko et al. (2008), se considera que la persona debe capturar el paso de peatones con la cámara del dispositivo en una única imagen para que el algoritmo produzca una detección. Esto puede resultar complicado para alguien con una discapacidad visual, que no tiene un conocimiento previo de dónde está situado el paso de peatones tal y como se apunta en Murali y Coughlan (2013).

En este trabajo, y siguiendo la filosofía de utilización de los smartphones y de los sensores que éste lleva integrados, se aborda una solución a este problema consistente en integrar la información obtenida por la cámara del teléfono móvil, con la de posicionamiento adquirida con su antena GPS y de orientación proporcionada por un magnetómetro.

El GPS podría parecer *a priori* una buena solución a la hora de orientar a una persona con una discapacidad visual al cruzar un paso de peatones. Sin embargo, como ya se apuntó en el capítulo 1 de este proyecto, los GPS arrojan importantes errores de medida a la hora de posicionar un punto en entornos urbanos. Debido a este motivo, la información del GPS exclusivamente no es lo suficientemente precisa como para ser utilizada a la hora de orientar a una persona con algún tipo de discapacidad visual al cruzar un paso de peatones. Teniendo en cuenta esto, el trabajo desarrollado en Murali y Coughlan (2013) toma primero una captura panorámica con la cámara del smartphone, solucionando así el problema de desconocer previamente la situación del cruce, y a a partir de ella obtiene una imagen aérea correspondiente a la captura, utilizando un cambio de perspectiva. A continuación, binariza esta imagen para quedarse exclusivamente con las marcas viales que aparecen en la misma y la orienta con respecto al norte geográfico utilizando la información del magnetómetro. Esta imagen es utilizada después como una plantilla que es comparada con una imagen de satélite obtenida de la cartografía de GoogleMaps posicionando así de forma más precisa a la persona. Este sistema está en una fase temprana de su desarrollo pero propone una alternativa conceptualmente diferente y muy novedosa frente al resto de trabajos citados.

2.1.2. El reconocimiento del paso de peatones en la prevención de accidentes de tráfico

Como se vio en el capítulo 1, los accidentes de tráfico suponen un importante problema socio-económico, ante el cual las principales marcas relacionadas con el sector de la automoción han hecho importantes inversiones, con el fin de desarrollar sistemas de seguridad que asistan al conductor durante la maniobra del vehículo, de tal forma que ayuden a reducir los accidentes de tráfico producidos por factores humanos. Estos sistemas, denominados **Advanced Driver Assistance Systems (ADAS)**, utilizan distintas tecnologías de sensores para detectar el entorno del vehículo e intervenir, informar o dar soporte al conductor en caso de producirse situaciones peligrosas que puedan tener como consecuencia un accidente.

Según lo anterior, el desarrollo de un algoritmo de visión por computador para el reconocimiento de pasos de peatones, es susceptible de ser utilizado en el desarrollo de ADAS y ésta es la motivación principal del segundo tipo de trabajos que se pueden encontrar al respecto en la bibliografía.

En este contexto, en Lausser et al. (2008) se propone un método de detección basado en la aproximación de Viola y Jones (2001), la cual utiliza un algoritmo de aprendizaje Ada-Boost (Freund et al., 1999) para entrenar varios clasificadores débiles que son agrupados luego en una arquitectura en cascada, obteniendo así un clasificador más robusto. Es un método utilizado en patrones más complejos como caras humanas y, como se apunta en Ahmetovic (2013) no parece de tanto interés a la hora de reconocer un paso de peatones.

Por otro lado, son de especial interés para el desarrollo de este proyecto los trabajos expuestos en Sichelschmidt et al. (2010) y en Collado et al. (2005), pues en ambos se utiliza el análisis frecuencial de la imagen en la clasificación de un patrón visual, aspecto que será clave para el sistema de reconocimiento desarrollado en la presente investigación.

En el primero de ellos se desarrolla un algoritmo que obtiene una segmentación de la imagen, que es utilizada después para obtener varios clasificadores usados como entrada de una SVM. La principal novedad de este algoritmo se refiere a los clasificadores que utiliza:

- **Transformada de Fourier**. Se utiliza una transformada de Fourier para analizar el espectro de frecuencias de las filas de dos imágenes distintas. Por un lado, de la imagen obtenida al aplicar a la captura original una transformación que elimina de ella todos aquellos píxeles que no pertenecen a la calzada y, por otro, de la imagen binaria

producto de aplicar a la anterior un detector de bordes de Canny, una dilatación y una operación de rellenado de los contornos cerrados. La transformada de Fourier de las filas pertenecientes a un paso de peatones van a presentar un pico en ambas imágenes en un rango de frecuencias determinado.

- **Bipolaridad aumentada**. Ya se vio en Uddin y Shioyama (2005) el concepto de bipolaridad del histograma como herramienta para caracterizar un paso de peatones. Se vieron también los problemas que los métodos basados en esta idea presentaban ante oclusiones de la señal en la imagen. La bipolaridad aumentada trata de solucionar este error, tomando el histograma ecualizado de la región de interés, obtenida del proceso de segmentación, y agrupando sus píxeles en 4 bloques. Para llevar a cabo este proceso se propone utilizar el método de Otsu (Otsu, 1975). Las características aportadas por este clasificador serán los ratios entre el número de píxeles de los dos grupos más bajos y de los dos grupos intermedios, con respecto al total de píxeles de la imagen. Para un paso de peatones, el primer ratio debe tener valores medios-altos, mientras que para el segundo los valores debe ser bajos.
- **Inverse Perspective Mapping (IPM) Template Matching**. Basándose en las ideas de Mallot et al. (1991) y Muad et al. (2004), se hace una re-mapeado de las imágenes a las que se aplicó la transformada de Fourier, que permite obtener una perspectiva a vista de pájaro de las mismas. Sobre estas imágenes se aplican un conjunto de plantillas obteniéndose unos valores de correlación. Los valores de correlación más altos de cada imagen son extraídos como características.
- **Ratio de orientación de bordes**. Se estudia el gradiente en los bordes para determinar su ángulo con respecto a la horizontal. Se toman los módulos del gradiente para ciertos rangos de ángulo como características.

Por su parte, el método desarrollado en Collado et al. (2005) es el único de los trabajos utilizados como referencia para esta sección que no trata de la detección de pasos de peatones, sino que pretende clasificar distintos tipos de líneas de delimitación de carril (continuas, discontinuas y de aceleración) en función de la frecuencia característica de cada una de estas marcas viales. Pese a esto, las ideas expuestas en este trabajo han sido de gran utilidad para el presente proyecto por las ideas que aporta en la clasificación de patrones geométricos utilizando la transformada de Fourier.

También en el contexto de los ADAS, son de gran interés en el desarrollo del presente trabajo las ideas desarrollados en Sebsadji et al. (2010) y Foucher et al. (2011). Ambos trabajos pertenecen al mismo proyecto, sirviendo la investigación desarrollada en el primero de base a la del segundo.

En Sebsadji et al. (2010) los autores proponen cuatro métodos orientados a la extracción de los píxeles pertenecientes a cualquier marca vial presentes en una imagen (no exclusivamente a pasos de peatones), haciendo un análisis comparativo entre estos métodos. Las aproximaciones a este problema se basan en la idea de que cualquier marca vial tiene un color mucho más claro que los píxeles alrededor de la misma y aplica por, filas a nivel local, filtros unidimensionales de anchura variable basados en la media o la mediana. Los resultados obtenidos del filtrado son comparados luego con la imagen original, decidiendo así que un píxel pertenece a una de estas señales pintadas en la calzada si su nivel de gris es mucho más alto que el correspondiente de la imagen filtrada. Al considerar en la extracción de características una propiedad común a cualquier marca vial, estos algoritmos tienen la ventaja de ser fácilmente adaptables al desarrollo de aplicaciones de reconocimiento de cualquier señal de tráfico pintada en la calzada, incluso a algoritmos multi-detección con una única etapa de extracción de características y clasificadores diferentes en función de la marca vial.

En este trabajo, también proponen una optimización para estos métodos utilizando un sistema estereoscópico de cámaras. La investigación desarrollada en Sebsadji et al. (2010) es clave para el desarrollo del presente proyecto, siendo una adaptación del algoritmo que se concluye más óptimo de los cuatro evaluados, el método que es utilizado aquí como una de las piezas fundamentales de la etapa de extracción de características del código implementado. Por este motivo, un análisis más detallado del trabajo desarrollado en Sebsadji et al. (2010) pueden encontrarse en 4.1.1.

Por su parte, en Foucher et al. (2011) se aplica el mejor algoritmo obtenido del trabajo anterior como primera parte de un proceso cuyo objetivo es el reconocimiento en la imagen de pasos de peatones y flechas pintadas en la calzada. Para ello, obtiene la IPM de la imagen y en ella agrupa píxeles conectados en función de características geométrica: las dimensiones del eje mayor y el eje menor de componentes conectados debe pertenecer a un cierto rango y el cociente entre el producto de las longitudes medias de los ejes y el producto de sus longitudes máximas debe estar en torno a un valor concreto (criterio de rectangularidad). El reconocimiento del paso de peatones es llevado a cabo teniendo en cuenta la posición de los componentes conectados en relación unos con otros: dos componentes conectados, al menos, pertenecen al mismo paso de peatones si tienen aproximadamente la misma orientación, sus centroides están localizados en el mismo eje menor y la distancia entre ellos es menor de un umbral.

Por último, en esta relación de trabajos en el contexto de los ADAS es interesante incluir el expuesto en Choi et al. (2013), el cual propone utili-

zar un algoritmo que aúne la detección de pasos de peatones y semáforos de manera conjunta, para obtener así un sistema muy robusto ante falsos positivos, partiendo para ello de la idea de que estos dos elementos aparecen normalmente próximos. Además introducen algunas novedades con respecto a otro trabajos en el reconocimiento del paso de peatones, desarrollando a tal fin un algoritmo que consta de las siguientes operaciones:

1. Transformar la imagen al espacio de color HSV, quedándose exclusivamente con la capa V. Según los autores, este espacio de color es más robusto a los cambios de iluminación que el tradicional RGB.

2. Utilizando un filtro basado en la media, similar al propuesto en Sebsadji et al. (2010), obtiene en esta ocasión dos imágenes binarias: una que contiene todos aquellos píxeles en que se cumple que $I_v - I_{filtrada} > umbral$ (potenciales bandas claras del paso de peatones) y otra en que $I_v - I_{filtrada} < -umbral$ (potenciales bandas oscuras del cruce).

3. Realizando un OR lógico de las dos imágenes anteriores se obtiene un rectángulo en caso de que en la imagen hubiese un paso de peatones. Este patrón es fácilmente localizable en la imagen utilizando un filtro de dos dimensiones basado en la media.

2.2. Utilización de información visual como apoyo a los sistemas de posicionamiento

Como se expuso en el capítulo 1, los sistemas de posicionamiento tradicionales, tales como el GPS, tienen un error de localización elevado en ciertas circunstancias, como son los entornos urbanos. Como también se dijo en dicho capítulo, distintas aproximaciones orientadas a obtener un dato más preciso relativo a la posición de un vehículo han sido propuestas en los últimos años, destacándose aquellas relacionadas con el uso de información visual obtenida por cámaras instaladas en el mismo. Son los sistemas vSLAM y de odometría visual, que se trataron en la primera parte de este proyecto.

La bibliografía referida a este tipo de sistemas es ya hoy en día bastante extensa. Algunos ejemplos de trabajos que utilizan vSLAM se pueden encontrar en Lemaire et al. (2007) y en Lategahn et al. (2011). Por su parte, algunos proyectos que tratan sobre la odometría visual son Nistér et al. (2004) y Guindel Gómez (2012).

Como también se dijo en la introducción de este proyecto, los dos sistemas anteriores tienen un problema importante: presentan derivas sustanciales en las medidas cuando funcionan a lo largo de distancias de apenas unos kilómetros. Debido a esto, durante los últimos años han surgido numerosos

algoritmos que tratan de superar los inconvenientes anteriores, combinando la información de imágenes aéreas obtenidas por satélite con la aportada por las cámaras instaladas en el vehículo. Son los llamados algoritmos de *map-matching* (Quddus et al., 2007).

El funcionamiento de este tipo de métodos es por tanto similar al expuesto en Murali y Coughlan (2013), que ya fue citado en la sección anterior, pero aplicado en este caso a la/s cámara/s instaladas en un vehículo: se obtiene una imagen de satélite, se extraen una serie de características de la misma y se utilizan éstas como una plantilla, que es comparada con las características extraídas a su vez de la captura tomada por la/s cámara/s instaladas en el coche. Gracias a esta comparación es posible obtener una estimación más exacta de la posición en la que se encuentra el vehículo.

Siguiendo esta idea Pink (2008) propone un método que consta de las siguientes operaciones:

1. Obtención de la imagen aérea de la zona y de la captada por una cámara estéreo instalada en el vehículo. Se da también un dato de GPS inicial.

2. Extracción de características. Las características utilizadas para ambas imágenes son los centroides de las marca viales que aparecen en la calzada. Para su extracción en la imagen aérea (realizada *offline*) se aplica un proceso robusto pero lento usando SVM o redes neuronales. En el caso de la captura de la cámara se usa un algoritmo mucho más eficiente, pero también propenso a más falsos positivos, basado en un detector de bordes de Canny.

3. Se transforman las características extraídas de la fotografía obtenida por la cámara, de coordenadas de la imagen a coordenadas del vehículo, utilizando para ello el v-disparity.

4. *Feature Matching*. Se utiliza la combinación de un algoritmo *iterative closest point* (Besl y McKay, 1992) con un *iterative recursive least squares*(Darche, 1989).

En Noda et al. (2011) se lleva a cabo un procedimiento similar, extrayendo en este caso las características de la imagen obtenida por la cámara del vehículo, una vez calculada la transformación IPM de ésta y mediante un detector de esquinas de Harris. Las características de la fotografía aérea, por su parte, se extraen de las imágenes mediante un descriptor de imágenes SURF (Bay et al., 2008).

Los sistemas anteriores arrojan buenos resultados, pero resultan lentos y muy alejados del tiempo real. El principal inconveniente en este aspecto es

el cálculo de las plantillas, a partir de las imágenes de satélite. Una aproximación interesante que soluciona este inconveniente, siguiendo la filosofía de los procesos de map-matching, es sustituir el uso de imágenes aéreas por datos de cartografía públicos disponibles, tales como GoogleMaps.

Un trabajo interesante en este sentido es el presentado en Floros et al. (2013), que combina la idea anterior con el uso de la odometría visual para proveer una localización precisa a lo largo del tiempo. La combinación de la estimación de la trayectoria seguida por el vehículo, obtenida por odometría visual, con la información de las rutas posibles, extraídas de los mapas, reduce en gran medida la deriva producida durante la estimación de movimiento. Se asume para ello que la odometría visual es suficientemente precisa en cortas distancias. Además en este caso, el sistema utiliza los mapas de código libre de OpenStreetMaps (OSM).

El funcionamiento del sistema expuesto en Floros et al. (2013) comienza con la obtención de una localización de GPS inicial. A partir de las coordenadas de esta posición, descarga de OSM el mapa del área circundante y extrae del mismo el dibujo de las calles, obteniendo así un conjunto de líneas que serán usadas como plantilla. A continuación, el sistema de odometría visual estima una trayectoria para el vehículo durante una mínima distancia. El dibujo de la trayectoria obtenido es utilizado entonces por un algoritmo de *chamfer matching* (Barrow et al., 1977) que compara dicha trayectoria con la plantilla, obteniendo una serie de hipótesis sobre la localización del vehículo. Estas hipótesis son utilizadas para inicializar un filtro de partículas o Método de Montecarlo, para estimar la posición del vehículo a lo largo del tiempo.

La ventajas de usar OSM en el map-matching usando el método anterior arrojan muy buenos resultados. El uso de OSM, ofrece además otras ventajas importantes frente a otras cartografías como es la disponibilidad de información semántica asociada con la cartografía. Esta información aporta datos más allá de los relativos al trazado de las calles, como puede ser la limitación de velocidad de éstas o la posición de un semáforo.

El algoritmo propuesto en el presente proyecto combina la idea desarrollada en Floros et al. (2013), con información semántica relativa a la posición de pasos de peatones que se puede obtener de OSM. Se pretende desarrollar de este modo un algoritmo más simple al descrito en Floros et al. (2013), que utilice únicamente la información de posición de estas marcas viales, de tal manera que los pasos de peatones sean tomadas como puntos de referencia a lo largo del recorrido de un vehículo que utiliza información GPS apoyada por un sistema de odometría visual para su localización. De esta manera al igual que en Floros et al. (2013), se espera obtener una buena corrección de

la deriva del sistema de odometría visual con una complejidad del algoritmo mucho menor.

Capítulo 3

Conceptos Teóricos y Herramientas

Antes de profundizar en el desarrollo del proyecto, es necesario aportar unas nociones sobre algunos conceptos teóricos y herramientas en que se fundamenta la resolución del problema planteado. A continuación se explican estos conceptos.

3.1. Visión por computador. Nociones y estructura básica.

La visión es el sentido más importante y complejo del que disponen los seres humanos. La información que llega al cerebro proveniente de la vista supone el 75 % de toda la que este órgano procesa, permitiéndo a las personas percibir el entorno de una manera prácticamente automática. Conocer la posición de los objetos para así interactuar con ellos o evitar obstáculos y peligros, son sólo algunas de las acciones que las personas realizan de forma sencilla gracias a este sentido (de la Escalera y Armingol, 2010).

Ya en la década de 1950, con los primeros pasos de la informática, se desarrolló la idea de la importancia de dotar a las máquinas con la capacidad de ver. Surgió entonces lo que se denominó ***visión artificial***, que pretendía imitar el sistema de visión de los seres humanos. Sin embargo, el exceso de confianza en las computadoras de la época y la falta de conocimiento sobre el modo en que los seres humanos ven, provocó que en la década de 1960 los avances producidos fueran aún escasos, abandonándose prácticamente la investigación en este ámbito.

No fue hasta la década de 1980 cuando, gracias al desarrollo de las computadoras y de la electrónica, así como a la disminución del coste de estos ele-

mentos, se retomó el problema bajo un enfoque más realista, basado en el análisis de imágenes en lugar de en tratar de emular el sistema de visión humana. Pasó entonces a denominarse ***visión por computador***. Actualmente, la visión por computador es un campo de investigación con una alta actividad y multitud de aplicaciones: control de calidad, biometría, seguridad, biomedicina, etc.

Un sistema de visión por computador está compuesto por varios elementos físicos: iluminación, cámaras, tarjetas de procesamiento de imágenes,etc., sin embargo, el corazón del sistema de visión es el algoritmo, o conjunto de algoritmos, encargados de llevar a cabo sobre las imágenes de entrada las operaciones necesarias para ser capaces de extraer de ellas una información útil. La creación de un algoritmo de visión puede abordarse desde dos enfoques: el enfoque denominado ***bottom-up***, o bien, desde el llamado ***top-down***. En el primero, partiendo de la imagen, se llega al reconocimiento del objeto utilizando diferentes técnicas procedentes del procesamiento de imágenes, las cuales obtienen en pasos sucesivos información sobre la captura. Son métodos basados fundamentalmente en el histograma y, por tanto, muy dependientes de la iluminación. El enfoque top-down, por su parte, busca en la imagen estructuras conocidas, es decir, partiendo del conocimiento del objeto busca éste en la imagen. Es el ejemplo de técnicas como las plantillas deformables (Yuille, 1991). Estos métodos, además de su coste computacional, presenta el problemas de la variabilidad de formas, colores, etc. que puede tener un objeto de la misma clase.

El algoritmo que se presenta en este proyecto, está basado en el enfoque bottom-up. Dentro de este método de abordar el reconocimiento de un objeto en la imagen, los algoritmos desarrollados son muy distintos los unos de los otros y no siempre repiten la misma secuencia lógica de operaciones. Si existen sin embargo una serie de etapas que se pueden identificar en gran parte de ellos. En este sentido, a continuación se enumeran algunas de las etapas básica de un algoritmo de visión por computador:

1. **Adquisición de la imagen.** Mediante el uso de cámaras se toman imágenes del entorno con el objetivo de extraer de ellas una información útil. En este proceso, además del tipo de cámara utilizado, son importantes elementos como el tipo de sensor de que dispone, su óptica (ver apartado 3.3), la iluminación bajo la cual se toman las imágenes, etc. Para las imágenes utilizadas en este proyecto se utilizó un sistema de cámaras estereoscópicas, Bumblebee, de la compañía Pointgrey (Pointgrey, 2014).

2. **Pre-procesamiento.** El objetivo de estas operaciones es el de mejorar características de la imagen. La meta final de alguna de estas opera-

ciones puede ser: mejorar el contraste (ver apartado 3.4), realzar los bordes que delimitan las figuras de la imagen o eliminar ruido. En este último caso, se entiende por ruido, píxeles de la imagen que toman valores muy diferentes a los de sus vecinos. El ruido se produce por una superposición de una señal no útil y de naturaleza aleatoria sobre la imagen. La generación de esta señal de ruido es debida al sensor de la cámara o al medio de transmisión de la señal.
Relacionada con esta etapa, en este trabajo se realiza una mejora de contraste (ver secciones 3.4 y 4.1.3).

3. **Extracción de características.** En esta etapa, se buscan en la imagen características capaces de definir el objeto. Las más habituales son: el color o el nivel de gris, los bordes que definen los objetos en la imagen, las texturas y el movimiento. En este sentido, en el presente proyecto se realizan operaciones de extracción de características orientadas a la obtención de las marcas viales de la imagen, basándose para ello en la diferencia de nivel de gris entre las marcas viales y la calzada (ver sección 4.1.1). También se obtienen las características que definen el patrón geométrico representado por los pasos de peatones (textura artificial), basándose para ello en la transformada de Fourier (ver secciones 3.9 y 4.1.5).

4. **Segmentación.** Es la etapa consistente en separar los objetos que aparecen en la imagen del fondo y entre sí, teniendo en cuenta para realizar esta operación las características extraídas de la imagen. Existen dos enfoques para las operaciones realizadas en esta etapa:

   - Métodos basados en buscar zonas en que se produce un cambio. Se utilizan para ello los bordes como características fundamental que separa unos objetos de otros y estos del fondo.
   - Métodos basados en buscar zonas de la imagen con características semejantes, ya sea de movimiento, color, nivel de gris etc.

   En el presente trabajo se utiliza una umbralización (ver apartado 3.6) para, utilizando las características basadas en la diferencia de niveles de gris obtenidas en la etapa de extracción de características, separar las marcas viales que aparecen en la imagen, del fondo (ver 4.1.1). Asimismo, se utiliza información relativa a la profundidad de los objetos en la imagen para afinar el resultado de la umbralización, eliminando todos aquellos píxeles obtenidos que además no pertenecen a la calzada (ver sección 3.7 y 4.1.2). No se realiza por tanto una segmentación como tal, puesto que se separan las marcas viales del fondo, pero no éstas entre sí. Aún así, en lo que respecta a este trabajo, el conjuntos de las dos operaciones son asimiladas a una etapa de segmentación.

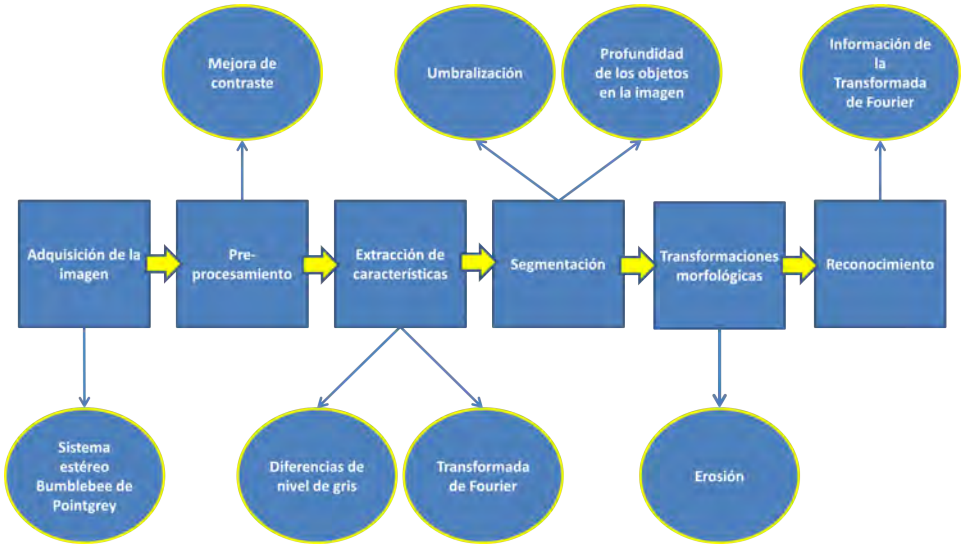


Figura 3.1: Esquema de las operaciones de un algoritmo de visión

5. **Transformaciones morfológicas.** Son operaciones aplicadas generalmente sobre imágenes binarias, que modifican la estructura de los objetos identificados durante el proceso de segmentación. Estas operaciones pueden usarse con dos objetivos: extraer características (descriptores) de los objetos identificados en la segmentación, tales como perímetro, área, etc., o corregir errores producidos durante la misma (píxeles mal clasificados). En este sentido en el presente trabajo se usará una erosión (ver apartado 3.8).

6. **Reconocimiento.** A partir de los descriptores obtenidos de las transformaciones morfológicas y/o de los las características obtenidas durante la etapa de extracción de características, se clasifican los objetos identificados en la imagen. En el presente proyecto se utilizarán la información obtenida de la transforma de Fourier de la imagen para saber si en ella existe o no un paso de peatones (ver secciones 3.9 y 4.1.5).

En la figura 3.1 se esquematizan estos pasos tomando como ejemplo el algoritmo objeto de este proyecto.

3.2. Imágenes Digitales. Conceptos.

Una imagen digital es la representación de una imagen bidimensional en una computadora u otro sistema electrónico. En este tipo de sistemas las imágenes están representadas generalmente por matrices de dimensión $[u \times v]$; siendo u el número de columnas de la matriz y v el número de filas

de la misma. Cada una de las entradas de estas matrices se denomina *píxel*. A igual superficie de representación, cuantos más píxeles compongan la matriz, mayor definición tendrán los objetos representados en la imagen. Este concepto es lo que se denomina **resolución de una imagen**. Las imágenes usadas en este proyecto en la detección del paso de peatones tienen una resolución de 640×480 píxeles.

Además, en la memoria de una computadora, cada uno de estos píxeles está representado por un número k de bits. El parámetro k se denomina **profundidad del color**, pues un número k de bits permite representar 2^k combinaciones, y por tanto, 2^k colores. Los valores más habituales que suele tomar k son 1, 2, 4, 8, 16, 24 y 32.

Otro concepto importante para definir una imagen digital es conocer el **modo de color** (Formación en red: Ministerio de educación) en que está representada (ver figura 3.2). Los más habituales son:

- **Escala de grises**. Una imagen en escala de grises sólo puede representar diferentes **tonos o niveles de gris**. En función de la profundidad del color la variedad de estos níveles es más o menos amplia. También se denomina a estos níveles de gris, niveles de intensidad, ya que su valor da una idea de la intensidad con que la cámara ha recibido la luz procedente de los objetos en la formación de la imagen. De esta forma, en una imagen con una profundidad del color de 8 bits, se podrán tener 256 níveles distintos, donde el valor 255 representará la intensidad máxima de luz que puede llegar a la cámara procedente del objeto, mientras que un valor de 0 supondrá una ausencia de luz. Debido a este hecho, un píxel con un valor de 255 estará representado en blanco y uno de valor 0 será negro (figura 3.2b).

- **Imagen binaria**. Es un caso particular del anterior con una profundidad de color de un bit, lo que quiere decir que los objetos en la imagen sólo pueden tomar dos valores 0 (negro) ó 1 (blanco). Un ejemplo de este tipo de imágenes se muestra en la figura 3.2a.

- **RGB**. Es el modo más habitual para imágenes en color y el más utilizado por cámaras digitales, televisores, etc. Está basado en la síntesis aditiva, con el que es posible representar cualquier color mediante la mezcla por adición de los tres colores luz primarios, cuyas siglas en inglés dan nombre al espacio RGB: Rojo (Red), Azul (Blue) y Verde (Green). En las imágenes digitales, cada píxel de una imagen en RGB está representado por tres canales (uno por cada uno de los colores primarios) que pueden tomar 256 valores distintos. Esto es equivalente a tener tres matrices con una profundidad de color de 8 bits, cuya información se superpone para codificar el color. Según esto, un píxel

(a) Imagen binaria (b) Imagen en escala de grises (c) Imagen en RGB

Figura 3.2: Imagen en tres modos de color distintos

de valor (255,0,0) sería rojo, otro de (0,255,0) sería azul y uno con valor (0,0,255) sería verde. Asimismo, los píxeles los valores (x,x,x) dan lugar a los diferentes níveles de gris, representando x=0 el color negro y x=255 el blanco. Lo anterior también es equivalente a decir que la imagen tiene una profundidad del color de 24 bits, por lo que en RGB se pueden representar $2^{24} = 16777216$ colores diferentes (figura 3.2c).

En este trabajo sólo se utilizan imágenes binarias e imágenes en escala de grises con una profundidad de color de 8 bits en el proceso de detección del paso de peatones. El cambio del espacio de color RGB a niveles de gris es sin embargo sencillo, por lo que la adaptación del código para trabajar con imágenes en RGB sería trivial. Aunque no se explican aquí, existen otros espacios de color menos habituales como serían HSI (Hue-Saturation-Intensity) o CMYK (Cyan-Magent-Yellow-blacK), este último con 4 canales y una profundidad de color de 32 bits.

Todo lo descrito anteriormente, está referido a imágenes con formato de mapa de bits, que son todos aquellos formatos que representan la imagen con una matriz de píxeles. Son los que producen cámaras digitales, escáneres, etc. Existen, sin embargo, otro tipo de formatos para imágenes digitales denominados vectoriales, que representan la imagen con trazos geométricos controlados por operaciones matemáticas que realiza la computadora. Las líneas que componen la imagen están definidas por vectores, lo que da nombre a estos formatos. La principal ventaja de las imágenes vectoriales es que pueden reducirse o ampliarse sin ningún tipo de pérdida de calidad. Algunos de lo formatos vectoriales más conocidos son SVG, WMF, SWF, EPS o PDF.

Este tipo de formatos no son usados en este proyecto.

En cuanto a los formatos matriciales, se caracterizan por sus diferentes calidades y capacidades de compresión. Algunos de los más conocidos son JPG, PNG, GIF, BMP o TIFF. En el presente trabajo se usan imágenes en formato TIFF en la detección del paso de peatones, ya que, frente a otros más habituales como JPG, no presenta compresión y tiene una calidad muy alta, con lo que no presenta pérdidas de información.

3.3. Principios ópticos. Formación de la imagen.

En el presente trabajo, es importante poder determinar la distancia real a la que se encuentran los objetos que aparecen en las imágenes, a partir de las coordenadas (u,v) que tienen estos en las capturas. Para poder determinar la relación entre las distancias reales y las distancias proyectadas en la imagen, es necesario conocer primeros algunos conceptos sobre la formación de las imágenes en una cámara (de la Escalera y Armingol, 2010).

El principal elemento en la formación de una imagen es la luz. Cualquier sistema encargado de tomar imágenes, como por ejemplo una cámara digital, tiene un conjunto de elementos que se encarga de captar los rayos de luz procedentes de los objetos y concentrarlos sobre un elemento sensible, ya sea una película fotográfica, como ocurre en las cámaras analógicas, un sensor CCD o CMOS, como en las cámaras digitales, o la retina en muchos seres vivos. Este conjunto de elementos encargados de concentrar la luz es lo que se denomina la **óptica del sistema**.

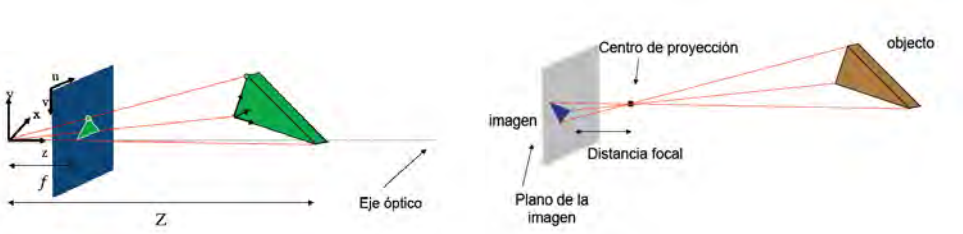
En las primeras cámaras que existieron, la óptica era muy sencilla, compuesta únicamente por una caja en la que se practicaba un pequeño agujero. Según la modelización de este tipo de sistema, cada punto perteneciente a un objeto emite un único rayo de luz que atraviesa el agujero, formándose la imagen de ese punto sobre un plano (plano de la imagen) perpendicular al eje del agujero y colocado a una distancia fija para la cual los objetos quedan enfocados sobre ese plano. Esta distancia con respecto al agujero a la que los objetos sobre el plano de la imagen quedan enfocados, es lo que se denomina **distancia focal** (f). El modelo descrito anteriormente se denomina modelo ***pin-hole***. En la figura 3.3 se muestra un esquema que representa éste modelo. Por semejanza de triángulos se puede determinar la relación entre las coordenadas de un punto del mundo y su proyección en el plano de

la imagen. Estas relaciones se describen en las ecuaciones 3.1 y 3.2:

$$u = \frac{f}{Z}X \tag{3.1}$$

$$v = \frac{f}{Z}Y \tag{3.2}$$

Donde X-Y-Z son las coordenadas de un punto del mundo (P) y u-v las coordenadas (columna-fila) de la proyección de ese punto en la imagen.



(a) (b)

Figura 3.3: Esquematizaciones del modelo pin-hole

Las cámaras actuales, sin embargo, no siguen este modelo tan sencillo, sino que su óptica está compuesta por un conjunto de lentes convergentes que se encargan de concentrar los rayos de luz procedentes de los objetos. Como una simplificación, estas lentes, se pueden modelizar como una lente ideal de espesor nulo, cuya dimensión principal se denomina eje de la lente. Se define además el eje óptico como la recta perpendicular al eje de la lente que la atraviesa por su centro. Esto es lo que se denomina **modelo de lente fina**, representado en la figura 3.4, según el cual, aquellos rayos de luz que atraviesan la lente paralelamente al eje óptico convergen en el mismo a la distancia focal. Aquel punto perteneciente al eje óptico en el que convergen todos los rayos de luz que atraviesan la lente paralelos al mismo, se denomina foco de la lente (F). Según este modelo, la relación entre la distancia de un punto perteneciente a un objeto con respecto al eje de la lente y la distancia a la que este punto se proyecta sobre el plano de la imagen se puede expresar como (ecuación 3.3):

$$\begin{aligned} \frac{1}{f} &= \frac{1}{Z} + \frac{1}{z} \\ Z &= \text{profundidad real del punto} \\ z &= \text{distancia a la que se proyecta el punto en la imagen} \end{aligned} \tag{3.3}$$

Para distancias suficientemente grandes el modelo de lente fina, es equivalente al modelo pin-hole.

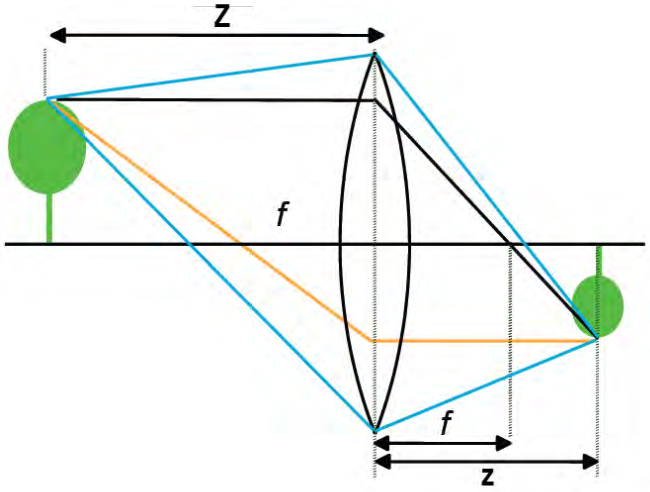


Figura 3.4: Esquema del modelo de lente fina

Según las ecuaciones del modelo pin-hole, es posible por tanto averiguar las coordenadas que tendrá la proyección de un punto P del mundo real sobre el plano de la imagen, conocidas sus coordenadas XYZ y la distancia focal de la cámara, pero no al contrario. Este inconveniente se puede solucionar si se conocen además algunos parámetros externos a la cámara, obtenidos durante un proceso de calibración: orientación y distancia de la cámara con respecto al plano que representa la superficie del objeto (coordenada Z). Si se tienen estos parámetros en cuenta, es posible calcular las coordenadas XY de un punto P del mundo, conocidas las coordenadas (u, v) de su proyección, utilizando las expresiones 3.4 y 3.5:

$$Y = Z\frac{f + v\,tan(\theta)}{f\,tan(\theta) - v} \tag{3.4}$$

$$X = \frac{Zu}{f, \text{sen}(\theta) - v\,cos(\theta)} \tag{3.5}$$

Donde θ es el ángulo definido por el eje Y y el eje óptico.

Según las ecuaciones proporcionadas por los modelos anteriores, como consecuencia de la proyección de la información tridimensional del mundo sobre el plano bidimensional de la imagen, se hace imposible conocer la profundidad real a la que se encuentran los puntos del mundo que aparecen proyectados en la captura (coordenada Z). Para solucionar este inconvenien-

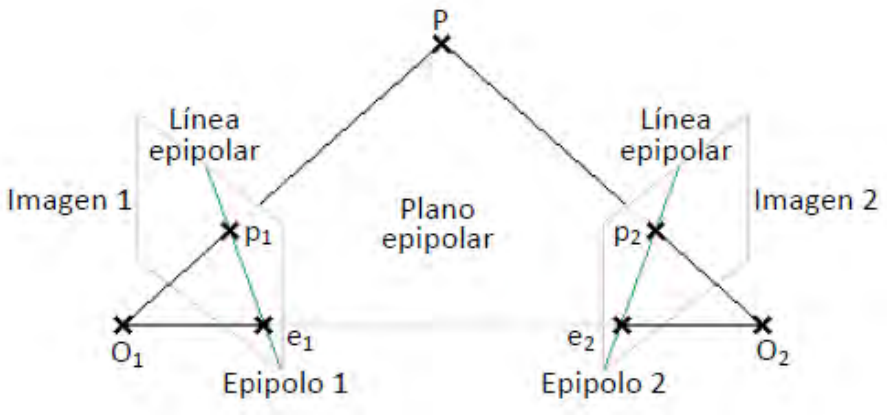


Figura 3.5: Representación de la geometría epipolar

te, se usan los sistemas estereoscópicos de cámaras (Guindel Gómez, 2012).

Un sistema estereoscópico está compuesto por dos cámaras supuestas de igual distancia focal, paralelas y separadas una distancia B, que se denomina **baseline**. En cada una de las imágenes un mismo punto del mundo (P) se ve proyectado en el punto de coordenadas (u_1, v_1) en la imagen tomada por una de las cámaras y en (u_2, v_2) en la otra. Si para el sistema de cámaras considerado se cumplen las hipótesis anteriores, el sistema estéreo se considera ideal, cumpliéndose en tal caso que $v1 = v2$. En cambio, las coordenadas horizontales, u_1 y u_2, serán en general diferentes. A la diferencia $u_1 - u_2$ es a lo que se denomina **disparidad** (d). A partir de B, f y d, sí se pueden deducir ya las coordenadas de un punto P, correspondientes a los puntos proyectados (u_1, v_1) y (u_2, v_2), usando para ello el modelo de **geometría epipolar**. En la figura 3.5 se representa este sistema, donde O_1 y O_2 representan los centros ópticos correspondientes a cada una de las cámaras, la unión del punto P con O_1 y O_2 representa el plano epipolar y la intersección de los planos de cada imagen con el plano epipolar representan las líneas epipolares. Los puntos de corte de estas líneas con la recta que une los centros ópticos de ambas cámaras dan como resultado los epipolos del sistema. Como se aprecia en la figura, para un sistema estéreo ideal, estás líneas son coincidentes y paralelas a las filas de la imagen. Este modelo, en combinación con las ecuaciones del modelo pin-hole (ecuaciones 3.1 y 3.2) permiten obtener las ecuaciones 3.6, 3.7 y 3.8:

$$X = Z\frac{u_1}{F} - \frac{B}{2} = Z\frac{u_1}{F} - \frac{B}{2} \tag{3.6}$$

$$Y = \frac{v_1 B}{u_1 - u_2} = \frac{v_1 B}{d} \tag{3.7}$$

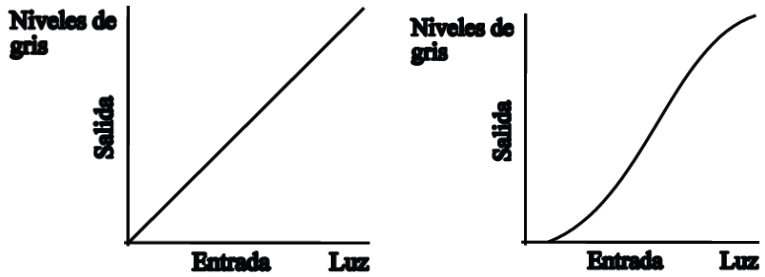
$$Z = \frac{fB}{u_1 - u_2} = \frac{fB}{d} \tag{3.8}$$

Las colecciones de imágenes utilizadas en el presente proyecto han sido obtenidas mediante un sistema estereoscópico de cámaras (Pointgrey, 2014), lo que nos permitirá conocer la distancia a la que los pasos de peatones detectados se encuentran de la cámara.

3.4. Mejora de contraste de una imagen

En una imagen ideal, la luz es uniforme y la relación entre los niveles de intensidad de la imagen y la luz de entrada a la óptica de la cámara (ganancia) es una recta (ver figura 3.6).

En general esta relación no se cumple, por lo que los niveles de intensidad que aparecen en la imagen son limitados. A partir de esta idea, se define un concepto muy importante para cualquier imagen, denominado contraste. Se define el contraste como la diferencia entre el mayor y el menor de los niveles de intensidad que existen en una imagen. Una mayor diferencia entre estos dos niveles de intensidad, dará lugar a una imagen con un mayor contraste, en la que los objetos que aparecen en la misma estarán mejor definidos y diferenciados los unos de los otros.

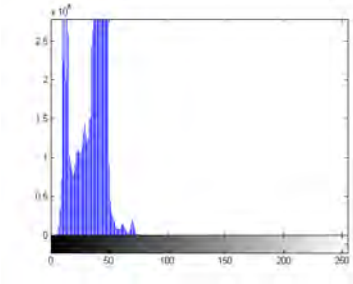


(a) Ganancia ideal (b) Ganancia real

Figura 3.6: Concepto de ganancia del sistema óptico

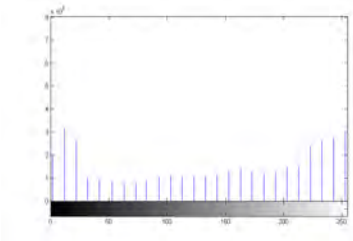
Dentro de las operaciones que se realizan en la etapa de pre-procesamiento de una imagen, una de las características más importantes que se pretende

(a) Ejemplo de imagen con contraste bajo



(b) Histograma de la imagen con contraste bajo

(c) Ejemplo de imagen con contraste alto



(d) Histograma de la imagen con contraste alto

Figura 3.7: Concepto de histograma de una imagen

mejorar es el contraste. Existen diferentes técnicas para modificar el contraste de una imagen, de las cuales, por su interés para este trabajo, se destacan dos: amplitud de la escala y ecualización del histograma. Para entender estos métodos, en primer lugar es necesario explicar qué es el histograma de una imagen.

En una imagen en escala de grises, el histograma es una representación de la misma en forma de gráfico, que acumula todos aquellos píxeles de la imagen con un mismo nivel de gris. Los valores del eje de abcisas de este gráfico, representan por tanto los posibles niveles de gris de la imagen (0 a 255), mientras que el eje de ordenadas determina el número de píxeles que existen para un determinado nivel de gris (ver figura 3.7). De esta manera, imágenes con un contraste bajo, tendrán un histograma con sus valores altamente concentrados en una zona del mismo, mientras que aquellas con un contraste alto tenderán a ocupar todo el rango de valores del histograma. Las dos técnicas de modificación del contraste que se explicarán a continuación, se basan en modificar el histograma de la imagen, con el fin de aumentar el rango de niveles de gris que aparecen en el mismo, pero con objetivos distintos. Estos métodos son:

- **Amplitud de la escala.** Busca obtener un histograma con una distribución de niveles de gris más amplia, pero sin modificar la forma

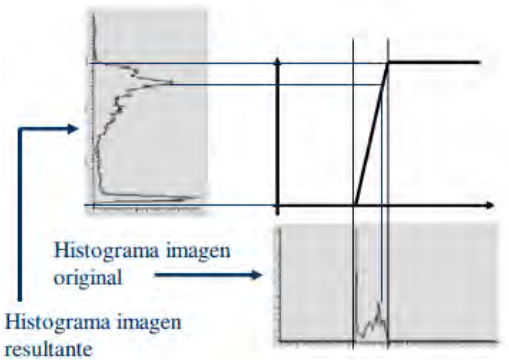


Figura 3.8: Proceso de amplitud de la escala aplicado sobre un histograma

del histograma original, es decir, manteniendo agrupados los píxeles como en el histograma de partida. Para ello aplica sobre la imagen una transformación del tipo (ecuación 3.9):

$$y = (Max - min)\frac{x-a}{b-a} + min; \tag{3.9}$$

y = Nivel de la imagen resultante
x = Nivel de gris de la imagen original
a = Nivel de gris mínimo de la imagen original
b = Nivel de gris máximo de la imagen original
Min = Nivel de gris mínimo de la imagen resultante
Max = Nivel de gris máximo de la imagen resultante

La ecuación 3.9 representa una transformación lineal como la mostrada en la figura 3.8.

También se puede escribir la expresión general de esta transformación como (ecuación 3.10):

$$y = \begin{cases} \alpha x & si \quad 0 \leq x \leq a \\ \beta(x-a) + v_a & si \quad a \leq x \leq b \\ \gamma(x-b) + v_b & si \quad b \leq x \leq L \end{cases} \tag{3.10}$$

y = Nivel de la imagen resultante
x = Nivel de gris de la imagen original
a, b, L = Intervalos de ganancia
α, β, γ = Ganancias de cada tramo
v_a, vb = Mínimo y máximo nivel de gris en la imagen resultante, para el tramo [a,b]

(a) Ejemplo de imagen con contraste bajo

(b) Histograma de la imagen con contraste bajo

(c) Imagen tras aplicar la amplitud de la escala

(d) Histograma de la imagen tras aplicar la amplitud de la escala

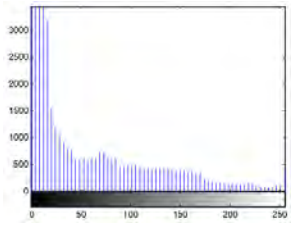
Figura 3.9: Ejemplo de aplicación del método de amplitud de la escala

De esta manera, se puede escoger el rango de valores del histograma al que aplicar la transformación y/o la importancia de cada tramo del histograma original en el histograma de la imagen de salida .
En definitiva, el método de amplitud de la escala hace algo similar a "estirar" el histograma. Esta filosofía es la aplicada en este trabajo a través de la función `imadjust` de MATLAB (ver secciones A.3.3.1 y 4.1.3).

Un ejemplo de aplicación del método de la amplitud de la escala sobre una imagen puede verse en la figura 3.9.

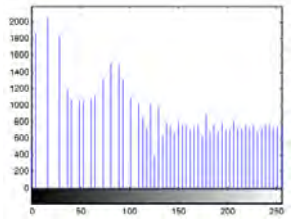
- **Ecualización del histograma.** El objetivo de esta operación es obtener una imagen nueva, a partir de la original, cuyo histograma tenga una distribución uniforme sobre toda la escala de grises, es decir, un histograma en el que todos los niveles de gris tengan la misma importancia. Por tanto, esta operación busca obtener un histograma de aspecto "plano", de tal forma que, si se calcula su histograma acumulado, éste sea equivalente al del caso ideal, es decir, una recta como la de

(a) Ejemplo de imagen con contraste bajo



(b) Histograma de la imagen con contraste bajo

(c) Imagen tras aplicar la ecualización del histograma



(d) Histograma de la imagen tras aplicar la ecualización del histograma

Figura 3.10: Ejemplo de aplicación dde la ecualización

la figura 3.6a. Con respecto a esto, el histograma acumulado es aquel gráfico obtenido a partir del histograma, en el que cada valor de gris lleva asociado el número de píxeles de la imagen que tienen ese nivel de gris o menor.
Para llevar a cabo la ecualización del histograma, se procede de la siguiente manera (Material web: González, Y.):

1. Calcular el histograma acumulado (ecuación 3.11):

$$\begin{aligned} H(i) &= \sum_{k=0}^{i} h(k) \qquad (3.11) \\ H(i) &= \text{Posición i del histograma acumulado} \\ h(k) &= \text{Posición k del histograma} \end{aligned}$$

2. Hallar los nuevos niveles de gris con la expresión 3.12:

$$\begin{aligned} i' &= \text{parte entera}(H(i)\frac{n}{N \times M} - 1) \qquad (3.12) \\ i' &= \text{Nuevo nivel de gris} \\ H(i) &= \text{Posición i del histograma acumulado} \\ n &= \text{Número de píxeles del histograma con el valor i} \\ N, M &= \text{Resolución vertical y horizontal de la imagen de partida} \end{aligned}$$

En la figura 3.10 se muestra el resultado de aplicar la ecualización del histograma sobre una imagen.

3.5. Convolución

En matemáticas la convolución es un operador que transforma dos funciones f y h en una tercera función g que, explicado de manera simplificada, representa la magnitud en la que se superponen f y una versión trasladada e invertida de g (Wikipedia, Convolución) (ver figura 3.11). La convolución de dos funciones se define como (ecuación 3.13):

$$g(x) = f(x) * h(x) = \int_{-\infty}^{\infty} f(\lambda)h(x-\lambda)d\lambda \tag{3.13}$$

Cuya expresión para funciones discretas viene dada por la ecuación 3.14:

$$g(x) = f(x) * h(x) = \sum_{i=-\infty}^{i=\infty} f(i)h(x-i) \tag{3.14}$$

Además, la convolución discreta bidimensional de dos funciones se expresa como (ecuación 3.15):

$$g(x,y) = h(x,y) * f(x,y) = \sum_{i=-\infty}^{i=\infty} \sum_{j=-\infty}^{j=\infty} f(i,j)h(x-i,y-j) \tag{3.15}$$

En el análisis de imágenes, la convolución es usada en el filtrado de las mismas, ya sea para resaltar detalles de la imagen (bordes), o eliminar de ella elementos no deseados (ruido). En este sentido, resulta de interés la expresión 3.15, al ser las imágenes definidas como un espacio discreto bidimensional. Para el caso de imágenes, f representa la matriz de píxeles de la imagen sobre la que se quiere aplicar la convolución, mientras que h es una segunda matriz denominada filtro o *kernel*. La operación de convolución en imágenes, por tanto, se simplifica a trasladar el kernel de convolución a través de la matriz de píxeles de la imagen, en un proceso iterativo compuesto por los siguientes pasos:

1. En cada iteración se sitúa el kernel sobre uno de los píxeles de la imagen, $f(u,v)$. Los valores u y v representan, respectivamente, el número de columna y fila sobre los que se apoya el filtro.

2. Se multiplica cada elemento del kernel situado encima de un píxel de la imagen, por el correspondiente elemento de la imagen. Se obtiene de esta manera un número de productos que depende de las dimensiones del filtro de convolución. Así, para un filtro de 3×3, se obtendrían 9 productos.

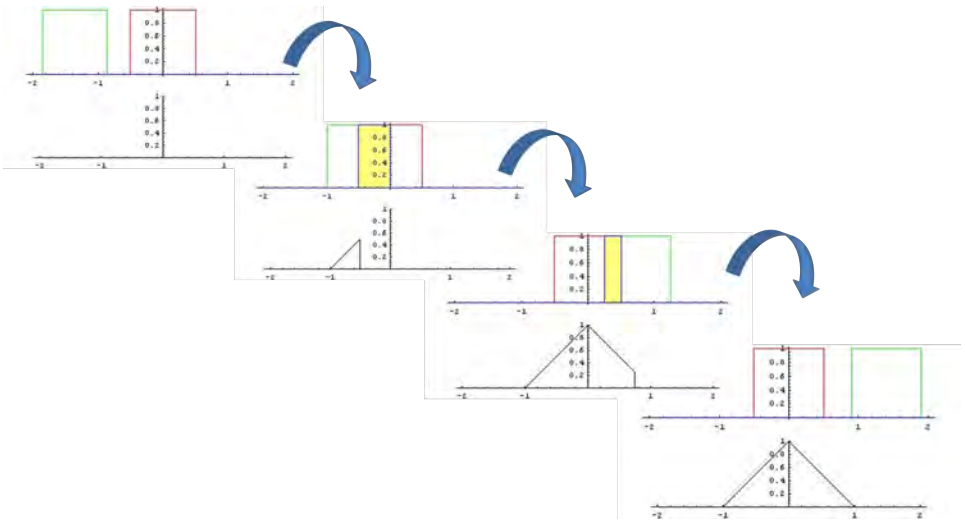


Figura 3.11: Ejemplo de convolución de dos funciones

3. Se suman los productos obtenidos y se asigna el valor calculado al píxel correspondiente de la imagen g resultante ($g(u, v)$).

4. Se desplaza el filtro a la posición $f(u + 1, v)$ y se repite el proceso. Si u es el número máximo de columna de la imagen, el kernel pasa a la siguiente fila, $v + 1$, repitiendo todo el proceso realizado sobre la fila anterior.

Esta es la lógica que subyace en la ecuación 3.15, que es de aplicación a los filtros lineales, como por ejemplo el representado en la ecuación 3.16, el cual calcula la media de los píxeles alrededor de uno dado:

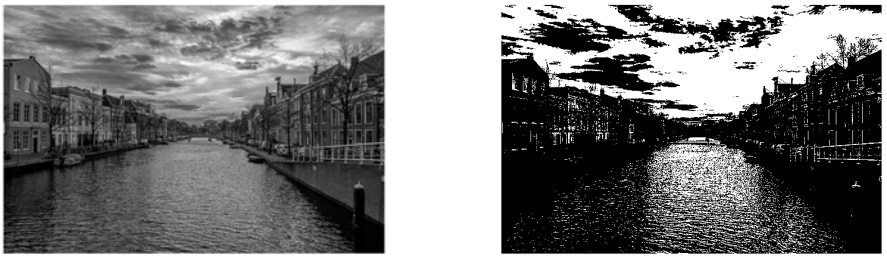
$$\begin{pmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{pmatrix} \tag{3.16}$$

Para filtros no lineales, como la mediana (ver sección 4.1.1), la forma en que se mueve el kernel a través de la imagen es equivalente, pero la operación difiere con respecto a la de los filtros lineales, ya que no se computan los productos de los píxeles del filtro y la imagen, obteniéndose después la suma, como se expuso anteriormente.

En la figura 4.4 del capítulo 4, se ejemplifica gráficamente la lógica de movimiento del filtro de convolución a través de la imagen, usando un kernel de dimensión [n×1].

3.6. Umbralización

La umbralización es una operación con la que se obtiene una imagen binaria a partir de otra en escala de grises, utilizando para ello el valor de una característica. Este valor se denomina umbral y aquellos píxeles, o grupos de píxeles, cuyo valor para esa característica supera el umbral son puestos a 1 (ó 0) en la imagen binaria resultante, mientras que aquellos que no lo superan son asignados a un valor 0 (ó 1) en la imagen de salida. En este trabajo la umbralización se utiliza en conjunción con una convolución con un filtro de mediana (ver sección 4.1.1). En la figura 3.13 se muestra un ejemplo del resultado de una umbralización sobre una imagen en escala de grises. Aquellos píxeles cuyo nível de gris es mayor de 127 se muestran en blanco (1 en la imagen binaria).



(a) Imagen original (b) Imagen umbralizada

Figura 3.12: Ejemplode umbralización de una imagen

3.7. Mapa de disparidad y u-v disparity

Una herramienta de gran utilidad a la hora de conocer la profundidad a la que se encuentran realmente los objetos representados en la imagen es el **mapa denso de disparidad**, o simplemente mapa de disparidad (Guindel Gómez, 2012), obtenido a partir de un sistema estéreo (ver apartado 3.3).

El mapa de disparidad está representado por una imagen en escala de grises, donde el nivel de intensidad de cada píxel de esta imagen representa la disparidad para ese píxel entre las dos imágenes del sistema estéreo, lo que teniendo en cuenta la ecuación 3.8, es equivalente a decir que representa la profundidad de ese píxel. Un ejemplo de este tipo de estructura, se muestra en la figura 3.13b.

El cálculo del mapa de disparidad es un tema recurrente en la bibliografía. La idea básica de estos algoritmos es tomar una de las dos imágenes obtenidas

(a) Imagen original (b) Mapa de disparidad de la imagen

Figura 3.13: Ejemplo de mapa de disparidad

por el sistema estéreo como referencia y recorrer la otra imagen en busca de píxeles correspondientes con los de ésta. En (Scharstein y Szeliski, 2002) se hace un análisis de la estructura general de los algoritmos destinados al cálculo del mapa de disparidad. Según este trabajo, los algoritmos de obtención del mismo cuentan en general con cuatro etapas:

- **Cálculo de la función de coste.** El coste es una medida de lo distintos que son dos píxeles correspondientes a un mismo punto en cada una de las dos imágenes del par. Hay numerosos métodos para evaluarlo; los más comunes hacen uso de la diferencia en el nivel de gris entre píxeles, ya sea absoluta, *AD* (*Absolute intensity Differences*, o cuadrática,*SSD* (*Sum of Squared Differences*).

- **Agregación del coste.** Habitualmente, la similitud de dos píxeles se evalúa teniendo en cuenta también aquellos otros pertenecientes a su entorno cercano; esto es, una ventana en torno al píxel analizado. También hay gran cantidad de posibilidades en este aspecto: ventanas rectangulares, adaptativas, con pesos adaptativos, métodos iterativos, etc.

- **Cálculo de disparidad.** Cálculo de la disparidad. En la etapa de cálculo de disparidades, hay varias posibilidades:

  - Métodos locales. Requieren concentrarse en las etapas de cálculo de costes y agregación, dado que calcular las disparidades finales es trivial; consiste en escoger en cada píxel la disparidad asociada al mínimo coste; es lo que se llama optimización *WTA* (*Winner-Take-All*,"el ganador se lleva todo").
  - Optimización global. Concentran el trabajo en el cálculo de la disparidad. Tratan de encontrar el valor de la misma que minimice

una energía global que considera toda la imagen.

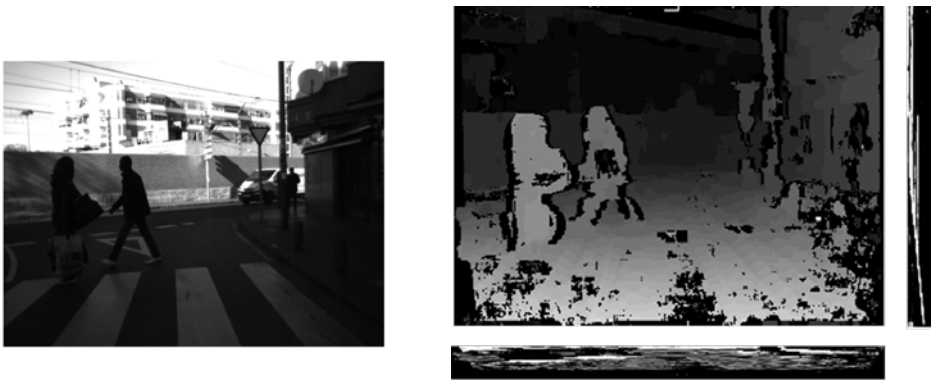
- Programación dinámica. Se basan en calcular un camino de coste mínimo.
- Algoritmos cooperativos. Tienen un comportamiento similar a los métodos de optimización global.

- **Refinado de disparidad.** Las etapas finales de estos algoritmos suelen consistir en refinar los resultados obtenidos, para corregir posibles discontinuidades en la disparidad. Una de las técnicas empleadas es el *cross-checking*, que compara el mapa de disparidad obtenido tomando como referencia la imagen izquierda con aquel que se construye tomando como referencia la imagen derecha. Los puntos con disparidad diferente se consideran ocluidos.

Otro concepto relevante para el desarrollo de este trabajo relacionado con el mapa de disparidad de una imagen, son los ***u-v disparity*** (Musleh et al., 2012b). El *u disparity* es la representación conjunta de los histogramas de cada una de las columnas del mapa de disparidad. La definición del *v disparity* es análoga la del *u disparity*, pero aplicada a las filas. Dicho de otra forma, el *u disparity* acumula todos aquellos píxeles con el mismo de valor de columna (u) y disparidad (d), mientras que el *v disparity* representa todos aquellos con el mismo valor de fila (v) y d. El aspecto final del *u disparity* es por tanto, una nueva imagen en escala de grises que tiene en cada una de sus columnas el valor de los acumulados de disparidad de la columna correspondiente del mapa de disparidad, representados estos acumulados con distintos niveles de gris en función de su valor. De nuevo, el caso del *v disparity* es equivalente, aplicado a la filas de la imagen. En la figura 3.14b se muestra un ejemplo de mapa de disparidad con sus respectivos *u-v disparity*.

Los *u-v disparity* son una herramienta muy útil por la información que se puede extraer de ellos. Así por ejemplo, los obstáculos que existan delante de la cámara aparecen en estas representaciones como líneas proporcionales a las dimensiones de los mismos y en su valor correspondiente de disparidad (Musleh et al., 2012a), teniendo en el *u disparity* el aspecto de líneas horizontales, mientras que en el *v disparity* aparecerán como rectas verticales. En el presente trabajo, esta idea se utilizará más adelante para filtrar la imagen, pues nos permitirá eliminar de ella todos aquellos píxeles que no pertenezcan a obstáculos (sección 4.1.2).

Otro ejemplo de información útil que se puede extraer de los *u-v disparity* es el perfil de la calzada, que estará representado en el *v disparity* por una línea oblicua. Esta información será utilizada también durante el desarrollo del proyecto (ver 4.1.8).



(a) Imagen original

(b) Mapa de disparidad de la imagen y *u disparity* (debajo) y *v disparity* (derecha) asociados

Figura 3.14: Ejemplo de *u-v disparity* de una imagen

3.8. Erosión

La erosión es una transformación morfológica destructiva, es decir, una transformación que cambia el aspecto de las formas que aparecen en una imagen binaria (elementos blancos sobre fondo negro o viceversa), mediante la eliminación de zonas pertenecientes a las mismas que no cumplen que un determinado subconjunto de la imagen está incluido en ellas. Dicho de otra forma, la idea básica que implementa la erosión es tomar una imagen binaria y comprobar en que zonas de la misma está incluida otra imagen binaria, que es un subconjunto de la primera, denominada elemento estructural. En un proceso similar al de la convolución, el píxel definido como centro del elemento estructural se posiciona sucesivamente en todos los píxeles de la imagen a erosionar, de tal forma que si el elemento estructural colocado en ese píxel coincide con la imagen, ese píxel se mantiene; eliminándose en caso contrario.

Según lo dicho anteriormente (Recurso web:Erosión), dada una imagen A y un elemento estructural B (ambas imágenes binarias), la erosión de la imagen A por B es el conjunto de todos los elementos x para los cuales B trasladado por x está incluido en A (ecuación 3.17):

$$A \ominus B = x/B_x \subset A \tag{3.17}$$

Donde el símbolo $\ominus$ representa la resta de Minkowski. Un ejemplo de lo anterior puede verse en la figura 3.15.
En el presente proyecto, la erosión es utilizada en la sección 4.1.4, utilizando para ello un elemento estructural cuadrado de 3 píxeles de lado (ecua-

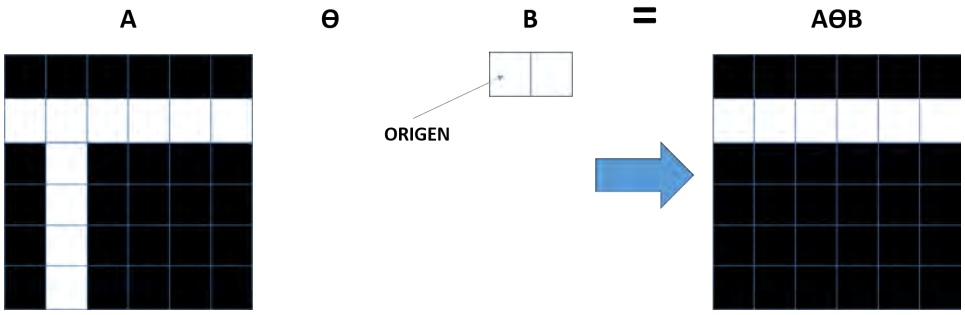


Figura 3.15: Ejemplo de operación de erosión sobre una imagen binaria

ción 3.18):

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{3.18}$$

3.9. Transformada de Fourier

Para entender la transformada de Fourier, resulta de gran ayuda comprender primero el concepto de las series de Fourier, cuya idea básica es que cualquier función periódica ($f(x)$) de periodo T puede ser expresada como la suma de infinitos senos y cosenos, según la expresión (ecuación 3.19):

$$f(x) = \sum_{0}^{\infty} \left(a_n cos(\frac{2n\pi}{T}x) + b_n sen(\frac{2n\pi}{T}x) \right) \tag{3.19}$$

Los coeficientes a_n y b_n, que dan una idea de la importancia de la contribución de cada $sen(\frac{2n\pi}{T}x)/cos(\frac{2n\pi}{T}x)$ a la función $f(x)$, se pueden expresar como (ecuación 3.20):

$$\begin{aligned} a_n &= \frac{2}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(x) cos(nx) dx \\ b_n &= \frac{2}{T} \int -\frac{T}{2}^{\frac{T}{2}} f(x) sen(nx) dx \end{aligned} \tag{3.20}$$

Estas contribuciones de cada uno de los infinitos coeficientes de la serie a la función $f(x)$ se pueden ver de una manera más clara si sobre la ecuación 3.19 se aplica la transformación sugerida en la figura 3.16. Haciendo

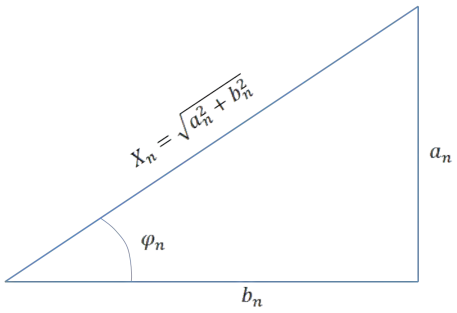


Figura 3.16: Lógica del cambio de senos y cosenos a senos con distinta fase

esto se obtiene (ecuación 3.21):

$$\begin{aligned} f(x) &= \sum_{n=0}^{\infty} \mathcal{X}_n \left(\frac{a_n}{\mathcal{X}_n} cos(\frac{2n\pi}{T}x) + \frac{b_n}{\mathcal{X}_n} sen(\frac{2n\pi}{T}x) \right) \xrightarrow{\frac{2n\pi}{T}=\omega_n} \\ f(x) &= \sum_{n=0}^{\infty} \mathcal{X}_n \left(sen(\varphi_n)cos(\omega_n x) + cos(\varphi_n)sen(\omega_n x) \right) \xrightarrow{sen(x\pm y)=sen(x)cos(y)\pm cos(x)sen(y)} \\ f(x) &= \sum_{n=0}^{\infty} \mathcal{X}_n sen(\omega_n x + \varphi_n) \end{aligned} \tag{3.21}$$

Donde:

$$\begin{aligned} \mathcal{X}_n &= \sqrt{a^2 + b2} \\ \varphi_n &= atan(\frac{a_n}{b_n}) \\ \omega_n &= \frac{2n\pi}{T} \end{aligned} \tag{3.22}$$

Por lo tanto, $f(x)$ se puede descomponer en la suma de infinitas senoides de amplitud $\mathcal{X}_n$, desfase φ_n y frecuencia angular ω_n. En la imagen de la figura 3.17 se puede observar este hecho. En ella, el gráfico que aparece más a la derecha representa la amplitud de algunos de los coeficientes de la serie de Fourier de $f(x)$ (primera imagen) con respecto a su frecuencia. A cada una de estas señales periódicas simples que componen $f(x)$ se les denomina también armónicos de $f(x)$.

Para ver el siguiente concepto importante sobre las series de Fourier es necesario transformar de nuevo la ecuación 3.19. Se sabe que (ecuación 3.23):

$$\begin{aligned} cos(x) &= \frac{\exp^{jx} + \exp^{-jx}}{2} \\ sen(x) &= \frac{\exp^{jx} - \exp^{-jx}}{2j} \end{aligned} \tag{3.23}$$

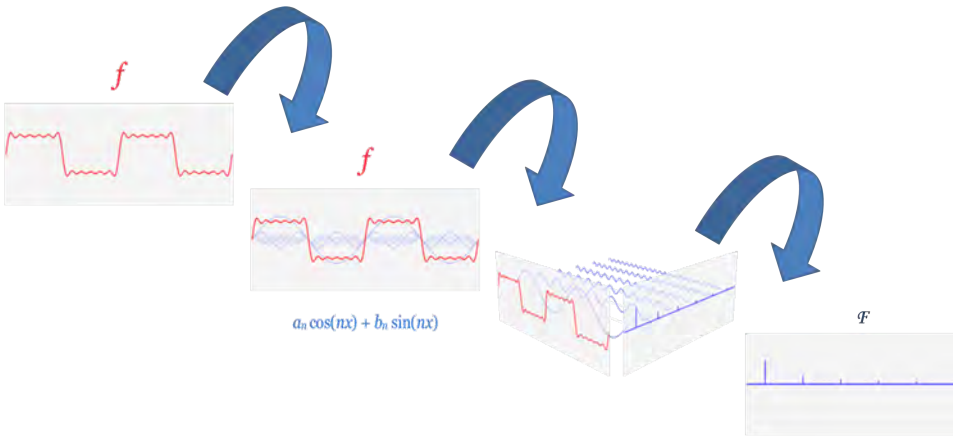


Figura 3.17: Descomposición de una función f en sus armónicos

Donde j representa $\sqrt{-1}$, es decir ,la unidad imaginaria. Sustituyendo esto en la ecuación 3.19 se obtiene la siguiente expresión (ecuación 3.24):

$$\begin{aligned} f(x) &= \sum_{n=0}^{\infty} \frac{a_n}{2}(\exp^{j\omega_n x} + \exp^{-j\omega_n x}) + \frac{b_n}{2j}(\exp^{j\omega_n x} - \exp^{-j\omega_n x}) \Longrightarrow \\ f(x) &= \frac{1}{2}\sum_{n=0}^{\infty} \exp^{j\omega_n x}(a_n - jb_n) + \exp^{-j\omega_n x}(a_n + jb_n) \end{aligned} \tag{3.24}$$

De la expresión 3.24 se puede extraer que (ecuación 3.25):

$$\begin{aligned} c_n &= \frac{1}{2}(a_n - jb_n) \\ c_{-n} &= \frac{1}{2}(a_n + jb_n) \end{aligned} \tag{3.25}$$

Sustituyendo esto en la ecuación 3.24 (ecuación 3.26):

$$\begin{aligned} f(x) &= \sum_{n=0}^{\infty} c_n \exp^{j\omega_n x} + \sum_{n=0}^{\infty} c_{-n} \exp^{-j\omega_n x} \Longrightarrow \\ f(x) &= \sum_{n=0}^{\infty} c_n \exp^{j\omega_n x} + \sum_{n=0}^{-\infty} c_n \exp^{-j\omega_n x} \Longrightarrow \\ f(x) &= \sum_{-\infty}^{\infty} c_n \exp^{j\omega_n x} \end{aligned} \tag{3.26}$$

Donde:

$$c_n = \frac{1}{2}(a_n - jb - n) = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(x) \exp^{-j\omega_n x}\, dx \tag{3.27}$$

De la ecuación 3.26 se deduce que cualquier función periódica $f(x) \in \mathcal{R}$ se puede expresar como la suma de infinitos números complejos. **La descomposición de una función en series de Fourier representa por tanto un cambio de espacio, $\mathcal{R} \rightarrow \mathcal{C}$.**

En la práctica, esto supone una transformación del dominio del espacio/tiempo al dominio de la frecuencia, pues como se dedujo en la ecuación 3.21, una función periódica se puede descomponer en infinitas funciones senoidales, que son caracterizables por su amplitud y su frecuencia. Esto hace de la transformada de Fourier una herramienta extraordinariamente útil para multitud de aplicaciones, como por ejemplo para el filtrado de señales, al permitir estudiar lo que se denomina el espectro de frecuencias de una señal. En este ámbito las series de Fourier permiten conocer cuáles son los componentes frecuenciales de una onda y su importancia relativa, haciendo posible así eliminar de la misma componentes no deseadas o ruido.

Las **series de Fourier son por tanto una herramienta muy útil, pero limitada en su aplicación a funciones periódicas**. La transformada de Fourier es la herramienta matemática, equivalente a las series de Fourier, para su aplicación en funciones no periódicas o, dicho de otra forma, de periodo infinito. Esta operación de define como (ecuación 3.28):

$$\mathcal{F}(\omega) = \mathcal{F}(f(t)) = \int_{-\infty}^{\infty} f(t) \exp^{j\omega t} dt \tag{3.28}$$

A su vez se define la operación inversa a la anterior, denominada transformada inversa de Fourier, como (ecuación 3.29):

$$f(t) = \mathcal{F}^{-1}(\mathcal{F}(\omega)) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \mathcal{F}(\omega) \exp^{j\omega t} d\omega \tag{3.29}$$

En este caso, esta ecuación relaciona la operación de transformación de $\mathcal{C} \rightarrow \mathcal{R}$.

El concepto de la transformada de Fourier es de gran importancia dentro de este proyecto (ver 4.1.5), al ser la herramienta matemática que permite decidir si en las imágenes existe un paso de peatones. Esto se debe a que este tipo de marcas viales forman un patrón periódico, que tiene por tanto una frecuencia característica identificable en el espectro de frecuencias que se obtiene a partir de la transformada de Fourier de la imagen.

3.10. MATLAB

MATLAB (Wikipedia, Matlab) es una herramienta de software matemático, la cual ofrece un entorno interactivo y un lenguaje de programación

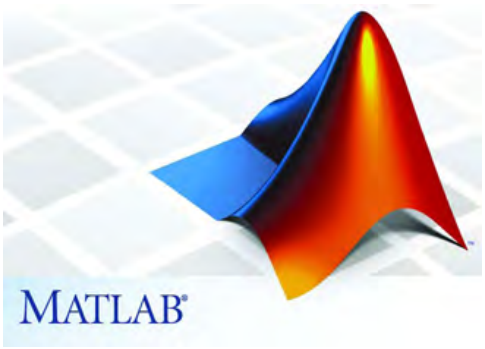


Figura 3.18: Logo de MATLAB

propio de alto nivel, similar a lenguajes como Fortran o C. Sin embargo, a diferencia de estos lenguajes, la unidad básica de programación en MATLAB son los arrays. MATLAB, o *MATtrix LABoratory*, recibe su nombre de esta característica que es también la mayor fortaleza de MATLAB: su potencia en el trabajo con formulaciones en matrices y vectores, permite resolver multitud de problemas numéricos de una manera mucho más eficiente y con un menor tiempo de desarrollo de código que cualquier otro lenguaje de programación tradicional.

Además de capacidades de cálculo numérico y programación de algoritmos, MATLAB permite otra muchas opciones a través de su entorno interactivo (Mathworks, a): visualización gráfica de datos, desarrollo de aplicaciones e interfaces de usuario, etc. Es por ello una herramienta muy versátil, utilizada por millones de ingenieros y científicos del ámbito académico y empresarial.

Las capacidades de MATLAB pueden ser además ampliadas a través de *toolboxes* (Mathworks, e), que son colecciones de funciones de MATLAB diseñadas para resolver problemas específicos: procesamiento de señales y comunicaciones, diseño y análisis de sistemas de control, biología computacional, etc.

Para el presente proyecto es de gran importancia la librería *Image Processing Toolbox* (Mathworks, d), que contiene una amplia colección de funciones para el procesamiento, análisis y visualización de imágenes. Algunas funciones de utilidad dentro de esta *toolbox* pueden encontrarse en el apéndice A.3.3.

3.11. OpenStreetMaps (OSM)

OpenStreetMap (OSM) es un proyecto colaborativo que pretende crear una cartografía mundial que pueda ser editada por cualquier individuo mediante datos de dispositivos GPS portátiles, fotografías aéreas o simplemente el conocimiento local. Una de las ventajas de OSM frente a otros sistemas de información geográfica, tales como GoogleMaps, es que los datos de OSM son publicados bajo una licencia de contenidos abiertos, con el fin de promover su libre uso y distribución, ya sea con fines comerciales o no.

La información de OSM se almacena en una base de datos que contiene las coordenadas geográficas (latitud/longitud) de puntos de la superficie terrestre, así como otras información asociada a estos puntos. Para una región del planeta dada, se puede descargar un archivo XML con los datos geográficos relativos a esa zona desde OpenStreetMaps. De acuerdo con las especificaciones de OSM citeposmfeatures, el modelo de datos de datos de este archivo está compuestos por tres estructuras básicas:

- **Nodos (*nodes*)**. Son elementos puntuales que se usan para representar puntos de interés, tales como gasolineras, señales de tráfico o restaurantes. Cada nodo se define con un identificador único, así como su longitud y latitud asociadas. Estas magnitudes se especifican en grados con siete cifras decimales, lo que en el peor de los casos da como resultado un error de posicionamiento de $\pm 1cm$.
- **Caminos (*ways*)**. Es una lista ordenada de nodos que, interconectados, modelan elementos lineales como carreteras, autopistas, ríos, etc. Análogamente a los nodos, se denominan con un identificador único. En los casos en que los identificadores del primer y último nodo sean iguales los caminos son interpretados como áreas. Estos elementos de área pueden usarse para modelar elementos como la huella de los edificios, los límites de un bosque, etc.
- **Relaciones (*relations*)**. Se usan para modelar asociaciones entre nodos y/o caminos. Un ejemplo de este tipo de elementos podría ser la ruta de un autobús, que asociaría a todos aquellos caminos a través de los que el autobús se desplaza en su recorrido y todos aquellos nodos que representan las paradas del mismo.

Todos los elementos anteriores pueden tener etiquetas asociadas (*tags*), las cuales aportan información semántica sobre dicho elemento. Las etiquetas están compuestas por dos partes: una clave (*key*), que representa el grupo de características al que pertenece esa etiqueta, y un valor (*value*), que detalla la característica específica que se referencia dentro del grupo clave. Un ejemplo de una etiqueta podría ser `tourism=museum` asociada a un nodo. En

Figura 3.19: Captura de pantalla de un mapa en OpenStreetMap

este caso, la clave sería `tourism` y el valor `museum`, lo que querría decir que el nodo representa la ubicación de un museo y que esto es una característica específica dentro del grupo de características existentes relacionadas con el turismo.

La principal ventaja de OSM frente a otros sistemas de información geográfica (*Geographic Information Systems, GIS*), como Google Maps, radica en que es una base de datos abierta y creada de forma colaborativa por cualquier usuario registrado, aprovechándose así de la experiencia y el conocimiento local por parte de los usuarios para componer la información existente en la base de datos.

Capítulo 4

Desarrollo del Proyecto

Como ya se ha comentado en otras partes de este documento, el resultado obtenido a partir de esta investigación es un algoritmo de visión por computador que detecta y posiciona respecto a un vehículo los pasos de peatones del entorno cercano al mismo, pretendiéndose utilizar esta información para corregir posibles errores de localización que puedan estar cometiendo sistemas como el GPS, o la Odometría Visual (GPS/VO), a la hora de determinar el posicionamiento del vehículo.

Desde este punto de vista, el trabajo que realizaría el sistema completo puede ser dividido en dos partes:

1. **Detección del paso de peatones en las imágenes y cálculo de su posición respecto al vehículo.**

2. **Análisis de la cartografía y corrección del error.** Comparación del dato de distancia del paso de peatones al vehículo, obtenido en la primera parte, con el que se obtiene de un sistema de posicionamiento cualquiera, y corrección de este último en caso necesario. En este trabajo, esta parte sólo es desarrollada a nivel teórico. Se explica, sin embargo, cómo se podría obtener de la cartografía la distancia al paso de peatones más próximo si se conociese la posición del vehículo determinada por GPS/VO.

Una vez sentadas en los capítulos anteriores las bases necesarias para entender el marco de referencia contextual y teórico del proyecto, en las secciones de este capítulo se describen con detalle cada una de sus partes y el funcionamiento de los distintos bloques que las componen.

La **implementación final del algoritmo es una colección de funciones escritas en MATLAB** cuyo código completo se puede consultar en el apéndice A de este documento. Como se dijo en la sección 1.2.2, el

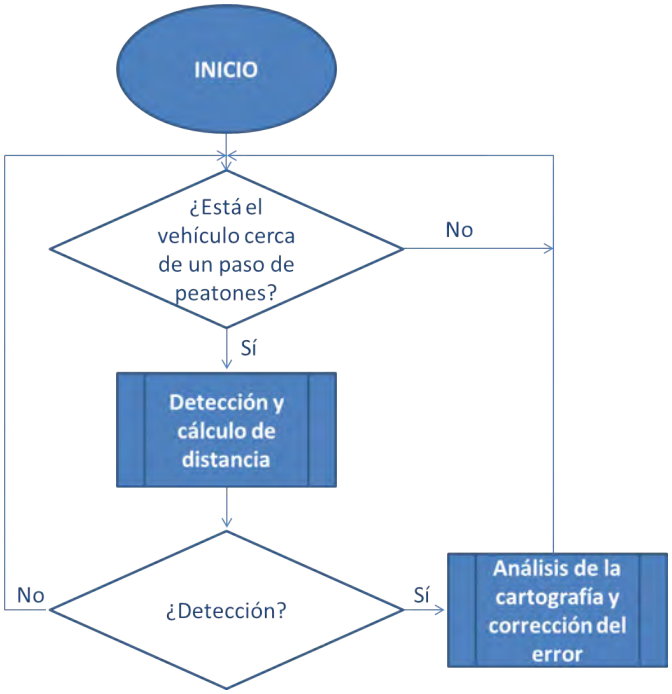


Figura 4.1: Lógica de funcionamiento general del algoritmo

objetivo de este proyecto no es por tanto la puesta a prueba de este sistema en un vehículo real, para lo cual habría que tener en cuenta consideraciones de tiempo real en la programación, sino realizar una evaluación de las posibilidades que tendría un sistema de este tipo.

Para la realización de las pruebas, aunque se explicará con más detalle en el capítulo 5, se han utilizado principalmente tres secuencias de **imágenes en escala de grises de 640x480 píxeles** obtenidas de un sistema de cámaras estéreo (Pointgrey, 2014) instalado en el Vehículo Inteligente basado en Información Visual (*Intelligent Vehicle based on Visual Information, IvvI*) perteneciente a la Universidad Carlos III de Madrid (UC3M) (ver apartado 1.1).

4.1. Detección y posicionamiento del paso de peatones

En esta sección se describen los procedimientos que son llevados a cabo en la primera de las partes en las se ha subdividido el funcionamiento del sistema planteado, para que éste sea capaz de detectar pasos de peatones en las imágenes captadas por las cámaras y, en caso de existir tal detección,

(a) Imagen original

(b) Imagen filtrada con la adaptación del algoritmo 43rd PLT

Figura 4.2: Aplicación del algoritmo de 43rd PLT.

medir la distancia que existe entre el vehículo y el paso de peatones.

Las operaciones que realiza el algoritmo con este fin, se pueden subdividir a su vez de la siguiente manera:

1. **Filtrado de la imagen original.** Engloba todas aquellas operaciones que se realizan sobre la imagen para extraer de ella todos los píxeles candidatos a ser una marca vial cualquiera (pasos de peatones, líneas de delimitación de carriles, señales horizontales de ceda el paso, etc.)

   a) **Mejora de contraste en caso necesario.** Si la parte inferior de la imagen es muy oscura se aplica un ajuste de contraste que resalta esta parte de la misma. Para ello se ha utilizado la función `imadjust` integrada en MATLAB (ver sección A.3.3.1).

   b) **Procesamiento de la imagen con el filtro 43rd Percentile Local Threshold (43rd PLT)** (Sebsadji et al., 2010). Filtra la imagen original para obtener todos aquellos píxeles que pertenecen potencialmente a cualquier marca vial. El funcionamiento de este filtro se basa en el hecho de que los píxeles pertenecientes a marcas viales tienen un nivel de gris mucho mayor que el de los píxeles circundantes. Su implementación, consiste en la combinación de una convolución con un filtro que actúa a nivel local por filas más una umbralización. Con esta operación se obtiene una imagen binaria que toma el nivel 1 (blanco) para aquellos píxeles considerados pertenecientes a una marca vial y 0 (negro) para el resto de píxeles (ver figura 4.2).

   c) **Uso del mapa denso de disparidad de la imagen**. Con esta etapa se busca eliminar todos aquellos píxeles de la imagen binaria

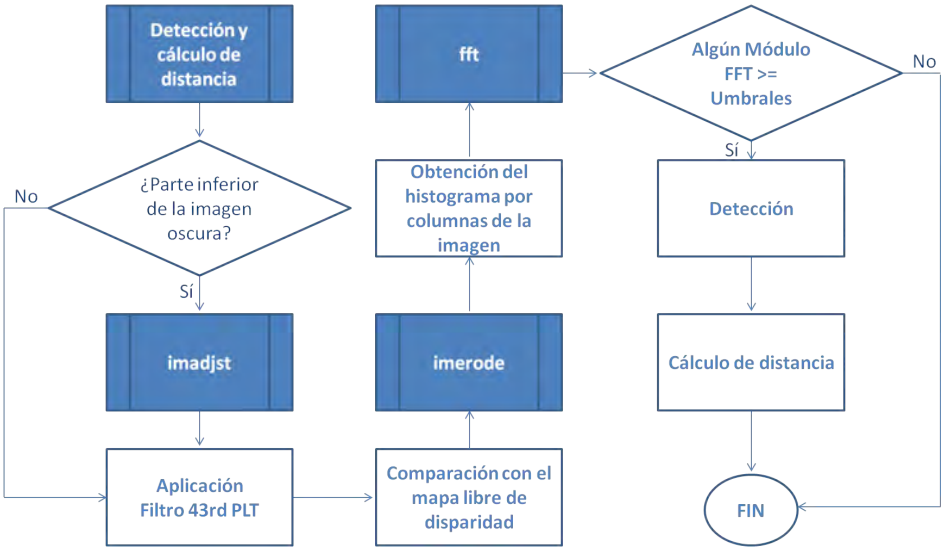


Figura 4.3: Lógica del módulo de detección y cálculo de distancias

obtenida en la operación anterior que no están sobre la calzada, utilizando para ello el mapa de disparidad generado a partir de la captura de las cámaras. Usando el mapa de disparidad se obtiene el u-disparity y, con él, el mapa libre disparidad (Musleh et al., 2012b). Como se verá en la sección 4.1.8 el mapa libre de disparidad es una representación de aquellos píxeles que no pertenecen a obstáculos dentro del mapa de disparidad. Por tanto, utilizando la información del mapa libre, se pueden eliminar de la imagen binaria todos aquellos píxeles considerados marcas viales que no se encuentran sobre la calzada. Las funciones utilizadas para la realización de estas operaciones no pertenecen a este proyecto y son sólo adaptaciones a las necesidades del desarrollo, de códigos pertenecientes a otros trabajos en la universidad.

d) **Operación de erosión.** En el caso de haber aplicado una mejora de contraste, se aplica una erosión sobre la imagen binaria para eliminar pequeños objetos que no pueden pertenecer a una marca vial. Se utiliza para ello la función `imerode` de MATLAB.

2. **Transformada Rápida de Fourier (*Fast Fourier Transform, FFT*) y reconocimiento.** En estos apartados se describe el conjunto de los procesos utilizados para distinguir los pasos de peatones del resto de marcas viales que aparecen en la imagen binaria obtenida en la etapa anterior. Se utiliza como elemento de clasificación la frecuencia características del paso de peatones en la imagen.

a) **Cálculo de la proyección horizontal de la imagen filtrada.** Para cada columna de la imagen binaria, se suman todos aquellos píxeles que en ella toman el valor 1 (marcas viales). Se obtiene así un vector de longitud igual al número de columnas de la imagen, que tiene en cada una de sus posiciones el número de píxeles pertenecientes a una marca vial en la correspondiente columna de esta captura. La representación del valor de cada una de las posiciones de este vector frente al número de columna de la imagen de la que procede, es lo que se denomina proyección horizontal de la imagen filtrada.

b) **Aplicación de la FFT a la proyección.** Se realiza con la función `fft` de MATLAB.

c) **Análisis de la amplitud del módulo de la FFT para distintas frecuencias y detección**. Como ya se ha dicho en la sección 3.9, la transformada de Fourier es una operación cuyos resultado son valores complejos, es decir, con módulo y fase. En el presente trabajo sólo se analiza el valor de los **módulos** de la transformada en algunas frecuencias concretas en las que se ha estudiado que puede aparecer el paso de peatones en la imagen. Si alguno de estos valores es mayor que unos umbrales prefijados de forma experimental, se puede considerar que en la captura aparece un paso de peatones.

3. **Cálculo de la distancia al paso de peatones si se ha producido una detección**. Para posicionar el paso de peatones en la imagen se utiliza una filosofía análoga a la utilizada para calcular la proyección horizontal de la imagen, aplicada en esta ocasión a sus filas (proyección vertical). Para aquellas imágenes en que el algoritmo ha determinado una detección, y en el rango de filas en que aparece éste, en la proyección vertical se puede apreciar algo similar a una función escalón. Para determinar la posición del paso de peatones en la captura se calcula el punto medio del escalón y se considera que éste representa la fila media del paso de peatones en la imagen. Una vez calculada la posición del paso de peatones en la imagen, para precisar la distancia real a la que éste se encuentra, se utilizará el perfil de la calzada, calculado éste a partir de de la Transformada de Hough y del *v-disparity* asociado al mapa libre de disparidad.

Este proceso se resume gráficamente en la figura 4.3. A continuación se describen con detalle sus operaciones.

4.1.1. Filtrado de la imagen original: 43rd PLT

Como se dijo en la sección anterior, esta operación es la primera de las que se realizan dentro de la etapa de filtrado de la imagen original y es la

encargada de hacer una primera selección entre aquellos píxeles que se consideran potencialmente pertenecientes a una marca vial y aquellos que no.

El procedimiento que se lleva a cabo, está inspirado en el trabajo presentado en Sebsadji et al. (2010). En este artículo se elabora un riguroso análisis comparativo entre cuatro métodos distintos de extracción de los píxeles pertenecientes a marcas viales que aparecen en una imagen: Local Threshold (LT), Symmetrical Local Threshold (SLT), Median Local Threshold (MLT) y 43rd Percentile Local Threshold (43rd PLT).

Todos estos algoritmos toman como principal característica de una marca vial el hecho de que estas señales son generalmente de color blanco, y más ocasionalmente amarillo o azul, pero en definitiva, de un color mucho más claro que el del pavimento de la calzada, lo que en la imagen en escala de grises captada por la cámara, se traduce en un nivel de gris elevado para estos elementos con respecto a la calzada a su alrededor. Por este motivo, si en un entorno de la marca vial se calcula el valor de gris medio, o la mediana del nivel de gris, el resultado obtenido va a ser significativamente menor que el de los píxeles pertenecientes a una marca vial. Los algoritmos anteriormente citados se basan en esta idea para filtrar las imágenes de la siguiente manera:

1. **Convolución** de las filas de la imagen original ($I_{original}$) con un filtro unidimensional $[n \times 1]$ que calcula la media o la mediana, según el método aplicado, de valores de gris en un entorno predefinido del píxel. Los métodos LT y SLT utilizan la media, mientras que MLT utiliza la mediana. 43rd PLT es un caso especial de MLT, que utiliza el valor del percentil 43 en lugar del de la mediana (percentil 50). Los motivos de esta particularidad se explicarán más adelante (ver 4.1.1).
   Cada uno de los algoritmos define el entorno del píxel alrededor del cual se aplica el filtro de manera diferente, pero siempre en función del ancho máximo de las marcas viales ($n = n(S_M)$). Sin embargo, debido a la perspectiva de las imágenes tomadas, los objetos que están más alejados de la cámara tienen un tamaño menor que los que se encuentran más próximos a ésta. Como consecuencia de este hecho, S_M es variable en función de la fila de la imagen y, por ello, la longitud del filtro no puede ser constante independientemente de la fila a la que sea aplicado, siendo realmente su dimensión $[n(S_M(v)) \times 1]$. El parámetro "v"se refiere a la coordenada vertical de la imagen. Así, para LT y SLT, la longitud sobre la que se aplica la convolución es $12S_M(v)$, centrada ésta en el píxel estudiado en cada momento, mientras que para MLT y 43rd PLT es de $2S_M(v)$

   Como resultado de este paso, se obtiene una imagen intermedia ($I_{filtrada}$) que contiene en cada una de sus posiciones el valor medio, o la mediana, de los valores de los píxeles a su alrededor (ver figura 4.5). En la

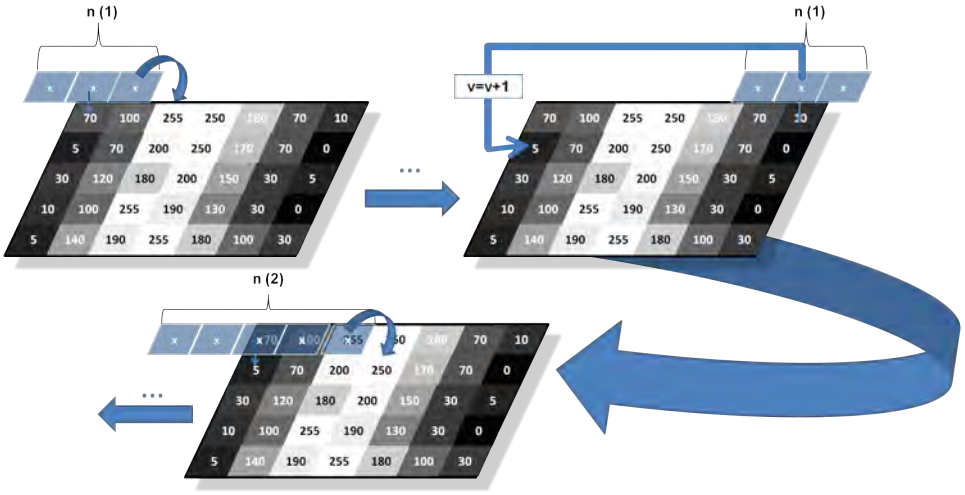


Figura 4.4: Representación del movimiento realizado por el filtro en la convolución: El filtro avanza por una fila y al llegar al final de la misma pasa a la siguiente aumentando de tamaño, según la función descrita por $S_M(v)$

figura 4.4[1] se muestra gráficamente en qué consiste el proceso de convolución (ver también sección 3.5).

2. **Umbralización** (ver sección 3.6). Estos mecanismos de filtrado tienen una característica en común, son la combinación de una convolución más una umbralización. Cada píxel de la imagen original es comparado con su correspondiente píxel de la imagen filtrada. Como se decía al principio de este apartado, un píxel perteneciente a una marca vial va a tener un valor bastante superior al de la media o mediana de los píxeles a su alrededor. Lo que se hace, por tanto, es comparar el valor de gris la imagen original con el de la imagen filtrada. Si $I_{original}(u,v) > I_{filtrada}(u,v) + Umbral$, el píxel de coordenada horizontal u y coordenada vertical v se considera como potencialmente perteneciente a una marca vial.

   Para todos los métodos se considera un único umbral a excepción de para SLT. Para este método el ancho de filtro es $12S_M(v)$ pero computa una media distinta a la izquierda y a la derecha del píxel. Para que éste sea considerado como potencial marca vial deberá cumplir simultáneamente que $I_{original}(u,v) > I_{filtradaIzquierda}(u,v) + Umbral$ y que

[1] En la imagen sólo se muestra el movimiento que realiza el filtro en la convolución, no el resultado de la misma. Las "x"del filtro representan que puede llevar a cabo distintas operaciones.

$I_{original}(u,v) > I_{filtradaDerecha}(u,v) + Umbral.$

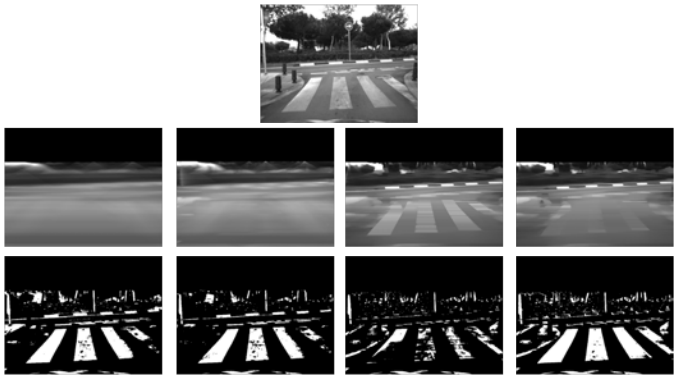
3. **Selección de píxeles conectados**. Todos aquellos píxeles con relación de vecindad que forman grupos de ancho mayor que el mínimo ancho admisible para una marca vial (s_m) son considerados pertenecientes a una de estas señales y son designados con el valor 1 en la imagen binaria resultante de este algoritmo (imagen umbralizada, I_u). El resto de píxeles toman el valor 0. Al igual que ocurre con el valor máximo de ancho de marca vial, el valor mínimo depende del número de fila en que sea considerado ($s_m(v)$).

Como ya se ha visto, en este proceso el rango de tamaños de una marca vial $[s_m(v), S_M(v)]$ depende de la coordenada vertical de la imagen en que sea considerado. Estas funciones se obtienen, para ambos casos, a partir de la definición de una recta, utilizando para ello dos valores de anchos de marca vial distintos.

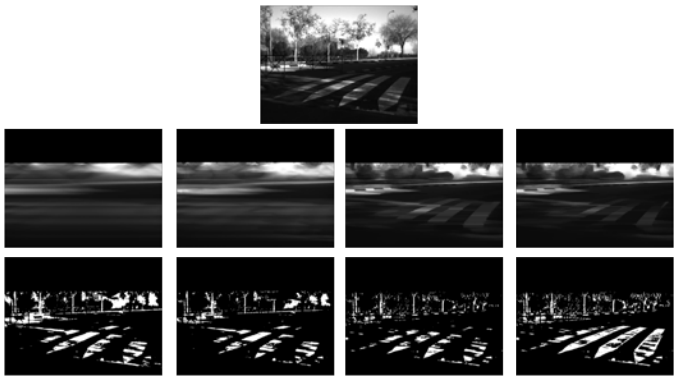
De los 4 algoritmos mencionados, el que ha sido utilizado para realizar esta operación en el presente trabajo es un código inspirado en 43rd PLT. El motivo de esta elección tiene dos razones:

- Por un lado, los autores de (Sebsadji et al., 2010) demuestran que éste es el mejor de los 4 filtros citados. Esto, según ellos, es debido a su vez a dos motivos:
  - La media se ve muy influenciada por valores extremos y los filtros que hacen uso de ella (LT y SLT) no obtienen los mejores resultados a la hora de calcular el nivel de gris más representativo del entorno del píxel. Los filtros basados en la mediana solucionan hasta un 50 % de los errores de clasificación de estos filtros.
  - En las zonas de la imagen en que aparece un paso de peatones, más del 50 % de los píxeles son claros. Es por esta razón por la que es más adecuado utilizar un percentil menor de 50 para calcular la mediana. Experimentalmente demostraron que el percentil que mejores resultado arroja en estos casos es el 43.
- Por otro lado, en este trabajo han sido implementados versiones de los 4 algoritmos, llegando a partir de los resultados a las mismas conclusiones que las que se ofrecen en el trabajo citado. En la figura 4.5 se puede observar un ejemplo de aplicación de estas funciones en 2 imágenes.

A pesar de estar inspirados en las ideas propuestas en Sebsadji et al. (2010), los algoritmos desarrollados para realizar estas pruebas en el proyecto tienen una serie de particularidades propias que cabe mencionar:



(a) Ejemplo sobre una imagen con buena iluminación



(b) Ejemplo sobre una imagen oscura

Figura 4.5: Aplicación de algoritmos de filtrado.
Fila 1: Imagen Original.
Fila 2: Imagen filtrada para (de izquierda a derecha) LT, SLT, MLT y 43rd PLT.
Fila3: imagen umbralizada en el mismo orden

1. No se realiza la operación de "Selección de píxeles conectados"que fue mencionada cuando se describieron estos algoritmos. Como se aprecia en las imágenes de la figura 4.5, para cualquiera de los métodos utilizados se obtienen grupos de píxeles más o menos grandes que no pertenecen a marcas viales y que generalmente no están a nivel del suelo. Una operación de este tipo podría eliminar parte de estos píxeles no pertenecientes a la calzada, sin embargo, como este proyecto está concebido para integrarse con otros módulos de código desarrollados en la universidad, se utiliza un método basado en el **mapa de disparidad** que será explicado más adelante (ver sección 4.1.2).

   Esta operación de agrupar píxeles vecinos también podría facilitar un reconocimiento posterior, al obtener de esta manera conjuntos de pí-

xeles ya conectados, pero esto no interesa de cara a este proyecto, pues no es una información que se considere necesaria para este método de detección y sólo serviría para complicar el código.

Como otra consecuencia de este hecho, el ancho mínimo admisible de una marca vial ($s_m(v)$) no es necesario y, por ello, no es calculado en este algoritmo.

2. En el artículo citado, se calcula la función que define $S_M(v)$ con dos valores de ancho elegidos, a partir de los cuales se define una recta. En dicho trabajo se mencionan los dos valores usados (350 píxeles en el borde inferior de la imagen y 1 pixel en la línea del horizonte). Sin embargo, en este trabajo no se usan esos mismos valores para calcular la función por varios motivos:

   - Por un lado, las imágenes usadas en el trabajo referenciado tienen una resolución de 1920x1080 píxeles, mientras que las usadas en el presente proyecto son de 640x480. Debido a ello, el ancho de las marcas viales va a aparecer en las imágenes de las que se dispone en el presente proyecto con un tamaño menor que en las imágenes usadas en esa investigación.
   - Dado que en el presente trabajo no son de interés todas las marcas viales, sino únicamente los pasos de peatones, se ha optimizado el cálculo de esta recta para estas señales.

   Por estas razones, para la definición de la función $S_M(v)$ en este trabajo, se midió el ancho de una banda de un paso de peatones en una imagen para dos filas distintas y se calculó la recta que definían estos dos puntos. Este es un cálculo simple pero que arroja buenos resultados.

   Según lo dicho anteriormente, la función calculada se define de la siguiente manera (ecuación 2):

$$\left.\begin{array}{l} S_M(480) = 100px \\ S_M(150) = 4px; \end{array}\right\} \Longrightarrow S_M(v) = \frac{16}{55}v - \frac{436}{11}; \qquad (4.1)$$

   Esta definición tiene una consecuencia: la función se anula en 136,25. Al ser una recta de pendiente positiva, para valores inferiores a este número, $S_M(v)$ será negativo, lo que no tiene un sentido físico. Ésta definición de $S_M(v)$ sólo se puede aplicar por ello para $v \geq 137$. En este trabajo, se ha considerado como coordenada de inicio de aplicación del filtro la fila 140 de la imagen y es por este motivo, por lo que en las imágenes de la figura 4.5 aparece una banda negra en la parte superior de las imágenes filtradas y umbralizadas.

3. Los autores de (Sebsadji et al., 2010) no especifican el umbral usado en el paso de "Umbralización". Tras varios ensayos experimentales, en este proyecto se decidió asignarle un valor de 20.

4. En su movimiento a través de las filas, el filtro siempre queda centrado en el píxel en el que se calcula la operación. En los bordes de la imagen la convolución sobrepasa estos bordes (figura 4.4). En estos casos sólo se calculan los valores del filtro que caen sobre valores de la imagen, dejando de haber una simetría respecto al píxel. Además, a la hora de considerar el ancho del filtro, en el algoritmo desarrollado en este proyecto no es n, como de decía anteriormente, sino n+1, pues se incluye en el cálculo el propio píxel sobre el que se aplica el filtro. Así por ejemplo, para 43rd PLT el ancho del filtro sería $2S_M(v)+1$, en lugar de $2S_M(v)$, dejando en el caso general (no en los bordes), $S_M(v)$ valores a cada lado del píxel sobre el que se aplica la operación de convolución.

5. Para calcular MLT y LT 43rd se ha desarrollado un único código que, de hecho, permitiría calcular el filtrado para cualquier percentil. Es por eso que en el apéndice A no se puede encontrar ningúna función desarrollada en MATLAB para el algoritmo MLT.

6. El hecho de haber seleccionado un filtro basado en la mediana para realizar esta operación tiene un inconveniente importante: su coste computacional es bastante mayor que para los algoritmos basados en la media y, consecuentemente, es bastante más lento. En la computadora en que ha sido desarrollado este proyecto (procesador Intel Core i7 a 1,6GHz y 4GB de RAM), el tiempo que tardan LT y SLT en completar este proceso es del orden de 10 veces menor al de MLT o 43rd PLT. En Sebsadji et al. (2010) se utiliza un algoritmo para agilizar este cálculo explicado en Perreault y Hébert (2007). Para este caso, al no ser el objetivo de este proyecto su implementación en un sistema real, no resulta algo crítico el tiempo de computación, aunque este aspecto deberá ser tenido en cuenta si en tal caso se quisiera hacer.

Todas estas modificaciones realizadas sobre el algoritmo expuesto en Sebsadji et al. (2010), definen la función final implementada en MATLAB que realiza esta operación de la etapa de filtrado en el presente trabajo. El código correspondiente puede consultarse en la sección A.1.3.

4.1.2. Filtrado de la imagen original: Mapa libre de disparidad

Como resultado de la operación anterior, se obtiene una imagen binaria en la que se representan todos aquellos píxeles considerados potencialmente

como marcas viales (I_u). Sin embargo, la etapa de extracción de estos píxeles, utilizando la adaptación del algoritmo 43rd PLT, impone como única restricción para que un píxel sea considerado como una marca vial, el hecho de que éste tenga un nivel de gris mayor que el del percentil 43 de los píxeles próximos a él en su misma fila. En una imagen, es fácil que otros píxeles que no pertenecen a marcas viales cumplan esta restricción. Debido a esto, la imagen resultante contiene muchos otros píxeles que no pertenecen a marcas viales (ver figura 4.2b).

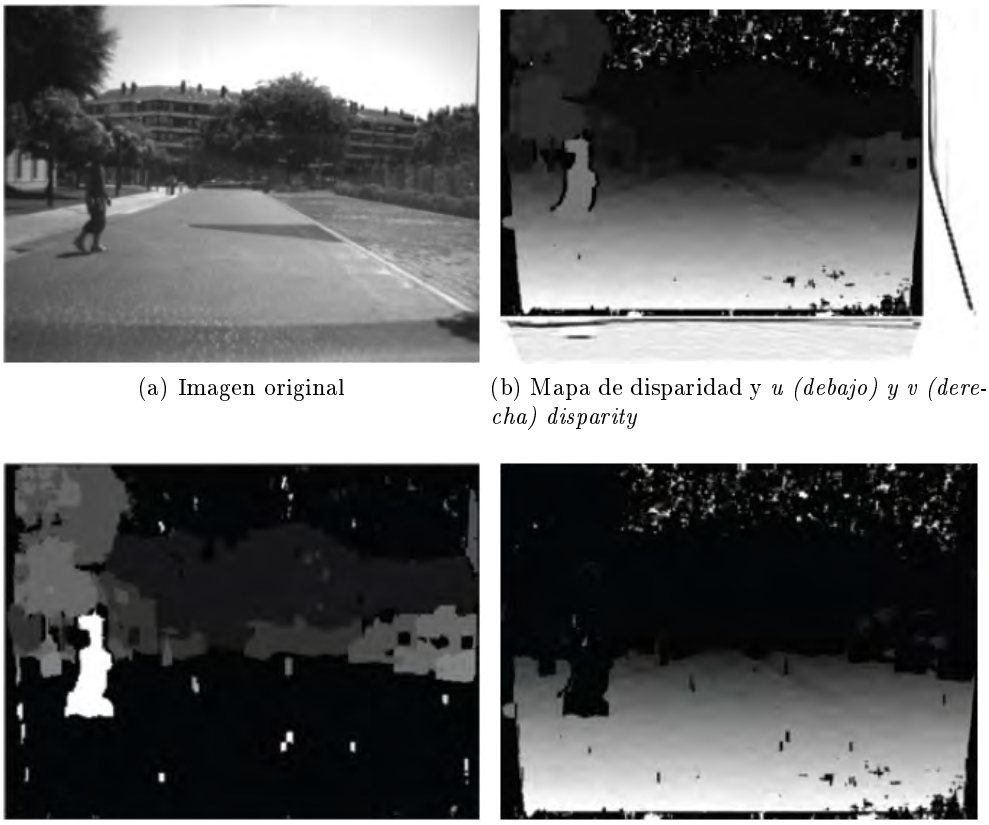
De lo anterior, se puede deducir que es necesario aplicar una condición más restrictiva a la hora de considerar un píxel como perteneciente a una marca vial. En este trabajo se propone extraer de la imagen binaria obtenida, únicamente aquellos píxeles que, además de cumplir con las restricciones impuestas por el algoritmo 43rd PLT, pertenezcan al plano de la calzada. Para determinar qué píxeles de la imagen pertenecen a la calzada, se utilizan la información proporcionada por el mapa denso de disparidad y los *u-v disparity* (ver sección 3.7), basándose en otros trabajos realizados en la universidad (Musleh et al., 2012a).

Según se expone en Musleh et al. (2012a), la información contenida en el mapa de disparidad puede dividirse en dos grupos: por un lado, aquellos píxeles del mismo que representan obstáculos delante del vehículo, es decir, objetos con una altura sobre el plano calzada. Por otro lado, aquellos que representan el espacio libre delante de la cámara, lo que en general, se corresponde con el plano de la carretera. Separando estas dos informaciones contenidas en el mapa de disparidad, se obtienen otros dos mapas densos de disparidad nuevos: **el mapa de obstáculos y el mapa libre de disparidad** (ver figura 4.6).

Para obtener estas dos nuevas estructuras, en Musleh et al. (2012a) se recurre al *u-disparity*. Como ya se explicó en la sección 3.7 de este trabajo, en el *u-disparity* los obstáculos aparecen representados como rectas horizontales, proporcionales a la anchura del mismo y en su correspondiente valor de disparidad. Para obtener a partir de esta representación del mapa de disparidad el mapa de obstáculos, en Musleh et al. (2012a) se propone un algoritmo que cuenta con dos etapas:

1. Umbralización del u-disparity, de tal manera que se obtiene un *u-disparity* binarizado, en el que los píxeles con valor 1 son aquellos pertenecientes a un obstáculo cuya altura es mayor que un umbral medido en píxeles.

2. Extracción del mapa de disparidad de todos aquellos píxeles que permanecen en el *u-disparity* umbralizado. Todos los píxeles extraídos



(a) Imagen original

(b) Mapa de disparidad y *u (debajo) y v (derecha) disparity*

(c) Mapa de obstáculos

(d) Mapa libre

Figura 4.6: Ejemplo de mapa de obstáculos y mapa libre obtenidos a partir del mapa de disparidad

componen el nuevo mapa de disparidad que representa el mapa de obstáculos.

La obtención del mapa libre a partir de lo anterior es inmediata, pues estará compuesto por todos aquellos píxeles del mapa de disparidad que no formen parte del mapa de obstáculos.

En el presente proyecto, a partir del procedimiento descrito anteriormente, el algoritmo calcula el mapa libre de disparidad y utiliza esta información para eliminar de la imagen binaria procedente de 43rd PLT (I_u), todos aquellos píxeles que no aparezcan a su vez en el mapa libre de disparidad. De esta manera, se eliminan de la imagen umbralizada (I_u) aquellos píxeles que no pertenecen al espacio libre delante del vehículo, lo que se asimila como la calzada.

A partir de los procedimientos mencionados en este apartado, se obtiene una imagen binaria nueva que será denominada "imagen limpi" (I_{limpia}).

Cabe mencionar también que el algoritmo perteneciente a este trabajo sí calcula el *u-disparity* y el mapa libre, pero el mapa de disparidad procedente del sistema estereocópico es tomado como dato de entrada al algoritmo.

Las funciones que realizan las operaciones citadas en esta sección no se muestran en el apéndice A, por pertenecer a otros trabajos.

4.1.3. Filtrado de la imagen original: Mejora de contraste

A pesar de que según la secuencia lógica del algoritmo esta operación es la primera en llevarse a cabo en la etapa de filtrado de la imagen, se ha preferido tratarla en este punto, ya que se ha considerado que de esta manera tiene más sentido para el entendimiento de la lógica de desarrollo del proyecto. Esto es así porque ésta fue una mejora que se realizó sobre el código durante una primera revisión de los resultados obtenidos.

Al estudiar los resultados durante una fase temprana del desarrollo, se observó que existía un cierto número de falsos negativos en las colecciones de imágenes tratadas. Analizando las capturas en que esto ocurría, se llegó a la conclusión de que éstas eran demasiado oscuras. Debido a este hecho, el contraste entre las marcas viales y la calzada a su alrededor no era lo suficientemente grande, y el filtro 43rd PLT no conseguía extraer de las imágenes un número suficiente de píxeles pertenecientes al paso de peatones, como para conseguir un adecuado reconocimiento del mismo. Un ejemplo de este tipo de imágenes se muestra en la fila 1 de la figura 4.7

Un primer planteamiento ante este problema fue tratar de mejorar el contraste de todas las imágenes previamente a la aplicación de 43rdPLT. Para ello se hicieron pruebas tanto ecualizando el histograma con la función `histeq` de MATLAB, como ajustando el contraste con la función `imadjust`, pero los resultados en aquel momento no fueron los esperados y no sólo no se redujeron significativamente los falsos negativos, sino que creció el número de falsos positivos.

A la vista de los resultados anteriores, se hizo un segundo análisis de las imágenes en que el algoritmo tenía problemas a la hora de detectar el paso de peatones y se dedujo que todas ellas presentaban un hecho en común: el problema no era tanto que las imágenes fueran oscuras, como se había deducido en un primer momento, sino que lo que en las imágenes aparecía con

(a) Imagen oscura sin ajuste de contraste

(b) Imagen (a) tras el proceso de filtrado con 43rdPLT

(c) Imagen oscura con ajuste de contraste

(d) Imagen (c) tras el proceso de filtrado con 43rdPLT

Figura 4.7: Ejemplo de imagen sobre la que aplicar la mejora de contraste

niveles de gris más bajos era únicamente la calzada y esto era consecuencia de que la parte correspondiente al cielo aparecía con valores de gris mucho más altos. En definitiva el problema era que la parte superior de la imagen era muy clara, y esto provocaba que la inferior fuera muy oscura y con un contraste muy bajo.

Para solucionar esto se decidió hacer una mejora de contraste únicamente de las zonas más oscuras de la imagen, para lo que se volvió a recurrir a la función de MATLAB `imadjust`, variando los parámetros que se usaron para su configuración en un primer momento. Para una explicación más detallada del funcionamiento y parámetros de `imadjust` se puede consultar el apéndice A.3.3.1

En resumen, lo que permite la función `imadjust` es aplicar el método de modificación del contraste denominado amplitud del histograma (ver apar-

tado 3.4), para "estirar" un rango de valores del histograma de la imagen, obteniendo una nueva con mayor contraste para ese rango.

En la primera aproximación que se hizo al problema usando `imadjust`, a `[low_in, high_in]` se le asignó el valor [0.01,0.99] usando la opción `stretchlim(I)` que ofrece la función, mientras que `[low_out, high_out]` fue fijado a [0,1] usando la sintaxis equivalente `[]`. Después de haber deducido que el problema estaba en que la parte inferior de la imagen tenía un contraste insuficiente, se modificaron estos valores de las entradas. Para el uso final de la función en el algoritmo de este proyecto `[low_in, high_in]` toma un valor de [0.05,0.15] y `[low_out, high_out]` de `[]`, es decir, los valores del histograma de `I` que dejan entre el 5 % y el 15 % de valores a su izquierda (valores oscuros) son extendidos a lo largo de todo el histograma de la nueva imagen `J`, mientras que el resto de valores inferiores y superiores a este rango son colocados a 0 y 255, respectivamente. Como se observa en la fila 2 de la figura 4.7[2], estos valores, obtenidos experimentalmente, mejoran enormemente el contraste de la calzada en este tipo de imágenes y, como consecuencia, el resultado del filtrado.

Esta solución tiene sin embargo un inconveniente: en imágenes que no cumplen con los supuestos deducidos anteriormente, es muy probable que la imagen quede inservible al aplicar esta transformación, ya que en capturas más claras, en las que la diferencia entre el nivel de gris medio de la parte superior e inferior no es tan acusada, la mayoría de los píxeles saturan a 255, dando como resultado una imagen prácticamente blanca, con muy poco contraste (figura 4.8).

Para solucionar este problema esta operación no se aplica sobre todas las imágenes obtenidas por las cámaras, sino que se usa sólo de manera selectiva. Para realizar esta criba entre unas imágenes y otras se utiliza el siguiente procedimiento:

1. Se calcula el nivel de gris medio de la mitad superior de la imagen.
2. Se calcula el nivel de gris medio de toda la imagen.
3. Se obtiene la diferencia entre ambas medias. Experimentalmente se ha deducido que si ese valor es superior a 40, no existe ningún problema al aplicar la mejora de contraste.

A la hora de obtener este método para diferenciar imágenes a las que se puede aplicar `imadjust` de las que no, se planteó también que la diferencia de medias fuera calculada entre la mitad superior e inferior de la imagen o

[2]Se muestra únicamente el resultado tras filtrar con LT 43rd. No se ha aplicado la operación de limpieza con el mapa de disparidad

(a) Imagen clara sin ajuste de contraste

(b) Imagen (a) tras el proceso de filtrado con 43rdPLT

(c) Imagen clara con ajuste de contraste

(d) Imagen (c) tras el proceso de filtrado con 43rdPLT

Figura 4.8: Ejemplo de imagen sobre la que no se debe aplicar la mejora de contraste

entre la media global y la parte inferior, pero finalmente se eligió el método aplicado por ser el más discriminante.

En este caso no existe una única función que realice los procesos aquí citados. La función encargada de hallar la diferencia entre las medias superior y global de la imagen se puede encontrar en la sección A.1.2. Por otro lado, la decisión de aplicación de `imadjust` en función del valor de esta diferencia de medias se puede encontrar en la función `CrossDetect` (ver sección A.1.1). La función `imadjust`, es aplicada en caso necesario en la función `LTprd_CrossDetect` (sección A.1.3).

(a) Imagen original tras aplicar una mejora de contraste

(b) Imagen (a) tras el proceso de filtrado con 43rdPLT

(c) Imagen (a) tras el proceso de filtrado con 43rdPLT y después de aplicar `imerode`

Figura 4.9: Ejemplo de imagen sobre la que se debe aplicar una erosión para eliminar errores de segmentación

4.1.4. Filtrado de la imagen original: Erosión

Al igual que ocurría con la operación de mejora de contraste, ésta fue una modificación que se hizo al algoritmo *a posteriori*, una vez analizados los resultados que arrojaba el mismo al ser probado sobre las colecciones de imágenes. De hecho, se hizo en conjunto con la anterior, pues cuando era aplicada la mejora de contraste se producía un incremento de píxeles mal clasificados como marca vial, dando lugar en ocasiones a falsos positivos en el reconocimiento final del paso de patones.

Para solucionar este problema de segmentación se decidió aplicar una operación que disminuyese la cantidad de píxeles mal clasificados en aquellas imágenes sobre las que previamente se hubiese aplicado `imadjust`. Un ejemplo de estas imágenes problemáticas se puede ver en la figura 4.9.

Estos errores de segmentación en las imágenes se presentan en forma de píxeles aislados o formando pequeños grupos. Por este motivo, para eliminarlos se decidió realizar una erosión en la imagen (ver apartado 3.8). Para ello se utiliza la función `imerode` de MATLAB. En el apéndice A.3.3.2 se hace una explicación más detallada de esta función y de los parámetros usados en su configuración para la utilización en el presente proyecto.

4.1.5. FFT y reconocimiento: Transformada Rápida de Fourier (FFT)

El conjunto de operaciones de la etapa de filtrado, da como resultado una imagen binaria (I_{limpia}) en la que se representan todos aquellos píxeles de la imagen original que potencialmente pertenecen a una marca vial cualquiera de las que existen dibujadas en la calzada: pasos de peatones, señales de ceda el paso, líneas de delimitación de carril, etc. El objetivo de toda esta nueva

etapa es discriminar entre aquellas marcas viales que son pasos de peatones y aquellas que no. Este y los siguiente epígrafes de esta sección explican las operaciones realizadas con este fin de manera independiente, aunque como se verá, todas ellas están mucho más relacionadas entre sí que las correspondientes a la etapa de filtrado, haciéndose constantes referencias entre ellas y siendo difícil una comprensión del conjunto sin el entendimiento e interrelación de las partes que lo componen. Las funciones relativas a este epígrafe pueden consultarse en el apéndice A en las secciones A.1.1 y A.1.4. En este primer punto, se explicará con detalle la Transformada Rápida de Fourier que será fundamental para el resto de la descripción de este bloque.

En la sección 3.9 se explicó el concepto de la transformada de Fourier. En resumen, esta transformación permite cambiar el espacio de trabajo del dominio temporal o espacial al frecuencial, teniendo en cuenta que cualquier señal en el dominio del tiempo o el espacio puede descomponerse en la suma de infinitas señales periódicas.

En visión por computador y análisis de imágenes, el uso más habitual de esta transformación es el filtrado de las capturas obtenidas, ya que por ejemplo algunos tipos de ruido, como la interferencia, son mucho más fáciles de eliminar en el dominio de la frecuencia. Su utilización en la extracción de características es menos habitual en la bibliografía y se utiliza sobre todo para la caracterización de texturas, es decir patrones homogéneos y periódicos que se repiten en la imagen y que pueden ser por ello representados con una frecuencia característica.

Una marca vial correspondiente a un paso de peatones se adecúa a esta descripción, componiendo un patrón periódico de franjas verticales a lo largo de la imagen, por lo que su frecuencia característica es útil para discernir una de estas señales de otra cualquiera de las que pudieran aparecer en la imagen limpia. Es por ello que será utilizada una **variante de la transformada de Fourier** para convertir la imagen (I_{limpia}) del dominio del espacio al de la frecuencia y caracterizar de esta forma el paso de peatones.

Debido a sus propiedades la transformada de Fourier es una operación utilizada en multitud de aplicaciones, pero su resultado es una función continua cuyo coste computacional de cálculo es muy alto. Por este motivo, la transformada de Fourier no puede ser utilizada en su definición tradicional para los cálculos en una computadora. Para una imagen, el cálculo de la transformada de Fourier se puede escribir de la siguiente forma (ecua-

ción 4.2):

$$\begin{aligned} \mathcal{F}(\omega_1, \omega_2) &= \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f(m,n) e^{-j\omega_1 m} e^{-j\omega_2 n} \\ m &= coordenada\, vertical\, de\, la\, imagen \\ n &= coordenada\, horizontal\, de\, la\, imagen \\ \omega_1 &= frec.\, filas[rad/s] \\ \omega_2 &= frec.\, columnas[rad/s] \end{aligned} \tag{4.2}$$

Lo cual representa la transformada de Fourier bidimensional del espacio discreto de la imagen a otro continuo en el dominio de la frecuencia.

Para una computadora, el tratar con funciones continuas no es posible y operan sólo con funciones discretas con más o menos semejanza a la función real a la que representan, en función del número de muestras de la función original utilizadas en su representación digital. Es por ello que para su cálculo en una computadora se utiliza la Transformada de Fourier Discreta (DFT) cuya expresión se puede escribir de la siguiente forma (ecuación 4.3):

$$\begin{aligned} \mathcal{F}(p,q) &= \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) e^{-j2\pi \frac{pm}{M}} e^{-j2\pi \frac{qn}{N}} \\ p &= 0, 1, ..., M-1\, M = resolucin\, vertical\, de\, la\, imagen \\ q &= 0, 1, ..., N-1\, N = resolucin\, horizontal\, de\, la\, imagen \end{aligned} \tag{4.3}$$

Cuya relación con la función continua es (ecuación 4.4):

$$\begin{aligned} \mathcal{F}(p,q) &= \mathcal{F}(\omega_1, \omega_2)|_{\omega_1 = \frac{2\pi p}{M};\, \omega_2 = \frac{2\pi q}{N}} \\ p &= 0, 1, ..., M-1 \\ q &= 0, 1, ..., N-1 \end{aligned} \tag{4.4}$$

La FFT es el algoritmo que permite el calculo optimizado de la DFT, y es por este motivo por el que se utiliza esta variante de la transformada de Fourier a la hora de ser aplicada en una computadora y, por tanto, en este trabajo.

A pesar de que en las líneas anteriores se ha descrito la FFT bidimensional por ser una imagen un espacio 2-D, en esta investigación y como se verá en el siguiente apartado, se usa la FFT unidimensional. La expresión para esta

transformación viene dada por la ecuación 4.5:

$$\begin{aligned} \mathcal{X}(k) &= \sum_{n=0}^{N-1} x(n)\omega_N^{nk} \Longrightarrow \\ \mathcal{X}(k) &= \sum_{n=1}^{N} x(n)\omega_N^{(n-1)(k-1)} \\ \omega_N &= e^{\frac{-2\pi j}{N}} \end{aligned} \tag{4.5}$$

De la expresión 4.5 es necesario apuntar varias cosas:

- El cambio realizado para comenzar en $n = 1$, en lugar de en $n = 0$ (ecuación 4.5), se debe a que $x(n)$, como se verá más tarde (ver 4.1.6), va a representar en MATLAB un vector cuya primera posición tiene como índice en este software el valor 1 y no 0.

- Para las colecciones de imágenes utilizadas, $N = 640$, por ser éste el número de columnas de las capturas.

- Si se sustituye en la expresión 4.5 $k = 1$ se obtiene (ecuación 4.6):

$$\mathcal{X}(1) = \sum_{n=1}^{N} x(n)\omega_N^0 \Longrightarrow \mathcal{X}(1) = \sum_{n=1}^{N} x(n) \tag{4.6}$$

  A partir de aquí se concluye que el primer coeficiente de la transformada ($k = 1$), correspondiente a la frecuencia 0, es un número real que representa la suma de todos los valores del vector $x(n)$.

- Las posibles frecuencias de las que la FFT va a tener valores serán (ecuación 4.7):

$$\begin{aligned} \mathcal{X}(k) &= \sum_{n=1}^{N} x(n)e^{-2\pi j\frac{(n-1)(k-1)}{N}} \Longrightarrow \\ \mathcal{X}(k) &= \sum_{n=1}^{N} x(n)(cos(\frac{2\pi(k-1)(n-1)}{N}) - j\ sen(\frac{2\pi(k-1)(n-1)}{N})) \xrightarrow{n-1=t} \\ \mathcal{X}(k) &= \sum_{n=1}^{N} x(n)(cos(\frac{2\pi(k-1)}{N}t) - j\ sen(\frac{2\pi(k-1))}{N}t)) \Longrightarrow \\ \omega(k) &= \frac{2\pi(k-1)}{N} \Longrightarrow f(k) = \frac{k-1}{N} \end{aligned} \tag{4.7}$$

  Lo que significa también que cada coeficiente $\mathcal{X}(k)$ está asociado a una frecuencia $f(k)$.

- Por otro lado de la expresión 4.7 se puede deducir que la frecuencia de muestreo (f_s) para el cálculo de la FFT es (ecuación 4.8):

$$f_s = f(k+1) - f(k) = \frac{(k+1)-1}{N} - \frac{k-1}{N} = \frac{1}{N} \tag{4.8}$$

  En el caso de este trabajo, para N=640, se tiene que (ecuación 4.9):

$$\begin{aligned} f_s = \frac{1}{640} = 1,5625 \times 10^{-3} \Longrightarrow f(1) &= \frac{1-1}{640} = 0\, pixel^{-1} \\ f(2) &= \frac{2-1}{640} = 1,5625 \times 10^{-3}\, pixel^{-1} \\ f(3) &= \frac{3-1}{640} = 3,125 \times 10^{-3}\, pixel^{-1} \\ f(4) &= \frac{4-1}{640} = 4,6875 \times 10^{-3}\, pixel^{-1} \\ f(5) &= \frac{5-1}{640} = 6,25 \times 10^{-3}\, pixel^{-1} \ ... \end{aligned} \tag{4.9}$$

Estos conceptos serán importantes más adelante.

Por otro lado, también es importante destacar que, como se puede deducir de las expresiones anteriores y como ya se explicó en el apartado 3.9, la transformada de Fourier devuelve para cada coeficiente de $\mathcal{X}(k)$ un valor complejo, es decir con módulo y fase. Como se verá más adelante, en este trabajo sólo se utilizará el módulo de este número.

Para el cálculo de la FFT en el presente proyecto se usará la función de MATLAB `fft`.

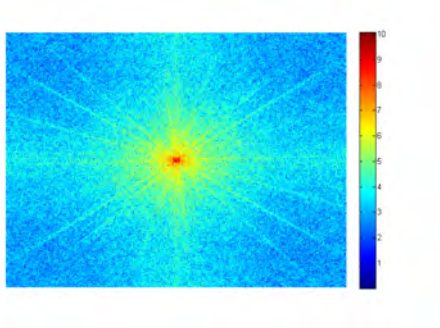
4.1.6. FFT y reconocimiento: Cálculo de la proyección horizontal

Una vez visto el concepto de la FFT, en ésta y en las siguiente secciones se tratará sobre cómo es aplicada esta operación al algoritmo y cómo se utilizan los datos que `fft` arroja con el objetivo de distinguir los pasos de peatones del resto de marcas viales que aparecen en la imagen (I_{limpia}); meta final de esta etapa.

En un primer momento a la hora de abordar este problema, se pensó en utilizar la FFT bidimensional, implementada en MATLAB con la función `fft2`, sobre las imágenes obtenidas de la etapa de filtrado.

Según la expresión 4.3, el cálculo de la FFT bidimensional es equivalente a realizar la FFT de las columnas de la imagen y a continuación aplicar ésta a las filas del resultado. La complejidad de interpretación de los valores

(a) Imagen limpia



(b) Transformada de Fourier bidimensional de 4.10b

Figura 4.10: Módulo de la Transformada de Fourier Bidimensional de una imagen

obtenidos de este proceso (ver figura 4.10) es por tanto mayor y, teniendo en cuenta el hecho de que el paso de peatones visto desde un vehículo es horizontal y tiene por tanto su componente frecuencial característica en las filas de la imagen, no parecía previsible que la información frecuencial de las columnas tuviera algún valor en esta aplicación. Es por esto que, una vez establecida esta hipótesis, se simplificó el problema a calcular la FFT unidimensional aplicada a las filas de la imagen.

La idea de procedimiento desarrollado a tal fin se basaba en tomar cada una de las M filas de la imagen y aplicar `fft` sobre cada una de ellas, analizando a continuación esos resultados. Si en alguna de las filas se detectaba la frecuencia característica del paso de peatones con un valor de módulo adecuado, se podía concluir que en la captura aparecía una de estas marcas viales. Este método sin embargo tiene varios inconvenientes:

- Lógicamente, *a priori* se desconoce en que fila/s va a aparecer el paso de peatones en la imagen por lo que se debe calcular `fft` para todas las filas de la misma o, en todo caso, concluir que este tipo de señales pintadas en el suelo pueden aparecer sólo en la mitad inferior de la captura y calcular FFT únicamente para la mitad de las filas. A pesar de esta simplificación el rendimiento del algoritmo dependería de la resolución vertical de la imagen, pues poniendo como ejemplo las imágenes utilizadas en este proyecto ($M = 480$), habría que aplicar `fft` 480 veces sin la simplificación y 240 veces con ella, analizando después sus correspondiente resultados, lo que haría muy costoso el cálculo del algoritmo.

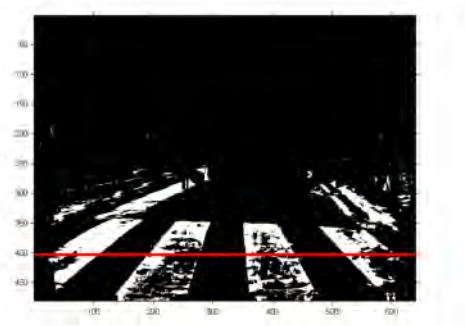
- Si el paso de peatones fuera tomado por las cámaras desde arriba,éste

(a) (b)

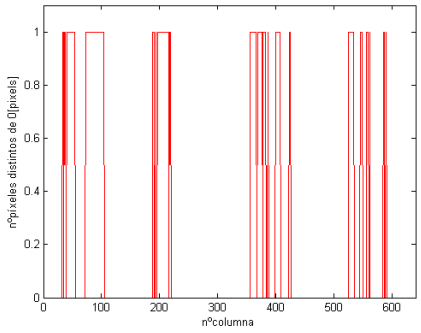
Figura 4.11: Paso de peatones visto según la perspectiva de la imagen (a) y visto desde arriba (b). Se demuestra que la distancia entre bandas es variable según la distancia al vehículo según la perspectiva de la cámara.

aparecería en la imagen como una secuencia de bandas blancas verticales y paralelas separadas entre sí por una distancia constante. Debido a la perspectiva de las capturas esto no es así, y el paso de peatones en las imágenes está representado por bandas oblicuas, que aparecen más separadas cuanto más cerca está la señal del vehículo y más próximas cuanto más alejado se encuentra éste (figura 4.11). Debido a este hecho, que ya se tuvo en cuenta a la hora de aplicar el filtrado con 43rd PLT, la frecuencia del paso de peatones es variable en función de su distancia al coche y además, para una distancia fijada, distinta para cada fila perteneciente al objeto. Como consecuencia de esto, no existe una frecuencia característica del paso de peatones sino una gama de frecuencias en función de la fila en la que aparezca éste y de la fila del mismo que este siendo estudiada. Para aplicar el método estos valores deberían ser también estudiados y conocidos lo que complica bastante su implementación.

- Si el proceso de segmentación fuera perfecto y se representara en una gráfica una fila de la imagen (I_{limpia}) que perteneciera a un paso de peatones, con el número de columna en el eje de abscisas y el valor del píxel (que puede ser 0 ó 1) en el de ordenadas, se obtendría algo parecido a un tren de pulsos de periodo $T = \frac{1}{f_{paso(fila)}}$. Esta es la idea que, cómo se decía antes, pretendía implementarse, sin embargo el proceso de segmentación no es perfecto y, debido a ello, el paso de peatones presenta en la imagen limpia píxeles vacíos dentro de las franjas blancas. Como consecuencia de estos errores de segmentación,



(a)



(b) PH de la fila de 4.12a marcada en rojo

Figura 4.12: Ejemplo de proyección horizontal para una fila de una imagen en que aparece un paso de peatones. Debido a errores de segmentación la PH no está compuesta por pulsos de anchura constante.

el tren de pulsos representado por la fila no esta compuesto por pulsos de anchura constante, con lo que ya no representa una señal periódica con una frecuencia característica siempre igual para cada par de valores distancia del vehículo-fila del objeto (ver figura 4.12).

Debido a estos problemas para obtener información de la imagen limpia aplicando `fft` fila a fila, se decidió usar lo que se denomina **proyección horizontal de la imagen**. Su obtención, como ya se introdujo en 4.1.7, se basa en conseguir un vector (PH) cuyas posiciones contienen la suma de todos los píxeles pertenecientes a una marca vial que aparecen en cada columna de la imagen (I_{limpia}), es decir, el proceso consiste en tomar una columna y contar en ella todos aquellos píxeles que tienen valor 1 repitiendo el proceso para todas las columnas(N) y almacenando cada uno de los valores obtenidos en PH. Si se representa el valor de este vector frente al número de columna de la imagen limpia se obtiene un gráfico con el número de píxeles pertenecientes a una marca vial en cada columna de la imagen (figura 4.13). Este vector es al que se le aplicará la FFT para realizar el análisis de sus frecuencias y

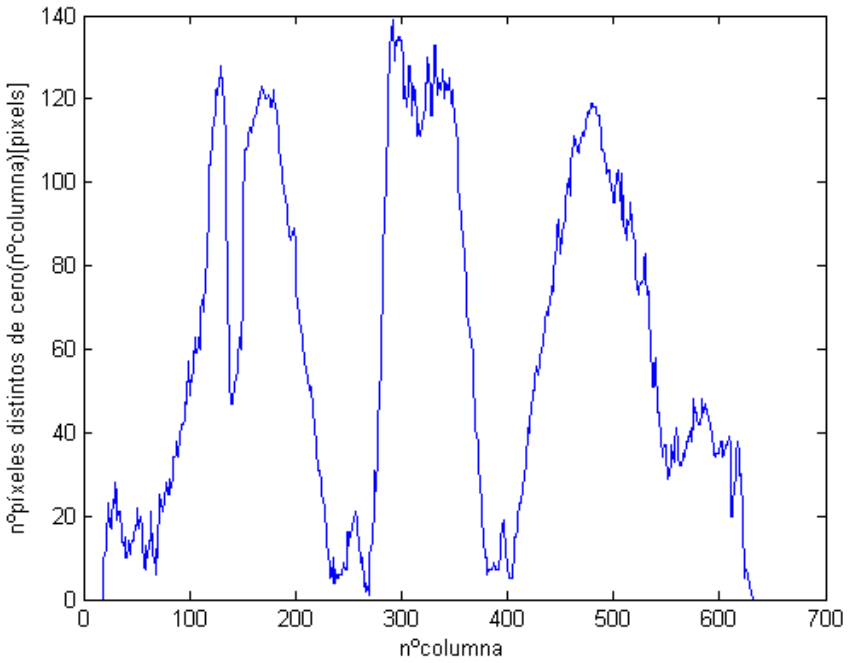


Figura 4.13: Ejemplo de proyección horizontal de una imagen

módulos asociados.

Esta herramienta soluciona de una forma muy simple los problemas que aparecían al aplicar `fft` fila a fila:

- Al obtener un único vector para representar toda la imagen, sólo es necesario aplicar `fft` una vez, analizando además con ello un único conjunto de resultados. Esto hace el algoritmo mucho más simple y aumenta su rendimiento.
- Al representar toda la imagen con una única fila de valores se pasa de tener un problema bidimensional a otro en una única dimensión. Con el método anterior se dijo que la frecuencia del paso de peatones dependía de la distancia del objeto al vehículo y, fijada ésta, de la fila que se considerase del mismo, por tratarse de un objeto con una superficie. Al unidimensionalizar el problema, la frecuencia característica obtenida para el paso de peatones va a depender sólo de la fila en que éste aparezca en la imagen, ya que el objeto no va a tener ya una superficie en su representación en PH. El rango de frecuencias que pueden caracterizar al objeto en la imagen va a seguir existiendo, pero va ser constante para todo él y variable únicamente en función de

la distancia a la que aparezca éste en la imagen (I_{limpia}). Este rango de frecuencias características va a ser diferente del que representaba a las filas del paso de peatones original y propio de la nueva "señal"que contiene PH. Esta señal ya no será un tren de pulsos, sino una señal periódica diferente. En la figura 4.13 se muestra un ejemplo de este tipo de señales cuando en la imagen (I_{limpia}) aparece un paso de peatones.

- Con el método anterior, los fallos de segmentación durante la etapa de filtrado de la imagen ($I_{original}$) eran determinantes, pues modificaban de forma importante la fila de la imagen tratada y de esta forma su frecuencia característica. Con el procedimiento implementado usando la proyección horizontal de la imagen, el algoritmo es mucho más robusto a este tipo de errores, pues los fallos de segmentación de una fila son compensados por otras, al obtenerse PH como suma de los píxeles de todas las filas para una columna concreta. Esto es similar a aplicar en cada columna un filtro paso bajo espacial que filtra este tipo de errores que son asimilables a altas frecuencias.

4.1.7. FFT y reconocimiento: Detección

Con el cálculo de PH, obtenido en la etapa anterior, toda la información frecuencial de las filas de la imagen (I_{limpia}) ha sido condensada en un único vector. A partir de aquí, se utilizará la información contenida en el mismo para discernir si en las imágenes aparece o no un paso de peatones.

Para extraer la información frecuencial contenida en PH, el primer paso es aplicar `fft` sobre PH. Una vez hecho esto, se obtiene un nuevo vector (FT), que contiene en cada una de sus posiciones los coeficientes de la FFT de PH. Estos valores nos dan una información doble:

- Por un lado, el módulo del número complejo dice la importancia de ese coeficiente y con ello, y según se extrae de la ecuación 4.7, de la correspondiente frecuencia asociada a éste.
- Por otro lado, el argumento de cada uno de los coeficientes indica la fase de las respectivas $f(k)$.

En este trabajo sólo se utiliza el módulo de FT. Debido a ello, una vez calculado este vector, se obtiene el módulo de sus coeficientes ($Modulo(k)$). Por otro lado, las frecuencias asociadas a cada coeficiente ($f(k)$) se obtienen de la expresión 4.9.

Una vez calculados estos datos, si se representa $Modulo(k)$ frente a $f(k)$, se obtiene un gráfico similar al de la figura 4.14. En él se aprecia cómo, en general, el valor de $Modulo(k)$ tiene una tendencia decreciente según aumenta

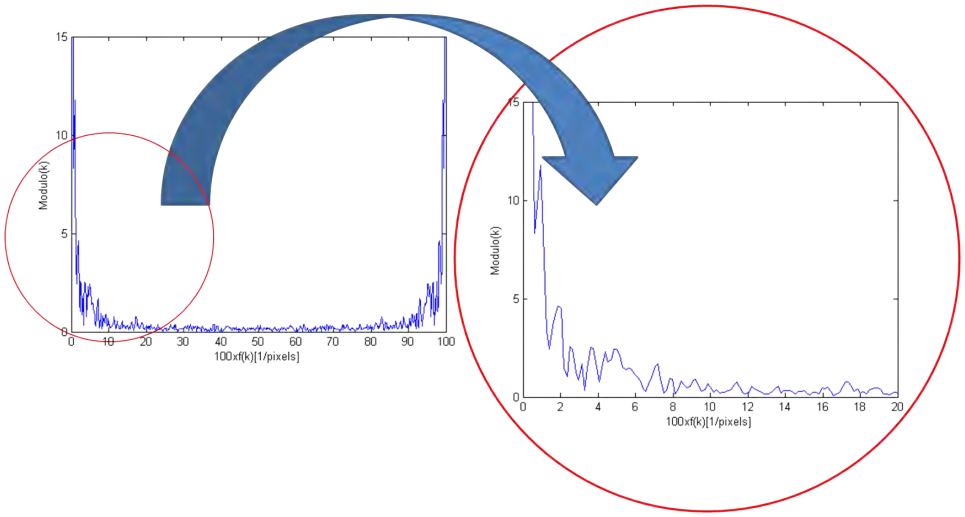


Figura 4.14: $Modulo(k)\,VS\,f(k)$

$f(k)$ y que a partir de $k = N/2$ el gráfico se repite. De lo anterior se puede deducir que los valores de los coeficientes son menos importantes a medida que aumenta la frecuencia, puesto que las bajas frecuencias representan la forma general de la función, mientras que las altas frecuencias determinan los "detalles"de la misma. También se concluye que FFT es simétrica, por lo que se pueden considerar únicamente los $N/2$ primeros valores de $Modulo(k)$. Hay que observar sin embargo que se ha dicho que la tendencia decreciente es una **tendencia general** en la gráfica de FFT. Ocurre sin embargo que, cuando una frecuencia tiene una presencia importante en la imagen (I_{limpia}), su valor $Modulo(k)$ es considerablemente mayor que los asociados a las $f(k)$ de su entorno, tomando la forma de un máximo local. Cabe recordar también que, aunque aquí se represente como una función continua por ofrecer una mayor claridad en la presentación, esta gráfica corresponde a la interpolación de los valores de una función que es en realidad discreta. Más adelante se presenta esta gráfica con su representación real. En ella se aprecia como existe un único valor $Modulo(k)$ para cada $f(k)$ (ver figura 4.15).

En un primer momento, las deducciones anteriores fueron concluidas de una forma experimental. Cogiendo una muestra de las imágenes disponibles, se calculó su FFT y se observaron los 6 valores mayores de $Modulo(k)$. De esta manera se comprobaron dos cosas:

- Por un lado, el valor del pico mayor (correspondiente a $Modulo(1)$) mantiene una relación lineal con el número de píxeles que aparecen en la imagen (I_{limpia}), tal y como se demostró en la ecuación 4.6. Haciendo un ajuste por mínimos cuadrados se probó que esta ecuación es de la

forma:

$$pico_{superior} = 0,003 * n^{o}pxeles_{I_{limpia}} - 3,2797; \quad (4.10)$$

Siendo la precisión del ajuste del 99 %. De esta conclusión se deduce que $Modulo(1)$ no nos da ninguna información acerca de si en la imagen aparece o no un paso de peatones.

- Como consecuencia de lo anterior, se buscó otro valor que pudiera determinar si en la imagen existía o no un paso de peatones. Para ello, en lugar de comprobar el valor del segundo $Modulo(k)$ más importante, se analizó el valor del segundo máximo local y la frecuencia a la que éste aparecía, tomando como referencia de este análisis la representación continua de $Modulo(k)\ VS\ f(k)$ que se vio en la figura 4.14. Se concluyó de esta manera que, cuando en la imagen aparecía un paso de peatones **cerca** del vehículo, este segundo máximo local se encontraba generalmente en $f(5) = 6{,}25 \times 10^{-3} pixels^{-1}$, asociado a un valor $Modulo(5)$ bastante alto. Además, aunque este segundo máximo local no estuviera situado en $f(5)$, la $f(k)$ a la que aparecía era cercana a ésta y $Modulo(5)$ seguía teniendo un valor relativamente alto, si en la imagen limpia aparecía un paso de peatones.

De estos resultados, se obtuvo **la primera versión de la parte del algoritmo correspondiente a la detección del paso de peatones en las imágenes** (ver capítulo 5), consistente en buscar el valor de $f(5)$ y el valor de $Modulo(5)$ asociado a esta frecuencia. Si $Modulo(5)$ era mayor de 7 se concluía que en la imagen (I_{limpia}) existía un paso de peatones. Una modificación posterior de una de las funciones asociadas al cálculo del mapa libre de disparidad mejoró el proceso de segmentación, haciendo que en el paso de peatones representado en las imágenes apareciesen menos "huecos"vacíos provocando que el valor de $Modulo(5)$ aumentase. La razón de esto es que al haber más píxeles relacionados con esa frecuencia el valor del $Modulo(5)$ también aumentaba. Debido a esto el umbral se pudo subir hasta un valor de 18.

Como se decía en los párrafos anteriores, las conclusiones que llevaron a utilizar $Modulo(5)$ para estas primera versiones del algoritmo fueron obtenidas únicamente a partir de una observación de los valores de $Modulo(k)$, desarrollándose posteriormente la demostración de por qué ocurría esto, tal y como sucedió en otras partes de esta investigación. Con el objetivo de llegar a una teoría que explicase las razones de poder utilizar $Modulo(5)$ a la hora de decidir si en las capturas existía o no un paso de peatones, se analizó la imagen de un verdadero positivo. Un ejemplo de este tipo de capturas se muestra en la figura 4.15. En la figura 4.15a se observa la imagen (I_{limpia}) de una captura para la cual se tiene un verdadero positivo usando $f(5)$. A

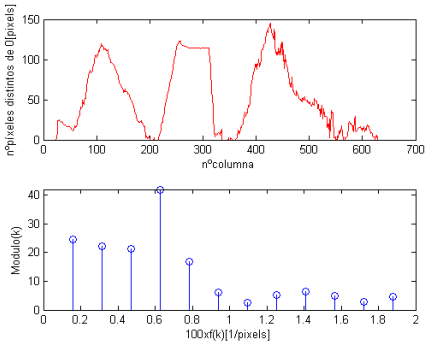
su derecha de muestran sus correspondientes PH y $Modulo(k)$ VS $f(k)$ (figura 4.15b), representada en este caso en su forma real de función discreta. En 4.15c aparece una imagen equivalente a un tren de pulsos (TP) de $T = 160 pixels \Longrightarrow f = \frac{1}{T} = f(5)$ y a su derecha sus respectivos PH y $Modulo(k)$ VS $f(k)$ (figura 4.15d). Se puede ver así como la componente frecuencial principal de TP se encuentra en la frecuencia que lo define. Se aprecia también como PH de la imagen limpia tiene un aspecto similar al correspondiente de TP y cómo de esta manera su componente frecuencial principal se encuentra también en $f(5)$. Se puede observar sin embargo que el $Modulo(k)$ VS $f(k)$ de la imagen tiene otras componentes frecuenciales que no tiene el correspondiente a TP. Esto se debe a que PH de la imagen es equivalente a una "deformación"del TP, por lo que tiene un mayor contenido armónico. De todo esto se concluye que las imágenes en que aparecen pasos de peatones tienen un PH similar a un tren de pulsos, motivo por el cual su frecuencia característica se encuentra en la frecuencia correspondiente al TP equivalente.

Teniendo en cuenta lo anterior se pueden deducir varias cosas:

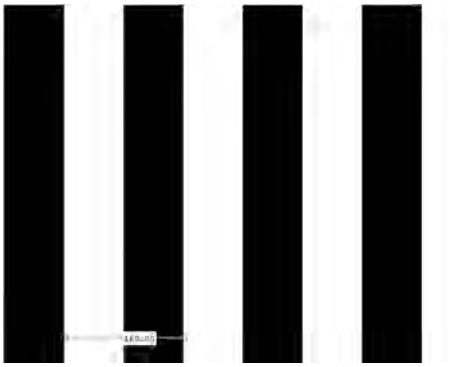
- Se pueden usar más valores de $Modulo(k)$ en la detección del paso de peatones, siempre y cuando su PH tenga un TP equivalente.
- Cuanto más lejos aparezca el paso de peatones en la imagen, más próximas aparecerán las bandas del paso de peatones en la captura (menor periodo) y, por tanto, mayor es la frecuencia. El TP equivalente tendrá por ello una mayor frecuencia característica.
- A mayor distancia entre el paso de peatones y el vehículo, menor tamaño tendrá el paso de peatones en las imágenes y un menor número de píxeles representarán al objeto en las mismas. Como consecuencia de esto PH tendrá valores más bajos y menor será el $Modulo(k)$ correspondiente a su $f(k)$ característica. Es por este motivo por el que a la hora de analizar los resultados experimentales que dieron como resultado el desarrollo del algoritmo con $Modulo(5)$ no se apreciaron otros valores característicos de las imágenes en las que aparecía un paso de peatones, pues sus $Modulos(k)$ no tenían valores tan relevantes.

El algoritmo con $Modulo(5)$ tenía un problema importante: el rango de detección del paso de peatones en una secuencia de imágenes en que el vehículo se fuera acercando progresivamente a una marca vial de este tipo era de pocas capturas. Debido a esto, el riesgo de falso negativo era mayor, puesto que si el paso de peatones no quedaba bien definido en las imágenes en las cuales la distancia al paso de peatones daba como resultado un PH con un TP equivalente de $f(5)$, el algoritmo no lo detectaba, ya que no había otras frecuencias alternativas de detección. Un mayor rango de frecuencias consideradas a la hora de detectar el paso de peatones en las imágenes, podía dar

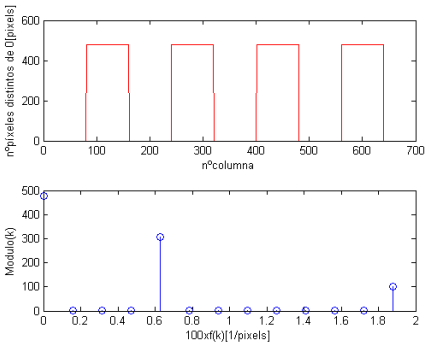
(a) Imagen limpia



(b) PH (arriba) y FFT de PH (abajo) de 4.15a



(c) Tren de pulsos ideal de frecuencia 6,25 $\times 10^{-3} pixels^{-1} = f(5)$



(d) PH (arriba) y FFT de PH (abajo) de 4.15c

Figura 4.15: Comparación de un verdadero positivo usando $f(5)$ con un tren de pulsos de frecuencia $6{,}25 \times 10^{-3} pixels^{-1} = f(5)$

por tanto como resultado un algoritmo más robusto.

A partir de la demostración de que $Modulo(5)$ era utilizable porque tenía un PH equivalente a un TP de $f(5)$ y de que esto era extrapolable a otras $f(k)$, se plantearon dos posibles mejoras al algoritmo desarrollado hasta ese momento, que podrían solucionar el problema anteriormente mencionado:

1. Rastrear el paso de peatones de frecuencias más altas a frecuencias más bajas. La idea de esta mejora consistía en buscar el paso de peatones a la $f(k)$ más alta posible. Una vez encontrado, se detectaría el paso de peatones a esta frecuencia hasta que se perdiera, momento en el cual se buscaría el objeto a $f(k-1)$. Este procedimiento se repetiría hasta llegar a la $f(k)$ de detección más baja posible.

2. Considerar un rango de frecuencias. Se utilizaría un conjunto de frecuencias concreto para la cuales se estarían analizando de forma con-

tinua sus respectivos módulos. Si alguno de estos módulos superara en cualquier momento un umbral se podría considerar que en la imagen existe un paso de peatones.

Para ambas aproximaciones el umbral debería ser variable puesto que, como se dedujo en la demostración, a mayores $f(k)$ características, menores $Modulo(k)$ asociados aparecerán, debido al menor tamaño del paso de peatones en las imágenes.

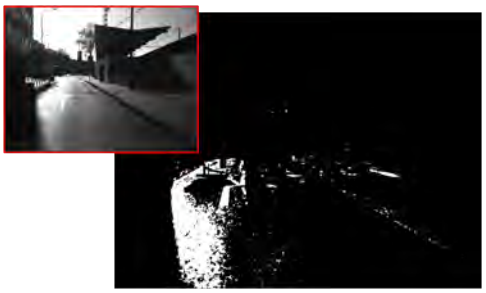
De los dos métodos planteados, finalmente se optó por desarrollar el segundo, debido a que el primero planteaba un problema fundamental: si para la $f(k)$ más alta no se localizaba el paso de peatones, éste no se localizaría ya para ninguna $f(k)$. Debido a este hecho, se concluyó que el primer método planteado no era una buena solución, pues si bien se superaría el rango de capturas en las que sería posible detectar el paso de peatones en una secuencia de imágenes, no se conseguiría un algoritmo más robusto.

Deducido lo anterior, se implementó el segundo método con los siguientes parámetros:

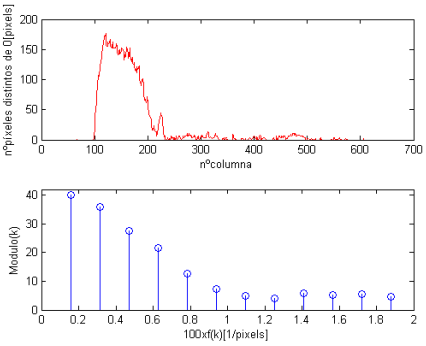
- El rango de frecuencias seleccionado son todas aquellas $f(k)$ con $k \in [4, 8]$. Para frecuencias más bajas de $f(4)$ no se encuentra un TP equivalente y para frecuencias mayores de $f(8)$ el $Modulo(k)$ asociado es demasiado bajo (el paso de peatones aparece lejos en las imágenes) y es difícil establecer un umbral que identifique correctamente el paso de peatones dando como resultado un algoritmo robusto.
- Los umbrales seleccionados para los módulos asociados a estas frecuencias se obtienen experimentalmente, analizando las secuencias de imágenes que se toman como muestra en esta investigación, estableciéndose en los siguientes valores:

$$\begin{aligned} f(4) &= 4{,}6875 \times 10^{-3} pixels^{-1} \Longrightarrow Modulo(4) \geq 20 \quad (4.11) \\ f(5) &= 6{,}25 \times 10^{-3} pixels^{-1} \Longrightarrow Modulo(5) \geq 17 \\ f(6) &= 7{,}8125 \times 10^{-3} pixels^{-1} \Longrightarrow Modulo(6) \geq 16{,}5 \\ f(7) &= 9{,}375 \times 10^{-3} pixels^{-1} \Longrightarrow Modulo(7) \geq 16 \\ f(8) &= 10{,}9375 \times 10^{-3} pixels^{-1} \Longrightarrow Modulo(8) \geq 15 \end{aligned}$$

El algoritmo implementado según lo visto hasta este momento, tiene sin embargo un cierto número de falsos positivos en imágenes del estilo de las mostradas en la figura 4.16. En 4.16a se aprecia como en la captura inicial hay un reflejo en la calzada que provoca un grupo de píxeles agrupados a lo largo de una franja de la imagen (I_{limpia}). En 4.16b se muestran su PH (gráfica superior) y $Modulo(k)$ VS $f(k)$ (gráfica inferior) correspondientes.

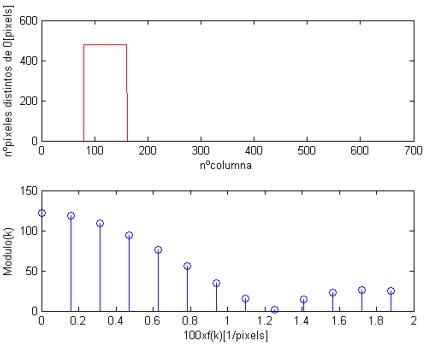


(a) Imagen original e imagen limpia correspon-diente



(b) PH (arriba) y FFT de PH (abajo) de 4.16a

(c) Señal tipo escalón



(d) PH (arriba) y FFT de PH (abajo) de 4.16c

Figura 4.16: Ejemplo de imagen que da un falso positivo si no se considera la restricción de máximo local

Se observa aquí como $Modulo(4) \simeq 29$. Como el umbral correspondiente a $f(4)$ es 20, se tendría un falso positivo en esta imagen puesto que no existe realmente un paso de peatones en esta captura. En 4.16d se demuestra como esta situación es equivalente a tener la FFT de un escalón(figura 4.16c). Este problema es fácil de solucionar si se tiene en cuenta la apreciación que se hizo al comienzo de esta sección, según la cual $Modulo(k)$ debía ser además máximo local de la función. Para tener en cuenta esta consideración se añade al algoritmo la condición de que además de superar el correspondiente umbral $Modulo(k-1) < Modulo(k) \bigcup Modulo(k) > Modulo(k+1)$, para lo cual se deben tener en cuenta también los valores de $Modulo(k)$ correspondientes a las frecuencias $f(3) = 3{,}125 \times 10^{-3} pixels^{-1}$ y $f(9) = 12{,}5 \times 10^{-}3 pixels^{-1}$ para poder así hacer comparaciones con $f(4)$ y $f(8)$, respectivamente.

Todas las consideraciones descritas en esta sección dan lugar a esta parte del algoritmo final cuyas funciones implementadas en MATLAB pueden verse en el apéndiceA.

4.1.8. FFT y reconocimiento: Posicionamiento y cálculo de distancia

Llegados a este punto, se tiene ya un algoritmo que es capaz de decidir de una manera eficaz si en las imágenes tomadas por las cámaras instaladas en el vehículo existe o no un paso de peatones. El problema ahora es que según el procedimiento que se ha llevado a cabo, el código es capaz de decidir si este tipo de marca vial aparece en las capturas o no, pero no puede posicionarlo dentro de la imagen y por tanto no puede darnos información de su distancia con respecto al vehículo. En esta sección se aborda este problema.

La filosofía utilizada para lograr posicionar el paso de peatones en las imágenes parte de una idea análoga a la que se utilizaba a la hora de calcular la proyección horizontal de la imagen (PH), pero en esta ocasión aplicada a las filas de la misma (veáse 4.1.6). Esto es: para cada fila de la imagen (I_{limpia}) se cuenta el número de píxeles de valor 1, los cuales representan aquellos píxeles potencialmente pertenecientes a marcas viales, y se coloca el valor obtenido en un vector (PV). Si se representa el valor de cada una de las posiciones de este vector frente al número de fila a partir del cual han sido calculadas, se obtiene una representación unidimensional de las filas de la imagen, que se denomina proyección vertical de la imagen.

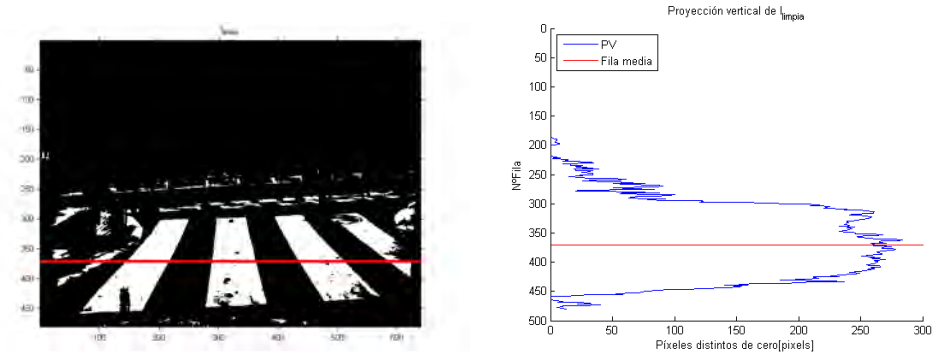
Si se aplica esta operación en las capturas en que un paso de peatones es detectado, se obtiene una gráfica similar a la que puede verse en la figura 4.17b. Como se observa en esta imagen, en las filas correspondientes al paso de peatones la proyección vertical tiene una forma similar a un escalón de ancho igual al del paso de peatones en la imagen y de alto un valor algo mayor de 200 píxeles. Lo que se pretende con esta función es calcular el punto medio de este escalón (línea representada en rojo en la gráfica), lo que corresponderá con la línea media del paso de peatones en la imagen (I_{limpia}).

Para realizar este cálculo, una vez obtenido PV, se procede de la siguiente forma:

1. Se recorre PV hasta encontrar un valor ($PV(fila)$) mayor de 90. Se ha comprobado experimentalmente que cualquier escalón que represente a un paso de peatones va a superar este valor en prácticamente la totalidad de su rango.

2. Una vez encontrado un $PV(fila) > 90$, se memoriza la fila (`fila_inicio`) y se cuenta el número de filas a partir de ésta en que se sigue cumpliendo la condición. Se termina de contar si se encuentra un $PV(fila) < 90$. El valor acumulado va a representar el ancho del escalón (`ancho`).

3. Si al seguir recorriendo PV se encuentra algún otro $PV(fila) > 90$, se



(a) Imagen mostrando una detección (b) Proyección vertical de la imagen

Figura 4.17: Proyección vertical de una imagen

procede de la manera anterior. Para que este nuevo valor de ancho sea considerado, debe superar el valor calculado anteriormente. En tal caso se almacenaría también la fila de inicio de este nuevo rango, sustituyendo a la almacenada previamente. Este procedimiento se lleva a cabo para impedir que picos puntuales de la proyección que no representen el paso de peatones, sino otras marcas viales, sean consideradas el escalón, pues se comprueba que el paso de peatones es la marca vial que cumple la condición $PV(fila) > 90$ durante un rango de filas mayor.

4. La posición del paso de peatones, representada por su fila media, se calcula como:

$$\frac{2 \times fila\_inicio + ancho}{2}$$

.

La función que implementa en MATLAB esta operación puede encontrarse en la sección A.1.5

Una vez determinada la posición del paso de peatones detectado en una imagen, el siguiente paso es determinar la distancia que existe entre éste y el vehículo. Para ello, y dado que se dispone de imágenes estereoscópicas donde el dato de disparidad de cada píxel es conocido gracias al mapa de disparidad, se puede aplicar la ecuación 3.8 para determinar la profundidad del paso de peatones detectado en la imagen (Z). Sin embargo, según Musleh et al. (2012a), la resolución de esta aproximación es bastante gruesa al depender únicamente del valor de disparidad, el cual está representado por valores enteros (ver figura 4.18). Como solución a este problema, en ese mismo trabajo se propone utilizar el perfil de la calzada.

Según se dijo en el apartado 3.7, el perfil de la calzada se muestra en el *v disparity* como una línea oblicua. Asumiendo la hipótesis de suelo plano

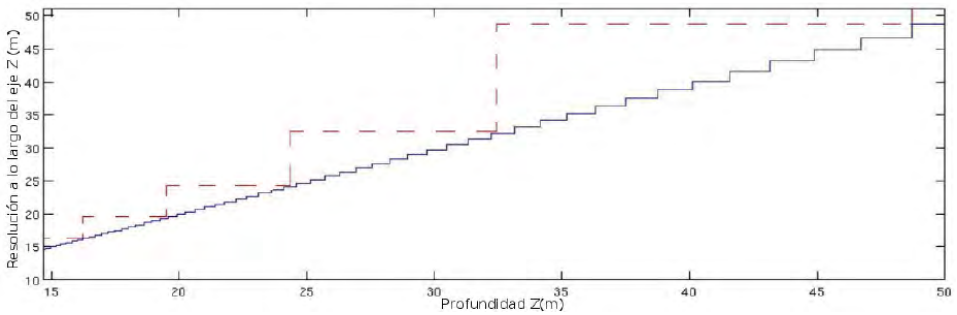


Figura 4.18: Resolución del cálculo de la profundidad usando el perfil de la calzada (línea azul) y usando la expresión 3.8 (línea roja)

delante del vehículo, esta línea se puede expresar con la ecuación de una recta (ecuación 4.12), donde m representa la pendiente de la recta, b la ordenada en el origen, la cual coincide con la línea del horizonte en la imagen, y v la coordenada vertical de la imagen. La variable dependiente d, representa el valor de disparidad.

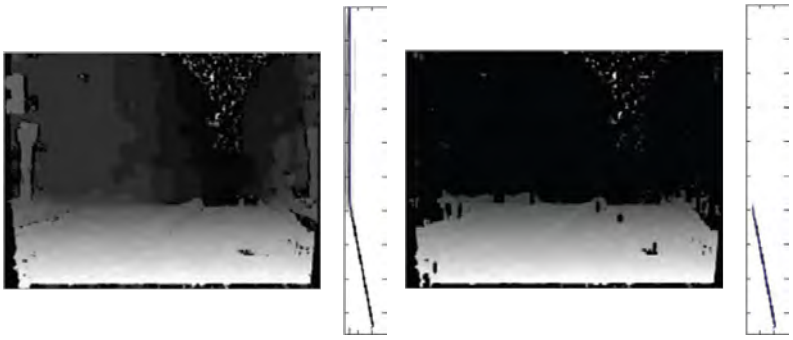
$$v = m \cdot d + b \tag{4.12}$$

Para obtener la ecuación de esta recta es posible aplicar la Transformada de Hough para rectas al *v disparity* asociado al mapa de disparidad. Este método obtiene como resultado la recta más votada o, lo que es lo mismo, la recta que contiene una mayor cantidad de los puntos existentes en el v disparity. Sin embargo, según el trabajo presentado en Musleh et al. (2012a), esto da lugar a errores cuando el perfil de la calzada no es la recta más votada debido a obstáculos de gran tamaño que aparecen delante de la cámara y que, como se dijo en la sección 3.7 aparecen como rectas verticales en el *v disparity*. Debido a esto, e inspirado en el trabajo citado anteriormente, en el presente trabajo no se utiliza el *v disparity* asociado al mapa de disparidad (ver figura 4.19), sino uno nuevo calculado a partir del mapa libre. Se elimina así el problema de que la recta más votada no sea el perfil de la calzada debido a los obstáculos.

Una vez calculada la ecuación de la recta, si se sustituye ésta en la expresión 3.8 se obtiene la ecuación 4.13:

$$\begin{aligned} Z &= \frac{f \cdot B}{d} \xrightarrow{v=m\cdot d+b} Z = \frac{f \cdot B \cdot m}{v - b} \\ f &= \text{Distancia focal} \\ B &= \text{Baseline, o separación entre los centros óptico de ambas cámaras} \end{aligned} \tag{4.13}$$

Sin embargo, como trata de representarse en la figura 4.20, la profundidad calculada con la expresión 4.13 representa la distancia entre el sistema estéreo y la calzada. Para calcular la distancia medida sobre la carretera, es



(a) Mapa de disparidad y v disparity de una imagen

(b) Mapa libre obtenido a partir de 4.19a y *v disparity* asociado

Figura 4.19: Comparación del v disparity obtenido del mapa de disparidad y del mapa libre

necesario utilizar el ángulo formado entre el sistema estéreo y la horizontal, denominado ángulo de cabeceo (α). El ángulo de cabeceo se puede obtener utilizando la ordenada en el origen de la recta del perfil de la calzada (b), la distancia focal (f) y la coordenada vertical del centro óptico del sistema estereoscópico (C_v) utilizando para ello la expresión 4.14.

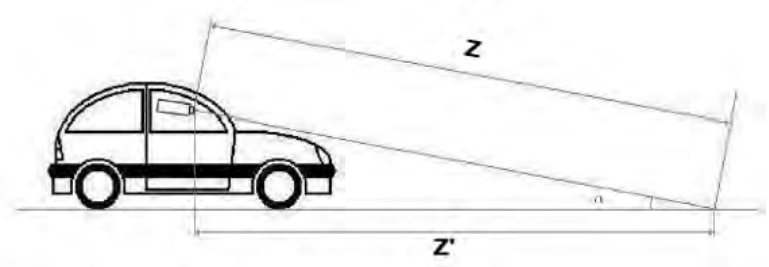


Figura 4.20: Distancia con respecto al sistema estéreo (Z) frente a la distancia con respecto al vehículo (Z')

$$\alpha = arctan(\frac{b - C_v}{f}) \tag{4.14}$$

Introduciendo este dato en la ecuación 4.13, se tiene que (ecuación 4.15):

$$Z' = Z \cdot cos(\alpha) = \frac{f \cdot B \cdot m}{v - b} \cdot cos(\alpha) \tag{4.15}$$

Al igual que en la sección 4.1.2 las funciones de cálculo de distancia no pertenecen a este proyecto, por lo que no aparecen reflejadas en el apéndice A.

4.2. Análisis de la cartografía y correción del error

En la primera parte de este capítulo se ha expuesto el desarrollo de un algoritmo de visión por computador capaz de detectar pasos de peatones en las imágenes y medir la distancia que existe entre estos y el vehículo. El objetivo de la segunda de las partes en que se ha subdividido el funcionamiento del sistema completo, sería utilizar la información obtenida para corregir el error de posicionamiento que pueda estar cometiendo un sistema de GPS, o de Odometría Visual (GPS/VO), a la hora de localizar el vehículo en el mapa. Además, ésta sería la parte del algoritmo completo encargada de llevar a cabo el control de flujo del sistema general.

Como se detalla en el diagrama de la figura 4.1, esta parte del algoritmo realizaría la tarea de comprobar de manera continua la distancia que existe entre la posición arrojada por el sistema GPS/VO y la posición en la cartografía del paso de peatones más próximo. Si esta distancia estuviera dentro de un rango de valores adecuado, lo cual significa que el sistema GPS/VO estima que el vehículo se encuentra lo suficientemente cerca del paso de peatones considerado, esta parte del algoritmo activaría el código correspondiente a la detección de pasos de peatones, desarrollado en la primera parte de este capítulo. El algoritmo de visión comenzaría entonces a realizar su tarea sobre las imágenes captadas, para determinar si en éstas aparece un paso de peatones. Si en algún momento antes de que el vehículo abandonase el rango de distancia establecido, se detectara una de estas marcas viales, se compararía el dato de distancia a la misma, calculado por el algoritmo de visión, con el dato de distancia arrojado por el sistema GPS/VO. Si la diferencia entre ambas medidas fuera lo suficientemente grande, el algoritmo re-posicionaría el vehículo en su localización correcta dentro de la cartografía. Se considera como posición correcta la determinada por la distancia calculada con el algoritmo de visión por computador.

Esta lógica de funcionamiento pretende conseguir que el algoritmo de visión no trabaje de forma continua, actuando sólo en caso necesario. Se consigue así no sobrecargar el sistema general cuando no es probable que se vaya a encontrar un paso de peatones en las proximidades del vehículo.

Como se expuso en el alcance del proyecto, debido a que el objetivo de este trabajo no es la implementación del desarrollo en un sistema real, la elaboración de un código relativo a esta parte no sirve para extraer ningún resultado útil sobre el funcionamiento del sistema, por lo que sólo se ha planteado la idea explicada en los párrafos anteriores, la cual se plantea desarrollar en futuros trabajos. Sin embargo, para poner a prueba las posibilidades que ofrece la cartografía de OpenStreetMaps, se ha desarrollado un

código en MATLAB capaz de extraer de una sección de mapa descargada de (OpenStreetMaps) la posición de los pasos de peatones existentes en la misma y calcular la distancia entre una posición simulada del vehículo y el más próximo de estos pasos de peatones, dentro de la carretera por la que el vehículo circula. El resultado obtenido puede encontrarse en el apéndice A.2.

La lógica de funcionamiento de estas funciones sería la siguiente:

- **Extracción de datos de la porción de mapa descargada de OpenStreetMaps**. El archivo descargado desde la web oficial del proyecto OpenStreetMaps (OSM) (OpenStreetMaps) es un archivo en XML que contiene datos sobre los nodos, caminos y relaciones de la sección de mapa descargada, así como de las etiquetas relacionadas con estos elementos (ver sección 3.11). Para poder trabajar con este archivo en MATLAB es necesario transferir los datos del archivo XML a este software. Para ello se usa una librería de funciones obtenidas de Ioannis et al. que cargan los datos del archivo XML a una estructura de MATLAB y representan el mapa en un plot. Las funciones de la librería usadas a tal fin son `parse_openstreetmaps`, para cargar los datos a la estructura de MATLAB, y `plot_way`, para representar el mapa. Esta librería de funciones ha sido analizada con detalle y una explicación detallada sobre su funcionamiento y posibilidades ha sido incluida en el apéndice B del presente proyecto. Es de especial de interés en este sentido el análisis sobre la organización de la estructura `osm_xml`, obtenida por la función `parse_openstreetmaps`, que se presenta en este apartado. Esta estructura será sobre la cual se realicen posteriormente las búsquedas de elementos del mapa.

- **Búsqueda y representación de los pasos de peatones presentes en la sección de mapa.** Como se explicó en el apartado 3.11, los nodos, caminos y relaciones que componen cualquier mapa de OSM, tienen etiquetas asociadas que aportan datos acerca de estos elementos. De esta manera, OSM puede representar a través de los puntos del mapa información como límites de velocidad de una carretera, posición de un semáforo, etc. Como también se explicó en el apartado 3.11, estas etiquetas están compuestas de dos partes la clave (*key, k*) y el valor (*value, v*).

  Los pasos de peatones en OSM están representados por nodos los cuales tiene una asociada una etiqueta con `k=highway` y `v=crossing`. Para buscar aquellos nodos de la estructura `osm_xml` que tienen asociada esta etiqueta, se realiza una búsqueda sobre la misma con la función `busqueda_CrossDetect` cuyo desarrollo se incluye dentro de este proyecto. Esta función se puede encontrar en el apartado A.2.1. También

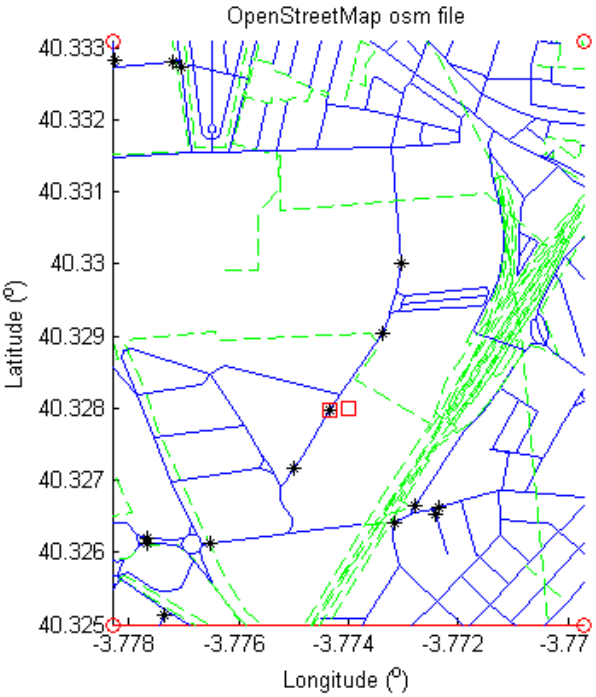


Figura 4.21: Aplicación de las funciones de cálculo de distancia del vehículo a pasos de peatones sobre la cartografía de OSM.
Las líneas verdes y azules representan los caminos del mapa.
El cuadrado rojo aislado representa la posición simulada para el vehículo.
Los asteriscos negros son las posiciones de los pasos de peatones.
El asterisco negro rodeado de un cuadrado rojo, representa el paso de peatones más próximo al vehículo.

con esta función, se representan en el mapa los nodos encontrados asociados a un paso de peatones.

- **Búsqueda del paso de peatones más cercano a la posición del vehículo, dentro de la carretera por la que circula éste**. Estas operaciones son realizadas por la función `distanciaOSM_CrossDetect`, que se puede encontrarse en el apartado A.2.2. El desarrollo de esta función también se incluye dentro del presente trabajo. Como se explicó en el apartado 3.11, los caminos (*way*) en OSM están compuestos por nodos cuya unión representa las carreteras y calles dentro de la cartografía. La función `distanciaOSM_CrossDetect` realiza la búsqueda del nodo más próximo a una posición simulada del vehículo (latitud, longitud), tomada como dato de entrada a la función, y determina a qué camino pertenece ese nodo. Se considera que este camino es por el

que circula el vehículo simulado. A continuación, determina si dentro de este camino existe algún paso de peatones, utilizando la información obtenida de `busqueda_CrossDetect`, y calcula la distancia al más próximo, en caso de existir más de uno.

Para calcular la distancia entre dos nodos, se usa la **Fórmula de Vincenty** (Vincenty), que es un método iterativo muy preciso para calcular la distancia entre dos puntos de un elipsoide geodésico, a partir de la longitud y latitud de los puntos sobre este elipsoide. La implementación del algoritmo que calcula esta fórmula, que puede encontrarse puede encontrarse citada ampliamente en la bibliografía (CPAN, Thaddeus Vincenty distance calculations), no ha sido desarrollada como parte de este proyecto. Se ha utilizado para ello una función ya implementada en MATLAB, que se puede encontrar en Kleder, M. (2014).

A través de la función `distanciaOSM_crossDetect` también se representa la posición simulada del vehículo y la posición del paso de peatones más cercano.

En la figura 4.21, se puede observar un ejemplo de aplicación del proceso descrito anteriormente sobre un mapa real.

Capítulo 5

Análisis de Resultados

Una vez planteada la solución para dar respuesta al alcance de este proyecto, en el presente capítulo se analizan los resultados obtenidos a partir del desarrollo expuesto.

Debido a que la parte fundamental de la solución es aquella relacionada con la detección del paso de peatones en las imágenes, en este apartado se pretende presentar y evaluar los resultados arrojados por los algoritmos desarrollados a tal fin.

Para ello se parte de **tres secuencias de imágenes en escala de grises** que sirven como dato de entrada al algoritmo. Estas imágenes han sido tomadas en entornos abiertos reales, por el sistema estereoscópico de cámaras de la compañía Pointgrey, Bumblebee (Pointgrey, 2014), instalado éste en el vehículo IvvI de la Universidad Carlos III de Madrid (ver apartado 1.1).

Estas secuencias están compuestas por un total de **1499 pares de imágenes** (obtenidas de las cámaras derecha e izquierda del sistema estereoscópico) con una **resolución de 640×480**. Este dato es importante porque el algoritmo desarrollado no es invariante a los cambios de escala; un cambio en el tamaño de las imágenes haría necesario una recalibración de los parámetros para garantizar un comportamiento adecuado. Este hecho se debe a que, como se dijo en la sección 4.1.1, se considera el ancho máximo de las marcas viales en la imagen ($S_M(v)$) durante la etapa de filtrado de la misma, parámetro que depende de la resolución de las capturas. Otra consecuencia, más importante si cabe, de tomar imágenes de un tamaño distinto a las utilizadas en este proyecto sería la variación de las frecuencias, y módulos asociados a éstas, a partir de las que se clasifica el paso de peatones (ver sección 4.1.7). Estos parámetros también cambiarían al modificar el tamaño de la imagen, puesto que también dependen del ancho con el que aparecen las bandas del paso de peatones en las capturas.

También se toman como dato de entrada al algoritmo los respectivos mapas de disparidad asociados a cada par de imágenes estereoscópicas que, como se decía en otras partes de este proyecto (ver sección 4.1.2), no son calculados por el código en MATLAB implementado para este trabajo.

Como se explicó en la sección 4.1.7, durante el desarrollo del algoritmo se implementó en primer lugar un código, que consideraba únicamente el valor del quinto módulo de frecuencia ($Modulo(5)$) de la FFT para determinar si en las imágenes aparecía un paso de peatones. Posteriormente, éste código fue mejorado para considerar los módulos que van desde el cuarto al octavo ($[Modulo(4), Modulo(8)]$), introduciendo además una restricción de máximo local que debían cumplir estos valores para decidir una detección. Este hecho permitió aumentar el rango de imágenes en que el paso de peatones era detectado dentro de una secuencia.

En este capítulo, se hará un análisis comparativo entre estas **dos versiones del código**, desde el punto de vista del rango de imágenes en que cada uno de ellos es capaz de detectar el paso de peatones que aparece en una secuencia, y desde el punto de vista de los ratios de acierto de cada uno. Además, para cada una de las versiones, se comprobarán estos ratios de acierto **en dos casos:** utilizando durante el proceso de filtrado de la imagen el mapa de disparidad y el *u disparity* asociado (como se hace en la versión final del algoritmo), y prescindiendo de estos datos. En este sentido, se pretenden determinar la necesidad de uso de la "limpieza", que se realiza utilizando el *mapa libre de disparidad* (ver sección 4.1.2), sobre la imagen binaria procedente del algoritmo 43rd PLT (ver apartado 4.1.1).

Para evaluar los ratios de acierto se utilizarán **matrices de confusión** (Recursos web:de la Escalera y Armingol, 2010), que son tablas en cuyas columnas aparecen las decisiones del algoritmo, mientras que sus filas contienen los resultados reales que debería haber obtenido éste. De esta forma es posible analizar los dos tipos de errores y las dos clases de aciertos que pueden producirse durante la clasificación de objetos en un algoritmo de visión: por una parte los aciertos, verdaderos positivos y verdaderos negativos, y por otra los errores, falsos negativos y falsos positivos. A este respecto, es necesario mencionar qué se entiende por esta clase de errores y aciertos en el presente trabajo.

Tradicionalmente, si en una imagen debiera detectarse un paso de peatones y el algoritmo decidiese que en la imagen no aparece una de estas marcas viales, se consideraría ese error un falso negativo, porque el objeto

Figura 5.1: Ejemplo de paso de peatones demasiado lejos para ser detectado por el algoritmo

aparece en la imagen pero el algoritmo no es capaz de detectarlo. En este trabajo, por el contrario, en multitud de ocasiones el paso de peatones aparece en las imágenes y el algoritmo no lo detecta, ni se pretende que sea así, debido a que la distancia entre el vehículo y el paso de peatones es demasiado alta para el algoritmo diseñado (ver figura 5.1). El objetivo del algoritmo de detección de pasos de peatones en este trabajo sólo es detectar estas marcas viales antes de que desaparezcan del campo de visión de las cámaras y, por este motivo, para no incrementar el número de falsos negativos con los errores que aportaría la aplicación directa de la definición, los cuales no aportarían ningún valor en el análisis de los resultados; **en este trabajo se considera que se ha producido un falso negativo únicamente si no se produce ninguna detección en toda la secuencia de imágenes en que aparece un mismo paso de peatones que se encuentra en el sentido de circulación del vehículo.** En tal caso, todas las imágenes computadas en ese rango de imágenes se considerarían falsos negativos. En caso contrario, si se produce una detección en este rango, se considera que la imagen correspondiente es un verdadero positivo. Por otro lado, si fuera de este rango no se produce una detección en alguna imagen, se computa este acierto como un verdadero negativo y, por el contrario, si sí se produce, se considera un falso positivo. También se considerarán falsos

positivos, aquellas imágenes en que se produce una detección, pero el paso de peatones se encuentra demasiado lejos del vehículo como para que realmente esta detección se deba a una correcta clasificación por parte del algoritmo.

Como se decía anteriormente, con el fin de comparar el rango de detección de las dos versiones del algoritmo, para cada paso de los identificados en la secuencias de imágenes (**18 en total en las tres secuencias**), se analizará también el número secuencial de capturas en que se producen verdaderos positivos en relación a ese paso de peatones.

A pesar de lo descrito anteriormente, ocurre en ocasiones que, a pesar de existir un paso peatones a una distancia adecuada, no en todas las imágenes de la secuencia se detecta esta marca vial, es decir, existen inconsistencias que producen que en la secuencia de imágenes en que debería haber únicamente verdaderos positivos, haya saltos de detección. Este tipo de errores se considerarán también falsos negativos, sin embargo, conocer la frecuencia con que ocurren estas inconsistencias se considera también de importancia para la evaluación de los resultados. Para determinar la importancia de estos errores, en la comparación de los rangos de detección de las dos versiones del algoritmo, se añadirá como dato adicional del análisis el número de inconsistencias en el rango de imágenes en que aparece un mismo paso de peatones.

5.1. Comparación de los rangos de detección de las dos versiones del algoritmo

En la tabla 5.1 se muestran los rangos de detección de los 18 pasos de peatones que aparecen en total en las tres secuencias de imágenes utilizadas, para cada una de las dos versiones del algoritmo. Se muestra también el número de inconsistencias del rango. La versión 1 se corresponde con el algoritmo que utiliza únicamente el *Modulo*(5), mientras que la versión dos es la que utiliza los módulos del 4 al 8.

A partir de los datos de la tabla 5.1 se pueden extraer las siguientes conclusiones:

- El paso de peatones con ID 16 no es detectado por ninguno de los algoritmos. Las imágenes que componen la secuencia de capturas en que aparece, serán consideradas en todos los casos falsos positivos. En la figura 5.2 se muestra un ejemplo de este tipo de imágenes. Como se observa en ella, la luz del Sol impacta directamente sobre la calzada y deslumbra a la cámara. Aunque en las secuencias no se dispone de más ejemplos de pasos de peatones sometidos a este tipo de circuns-

	Versión 1		Versión 2	
ID	Rango	Inconsistencias	Rango	Inconsistencias
1	11	0	23	0
2	5	0	10	2
3	18	2	22	1
4	7	0	8	0
5	10	0	12	0
6	10	0	10	0
7	12	0	19	0
8	1	0	10	1
9	0	0	6	0
10	9	6	12	5
11	7	0	9	0
12	12	1	16	0
13	11	0	16	0
14	9	0	12	0
15	82	0	91	0
16	0	0	0	0
17	10	4	11	0
18	18	0	25	0

Tabla 5.1: Rango de imágenes de detección de los pasos de peatones para las dos versiones del algoritmo

tancias, se piensa que podría suponer una debilidad importante para el algoritmo final, que no ha podido ser solucionada.

- El paso de peatones con ID 9 no es detectado por la versión 1 del algoritmo, pero sí por la 2. Este hecho demuestra la robustez que proporciona al algoritmo el tener varias frecuencias como opción de detección, en lugar de sólo $f(5)$. Una imagen de la secuencia en que aparece este paso de peatones puede encontrarse en la figura 5.3. En ella se puede apreciar como el paso de peatones aparece inclinado con respecto a la vertical, en lugar de perpendicularmente al vehículo, lo que da como resultado una proyección horizontal que no se asemeja a un tren de pulsos, como es condición para la detección (ver sección 4.1.7). El hecho de que el algoritmo tenga dificultades para detectar pasos de peatones con un cierto angulo, podría suponer un inconveniente, como evidencia este error, sin embargo, se considera que el vehículo se colocará lo suficientemente perpendicular al paso de peatones en algún momento de su paso a través de él y que el rango de frecuencias utilizado será suficiente para lograr la detección de la marca vial en alguna de las imágenes tomadas. Esta hipótesis se considera suficiente para cubrir satisfactoriamente las necesidades de la aplicación objetivo de este trabajo.

- Los pasos de peatones con ID 15 y 18 son detectados en un rango

(a) Imagen original

(b) Imagen tras la aplicación de la etapa de filtrado

Figura 5.2: Imagen del paso de peatones ID 16

de imágenes mucho más amplio que en el resto de casos. Se observa aún mejor en el caso de ID 15. Esto se debe a que en la secuencia de imágenes, el vehículo está parado durante un tiempo frente al paso de peatones debido al cruce de viandantes a través del mismo. Se puede observar sin embargo, que las inconsistencias son, en ambos casos y para ambas versiones del algoritmo, nulas. Este hecho demuestra la robustez del algoritmo desarrollado ante oclusiones parciales del paso de peatones en las imágenes, pues el algoritmo no deja de detectar el mismo a pesar de existir peatones tapando parte de la marca vial.

- Por separado, los valores de rango de detección de ambas versiones del algoritmo no aportan demasiada información, ya que el número de imágenes en el que va a aparecer el mismo depende de factores como la velocidad del vehículo al pasar por el paso de peatones, de si lo hace de manera continua o se detiene ante él, etc. Si es interesante sin embargo, ver las diferencias entre ambos valores. Para realizar una comparación se puede calcular la diferencia de la media de rangos (ecuación 5.1):

$$\begin{aligned} Media_{version\,1} &= \frac{(11-0)+(5-0)+...+(18-0)}{18} = 13\,imagenes \\ Media_{version\,2} &= \frac{(23-0)+((10-2)+...+(25-0))}{18} = 17\,imagenes \\ Diferencia\,medias &= 17-13 = 4\,imagenes \end{aligned} \tag{5.1}$$

  El algoritmo de la versión 2, mejora por tanto el rango de detección de la versión 1, lo que fue el objetivo de su implementación.

- Por otro lado, se puede apreciar que el algoritmo de la versión 2 arroja resultados mucho más consistentes que el de la versión 1. Para analizar

(a) Imagen original (b) Imagen tras la aplicación de la etapa de filtrado (c) Proyección horizontal de la imagen filtrada

Figura 5.3: Imagen del paso de peatones ID 9

este hecho, se calcula el porcentaje de inconsistencias de cada uno (ecuación 5.2):

$$\begin{aligned} \%inconsistencias(version_1) &= \frac{0+2+0+...+0}{5+18+7+...+18} \times 100 = 5{,}6\,\% \\ \%inconsistencias(version_2) &= \frac{2+1+...+0}{23+...+25} \times 100 = 2{,}88\,\% \end{aligned} \quad (5.2)$$

Se puede observar que la versión 2 del algoritmo es más consistente que la versión 1.

Del análisis realizado en este apartado, se puede concluir que el algoritmo de la versión 2 supera los resultados obtenidos por el de la versión 1, en cuanto al rango de detección, resultando además más consistente, por lo que la mejora realizada se considera de utilidad para una correcta detección de los paso de peatones en las imágenes.

5.2. Ratios de acierto de los algoritmos con y sin aplicación del mapa libre de disparidad

En la tabla 5.2 se muestran las matrices de confusión para las dos versiones del algoritmo en el caso de aplicar y en el caso de no aplicar la operación de filtrado con el mapa libre de disparidad (ver sección 4.1.2).

En la tabla 5.3 se muestran además los porcentajes de acierto y error obtenidos a partir de estos datos. En el porcentaje de errores, se distingue también entre la proporción debida a falsos negativos (FN) y la debida a falsos positivos (FP), sobre el total de imágenes.

De estos resultados se pueden obtener las siguientes conclusiones:

- Los algoritmos que no utilizan el mapa libre de disparidad tienen un mayor número de verdaderos positivos que sus correspondientes que sí hacen uso del mismo. Estas versiones tienen sin embargo una proporción de falsos positivos más alta que aquellos que sí utilizan el mapa libre. Se plantea que esto pueda ser debido a dos motivos:

		Versión 1 usando mapa libre de disparidad	
		Clase asignada	
		Paso de peatones	No paso de peatones
Clase real	Paso de peatones	219	88
	No paso de peatones	4	4186
		Versión 1 sin usar el mapa libre de disparidad	
		Clase asignada	
		Paso de peatones	No paso de peatones
Clase real	Paso de peatones	280	24
	No paso de peatones	14	4179
		Versión 2 usando mapa libre de disparidad	
		Clase asignada	
		Paso de peatones	No paso de peatones
Clase real	Paso de peatones	303	26
	No paso de peatones	0	4168
		Versión 2 sin usar el mapa libre de disparidad	
		Clase asignada	
		Paso de peatones	No paso de peatones
Clase real	Paso de peatones	390	25
	No paso de peatones	26	4056

Tabla 5.2: Matrices de confusión de los 4 casos

	Versión 1 + mapa libre	Versión 1 sin mapa libre	Versión 2 + mapa libre	Versión 2 sin mapa libre
% Acierto	97.95 %	99.15 %	99.42 %	98.87 %
% Error	2.05 %	0.85 %	0.58 %	1.13 %
% FP	0.09 %	0.31 %	0 %	0.58 %
% FN	1.96 %	0.53 %	0.58 %	0.56 %

Tabla 5.3: Porcentajes de acierto y error para cada versión del algoritmo propuesto

1. El incremento del número de píxeles clasificados como marca vial en la imagen binaria sobre la que se calcula la FFT, debido al hecho de no eliminar de ella aquellos elementos no pertenecientes a la calzada utilizando el mapa libre. Esto introduce frecuencias adicionales en la proyección horizontal de la imagen que se acoplan con las frecuencias que son tomadas como características del paso de peatones en las capturas.
2. Este incremento supone un aumento general de píxeles en la imagen binaria (tanto de aquellos pertenecientes a marcas viales, como de aquellos que no están sobre la calzada) produciendo un incremento general en los módulos de FFT. La solución a este problema pasaría por incrementar los umbrales de detección.

Sobre estas dos hipótesis se han hecho pruebas preliminares, que pa-

recen apuntar a que una modificación de los umbrales no solucionaría estos problemas de detección sin disminuir también drásticamente el número de verdaderos positivos, haciendo incluso inviable el uso de algunas de las frecuencias características utilizadas en la detección. Este problema debería ser estudiado más a fondo, pero todo apunta a que es clave el hecho de eliminar aquellos píxeles no pertenecientes a la calzada si se quiere obtener un algoritmo robusto de detección de pasos de peatones basado en sus frecuencias características.

- El 0.09 % de falsos positivos obtenidos para la primera versión del algoritmo usando el mapa libre de disparidad, se debe a no considerar la restricción de máximo local (ver sección 4.1.7). Si se aplica esta restricción se elimina ese porcentaje de errores. No ocurre lo mismo con su versión correspondiente que no usa el mapa libre.

- La versión 2 del algoritmo que utiliza además el mapa libre de disparidad durante el proceso de segmentación, es la más robusta de las cuatro comparadas, con un 99,42 % de aciertos y sólo un 0,58 % de errores, de los cuales la proporción de falsos positivos es nula. Este hecho es importante, pues de cara a la aplicación objetivo de este proyecto los falsos positivos deben ser reducidos al mínimo ya que, en caso de producirse estos, podrían aumentar el error de posicionamiento del vehículo en lugar de reducirlo.

Se concluye por tanto, que el algoritmo que combina mejor los parámetros de rango de detección, consistencia y robustez, especialmente ante falsos positivos, es la versión 2 del algoritmo propuesto que hace uso del mapa libre de disparidad. Las funciones de MATLAB correspondientes a este algoritmo se incluyen en el apéndice A.

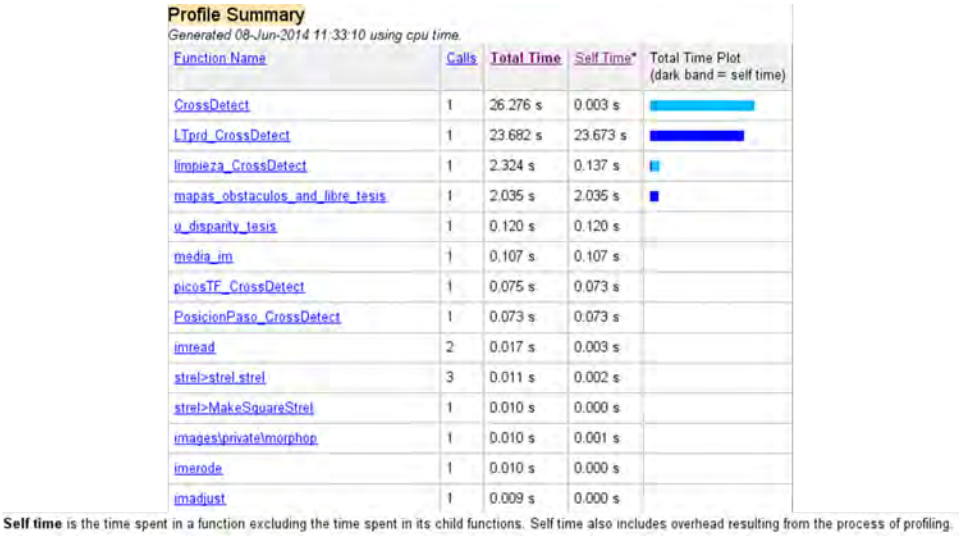
En cuanto al uso del mapa libre, no se obtiene una conclusión determinante acerca de la necesidad de su utilización, aunque parece probable que sea necesario eliminar los elementos pertenecientes a la calzada para obtener una correcta detección con los datos obtenidos de FFT.

Capítulo 6

Conclusiones y Trabajos Futuros

Partiendo del contexto de los vehículos inteligentes y con el objetivo de desarrollar un sistema de posicionamiento robusto para estos, durante el presente trabajo se ha expuesto el desarrollo de un algoritmo capaz de identificar y posicionar con respecto al vehículo pasos de peatones, los cuales se pretende que sean utilizados como puntos de referencia dentro del entorno. El objetivo final es que esta información sirva de apoyo a un sistema basado en GPS y/o Odometría Visual (GPS/VO), para lo cual se han dado los primeros pasos basándose en la cartografía de OpenStreetMaps.

Como se ha visto en el capítulo 4 el algoritmo desarrollado para cubrir el alcance del proyecto, cuyas funciones completas en MATLAB pueden encontrarse en el apéndice A, utiliza un filtro basado en la mediana y el mapa libre de disparidad para así obtener de la imagen aquellos píxeles pertenecientes a marcas viales, el módulo de la Transformada Rápida de Fourier de la proyección horizontal para la clasificación y la proyección vertical, junto con el perfil de la calzada y la información del mapa de disparidad, para el posicionamiento y cálculo de la distancia a los pasos de peatones detectados. Partiendo de estas ideas se ha desarrollado un algoritmo del que se demuestra que, para las imágenes utilizadas, tiene un porcentaje de acierto del 99,42 %, siendo el porcentaje de fallos debido a falsos positivos nulo (ver capítulo 5). Este dato se considera de importancia de cara a la implementación en un sistema real, para evitar fallos en el sistema de posicionamiento. En cuanto al resto de errores, correspondientes a los falsos negativos, los más representativos son aquellos en los que la luz incide directamente en el paso de peatones, provocando reflejos en la pintura de los mismos. Se propone estudiar este caso en futuros trabajos de cara a mejorar el comportamiento del algoritmo ante estos tipos de iluminación. En esta línea de acción, también se propone comprobar el comportamiento del algoritmo ante otras condicio-



Profile Summary

Generated 08-Jun-2014 11:33:10 using cpu time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
CrossDetect	1	26.276 s	0.003 s	
LTprd_CrossDetect	1	23.682 s	23.673 s	
limpieza_CrossDetect	1	2.324 s	0.137 s	
mapas_obstaculos_and_libre_tesis	1	2.035 s	2.035 s	
u_disparity_tesis	1	0.120 s	0.120 s	
media_im	1	0.107 s	0.107 s	
picosTF_CrossDetect	1	0.075 s	0.073 s	
PosicionPaso_CrossDetect	1	0.073 s	0.073 s	
imread	2	0.017 s	0.003 s	
strel>strel.strel	3	0.011 s	0.002 s	
strel>MakeSquareStrel	1	0.010 s	0.000 s	
images\private\morphop	1	0.010 s	0.001 s	
imerode	1	0.010 s	0.000 s	
imadjust	1	0.009 s	0.000 s	

Self time is the time spent in a function excluding the time spent in its child functions. Self time also includes overhead resulting from the process of profiling.

Figura 6.1: Tabla de tiempos calculada con MATLAB para el procesamiento de una imagen utilizando el algoritmo desarrollado en este trabajo

nes de iluminación, como en imágenes tomadas de noche o con lluvia, ya que todas las imágenes utilizadas en el presente trabajo han sido capturadas de día y con la iluminación correspondiente a un día despejado. Se destaca también la robustez del algoritmo ante oclusiones parciales del paso de peatones, como se deduce de las imágenes en que existen peatones cruzando a través del mismo, mejorando otros trabajos en el ámbito del reconocimiento visual de pasos de peatones (Se y Brady, 2003).

Para trabajos futuros que tomen de base este proyecto, también se propone estudiar más en detalle la necesidad de utilización del mapa libre de disparidad en el filtrado, ya que los resultados obtenidos no han sido concluyentes en este sentido. Además se propone incluir las ideas expuestas en Perreault y Hébert (2007) para optimizar el cálculo de la mediana, ya que se ha visto que actualmente es la operación más costosa del algoritmo, muy alejada de las restricciones de tiempo necesarias para la implementación del concepto desarrollado en un vehículo real (ver figura 6.1[1]). En esta línea de mejora de tiempos, también se propone la implementación en una GPU, de tal manera que sea posible la paralelización de procesos. Se plantea además el trabajar con matrices dispersas en aquellas partes del algoritmo en que se tiene una imagen con la mayor parte de los elementos nulos.

[1] Ordenador Utilizado: Procesador Intel Core 2 Duo a 2,67GHz, 8GB de RAM, Sistema Operativo Windows 7 de 64bits

También se propone de cara a futuros trabajos la utilización de un método de aprendizaje, como Support Vector Machines (SVM) o Redes Neuronales, para realizar el proceso de clasificación de las marcas viales, tal y como se ha visto en otros trabajos presentes en la bibliografía (Sichelschmidt et al., 2010), en lugar de establecer los criterios de clasificación de forma manual como se ha hecho en el desarrollo planteado. De esta manera, se pretende que sea más sencilla la modificación de parámetros tales como la resolución de las imágenes.

Por último, también de cara a futuras versiones del sistema, se contempla la posibilidad de utilización de otros elementos del entorno para la mejora del posicionamiento, tal y como se ha pretendido hacer con los pasos de peatones en este trabajo. Así, en esta línea de acción, se podrían utilizar semáforos, señales verticales, etc. cuya posición también está modelada en OpenStreetMaps. Asimismo, en trabajos futuros se tratará de implementar el sistema completo desarrollado en este trabajo con el fin de poner a prueba la parte del mismo sobre la correción de errores del sistema GPS/VO planteada en la segunda parte del desarrollo.

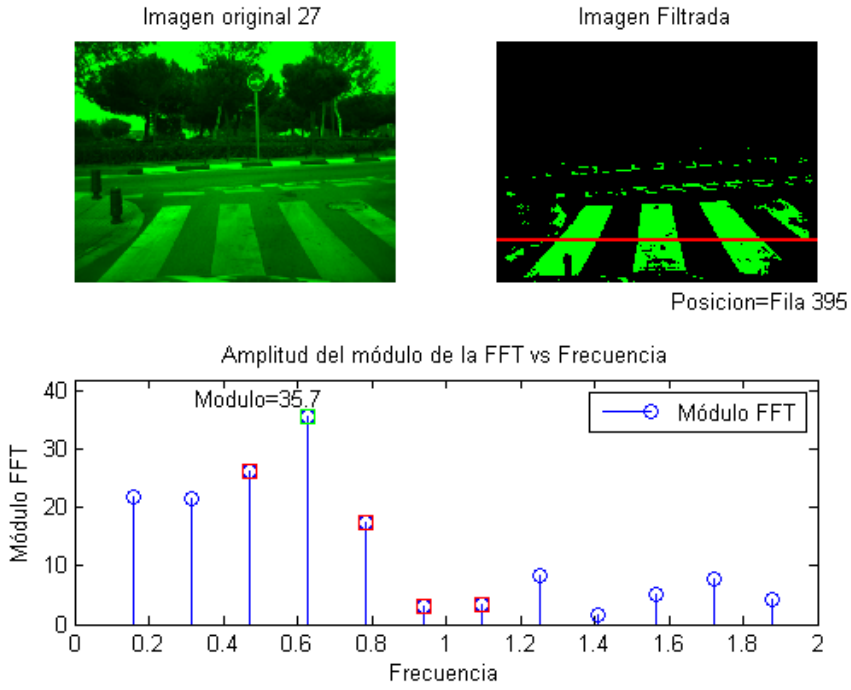


Figura 6.2: Ejemplo de resultado del algoritmo de reconocimiento de pasos de peatones

En conclusión en el presente trabajo se han dado los primeros para desarrollar un sistema de posicionamiento eficaz basado en información visual, que puede suponer una ventaja frente a otros algoritmos de map-matching por su sencillez, al establecer la idea de utilizar sólo ciertos hitos a la hora de determinar la correlación entre las imágenes y la cartografía.

Capítulo 7

Costes del Proyecto

En este capítulo, se pretende estimar los costes asociados al desarrollo de este proyecto. Para ello, por un lado, se hará un análisis de los tiempos dedicados a su elaboración, con el fin de valorar el coste asociado a recursos humanos. Por otro lado, se estimará el coste de los recursos materiales utilizados.

Los tiempos estimados para la elaboración del presente trabajo[1], se muestran en la tabla 7.1:

Tarea	**Tiempo**
Comprensión y estudio del problema (incluye recopilación de bibliografía)	80h
Familiarización con las herramientas utilizadas(MATLAB, OSM y LaTeX)	30h
Desarrollo del algoritmo: implementación, mejoras y análisis de resultados	290h
Planteamiento teórico del sistema completo	10h
Redacción de la memoria	220h
TOTAL	**640h**

Tabla 7.1: Tiempos asignados a cada una de las tareas realizadas

Teniendo en cuenta los tiempos anteriores y asumiendo que el sueldo de un ingeniero junior es de 21000€/año, distribuidos en 14 pagas, con una jornada de 8h/dia, se tiene (ecuación 7.1):

$$\text{Gasto Ingeniería/h} = \frac{21000\,€/\text{año}}{14\,pagas/\text{año}} = \frac{1500\,€/mes}{30\,dias/mes} = \frac{50\,€/dia}{8\,h/dia} = 6{,}25\,€/h \quad (7.1)$$

Teniendo en cuenta el total de horas y el precio de cada hora, el coste total de ingeniería sería (ecuación 7.2)

$$\text{Gasto Ingeniería}_{Total} = 6{,}25€/h \times 640h = 4160€ \quad (7.2)$$

[1]Para calcular estos tiempos se ha considerado aproximadamente una dedicación de media jornada, durante ocho meses y de lunes a viernes

Por otro lado, lo gastos asociados a material se muestran en la tabla 7.2:

Concepto	**Coste**
Ordenador portátil	700€
Gastos de mantenimiento ordenador	100€
Licencia de MATLAB	2000€
Sistema de cámaras estéreo (modelo *Bumblebee* de *Pointgrey*	3000€
Otros: Desplazamientos, consumo eléctrico, etc.	100€
TOTAL	5900€

Tabla 7.2: Gastos asociados a material

De acuerdo con el resultado de la ecuación 7.2, asociado al coste de recursos humanos, y el total mostrado en la tabla 7.2, asociado al coste de material, se tiene que los costes totales asociados al proyecto son (ecuación 7.3):

$$\text{Coste}_{Total} = 4160 + 5900 = 10060€ \tag{7.3}$$

Apéndice A

Funciones implementadas en MATLAB

En este apéndice se muestran las distintas funciones implementadas en MATLAB, las cuales componen el algoritmo objeto de este proyecto, así como otras desarrolladas durante el mismo pero que no se incluyen en su versión final. También se mencionan algunas funciones propias de MATLAB que a lo largo de esta investigación han resultado útiles en el tratamiento de imágenes y que pueden resultar de interés para trabajos futuros.

Como en el capítulo 4, las funciones principales están clasificadas en dos secciones, correspondientes a las dos partes en que fue subdividido el algoritmo. Todas las operaciones que aparecen en estos códigos están descritas con detalle en dicho capítulo. Además, todas las funciones aparecen comentadas (líneas marcadas con %).

En una tercera sección se especifican otras funciones que han sido utilizadas en el desarrollo del algoritmo pero que, o bien son propiedad de terceros, o bien no se han utilizado en el algoritmo final. Sólo algunas de ellas se encuentran explicadas con detalle en el capítuloñefcap4. En este caso únicamente las de desarrollo propio se encuentran comentadas.

A.1. Funciones del bloque de "Detección y posicionamiento del paso de peatones"

A.1.1. `CrossDetect`

CrossDetect es la función principal del algoritmo de visión, la cual realiza la secuencia lógica de éste llamando a las demás funciones cuando es

necesario y gestionando sus resultados. Su código en MATLAB se muestra a continuación:

```
%ENTRADAS: imagen-> Especifica la ruta del archivo de imagen a tratar
%con extensión. Debe ser una imagen en escala de grises. Probada con capturas
%de 480x640. Para otras no se garantiza un correcto funcionamiento.
%SALIDAS: deteccion-> Devuelve el string 'DETECTADO' si en imagen se encuentra
%un paso de cebra y 'NEGATIVO' en caso contrario.
%distancia-> Devuelve Inf si deteccion='NEGATIVO' y un entero indicando
%la distancia al paso de cebra en caso contrario.

function [deteccion,posicion]=CrossDetect(imagen)

%%%INICIALIZACIONES%%%
contraste=0;
posicion=Inf;  %Si no se detecta paso de cebra en la imagen, la distancia es
%considerada infinita.
%%%%%%%%%%%%%%%%%%%%%%

%%%CÓDIGO%%%

Ig=imread(imagen); %Lectura del archivo de imagen

%Cálculo de la diferencia entre la media de nivel
%de gris de la mitad superior de la imagen y la
%media de nivel de gris de la imagen total.
dif_medias=media_im(Ioriginal);


if dif_medias>40         %Establece que se debe aplicar mejora de contraste
                         %con imadjust si la diferencia de medias calculada
                         %es mayor de 40-> Ver función LTprd_CrossDetect.
    contraste=1;
end

%Filtrado de Ioriginal con el algoritmo 43rd PLT para extraer píxeles
%pertenecientes a marcas viales. %Aplica una mejora de contraste
%en la parte inferior de la imagen si contraste=1.
Iu=LTprd_CrossDetect(Ioriginal,43,20,contraste);


%Uso del mapa libre de disparidad para eliminar píxeles no pertenecientes
%a la calzada. Devuelve también el mapa de disparidad, la distancia focal y
%la distancia entre cámaras, necesarios para el cálculo de la distancia.
```

```
[Ilimpia,mapa_disparidad,d_focal,B]=limpieza_CrossDetect(numero_imagen,Iu,
path,contraste);

%Calculo de la proyección horizontal de la imagen (PH), módulos de la FFT
%(Modulo(k)) y frecuencias asociadas f(k).
[f,Modulo]=picosTF_CrossDetect(Ilimpia);

%Comparación de los valores de Modulo(4) a Modulo(8) con sus respectivos
%umbrales y comprobación de la condición de máximo local
%((Modulo(k-1)<Modulo(k)) &  (Modulo(k)>Modulo(k+1))). Si alguno de los
%Modulo(k) cumple la condición se produce una detección.
if ((Modulo(4)>=20)&&(Modulo(4)>Modulo(5))&&(Modulo(4)>Modulo(3)))
    deteccion='DETECTADO';
elseif ((Modulo(5)>=17)&&(Modulo(5)>Modulo(4))&&(Modulo(5)>Modulo(6)))
    deteccion='DETECTADO';
elseif ((Modulo(6)>=16)&&(Modulo(6)>Modulo(5))&&(Modulo(6)>Modulo(7)))
    deteccion='DETECTADO';
elseif ((Modulo(7)>=16.5)&&(Modulo(7)>Modulo(6))&&(Modulo(7)>Modulo(8)))
    deteccion='DETECTADO';
elseif ((Modulo(8)>=15)&&(Modulo(8)>Modulo(7))&&(Modulo(8)>Modulo(9)))
    deteccion='DETECTADO';
else
    deteccion='NEGATIVO';
end

%Posicionamiento del paso de cebra en la imagen
if  strcmp('DETECTADO',deteccion)
    posicion=PosicionPaso_CrossDetect(limpia);
else
    posicion=Inf;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

A.1.2. media_im

Esta función es la encargada de realizar el cálculo de la diferencia entre los valores de gris medios de la imagen original completa y de su mitad superior, dato que se usa para discernir si a la fotografía se le debe aplicar o no una mejora de contraste con `imadjust` y una erosión con `imerode`. El código de esta función en MATLAB es el siguiente:

```
%ENTRADAS: Ioriginal-> imagen de entrada. Debe ser una imagen en escala
%de grises.
```

```
%SALIDAS: dif_medias-> diferencia entre la media de la media de la mitad
%inferior de la imagen y la media global de la misma. A partir de este
%valor se decide si se aplica o no imadjust sobre la imagen->Ver
%LTprd_CrossDetect.dif_medias>40 para aplicar imadjust.

function dif_medias=media_im(Ioriginal)

%%%INICIALIZACIONES%%%
Ioriginal=double(Ioriginal);
[m,n]=size(Ioriginal);
suma=0;
acum=0;
%%%%%%%%%%%%%%%%%%%%%

%%%CÓDIGO%%%

for i=1:m                                   %Se recorren las filas
    if(i==m/2+1)                            %Si hemos llegado a la fila
                                            %central se calcula la media
                                            %de la parte superior de la
                                            %imagen con el valor de suma
                                            %acumulado hasta ese momento.
        media_superior=ceil(suma/acum);
    end
    for j=1:n                               %Se recorren las colomnas de
                                            %la imagen.
        suma=suma+Ioriginal(i,j);           %Se suma el valor de todos los
                                            %píxeles.
        acum=acum+1;                        %Se cuenta el número de píxeles
                                            %sumados.
    end
end
media_global=ceil(suma/acum);              %Se calcula la media de nivel de
                                            %gris de toda la imagen con el
                                            %cociente de la suma de todos
                                            %los píxeles dividido por el
                                            %número de píxeles

dif_medias=media_superior-media_global; %Diferencia entre ambas medias

%%%%%%%%%%%%%%%%%%%%%%%%
```

A.1.3. LTprd_CrossDetect

Esta función desarrolla el algoritmo adaptado de 43rd Percentile Local Threshold utilizado en el código final. Si la entrada percentil se fijase a 50 también podría implementar el algoritmo Median Local Threshold. También es el código encargado de realizar la mejora de contraste con `imadjust`.

```
%ENTRADAS: Ioriginal-> Imagen en escala de grises. Optimizados los
%parámetros para imágenes de 480x640. SM debería ser recalibrado si se
%desean utilizar imágenes de otro tamaño, para que funcione correctamente.
%percentil-> Percentil que se desea aplicar en el cálculo de Iu. El valor
%50 calcula la mediana. En este trabajo se usa el percentil 43.
%umbral-> valor que el pixel de Ifiltrada debe estar por encima de
%Ioriginal para ser considerado un pixel de marca vial. Se recomienda 20.
%contraste-> Proviene de la función CrossDetect. Toma el valor 1 si la
%función media_im da una diferencia de medias mayor de 40 y cero en caso
%contrario.
%SALIDAS:
%Iu-> Imagen final "útil" del algoritmo. Pixeles de Ifiltrada que cumplen
%que Ioriginal(i,j)>Ifiltrada(i,j)+umbral, siendo i el número de fila y j
%el número de columna de un píxel cualquiera dentro de la imagen.

function Iu=LTprd_CrossDetect(Ioriginal,percentil,umbral,contraste)

%%%INICIALIZACIONES%%%
Ifiltrada=[];
Iu=[];
v=1:480;

SM=(16/55)*v-(436/11); %Función que calcula el ancho de las bandas
                       %del paso de cebra en las imágenes en función de la
                       %fila. Interpolación lineal calculada teniendo en
                       %cuenta SM(480)=100px y SM(150)=4px. Como
                       %consecuencia la fila mínima de aplicación es 137.
                       %Por debajo de este valor es negativo.
[n,m]=size(Ioriginal);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%CÓDIGO%%%

if contraste                           %Aplicación de la mejora de
                                       %contraste en las zonas más oscuras
                                       %de la imagen con imadjust.

    Ioriginal=imadjust(Ioriginal,[0.05 0.15],[]);
```

```
end

for i=140:n                                 %Recorre las filas de la 140
                                            %hasta la última.

    ancho=floor(SM(i));                     %Rango de cálculo de la media
                                            %para la fila correspondiente.

    for j=1:m                               %Recorre las columnas de la
                                            %imagen.

        vector_mediana=[];                  %Inicialización del vector para
                                            %el cálculo del percentil.
        z=0;                                %Índice del vector de percentil.

        rangoizq=j-ancho;                   %Numero de píxeles a la
                                            %izquierda del considerado a
                                            %computar en el cálculo del
                                            %percentil
        rangodcha=j+ancho;                  %Numero de píxeles a la derecha
                                            %del considerado a computar en
                                            %el cálculo del percentil. Como
                                            %se ve el intervalo es simétrico
                                            %respecto al píxel.

        if ((rangoizq<1)&&(rangodcha>m))    %Si el rango considerado se sale
                                            %de la imagen se halla el
                                            %percentil de toda la fila.
            for k=1:m
                z=z+1;
                vector_mediana(z)=Ig(i,k);
            end
        else
            if rangoizq<1                   %Si el rango se sale sólo por la
                                            %izquierda computa el percentil
                                            %desde la primer elemento de la
                                            %fila hasta alcanzar el final
                                            %del rango considerado.
                for k=1:rangodcha
                     z=z+1;
                    vector_mediana(z)=Ig(i,k);
                end
```

```
        elseif rangodcha>m              %Si el rango se sale sólo por la
                                        %derecha computa el percentil
                                        %desde la primer elemento de la
                                        %fila hasta alcanzar el final
                                        %del rango considerado.
            for k=rangoizq:m
                 z=z+1;
                vector_mediana(z)=Ig(i,k);
            end

        else
            for k=rangoizq:rangodcha    %Si el rango considerado queda
                                        %contenido en la imagen se
                                        %calcula el percentil de los
                                        %valores de la fila dentro de
                                        %ese rango.
                 z=z+1;
                vector_mediana(z)=Ig(i,k);
            end
        end
    end

    z=(percentil/100)*z;                %Cálculo del percentil
    vector_mediana=sort(vector_mediana);
    entero=fix(z);
    if ((z-entero)==0)                  %Si el número de elementos es par
                                        %se coge la media de los
                                        %elementos centrales como
                                        %percentil. %Si es impar, se coge
                                        %el valor central.
        percentil43=floor((vector_mediana(entero)
        +vector_mediana(entero+1))/2);
    else
        percentil43=vector_mediana(entero+1);
    end
    Ifiltrada(i,j)=percentil43;

    if ((Ig(i,j)>(Ifiltrada(i,j)+umbral))) %Si el valor de gris original
                                           %es mayor que el de
                                           %Ifiltrada más un umbral ese
                                           %píxel es potencialmente
                                           %perteneciente a una marca
                                           %vial y se pone blanco en Iu
                                           %arrojada por el algoritmo.
```

```
                                                %Si no es así se coloca en
                                                %negro.
            Iu(i,j)=255;
        else
            Iu(i,j)=0;
        end
    end
end

Iu=uint8(Iu);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

A.1.4. picosTF_CrossDetect

La función picosTF_CrossDetect se encarga del cálculo de la proyección horizontal (PH) y de los Modulo(k) y f(k) obtenidos a partir de la FFT del mismo.

```
%ENTRADAS: I_limpia-> Imagen que da como resultado la función
%limpieza_CrossDetect. Es una imagen binaria que representa todos los
%píxeles candidatos a marca vial.
%SALIDAS: f-> Frecuencias de la FFT de la proyección horizontal (PH)
%Modulo-> Módulos asociados a cada frecuencia de la FFT de PH.

function [f,Modulo]=picosTF_CrossDetect(Ilimpia)

%%%INICIALIZACIONES%%%
Ilimpia=im2double(Ilimpia);
[n,m]=size(I);
PH=zeros(1,m);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%CÓDIGO%%%

%Para cada columna de Ilimpia, se suman todos los píxeles que hay con
%valor 1 en ella obteniendo PH.
for i=1:n
    for j=1:m
        PH(j)=PH(j)+Ilimpia(i,j);
    end
end

%Cálculo de las frecuencias y Modulos asociados a FFT de PH.
```

```
TF=fft(PH);
Modulo=2*abs(TF)/length(TF); %Para obtener el verdadero valor de Modulo es
                             %necesario multiplicar por 2 y dividir por la
                             %longitud del vector, tal y como se explica en
                             %la ayuda de MATLAB.
f=linspace(0,100,length(TF));%El valor de las frecuencias está multiplicado
                             %por 100 para ver mejor los resultados.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

A.1.5. PosicionPaso_CrossDetect

Con esta función se calcula la posición del paso de peatones, representada por su línea media, haciendo uso de la proyección vertical de la imagen, PV.

```
%ENTRADAS: I_limpia-> Imagen que da como resultado la función
%limpieza_CrossDetect. Es una imagen binaria que representa todos los
%píxeles candidatos a marca vial.
%SALIDAS: posicion-> numero de fila en que se encuentra la línea media del
%paso de peatones.
function posicion=PosicionPaso_CrossDetect(limpia)

%%%INICIALIZACIONES%%%
[m,n]=size(limpia);
PV=zeros(1,m);
ancho=0;
ancho_anterior=0;
fila_inicio=0;
fila_inicio_anterior=0;
posicion=0;
%%%%%%%%%%%%%%%%%%%%%%%

%%%CÓDIGO%%%

%Cálculo de la proyección vertical de la imagen, PV
for i=1:m
    for j=1:n
        PV(i)=PV(i)+limpia(i,j);
    end
end

%Cálculo de la línea media del paso de peatones
for k=1:m
    if PV(k)>=90
```

```
        fila_inicio=k;
        ancho=0;
        while ((PV(k)>=90) && (k<480))  %Si algún PV(fila)>90, almacenamos
                ancho=ancho+1;          %su fila_inicio y contamos el
                k=k+1;                  %número de filas posteriores que
        end                             %repiten esta condición (ancho).
    end
    if ancho<ancho_anterior                %Si la condición PV>90 se repite
        ancho=ancho_anterior;              %en más de una ocasión, su ancho
        fila_inicio=fila_inicio_anterior;  %debe superar al almacenado para
    else                                   %modificar los valores de ancho
        ancho_anterior=ancho;              %fila_inicio anteriores.
        fila_inicio_anterior=fila_inicio;
    end
end

posicion=ceil((fila_inicio+fila_inicio+ancho)/2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

A.2. Funciones del bloque de "Análisis de la cartografía y correción del error"

A.2.1. busqueda_CrossDetect

Con esta función se realiza la búsqueda de los nodos que representan pasos de peatones en la estructura `osm_xml`, obtenida de un archivo de extensión .osm descargado de OpenStreetMaps. La estructura `osm_xml` se obtiene a partir de la función `parse_openstreetmaps`, perteneciente a la librería de Ioannis et al..

```
%ENTRADAS: osm_xml-> Estructura devuelta por la función
%parse_openstreetmap.m contenida en la librería de funciones descargada de
%http://www.mathworks.es/matlabcentral/fileexchange/
35819-openstreetmap-functions
%SALIDAS: cross->[1xnum_nodos struct]. Vector de estructuras que contiene
%datos sobre los nodos que representan pasos de peatones en el mapa. Cada
%posición del vector contiene una estructura con los siguientes campos:
%+id. String que representa el identificador del nodo.
%+lat. String que representa la latitud del nodo en el mapa.
%+lon. String que representa la longitud del nodo en el mapa.

function cross=busqueda_CrossDetect(osm_xml)
```

```
%%%%%%%INICIALIZACIONES%%%%%%%%%%%%
num_nodos=size(osm_xml.node,2);
j=0;
cross(num_nodos)=struct('id','identificador','lat','latitud','lon','longitud');
antiguo=0;
h=1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%CÓDIGO%%%%%%%%%%%%%%%%%%%%%%%%
%Busqueda de los nodos con etiqueta v=crossing o k=crossing, dentro del
%vector de celdas osm_xml.node{i}, perteneciente a la estructura osm_xml.
for i=1:num_nodos
    campos=fieldnames(osm_xml.node{i});
    tamano=size(campos,1);
    if tamano==2
        num_etiquetas=size(osm_xml.node{i}.tag,2);
        if num_etiquetas>1
            for etiqueta=1:num_etiquetas
                if strcmp(osm_xml.node{i}.tag{etiqueta}.Attributes.v,'crossing')
                 || strcmp(osm_xml.node{i}.tag{etiqueta}.Attributes.k,'crossing')
                    if i==antiguo
                    else
                        j=j+1;
                    end
                    cross(j).id=osm_xml.node{i}.Attributes.id;
                    cross(j).lat=osm_xml.node{i}.Attributes.lat;
                    cross(j).lon=osm_xml.node{i}.Attributes.lon;
                    antiguo=i;
                end
            end
        else
            if strcmp(osm_xml.node{i}.tag.Attributes.v,'crossing')
             || strcmp(osm_xml.node{i}.tag.Attributes.k,'crossing')
                j=j+1;
                cross(j).id=osm_xml.node{i}.Attributes.id;
                cross(j).lat=osm_xml.node{i}.Attributes.lat;
                cross(j).lon=osm_xml.node{i}.Attributes.lon;
            end
        end
    end
end

%Representación de los nodos encontrados en el mapa que representan un paso
%de peatones.
```

```
while size(cross(h).id,1)~=0
    plot(str2double(cross(h).lon),str2double(cross(h).lat),'k*');
    h=h+1;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

A.2.2. distanciaOSM_CrossDetect

Esta función calcula, en caso de existir, el nodo que representa en el mapa el paso de peatones más cercano a una posición simulada para el vehículo. Este paso de peatones debe cumplir la condición de pertenecer al camino por el que circula el mismo. Este camino por el que circula el vehículo también es calculado por la función, como aquel que contiene el nodo más cercano a su posición simulada dentro del mapa. Todas estas búsquedas son realizadas sobre la estructura `osm_xml`.

```
%ENTRADAS: nombre_mapa-> archivo de extensión .osm descargado de
%http://www.openstreetmap.org/
%latitud-> latitud de la posición simulada del vehículo.
%longitud-> longitud de la posición simulada del vehículo.
%SALIDAS: s_paso_cercano. Distancia entre la posición simulada del vehículo
%(latitud, longitud) y el nodo más cercano a esta posición de los que
%representan un paso de peatones (obtenidos de la función
%busqueda_CrossDetect.m). Esta distancia entre nodos, se calcula
%con la Fórmula de Vincenty para el elipsoide WGS84.

function s_paso_cercano=distanciaOSM_CrossDetect(nombre_mapa,latitud,longitud)

%%%%%%INICIALIZACIONES%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
s_menor_camino=Inf;
%Volcado a  una estructura de MATLAB de la información representada en el
%archivo .osm descargado de http:\\www.openstreetmap.org. Esta función
%pertenece a la librería que se puede descargar de
%http://www.mathworks.es/matlabcentral/fileexchange/
35819-openstreetmap-functions
[parsed_osm, osm_xml] = parse_openstreetmap(nombre_mapa);
fig = figure;
ax = axes('Parent', fig);
hold(ax, 'on')
%Representación del mapa. Esta función pertenece a la librería descargada
%de http://www.mathworks.es/matlabcentral/fileexchange/35819-openstreetmap
-functions
plot_way(ax, parsed_osm)
plot(longitud,latitud,'rs')
```

```
num_caminos=size(osm_xml.way,2);
num_nodos=size(osm_xml.node,2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%CÓDIGO%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Búsqueda del nodo más próximo a la posición simulada del vehículo. Se
%comprueba el camino al que pertenece ese nodo y se obtiene su
%identificador. Se asume que éste es el camino por el que circula el
%vehículo simulado.
for i=1:num_caminos
    id_camino=osm_xml.way{i}.Attributes.id;
    nodos_camino=size(osm_xml.way{i}.nd,2);
    for j=1:nodos_camino
        for k=1:num_nodos
            if strcmp(osm_xml.node{k}.Attributes.id,
            osm_xml.way{i}.nd{j}.Attributes.ref)
                s_nodo = vdist(latitud,longitud,
                str2double(osm_xml.node{k}.Attributes.lat),
                str2double(osm_xml.node{k}.Attributes.lon));
                if s_nodo<s_menor_camino
                    camino_mas_proximo=id_camino;
                    nodo_mas_proximo=osm_xml.node{k}.Attributes.id;
                    nodo_mas_proximo_latitud=
                    str2double(osm_xml.node{k}.Attributes.lat);
                    nodo_mas_proximo_longitud=
                    str2double(osm_xml.node{k}.Attributes.lon);
                    s_menor_camino=s_nodo;
                end
            end
        end
    end
end

%plot(nodo_mas_proximo_longitud,nodo_mas_proximo_latitud,'m*')
%Búsqueda de los nodos que representan pasos de peatones en el mapa.
cross=busqueda_CrossDetect(osm_xml);

%Comprueba a cuántos caminos pertenece el nodo más próximo calculado y
%obtiene los identificadores de estos caminos.
caminos_lo_contienen=0;
id_caminos=zeros(1,num_caminos);
for i=1:num_caminos
    nodos_camino=size(osm_xml.way{i}.nd,2);
    for j=1:nodos_camino
```

```
        if strcmp(osm_xml.way{i}.nd{j}.Attributes.ref,nodo_mas_proximo);
            caminos_lo_contienen=caminos_lo_contienen+1;
            id_caminos(1,caminos_lo_contienen)=i;
        end
    end
end

%Comprueba si alguno de los pasos de peatones obtenidos con la función
%busqueda_CrossDetect.m pertenece a alguno de los caminos obtenidos en el
%paso anterior. De ser así calcula la distancia al mismo, devolviendo este
%dato. Si existiese más de un nodo de estas características, devuelve el
%situado a una menor distancia de la posición simulada del vehículo.
numero_pasos=size(cross,2);
s_paso_cercano=Inf;
paso=0;
for i=1:caminos_lo_contienen
    nodos_camino=size(osm_xml.way{id_caminos(i)}.nd,2);
    for j=1:nodos_camino
        for k=1:numero_pasos
            if strcmp(cross(k).id,
            osm_xml.way{id_caminos(i)}.nd{j}.Attributes.ref)
                s_paso = vdist(latitud,longitud,
                str2double(cross(k).lat),str2double(cross(k).lon));
                if s_paso<s_paso_cercano
                    s_paso_cercano=s_paso;
                    paso=k;
                end
            end
        end
    end
end

%Si se ha encontrado algún paso de peatones en el camino por el que circula
%el vehículo, representa su nodo en el mapa.
if paso~=0
    plot(str2double(cross(paso).lon),str2double(cross(paso).lat),'rs');
end
hold(ax, 'off')
```

A.3. Otras funciones

A.3.1. LocalThreshold

Esta función implementa la adaptación del algoritmo Local Threshold. Como ya se comentaba en la sección 4.1.1 esta función no es usada en la implementación final del algoritmo.

```
%ENTRADAS: nombre_archivo->ruta del archivo de imagen que se desea filtrar
%con extensión. Optimizados los parámetros para imágenes de 480x640. SM
%debería ser recalibrado si se desean utilizar imágenes de otro tamaño,
%para que funcione correctamente.
%umbral-> valor que el pixel de Ifiltrada debe estar por encima de
%Ioriginal para ser considerado un pixel de marca vial. Se recomienda 20.
%SALIDAS: Ioriginal-> imagen de escala de grises que contiene
%nombre_archivo.
%Ifiltrada-> Imagen obtenida al sustituir cada píxel de Ioriginal por la
%media de los píxeles en un entorno 12SM dentro de su fila. SM=ancho de
%marca vial considerado en función de la fila de la imagen.
%Iu-> Imagen final "útil" del algoritmo. Pixeles de Ifiltrada que cumplen
%que Ioriginal(i,j)>Ifiltrada(i,j)+umbral, siendo i el número de fila y j
%el número de columna de un píxel cualquiera.
function [Ig,Ifiltrada,Iu]=LocalThreshold(nombre_archivo,umbral)

%%%INICIALIZACIONES%%%
t=cputime;
Ifiltrada=[];
Iu=[];
v=1:480;

SM=(16/55)*v-(436/11); %Función que calcula el ancho de las bandas
                       %del paso de cebra en las imágenes en función de la
                       %fila. Interpolación lineal calculada teniendo en
                       %cuenta SM(480)=100px y SM(150)=4px. Como
                       %consecuencia la fila mínima de aplicación es 137.
                       %Por debajo de este valor es negativo.

Ioriginal=imread(nombre_archivo); %Lectura del archivo de imagen.
[n,m]=size(Ioriginal);        %Dimensiones de la imagen.
%%%%%%%%%%%%%%%%%%%%%%

%%%CÓDIGO%%%

for i=140:n                %Recorre las filas de la 140 hasta la última.
```

```
ancho=floor(6*SM(i));  %Rango de cálculo de la media para la fila
                       %correspondiente.

for j=1:m              %Recorre las columnas de la imagen.

    suma=0;            %Inicializaciones para calcular la media del rango
    n_valores=0;

    rangoizq=j-ancho;  %Numero de píxeles a la izquierda del considerado
                       %a computar en el cálculo de la media
    rangodcha=j+ancho; %Numero de píxeles a la derecha del considerado
                       %a computar en el cálculo de la media. Como se
                       %ve el intervalo es simétrico respecto al píxel.

    if ((rangoizq<1)&&(rangodcha>m)) %Si el rango considerado se sale
                                     %de la imagen se halla la media
                                     %de toda la fila.
        for k=1:m
            suma=suma+double(Ioriginal(i,k));
            n_valores=n_valores+1;
        end

    else
        if rangoizq<1                %Si el rango se sale sólo por la
                                     %izquierda computa la media desde
                                     %la primer elemento de la fila
                                     %hasta alcanzar el final del rango
                                     %considerado.
            for k=1:rangodcha
                suma=suma+double(Ioriginal(i,k));
                n_valores=n_valores+1;
            end

        elseif rangodcha>m           %Si el rango se sale sólo por la
                                     %derecha computa la media desde el
                                     %inicio del rango considerado
                                     %hasta la último elemento de la
                                     %fila.
            for k=rangoizq:m
                suma=suma+double(Ioriginal(i,k));
                n_valores=n_valores+1;
            end
```

```
            else                            %Si el rango considerado queda
                                            %contenido en la imagen se calcula
                                            %media de los valores de la fila
                                            %dentro de ese rango.
                for k=rangoizq:rangodcha
                    suma=suma+double(Ioriginal(i,k));
                    n_valores=n_valores+1;
                end
            end
        end

        %Obtención de la media en el entorno correspondiente del pixel i,j.
        media=floor(suma/n_valores);
        Ifiltrada(i,j)=media;

        if ((Ioriginal(i,j)>(Ifiltrada(i,j)+umbral)))  %Si el valor de gris original
                                                    %es mayor que el de
                                                    %Ifiltrada más un umbral ese
                                                    %píxel es potencialmente
                                                    %perteneciente a una marca
                                                    %vial y se pone blanco en Iu
                                                    %arrojada por el algoritmo.
                                                    %Si no es así se coloca en
                                                    %negro.
            Iu(i,j)=255;
        else
            Iu(i,j)=0;
        end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

A.3.2. SLT

Al igual que la función anterior, ésta no es usada en la implementación final del algoritmo y sólo se utilizó en la fase de pruebas. Implementa la adaptación del algoritmo denominado Symmetrical Local Threshold.

```
%ENTRADAS: nombre_archivo->ruta del archivo de imagen que se desea filtrar
%con extensión. Optimizados los parámetros para imágenes de 480x640. SM
%debería ser recalibrado si se desean utilizar imágenes de otro tamaño,
%para que funcione correctamente.
%umbral-> valor que el pixel de Ifiltrada debe estar por encima de
```

```
%Ioriginal para ser considerado un pixel de marca vial. Se recomienda 20.
%SALIDAS: Ioriginal-> imagen de escala de grises que contiene
%nombre_archivo.
%Ifiltrada-> Imagen obtenida al sustituir cada píxel de Ioriginal por la
%la media de nivel de gris de los píxeles en un entorno 6SM a la izquierda
%del píxel considerado o a la media de nivel de gris en un entorno 6SM a la
%derecha del píxel considerado, dentro de su misma fila. Si la media de la
%izquierda es mayor que la de la derecha, se sustituye por la media de la
%izquierda y al contrario. SM=ancho de marca vial considerado en función de
%la fila de la imagen.
%Iu-> Imagen final "útil" del algoritmo. Pixeles de Ifiltrada que cumplen
%que Ioriginal(i,j)>Ifiltrada(i,j)+umbral, siendo i el número de fila y j
%el número de columna de un píxel cualquiera dentro de la imagen.

function [Ig,Ifiltrada,Iu]=SLT(nombre_archivo,umbral)

%%%INICIALIZACIONES%%%
Ifiltrada=[];
Iu=[];
v=1:480;

SM=(16/55)*v-(436/11); %Función que calcula el ancho de las bandas
                       %del paso de cebra en las imágenes en función de la
                       %fila. Interpolación lineal calculada teniendo en
                       %cuenta SM(480)=100px y SM(150)=4px. Como
                       %consecuencia la fila mínima de aplicación es 137.
                       %Por debajo de este valor es negativo.

Ioriginal=imread(nombre_archivo); %Lectura del archivo de imagen.

[n,m]=size(Ioriginal);            %Dimensiones de la imagen.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%CÓDIGO%%%

for i=40:n                    %Recorre las filas de la 140 hasta la última.

    ancho=floor(6*SM(i));     %Rango de cálculo de las medias para la fila
                              %correspondiente.

    for j=1:m                 %Recorre las columnas de la imagen.

        sumaizq=0;            %Inicializaciones para calcular las medias
                              %del rango.
```

```
sumadcha=0;
n_valoresizq=0;
n_valoresdcha=0;

rangoizq=j-ancho;   %Numero de píxeles a la izquierda del considerado
                    %a computar en el cálculo de la media
                    %izquierda.
rangodcha=j+ancho;  %Numero de píxeles a la derecha del considerado
                    %a computar en el cálculo de la media derecha.

if ((rangoizq<1)&&(rangodcha>m)) %Si el rango considerado se sale
                                 %de la imagen se hallan las medias
                                 %de toda la fila a izquierda y a
                                 %derecha del píxel considerado.
    for k=1:j
        sumaizq=sumaizq+double(Ioriginal(i,k));
        n_valoresizq=n_valoresizq+1;
    end
    for k=j:m
        sumadcha=sumadcha+double(Ioriginal(i,k));
        n_valoresdcha=n_valoresdcha+1;
    end

else                             %Si el rango se sale sólo por la
                                 %izquierda, la media de la
                                 %izquierda se computa sólo desde
                                 %el primer elemento de la fila
                                 %considerada hasta el píxel. La
                                 %media de la derecha se calcula en
                                 %6M.
    if rangoizq<1
        for k=1:j
            sumaizq=sumaizq+double(Ioriginal(i,k));
            n_valoresizq=n_valoresizq+1;
        end
        for k=j:rangodcha
            sumadcha=sumadcha+double(Ioriginal(i,k));
            n_valoresdcha=n_valoresdcha+1;
        end

    elseif rangodcha>m           %Si el rango se sale sólo por la
                                 %derecha, la media de la
                                 %derecha se calcula desde el pixel
                                 %considerado hasta el último
```

```
                                    %elemento de su fila. La media de
                                    %la izquierda se calcula en 6SM.
            for k=rangoizq:j
                sumaizq=sumaizq+double(Ioriginal(i,k));
                n_valoresizq=n_valoresizq+1;
            end
            for k=j:m
                sumadcha=sumadcha+double(Ioriginal(i,k));
                n_valoresdcha=n_valoresdcha+1;
            end

        else                        %Si el rango considerado queda
                                    %contenido en la imagen se calcula
                                    %la media de los valores en un
                                    %rango de 6SM a la izquierda del
                                    %píxel considerado y la media de
                                    %los valores en un rango 6SM a la
                                    %derecha del pixel considerado.
            for k=rangoizq:j
                sumaizq=sumaizq+double(Ioriginal(i,k));
                n_valoresizq=n_valoresizq+1;
            end
            for k=j:rangodcha
                sumadcha=sumadcha+double(Ioriginal(i,k));
                n_valoresdcha=n_valoresdcha+1;
            end
        end
    end
    mediaizq=floor(sumaizq/n_valoresizq);    %Cálculo de la media
                                             %de nivel de gris en el
                                             %intervalo a la izquierda
                                             %del pixel i,j.
    mediadcha=floor(sumadcha/n_valoresdcha); %Cálculo de la media
                                             %de nivel de gris en el
                                             %intervalo a la derecha
                                             %del pixel i,j.

    if (mediaizq>mediadcha)                  %Se comprueba que media es
                                             %mayor, quedándonos con
                                             %ella como media para el
                                             %píxel i,j.
        Ifiltrada(i,j)=mediaizq;
    else
        Ifiltrada(i,j)=mediadcha;
```

```
        end

        if ((Ioriginal(i,j)>(Ifiltrada(i,j)+umbral)))
            %Si el valor de gris original
                                                %es mayor que el de
                                                %Ifiltrada más un umbral ese
                                                %píxel es potencialmente
                                                %perteneciente a una marca
                                                %vial y se pone blanco en Iu
                                                %arrojada por el algoritmo.
                                                %Si no es así se coloca en
                                                %negro.
            Iu(i,j)=255;
        else
            Iu(i,j)=0;
        end
    end
end
Ifiltrada=uint8(Ifiltrada);
Iu=uint8(Iu);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

A.3.3. Otras funciones de MATLAB útiles en el tratamiento de imágenes

Se especifican a continuación algunas funciones de MATLAB que han resultado útiles a la hora de tratar con imágenes durante el desarrollo de esta investigación, de tal forma que esto sirva de punto de referencia para trabajos futuros. En este apartado, por tanto, se hace hincapié en para qué han servido esas funciones en la elaboración del presente trabajo, pero no se explican todas las opciones que éstas permiten. Para información detallada sobre el uso de estas funciones consultar la ayuda del software o Material web: Mathworks:

A.3.3.1. `imadjust`

La función `imadjust` aplica el método de la amplitud del histograma (ver apartado 3.4) para la modificación del contraste de una imagen. Según la ayuda de MATLAB (Mathworks, b), la función `imadjust` para imágenes en niveles de gris, mapea los valores de intensidad de la imagen original a otros nuevos en la imagen de salida, en función de los valores de entrada

que se le apliquen. La configuración de `imadjust` de interés para el caso de estudio en este proyecto es la siguiente:

```
J = imadjust(I,[low_in, high_in],[low_out, high_out])
```

A continuación se definen cada uno de los parámetros de esta función:

- `J`: Imagen de salida
- `I`: Imagen de entrada
- `[low_in, high_in]`: define el rango del histograma de `I` que es mapeado. El valor `low_in` determina el límite inferior del intervalo y `high_in` el superior. Este rango se define como un tanto por uno del histograma de `I`, de tal forma que si se fija en [0,1] equivaldría a utilizar el histograma completo,[0,0.5] la mitad inferior del mismo (valores de gris más bajos) y [0.5,1] la mitad superior (valores de gris más altos).
- `[low_out, high_out]`. Define el rango del histograma de salida en que son distribuidos los valores del rango de entrada. Igual que el `[low_in, high_in]` se expresa en tanto por uno del histograma de salida. Cualquier valor del histograma de `I` por encima de `high_in` se mapea a `high_out` en `J` y cualquiera por debajo de `low_in` a `low_out`.

A.3.3.2. `imerode`

Según la ayuda de Matlab (Mathworks, c), la función `imerode` se define como:

```
J=imerode(I,se)
```

Donde:

- `J`. Imagen de salida ya erosionada.
- `I`. Imagen binaria de entrada
- `se`. Elemento estructural con el que se realiza la operación de erosión. Se define con la función `strel`, asignándole como entradas 'square'y 3, es decir, se define un elemento estructural de forma cuadrada de 3x3 píxeles (ver sección 3.8).

A.3.3.3. Otras

- `imread`. Lee un archivo de imagen del origen especificado y lo transforma en una matriz de MATLAB con formato `uint8`. Si la imagen es en escala de grises la matriz tiene dimensiones `resolución_vertical`×`resolución_horizonta`
  Si es en RGB sus dimensiones son `resolución_vertical`×`resolución_horizontal`×3.

- `imwrite`. Guarda una matriz que representa una imagen en MATLAB en el destino especificado y con un formato de imágenes estándar (.png, .jpg, etc).
- `rgb2gray`. Transforma una imagen de color RGB a escala de grises.
- `im2bw`. Transforma una imagen en escala de grises en una imagen binaria. Los valores de la imagen resultante sólo pueden ser 0 ó 1.
- `im2double`. Transforma una imagen a doble precisión. Si la imagen de partida es de tipo binaria, los valores obtenidos son 0 ó 1, pero considerados por el software como números de doble precisión y no como valores lógicos. Esto es importante a la hora de sumar valores en matrices lógicas como se hace a la hora de obtener PH en A.1.4. Hay que tener cuidado si se aplica sobre una matriz de tipo `uint8`, pues los valores iniciales de la matriz (enteros entre 0 y 255) son convertidos a números entre 0 y 1, y no a los correspondientes números entre 0 y 255. En este caso se debe utilizar la función `double` y no `im2double` para obtener el resultado deseado.
- `im2uint8`. Transforma una imagen a enteros de 8 bits (valores enteros entre 0 y 255). Si se aplica a una imagen binaria el resultado es 255 para los píxeles que antes valían 1 y 0 para el resto. Si se aplica a una imagen de tipo `double` ocurre lo mismo que ocurría con `im2double`; para cualquier valor distinto de cero devuelve 255. Si se quieren "traducir" los números tal cual a enteros de 8 bits, se debe utilizar la función `uint8`.
- `imshow`. Muestra en una ventana gráfica una matriz en forma de imagen.
- `imtool`. Muestra en una ventana gráfica una matriz en forma de imagen, pero permite ver los valores de ésta con deslizar el ratón sobre la ventana gráfica y ofrece otras herramientas útiles a la hora de trabajar con imágenes.

Además, durante la elaboración de este trabajo ha sido de utilidad la creación de vídeos a partir de secuencias de imágenes para ver de una manera global la evolución de los datos que arrojaba el algoritmo al procesar éstas. La secuencia de comandos para crear un los vídeos en MATLAB ha sido:

1. `vid=VideoWriter(nombre_archivo)`. Crea el archivo de vídeo.
2. `vid.FrameRate=10`. Selección de la velocidad en frames/s
3. `open(vid)`. Abre el archivo de vídeo dejándolo listo para grabar sobre él.

4. Comandos para mostrar ventanas gráficas que almacenar como frames: `plot`, `subplot`, `stem`, etc.

5. `curr=getframe(gcf)`. Guarda la ventana mostrada.

6. `writeVideo(vid,curr)`. Almacena la ventana guardada como un frame en el vídeo.

7. `close(vid)`. Cierra el vídeo antes de salir de la función que lo ha creado.

Apéndice B

Análisis de la librería utilizada para trabajar con los archivos de OpenStreetMap en MATLAB

En este apéndice se realiza un análisis del funcionamiento, aplicaciones e interrelación de las funciones que permiten trabajar con los archivos XML descargados de OpenStreetMaps en MATLAB. Esta colección de funciones, con sus respectiva licencias de uso y distribución, puede descargarse en Ioannis et al.. Además de las funciones de la librería y otras de las que dependen éstas, el paquete contiene un breve manual de uso, la licencia, un documento readme y un ejemplo de utilización en Matlab (`usage_example.m`).

B.1. Usos de la colección de funciones

Esta librería de funciones permite, a partir de los archivos XML (extensión `.osm`), que se pueden obtener de OpenStreetMaps:

- **Importar y convertir el archivo XML descargado y almacenar los datos obtenidos en una estructura de Matlab.** Estos datos son los que componen el gráfico de la red de transporte.

- **Pintar la estructura de Matlab para visualizar la red de transporte**, los nodos que la componen y las etiquetas correspondientes a estos.

- **Extraer la matriz de adyacencias de la red**, es decir, obtener las intersecciones entre las carreteras.

- A partir de lo anterior, encontrar el **camino más corto entre nodos de la red y pintar las rutas**.

B.2. Funciones de la librería y su jerarquía

La librería descargada está compuesta por las siguientes funciones:

- `assign_from_parsed.m`
- `extract_connectivity.m`
- `get_unique_node_xy.m`
- `get_way_tag_key.m`
- `load_osm_xml.m`
- `parse_openstreetmap.m`
- `parse_osm.m`
- `plot_nodes.m`
- `plot_road_network.m`
- `plot_route.m`
- `plot_way.m`
- `route_planner.m`
- `show_map.m`

La jerarquía entre estas funciones se muestra en la figura B.1. Las funciones que se utilizan para obtener de la librería las utilidades descritas en el apartado B.1 de este documento, son exclusivamente las recuadradas en amarillo en la figura B.1. Por este motivo, a continuación sólo se analizará y se describirá el uso de estas funciones.

B.3. Funciones de dependencia

El archivo comprimido en el que se descarga la librería (Ioannis et al.), contiene otras funciones necesarias para el correcto funcionamiento de las funciones, además de las de las especificadas en el apartado B.2. Estas funciones adjuntas, que están alojadas en carpetas una vez descomprimido el paquete, están también disponibles en *MathWorks* pero son de autores diferentes a las de la librería a la que se refiere este documento. Se adjuntan también en el paquete las licencias correspondientes. Estas funciones son:

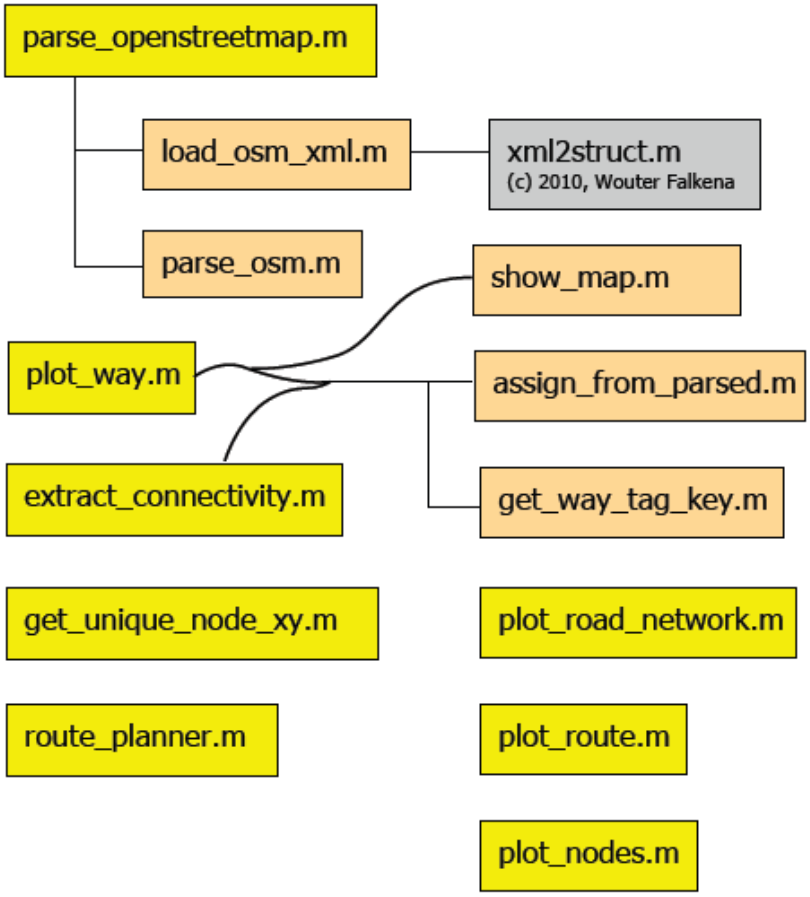


Figura B.1: Funciones de la librería y su jerarquía

- `dijkstra.m`
- `sparse_to_csr.m`
- `givehold.m`
- `takehold.m`
- `lat_lon_proportions.m`
- `plotmd.m`
- `textmd.m`
- `xml2struct.m`

B.4. NOTA IMPORTANTE

Debido a esta jerarquía de funciones y dependencias, es muy recomendable para evitar errores añadir la carpeta en la que se ubique la librería descomprimida al path de MATLAB. Para ello introducir en MATLAB el comando `pathtool` y buscar y seleccionar la carpeta correspondiente. También se puede hacer clic con el botón secundario del ratón en el icono de la carpeta, dentro del navegador de la parte izquierda de la ventana gráfica de MATLAB, y seleccionar la opción *Add to path*.

B.5. Descripción y uso de las funciones principales

B.5.1. `parse_openstreetmaps`

- **Utilidad.** Es la función más importante. Toma del archivo con extensión `.osm` los datos necesarios para el resto de funciones y los almacena en una estructura de MATLAB con los tipos de dato adecuados.
- **Sintaxis.**
  `[parsed_osm,osm_xml]=parse_openstreetmap(nombre_de_archivo)`
- **Entradas.**
  `nombre_de_archivo`. Es un *string* de la forma `nombre_archivo.osm`. Es el documento correspondiente al mapa descargado.
- **Salidas.**

  - `parsed_osm`. Estructura de MATLAB que almacena toda la información necesaria, una vez extraída del archivo `.osm` y convertida al tipo de datos correcto, para que el resto de funciones puedan procesarlos. Tiene los siguientes campos:
    - `parsed_osm.bounds`. [2×2 `double`]. Matriz que contiene los límites del mapa (latitudes y longitudes) contenido en el archivo `.osm`.
    - `parsed_osm.node`. [1×1 struct]. Estructura formada a su vez por:
      - `parsed_osm.node.id`. [1×n^{o}nodos `double`]. Vector con los identificadores numéricos de todos los nodos del mapa.
      - `parsed_osm.node.xy`. [2×n^{o}nodos `double`]. Matriz con las coordenadas x e y de cada uno de los nodos.
    - `parsed_osm.way`. [1×1 `struct`]. Estructura que contiene todos los datos sobre los nodos pertenecientes a caminos (way) del mapa. Está formada por:

- ◇ `parsed_osm.way.id`. [1×nºways `double`]. Vector con los identificadores de todos los caminos.
- ◇ `parsed_osm.way.nd`. {1×nºways `cell`}. Vector de celdas que contienen vectores con los identificadores de los nodos pertenecientes a cada camino.
- ◇ `parsed_osm.way.tag`. {1×nºway cell}. Vector de celdas que contienen bien estructuras, bien vectores de celdas que contienen estructuras. Esas estructuras son de la forma:
  `...Attributes`. [1×1 `struct`]. Contiene la/s etiqueta/s del camino correspondiente. Contiene a su vez:
  `...Attributes.k`. String con la clave de la etiqueta.
  `...Attributes.v`. String con el valor de la etiqueta.

- ○ `parsed_osm.Attributes`. [1×1 `struct`].Estructura con datos sobre copyright, versión de OSM, etc.

- `osm_xml`. Estructura que contiene los datos del mapa volcados directamente a una estructura de MATLAB y sin conversión de formato. Son los datos en bruto únicamente estructurados y con formato de cadena de caracteres. Contiene los siguientes campos:
  - ○ `osm_xml.bounds`=[1×1 `struct`]. Estructura que contiene:
    - ◇ `osm_xml.bounds.Attributes`=[1×1 `struct`]. Estructura que contiene:
      `...maxlat/maxlong/minlong/minlat`. Números en forma de cadena de caracteres que indican los límites del mapa.
  - ○ `osm_xml.node`. {1×nºnodos_mapa `cell`}. Vector de celdas que contiene:
    - ◇ `osm_xml.node{nºnodo}`. [1×1 `struct`]. Estructura que contiene:
      `...Attributes`. [1×1 `struct`]. Estructura que contiene todos los datos del nodo a excepción de las etiquetas.
      `...tag`. {1×nºetiquetas `cell`}. Este campo sólo aparece si el nodo tiene etiquetas. Vector de celdas que contienen todas las etiquetas del nodo.
  - ○ `osm_xml.way`. {1×nºways `cell`}. Vector de celdas que contiene estructuras del tipo:
    - ◇ `osm_xml.way{nºway}`. [1×1 `struct`]. Estructura con los siguientes campos:
      `...Attributes`. [1×1 `struct`]. Estructura con toda la información sobre el camino.
      `...nd`.{1×nºnodos_way `cell`}. Vector de celdas que contienen estructuras con el campo `Attributes`, en el cual

está el identificador de cada nodo del camino.
`...tag`. {1×nºetiquetas_way `cell`}. Vector de celdas que contienen estructuras con el campo `Attributes`, que es a su vez una estructura formada por los campos `k` y `v`, clave y valor, respectivamente, de la etiqueta del camino.

- `osm_xml.relation`. {1×nºrelacion `cell`}. Vector de celdas que contienen estructuras de la forma:
  - `osm_xml.relation{nºrelacion}`. [1×1 `struct`]. Estructura que contiene los siguientes campos:
    `...member`. {1×nºmiembros_relacion `cell`}. Vector de celdas con estructuras de la forma:
    `.....{nºmiembro_relacion}`. [1×1 struct]. Estructura que contiene el campo `Attributes`, que a su vez es una estructura con toda la información sobre la relación.
    `...tag`. {1×nºetiquetas_relation `cell`}. Vector de celdas que contienen estructuras que tienen en su interior el campo`Attributes`. Éste es a su vez una estructura con los campos `k` y `v`, clave y valor de la etiqueta, respectivamente.
    `...Attributes`. [1×1 `struct`]. Estructura que contiene todos los datos sobre la relación.
- `osm_xml.Attributes`. [1×1 `struct`]. Estructura que contiene todos los datos sobre la versión de OSM,etc.

B.5.1.1. Ejemplos de acceso a las partes más complejas de la estructura

- **Acceder a un campo de una estructura.**

  Estructura.campo

- **Acceder a una celda de un vector de celdas.**
  - Vector_celdas{nºcelda}. Accede al contenido de la celda.
  - Vector_celdas(nºcelda). Muestra el tipo de contenido de la celda.
- **Acceder a un vector dentro de una celda.**
  - Vector_celdas{celda_vector}. Accede a todo el vector contenido en la celda celda_vector de vector_celdas.
  - Vector_celdas{celda_vector}(posición_vector). Accede a la posición posición_vector dentro de la celda celda_vector del vector de celdas vector_celdas.

  - Vector_celdas(celda_vector). Muestra el tamaño del vector contenido en vector_celdas.

- **Acceder a una estructura dentro de un vector de celdas.**

  - Vector_celdas{celda_estructura}. Muestra los campos de la estructura que contiene la celda celda_estructura del vector de celdas vector_celdas.
  - Vector_celdas{celda_estructura}.campo_estructura. Accede al campo campo_esructura que está contenido en la estructura celda_estructura dentro de vector_celdas.

- **Acceder a un vector de celdas dentro de un vector de celdas**.

  - Vector_celdas{vector_celdas_int}. Accede el vector de celdas más interno, vector_celdas_int, y muestra el tipo de su contenido.
  - Vector_celdas{vector_celdas_int}{nºcelda}. Accede a la celda nºcelda dentro del vector de celdas más interno, vector_celdas_int, que está a su vez contenido en una celda de vector_celdas.
  - Vector_celdas{vector_celdas_int}(nºcelda). Tipo de contenido de la celda nºcelda dentro del vector de celdas más interno (vector_celdas_int),contenido en el vector de celdas vector_celdas.

B.5.2. `extract_connectivity.m`

- **Utilidad.** Extrae la matriz de conectividad de la red de carreteras. Esto da una serie de nodos en los que las carreteras intersectan. Algunas intersecciones pueden aparecer varias veces, bien porque diferentes carreteras pueden encontrarse en la misma intersección, o bien porque varias direcciones son consideradas distintas carreteras. Por este motivo, además de la matriz de conectividad, se identifican también los nodos únicos.

- **Sintaxis.**
  `[matrix_connectivity, intersection_node_indices]=extract_connectivity(parsed_osm)`

- **Entradas.**
  `parsed_osm`. Estructura de salida de la función parse_openstreetmap.m

- **Salidas.**

  - `matrix_connectivity`. Matriz dispersa que contiene unos en las posiciones i,j (índices globales) correspondientes a una intersección y ceros en el resto.
  - `intersection_node_indices`. Vector que contiene los índices de la matriz de conectividad de valor 1.

B.5.3. `route_planner.m`

- **Utilidad.** Planificar una ruta que una dos nodos.
- **Sintaxis.**
  ```
  dg = or(connectivity_matrix, connectivity_matrix.');(solo si
  hay carreteras de doble sentido)
  [route, dist] = route_planner(dg, start, target);
  ```
- **Entrada.**
  - `dg`. Matriz de conectividad obtenida de `extract_connectivity.m` o matriz de conectividad hecha simétrica (sólo si hay caminos de doble sentido).
  - `start`. Índice global (el obtenido en la matriz de conectividad) del nodo de partida.
  - `target`. Índice global del nodo de llegada.
- **Salidas.**
  - `route`. Índice global de los nodos por los que pasa la ruta calculada.
  - `dist`. Número de nodos entre `start` y `target` (distancia en número de nodos).

B.5.4. `plot_way.m`

- **Utilidad**. Pinta los caminos extraídos del mapa.
- **Sintaxis.**
  ```
  fig = figure;
  ax = axes('Parent', fig);
  hold(ax, ón')
  plot_way(ax,parsed_osm,ñombre_imagen.png')
  ```
- **Entradas.**
  - `ax`. Puntero a objeto de tipo axes.
  - `parsed_osm`. Estructura obtenida de la función `parse_openstreetmap.m`
  - `ñombre_imagen.png'`. Cadena de caracteres que representa un archivo de imagen con formato .png sobre el que pintar la red de carreteras. Este parámetro es opcional.

B.5.5. `plot_route`

- **Utilidad.** Pintar la ruta calculada con `route_planner.m` en el mapa.
- **Sintaxis.**
  `plot_route(ax,route,parsed_osm)`
- **Entradas.**
  - `ax`. Puntero a un objeto de tipo `axes`
  - `route`. Salida de la función `route_planner.m`
  - `parsed_osm`. Estructura obtenida como salida de `parse_openstreetmap.m`

B.5.6. `plot_nodes`

- **Utilidad.** Pintar en el mapa los nodos seleccionados.
- **Sintaxis.**
  `plot_nodes(ax, parsed_osm, only_node_indices, show_id);`
- **Entradas.**
  - `ax`. Puntero a un objeto de tipo axes.
  - `parsed_osm`. Estructura obtenida como salida de `parse_openstreetmap.m`
  - `only_node_indices`. Vector con los índices globales de los nodos que se quieran pintar en el mapa.
  - `show_id`. Parámetro opcional que determina si se quieren o no mostrar etiquetas con los identificadores de los nodos pintados.

B.5.7. `get_unique_node_xy.m`

- **Utilidad.** Obtener el identiicador y las coordenadas xy de un nodo a partir de sus coordenadas globales.
- **Sintaxis.**
  `nodos=get_unique_node_xy(parsed_osm,indices)`
- **Entradas.**
  - `parsed_osm`. Estructura obtenida como salida de la función `parse_openstreetmap.m`
  - `indices`. Vector de índices globales de los nodos de los que se quiere obtener el identificador y las coordenadas xy.
- **Salidas.**
  - `nodos`. Estructura compuesta por los campos `id` e `xy` que contiene los identificadores y las coordenadas xy de los nodos, respectivamente.

Bibliografía

20 MINUTOS.ES, (2010) (). Entrada: "El tiempo perdido al ano en los atascos equivale a una semana de vacaciones". Disponible en `http://consejosconducir.racc.es/es/coste_victimas` (último acceso, Mayo, 2014).

AHMETOVIC, D. Smartphone-assisted mobility in urban environments for visually impaired users through computer vision and sensor fusion. En *Proceedings-IEEE international conference on mobile data management*, vol. 2, páginas 15–18. 2013.

AHMETOVIC, D., BERNAREGGI, C. y MASCETTI, S. Zebralocalizer: identification and localization of pedestrian crossings. En *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, páginas 275–284. ACM, 2011.

BARROW, H. G., TENENBAUM, J. M., BOLLES, R. C. y WOLF, H. C. Parametric correspondence and chamfer matching: Two new techniques for image matching. Informe técnico, DTIC Document, 1977.

BAY, H., ESS, A., TUYTELAARS, T. y VAN GOOL, L. Speeded-up robust features (surf). *Computer vision and image understanding*, vol. 110(3), páginas 346–359, 2008.

BESL, P. J. y MCKAY, N. D. Method for registration of 3-d shapes. En *Robotics-DL tentative*, páginas 586–606. International Society for Optics and Photonics, 1992.

BLOGSPOT DE TOPOGRAFÍA (2012) (). Entrada: "Principales fuentes de error en GPS". Disponible en `http://detopografia.blogspot.com.es/search/label/GPS` (último acceso, Mayo, 2014).

BOSCH (2014) (). Entrada: "ACC". Disponible en `http://www.xn--bosch-tecnologadelautomvil-roc1p.es/es/es/driving_comfort_8/driving_comfort_systems_for_commercial_vehicles_9/driver_assistance_systems_35/driver_assistance_systems_7.html` (último acceso, Mayo, 2014).

BRAVO CORRALES, D. (2014) (). Entrada: "Next two, el coche autónomo de Renault para 2020". Disponible en `http://www.elmundo.es/motor/2014/02/07/52f4b59022601dbc708b4570.html` (último acceso, Mayo, 2014).

CANNY, J. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6), páginas 679–698, 1986.

CHOI, J., AHN, B. T. y KWEON, I. S. Crosswalk and traffic light detection via integral framework. En *Frontiers of Computer Vision,(FCV), 2013 19th Korea-Japan Joint Workshop on*, páginas 309–312. IEEE, 2013.

COLLADO, J., HILARIO, C., DE LA ESCALERA, A. y ARMINGOL, J. M. Detección y clasificación de líneas viales mediante análisis frecuencial. 2005.

COLLADO, J. M., HILARIO, C., ARMINGOL, J. y DE LA ESCALERA, A. Visión por computador para vehículos inteligentes. *XXIV Jornadas de Automática, León, España*, 2003.

COUGHLAN, J. y SHEN, H. A fast algorithm for finding crosswalks using figure-ground segmentation. En *2nd Workshop on Applications of Computer Vision, in conjunction with ECCV*, vol. 5. 2006.

CPAN (Thaddeus Vincenty distance calculations). Entrada: "Thaddeus vincenty distance calculations.". Disponible en `http://search.cpan.org/~bluefeet/GIS-Distance-0.08/lib/GIS/Distance/Formula/Vincenty.pm` (último acceso, Mayo, 2014).

DARCHE, G. Iterative l1 deconvolution. *SEP Annual Report*, vol. 61, páginas 281–301, 1989.

DE LA ESCALERA, A. y ARMINGOL, 2010, J. M. (). Entrada: "Teoría de sistemas de percepción". Disponible en `http://ocw.uc3m.es/ingenieria-de-sistemas-y-automatica/sistemas-de-percepcion/material-de-clase` (último acceso, Mayo, 2014).

EURO NCAP (2014) (). Entrada: "Sistemas de control del punto ciego". Disponible en `http://es.euroncap.com/es/rewards/technologies/blind.aspx` (último acceso, Mayo, 2014).

FISCHLER, M. A. y BOLLES, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, vol. 24(6), páginas 381–395, 1981.

FLOROS, G., VAN DER ZANDER, B. y LEIBE, B. Openstreetslam: Global vehicle localization using openstreetmaps. En *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, páginas 1054–1059. IEEE, 2013.

FORMACIÓN EN RED: MINISTERIO DE EDUCACIÓN (). Entrada: "Recursos web: ImÃ¡genes digitales". Disponible en `http://www.ite.educacion.es/formacion/materiales/86/cd/pdf/m2_caracteristicas_de_la_imagen_digital.pdf` (último acceso, Mayo, 2014).

FOUCHER, P., SEBSADJI, Y., TAREL, J.-P., CHARBONNIER, P. y NICOLLE, P. Detection and recognition of urban road markings using images. En *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, páginas 1747–1752. IEEE, 2011.

FREUND, Y., SCHAPIRE, R. y ABE, N. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, vol. 14(771-780), página 1612, 1999.

G. BEJARANO, P. (2014) (). Entrada: "Volvo experimenta con coches autónomos en carreteras magnéticas". Disponible en `http://blogthinkbig.com/carreteras-magneticas/` (último acceso, Mayo, 2014).

G. BEJARANO, P. (2014) (). Entrada: "Volvo prueba 100 coches autónomos en la carretera". Disponible en `http://blogthinkbig.com/coches-autonomos-de-volvo/` (último acceso, Mayo, 2014).

GARCÍA, R. G. y VÁZQUEZ, M. Á. S. *Sistema de Odometría Visual para la Mejora del Posicionamiento Global de un Vehículo*. 2007.

GOBIERNO DE ESTADOS UNIDOS (2014) (). Entrada: "GPS.gov". Disponible en `http://www.gps.gov/spanish.php` (último acceso, Mayo, 2014).

GROMPONE, V. G. R., JAKUBOWICZ, J., MOREL, J. y RANDALL, G. Lsd: a fast line segment detector with a false detection control. *IEEE transactions on pattern analysis and machine intelligence*, vol. 32(4), páginas 722–732, 2010.

GUINDEL GÓMEZ, C. Algoritmo de odometría visual estéreo para sistemas de ayuda a la conducción: implementación en gpu mediante cuda. 2012.

HE, X., ZEMEL, R. S. y CARREIRA-PERPINDN, M. Multiscale conditional random fields for image labeling. En *Computer vision and pattern recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE computer society conference on*, vol. 2, páginas II–695. IEEE, 2004.

IOANNIS, F., WOUTER, F. y SULLIVAN, J. (). Entrada: "OpenStreetMaps Functions". Disponible en http://www.mathworks.es/matlabcentral/fileexchange/35819-openstreetmap-functions (último acceso, Mayo, 2014).

IVANCHENKO, V., COUGHLAN, J. y SHEN, H. Detecting and locating crosswalks using a camera phone. En *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*, páginas 1–8. IEEE, 2008.

JIMÉNEZ (2013), J. D. D. (). Entrada: "Una valoración económica de los accidentes de tráfico: la necesidad del peritaje actuarial en las sentencias del tribunal supremo". Disponible en http://www.asociacionabogadosrcs.org/doctrina/Una%20valoraci%C3%B3n%20econ%C3%B3mica%20de%20los%20accidentes%20de%20tr%C3%A1ficola%20necesidad%20del%20peritaje.pdf (último acceso, Mayo, 2014).

JIMÉNEZ DE LUIS, Á. (2014) (). Entrada: "Google fabricará su propio coche autónomo". Disponible en http://www.elmundo.es/tecnologia/2014/05/28/538565f0268e3e58098b456c.html (último acceso, Mayo, 2014).

KLEDER, M. (2014), M. (). Entrada: "Geodetic distance on WGS84 earth ellipsoid". Disponible en http://www.mathworks.com/matlabcentral/fileexchange/5379-geodetic-distance-on-wgs84-earth-ellipsoid/content/vdist.m (último acceso, Mayo, 2014).

KSCHISCHANG, F. R., FREY, B. J. y LOELIGER, H.-A. Factor graphs and the sum-product algorithm. *Information Theory, IEEE Transactions on*, vol. 47(2), páginas 498–519, 2001.

LABORATORIO DE SISTEMAS INTELIGENTES UC3M (). Entrada: "Ivvi-2.0". Disponible en http://portal.uc3m.es/portal/page/portal/dpto_ing_sistemas_automatica/investigacion/lab_sist_inteligentes/sis_int_transporte/vehiculos/IvvI20/ (último acceso, Mayo, 2014).

LATEGAHN, H., GEIGER, A. y KITT, B. Visual slam for autonomous ground vehicles. En *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, páginas 1732–1737. IEEE, 2011.

LAUSSER, L., SCHWENKER, F. y PALM, G. Detecting zebra crossings utilizing adaboost. En *ESANN*, páginas 535–540. 2008.

LEMAIRE, T., BERGER, C., JUNG, I.-K. y LACROIX, S. Vision-based slam: Stereo and monocular approaches. *International Journal of Computer Vision*, vol. 74(3), páginas 343–364, 2007.

MALLOT, H. A., BÜLTHOFF, H. H., LITTLE, J. y BOHRER, S. Inverse perspective mapping simplifies optical flow computation and obstacle detection. *Biological cybernetics*, vol. 64(3), páginas 177–185, 1991.

MARKOFF, J. (2010) (). Entrada: "Guided by computers and sensors, a smooth ride at 60 miles per hour". Disponible en `http://www.nytimes.com/2010/10/10/science/10googleside.html?scp=1&sq=google%20toyota&st=Search&_r=0` (último acceso, Mayo, 2014).

MATERIAL WEB: GONZÁLEZ, Y. (). Entrada: "Manipulación del contraste". Disponible en `http://dmi.uib.es/~ygonzalez/VI/Material_del_Curso/Teoria/Tema4_Contraste.pdf` (último acceso, Mayo, 2014).

MATERIAL WEB: MATHWORKS (). Entrada: "Lista de funciones de tratamiento de imagenes". Disponible en `http://www.mathwork.es/es/help/images/functionlist.html` (último acceso, Mayo, 2014).

MATHWORKS (). Entrada: "Características principales de MATLAB". Disponible en `http://www.mathworks.es/products/matlab/description1.html` (último acceso, Mayo, 2014).

MATHWORKS (). Entrada: "Descripcion de la funcion imadjust". Disponible en `http://www.mathwork.es/es/help/images/ref/imadjust.html` (último acceso, Mayo, 2014).

MATHWORKS (). Entrada: "Descripcion de la funcion imerode". Disponible en `http://www.mathwork.es/es/help/images/ref/imerode.html` (último acceso, Mayo, 2014).

MATHWORKS (). Entrada: "Image Processing Toolbox". Disponible en `http://www.mathworks.es/products/image/index.html` (último acceso, Mayo, 2014).

MATHWORKS (). Entrada: "Productos y servicios". Disponible en `http://www.mathworks.es/products/?s_tid=gn_ps` (último acceso, Mayo, 2014).

MOTORPASIÓN FUTURO (2014) (). Entrada: "eCall: obligatorio en los coches nuevos a partir de octubre de 2015". Disponible en `http://www.motorpasionfuturo.com/ayudas-a-la-conduccion/ecall-obligatorio-en-los-coches-nuevos-en-2015` (último acceso, Mayo, 2014).

MUAD, A. M., HUSSAIN, A., SAMAD, S. A., MUSTAFFA, M. M. y MAJLIS, B. Y. Implementation of inverse perspective mapping algorithm for the development of an automatic lane tracking system. En *TENCON 2004. 2004 IEEE Region 10 Conference*, páginas 207–210. IEEE, 2004.

MUÑOZ-REPISO, C. Prefacio (13-14). *L. Montoro y cols. Manual de Seguridad Vial: El factor humano. Barcelona. Ariel Intras*, 2000.

MURALI, V. N. y COUGHLAN, J. M. Smartphone-based crosswalk detection and localization for visually impaired pedestrians. En *Multimedia and Expo Workshops (ICMEW), 2013 IEEE International Conference on*, páginas 1–7. IEEE, 2013.

MUSLEH, B., DE LA ESCALERA, A. y ARMINGOL, J. Detección de obstáculos y espacios transitables en entornos urbanos para sistemas de ayuda a la conducción basados en algoritmos de visión estéreo implementados en gpu. *Revista Iberoamericana de Automática e Informática Industrial RIAI*, vol. 9(4), páginas 462–473, 2012a.

MUSLEH, B., DE LA ESCALERA, A. y ARMINGOL, J. M. Uv disparity analysis in urban environments. En *Computer Aided Systems Theory–EUROCAST 2011*, páginas 426–432. Springer, 2012b.

NISSAN (2014) (). Entrada: "Safety Shield". Disponible en `http://www.nissan-global.com/EN/DOCUMENT/PDF/TECHNOLOGY/safety_activities/safety_activities_E-2.pdf` (último acceso, Mayo, 2014).

NISTÉR, D., NARODITSKY, O. y BERGEN, J. Visual odometry. En *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 1, páginas I–652. IEEE, 2004.

NODA, M., TAKAHASHI, T., DEGUCHI, D., IDE, I., MURASE, H., KOJIMA, Y. y NAITO, T. Vehicle ego-localization by matching in-vehicle camera images to an aerial image. En *Computer Vision–ACCV 2010 Workshops*, páginas 163–173. Springer, 2011.

OPENSTREETMAPS (). Entrada: "Página principal de OpenStreetMaps". Disponible en `http://www.openstreetmap.org/` (último acceso, Mayo, 2014).

ORGANIZACIÓN MUNDIAL DE LA SALUD (2013) (). Entrada: "Las diez principales causas de muerte en el mundo según la OMS". Disponible en `http://www.rpp.com.pe/2013-08-27-las-diez-principales-causas-de-muerte-en-el-mundo-segun-la-oms-fot625895_2.html#foto` (último acceso, Mayo, 2014).

OTSU, N. A threshold selection method from gray-level histograms. *Automatica*, vol. 11(285-296), páginas 23–27, 1975.

PERREAULT, S. y HÉBERT, P. Median filtering in constant time. *Image Processing, IEEE Transactions on*, vol. 16(9), páginas 2389–2394, 2007.

PINK, O. Visual map matching and localization using a global feature map. En *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*, páginas 1–7. IEEE, 2008.

POINTGREY, 2014 (). Entrada: "Bumblebee stereo vision camera systems". Disponible en `http://www.ptgrey.com/products/bumblebee2/bumblebee2_xb3_datasheet.pdf` (último acceso, Mayo, 2014).

QUDDUS, M. A., OCHIENG, W. Y. y NOLAND, R. B. Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation Research Part C: Emerging Technologies*, vol. 15(5), páginas 312–328, 2007.

RACC. Los costes de la accidentalidad en la Red de Carreteras del Estado. `http://consejosconducir.racc.es/es/coste_victimas`, 2013.

RACC, A. C. Estudio eurotest. comparativa europea de pasos de peatones. 2008.

RECURSO WEB:EROSIÓN (). Entrada: "Recurso web:Erosión". Disponible en `http://alojamientos.us.es/gtocoma/pid/tema5-1.pdf` (último acceso, Mayo, 2014).

SÁNCHEZ J.M. (2014) (). Entrada: "El coche autónomo de Google: el futuro se topará con la ley y los seguros". Disponible en `http://www.abc.es/tecnologia/informatica-hardware/20140529/abci-google-coche-autonomo-201405281636.html` (último acceso, Mayo, 2014).

SCHARSTEIN, D. y SZELISKI, R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, vol. 47(1-3), páginas 7–42, 2002.

SE, S. y BRADY, M. Road feature detection and estimation. *Machine Vision and Applications*, vol. 14(3), páginas 157–165, 2003.

SEBSADJI, Y., TAREL, J.-P., FOUCHER, P. y CHARBONNIER, P. Robust road marking extraction in urban environments using stereo images. En *Intelligent Vehicles Symposium (IV), 2010 IEEE*, páginas 394–400. IEEE, 2010.

SICHELSCHMIDT, S., HASELHOFF, A., KUMMERT, A., ROEHDER, M., ELIAS, B. y BERNS, K. Pedestrian crossing detecting as a part of an urban pedestrian safety system. En *Intelligent Vehicles Symposium (IV), 2010 IEEE*, páginas 840–844. IEEE, 2010.

SMART LIFE (2014) (). Entrada: "Renault next two, el coche autónomo". Disponible en http://cincodias.com/cincodias/2014/05/19/motor/1400508083_590137.html (último acceso, Mayo, 2014).

UDDIN, M. S. y SHIOYAMA, T. Robust zebra-crossing detection using bipolarity and projective invariant. En *ISSPA*, páginas 571–574. 2005.

VINCENTY, T. (). Entrada: "Direct and inverse solutions of geodesics on the ellipsoid wlth application of nested equations". Disponible en http://www.ngs.noaa.gov/PUBS_LIB/inverse.pdf (último acceso, Mayo, 2014).

VIOLA, P. y JONES, M. Robust real-time object detection. *International Journal of Computer Vision*, vol. 4, páginas 34–47, 2001.

WANG, S. y TIAN, Y. Detecting stairs and pedestrian crosswalks for the blind by rgbd camera. En *Bioinformatics and Biomedicine Workshops (BIBMW), 2012 IEEE International Conference on*, páginas 732–739. IEEE, 2012.

WIKIPEDIA (Convolución). Entrada: "Convolución". Disponible en http://es.wikipedia.org/wiki/Convoluci%C3%B3n (último acceso, Mayo, 2014).

WIKIPEDIA (Cross ratio). Entrada: "Cross ratio". Disponible en http://en.wikipedia.org/wiki/Cross-ratio (último acceso, Mayo, 2014).

WIKIPEDIA (Matlab). Entrada: "Matlab". Disponible en http://es.wikipedia.org/wiki/MATLAB (último acceso, Mayo, 2014).

WIKIPEDIA (Sistema de posicionamiento global). Entrada: "Sistema de posicionamiento global". Disponible en http://es.wikipedia.org/wiki/Sistema_de_posicionamiento_global#DGPS_o_GPS_diferencial (último acceso, Mayo, 2014).

WIKIPEDIA (Unidad de medición inercial). Entrada: "Unidad de medición inercial". Disponible en http://es.wikipedia.org/wiki/Unidad_de_medici%C3%B3n_inercial (último acceso, Mayo, 2014).

WORLD HEALTH ORGANISATION (2013) (). Entrada: "Global status report on road safety". Disponible en http://www.who.int/violence_injury_prevention/road_safety_status/2013/en/ (último acceso, Mayo, 2014).

YU, S. y SHI, J. Object-specific figure-ground segregation. En *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 2, páginas II–39. IEEE, 2003.

YUILLE, A. L. Deformable templates for face recognition. *Journal of Cognitive Neuroscience*, vol. 3(1), páginas 59–70, 1991.