UNIVERSIDAD CARLOS III DE MADRID

INGENIERÍA INDUSTRIAL

PROYECTO FIN DE CARRERA

DESARROLLO DE UN SISTEMA AVANZADO DE ASISTENCIA A LA CONDUCCIÓN EN TIEMPO REAL PARA LA DETECCIÓN DE PEATONES EN ENTORNOS URBANOS COMPLEJOS

Autora: Natalia Morán Cruz
Tutor: Basam Musleh Lancis

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

Leganés, Marzo de 2013

En primer lugar, agradezco a D. Basam Musleh, el tutor de este proyecto, haberme dado a conocer este campo de mi especialidad, así como por haberme ofrecido su ayuda en todo momento. Por otro lado, también quiero darle las gracias a mi familia y amigos que han tenido tanta paciencia, no sólo durante la elaboración del PFC si no en todos los años que ha durado mi periodo académico.

RESUMEN

Hoy en día, todavía son muchas las personas que pierden la vida en accidentes de tráfico. A pesar de que los vehículos que se desarrollan ahora son mucho más seguros que los que se fabricaban en un principio, también son más rápidos y más numerosos. Los nuevos sistemas de seguridad van más allá de la implantación de materiales que absorban mejor los impactos o componentes que reduzcan las lesiones de los ocupantes del vehículo producidas durante el accidente. En la actualidad, los ingenieros de la industria automovilística tratan de evitar que se produzcan estos accidentes.

El presente proyecto se centra en proteger a los peatones de las vías urbanas, pues son los mayores afectados en los accidentes producidos en este tipo de carreteras. El objetivo es diseñar un algoritmo basado en la visión estéreo capaz de detectar a los usuarios de vías urbanas complejas de forma rápida y precisa de tal forma que el conductor tenga constancia en todo momento de los peatones que se encuentran delante de su vehículo.

La técnica que se va a utilizar para la localización de los peatones se basa en los histogramas de gradientes orientados (HOG). Se trata de un método que ofrece resultados robustos gracias a su invariancia ante cambios en la iluminación, en el fondo o en las posturas de los peatones. Mediante una serie de operaciones previas se busca conseguir que esta detección se realice en tiempo real. Es necesario realizar un estudio de los distintos parámetros del sistema para alcanzar buenos resultados tanto en el tiempo de cómputo del algoritmo como en la eficacia de la detección. Para el desarrollo del algoritmo se recurre a las librerías OpenCV, muy útiles para el procesamiento de imágenes y la visión artificial.

Palabras clave: peatones, visión estéreo, HOG, uv_disparity.

ABSTRACT

Nowadays, there are still many people who lose their lives in crash accidents. Although modern vehicles are safer than older ones, they are faster and more numerous. New security systems go further than only the introduction of materials that could better absorb the impacts or components that can reduce injuries to the passengers of the vehicle during an accident.

The aim of this project is to protect pedestrians in urban roads, because they are the most affected people in accidents that occur in this kind of roads. An algorithm has been designed based on stereovision, which is able to detect people walking in complex urban roads in a quick and precise manner, so the driver can be aware at every moment of the pedestrians that are in front of his or her vehicle.

The technic used to locate pedestrians is based on Histograms of Orientated Gradients. This method offers strong results thanks to its invariance with light changes, backgrounds or the pedestrians postures. By means of some previous operations, detection will be in real time. It is necessary to make a survey of the different parameters of the system to get good results, as much in the algorithm calculation time as in the efficacy of detection. The OpenCV libraries has been used to develop the algorithm, as they are very useful for processing images and artificial vision.

Keywords: pedestrians, stereo visión, HOG, uv_disparity.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

LISTA DE SIGLAS Y ACRÓNIMOS

ABS	**A**ntilock **B**racking **S**ystem.
ACC	**A**daptative **C**ruise **C**ontrol.
ADAS	**A**dvanced **D**river **A**ssistance **S**ystem.
AGV	**A**utomatic **G**uided **V**ehicle.
Blob	**B**inary **L**arge **Ob**ject.
DGT	**D**irección **G**eneral de **T**ráfico.
GPS	**G**lobal **P**ositioning **S**ystem.
HOG	**H**istograms of **O**riented **G**radients.
HSV	**H**ue, **S**aturation, **V**alue.
iCab	**I**ntelligent **Ca**mpus Automo**b**ile.
ISA	**I**ntelligent **S**peed **A**daptation.
IVIS	**I**n-**V**ehicle **I**nformation **S**ystem.
IvvI	**I**ntelligent **V**ehicle based on **V**isual **I**nformation.
lidar	**Li**ght **D**etection **a**nd **R**anging.
LKA	**L**ane **K**eeping **A**ssistance.
OpenCV	**Open** Source **C**omputer **V**ision Library.
PCS	**P**re-**C**rash **S**ystem.
ROI	**R**egion **o**f **I**nterest.
RGB	**R**ed, **G**reen, **B**lue.
SIFT	**S**cale **I**nvariant **F**eature **T**ransform.

Capítulo 1

INTRODUCCIÓN

1.1. INTRODUCCIÓN

En 1886, Karl Benz inventó el automóvil. Al principio, su precio era tal que muy pocos podían permitirse el lujo de poseer uno; además, la velocidad máxima que alcanzaba era de 20 km por hora. Por ambas razones, los accidentes eran poco frecuentes. Sin embargo, los automóviles comenzaron a popularizarse en 1910 debido a que Henry Ford introdujo la producción en cadena. Esto hizo que la fabricación de los coches fuera más rápida y sencilla lo que propició un abaratamiento de los mismos, con lo que aumentó el número de personas que podían permitirse la compra de uno o más vehículos para uso particular. El rápido avance de la sociedad y de la tecnología durante estas últimas décadas, así como el aumento de población, ha dado lugar a un importante incremento del número de vehículos que circulan por las carreteras, lo que ha dado lugar a un incremento del número de accidentes.

Son miles las personas que fallecen por accidente de tráfico. Sólo en España, según la Dirección General de Tráfico [1] se produjeron un total de 85.503

accidentes con víctimas en el año 2010 (último año estudiado) donde el número de víctimas fue de 122.823, de las cuales 2.478 fallecieron, 11.995 fueron heridos graves y 108.350 heridos leves. Sin embargo, en la Tabla 1.1 se puede apreciar que las cifras de fallecidos y heridos graves se han reducido considerablemente con respecto a las de principios de siglo. En 2010 fallecieron 3.039 personas menos en accidentes de circulación que en 2001, suponiendo una reducción del 55%. Este mismo porcentaje se encuentra a la hora de comparar los heridos graves, siendo la diferencia entre estos años de 14.571 heridos graves. La cifra de heridos leves ha descendido un 12% durante este periodo, registrándose 14.683 heridos leves menos.

AÑOS	ACCIDENTES CON VICTIMAS	FALLECIDOS	HERIDOS GRAVES	HERIDOS LEVES	TOTAL
2001	100.393	5.517	26.566	123.033	155.116
2002	98.433	5.347	26.156	120.761	152.264
2003	99.987	5.399	26.305	124.330	156.034
2004	94.009	4.741	21.805	116.578	143.124
2005	91.187	4.442	21.859	110.950	137.251
2006	99.797	4.104	21.382	122.068	147.554
2007	100.508	3.823	19.295	123.226	146.344
2008	93.161	3.100	16.488	114.459	134.047
2009	88.251	2.714	13.923	111.043	127.680
2010	85.503	2.478	11.995	108.350	122.823

Tabla 1.1: Evolución de las cifras de accidente de tráfico y siniestralidad durante el periodo 2001-2010 tanto en carretera como en zona urbana.

Las causas más habituales de accidente de tráfico son los factores climatológicos, el factor mecánico pero, sobre todo, los errores humanos, ya que cualquier distracción o pérdida del control del vehículo puede dar lugar a una colisión con graves efectos tanto para las personas que se encuentran en el interior del vehículo como para los localizados en las cercanías del mismo. Los fabricantes buscan reducir este número de accidentes desarrollando vehículos más seguros así

como eficientes y confortables. En un principio, esta seguridad se basaba en la fabricación de vehículos que protegieran a las pasajeros ante un impacto (seguridad pasiva); sin embargo, el gran avance de la tecnología en estos últimos años ha permitido llevar esta seguridad mucho más allá, diseñando sistemas que reduzcan la probabilidad de impacto o sus consecuencias si éste se produce (seguridad activa). Con el fin de incrementar la seguridad global del vehículo durante la conducción surgieron los sistemas ADAS (Advanced Driver Assistance System) [2], sistemas capaces de interactuar con el entorno del vehículo y de actuar sobre el mismo, permitiendo en la actualidad que se hable de vehículos inteligentes.

1.2. OBJETIVO DEL PROYECTO

De los datos de la DGT [1] también se puede extraer que en el periodo 2001-2010, la mayor reducción de fallecidos se produjo en las carreteras, un 9%, mientras que en las zonas urbanas fue de un 6%. Los peatones son los más afectados por los accidentes de tráfico con víctimas en zona urbana. El 50,5% de los fallecidos fueron peatones, es decir, 278 peatones murieron por atropello en el año 2010, resultando, por otro lado, 1.586 heridos graves y 7.841 leves. Si se comparan estas cifras con las del año 2001, se deduce que el descenso de víctimas no se ha reducido de forma tan destacada como en el caso general, siendo, por ejemplo, el número de peatones fallecidos este año de 377. A continuación se presenta la Figura 1.1 y la Figura 1.2 que muestran la evolución de las cifras de peatones fallecidos y heridos en el periodo comprendido entre el 2001 y el 2010:

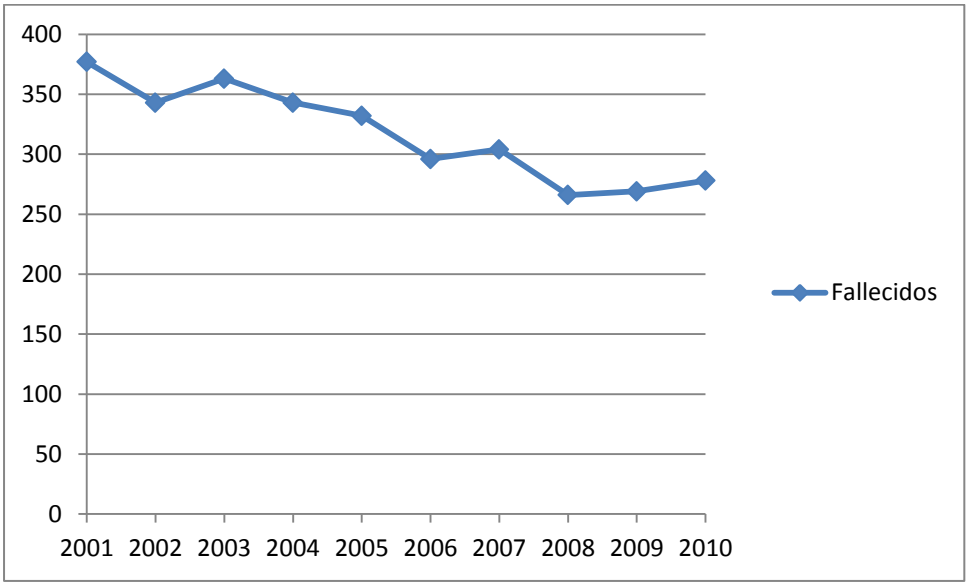


Figura 1.1: Evolución de las cifras de peatones fallecidos en zonas urbanas en el periodo comprendido entre los años 2001-2010.

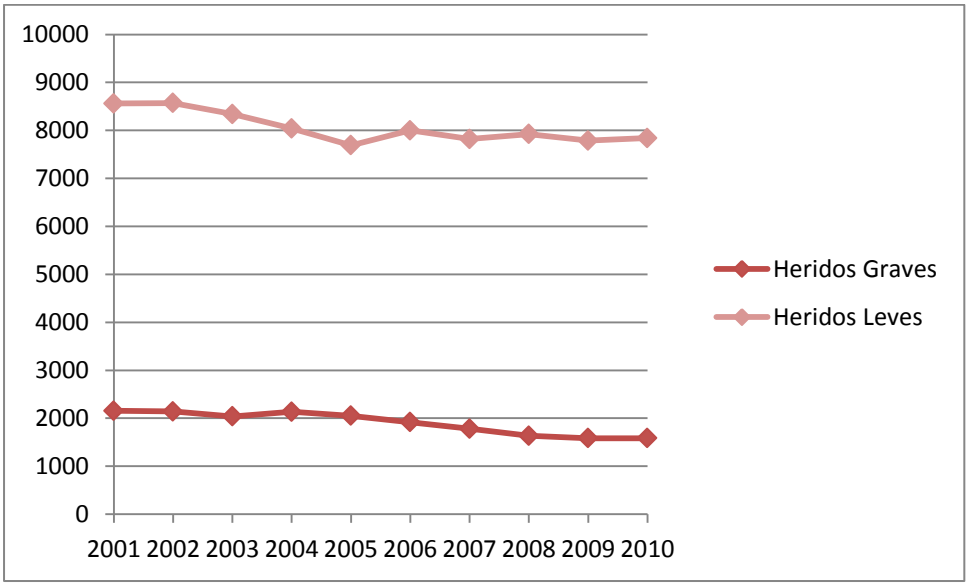


Figura 1.2: Evolución de las cifras de peatones que han sido heridos de forma leve o grave en zonas urbanas en el periodo comprendido entre los años 2001-2010.

Este leve descenso puede ser debido a que los fabricantes de vehículos se han centrado hasta hace poco en proteger a las personas integrantes del vehículo, mientras que son los peatones la parte más indefensa en la zona urbana pues, en el caso de un atropello, se enfrentan a un golpe directo sin carrocería que les proteja.

Debido al elevado número de víctimas mortales por atropello, en la actualidad se están diseñando vehículos pensando también en el resto de los viandantes, mediante la elaboración de airbags externos y carrocerías más flexibles o elevables. Con esto simplemente se consigue reducir en parte las lesiones sufridas por el peatón. Los últimos estudios realizados van más allá, ideando sistemas que indiquen al conductor la presencia de peatones en las cercanías del vehículo. De esta manera, el conductor es consciente de la situación de peligro con la suficiente anterioridad como para poder reaccionar en condiciones de seguridad reduciendo la velocidad. Los estudios realizados por la DGT recogen que un atropello producido a 70 km/h o más, supone generalmente la muerte segura del peatón, mientras que a 50 km/h el riesgo de muerte se reduce y a 30km/h queda demostrado que se evitan tres de cada cuatro atropellos.

El objetivo del presente proyecto es el desarrollo de un algoritmo que permita detectar peatones en el entorno de un vehículo que circula por una zona urbana compleja mediante un sistema estéreo instalado en el interior del vehículo. Dicho algoritmo se va a encargar de diferenciar aquellos peatones que se encuentren próximos al vehículo de aquéllos que se encuentren más alejados y que no corran el riesgo de ser atropellados. También va a permitir conocer las coordenadas reales de cada uno de los peatones con respecto al vehículo, de tal manera que el conductor tenga plena conciencia de la situación de los mismos. Se busca que el tiempo de computación sea el menor posible, ya que se trabaja con requisitos de tiempo real y cuanto antes se obtengan los resultados, antes podrá actuar el conductor ante un posible atropello.

El algoritmo ha sido desarrollado en los lenguajes de programación C/C++, basándose en las librerías OpenCV [3] en el entorno de desarrollo integrado Microsoft Visual Studio C++.

Capítulo 2

ESTADO DEL ARTE

El Estado del Arte del presente proyecto se centra en los sistemas de seguridad que presentan los vehículos que se encuentran en el mercado y aquellos que se están desarrollando actualmente para reducir el índice de mortalidad existente en las carreteras. También se hace referencia a los vehículos capaces de desplazarse de modo completamente autónomo ya existentes y a las plataformas de investigación llevadas a cabo en la Universidad Carlos III de Madrid.

2.1. SEGURIDAD EN EL AUTOMOVIL

2.1.1. SEGURIDAD PASIVA Y SEGURIDAD ACTIVA

Debido a las elevadas cifras de accidentalidad que presentan los automóviles, los fabricantes y diseñadores de estos productos a la hora de vender un vehículo no se centran sólo en características como la velocidad o la ergonomicidad sino que también emplean la seguridad del propio vehículo como modo de captar la atención del comprador. Los vehículos van equipados con una

serie de sistemas tanto internos como externos que protegen a los integrantes del mismo de posibles accidentes. Sin embargo, la tendencia actual es proteger también a las personas que se encuentran fuera del vehículo pero que también pueden formar parte de la incidencia, como pueden ser peatones, ciclistas, motoristas, etc. En materia de seguridad de un vehículo se puede hablar de seguridad pasiva y de seguridad activa.

- **SEGURIDAD PASIVA:**

Son los elementos que tratan de minimizar los posibles daños de los integrantes del vehículo en el caso de que el accidente sea inevitable, es decir, no evitan el accidente pero sí reducen sus consecuencias. Dentro de este grupo se engloban los cinturones de seguridad, los airbags, los reposacabezas, los cristales así como el chasis y la carrocería capaces de absorber la mayor energía posible en el impacto. Más adelante se explica como se están diseñando estas carrocerías para que protejan al peatón en caso de que se produzca un atropello.

- **SEGURIDAD ACTIVA**

Con el progreso de la tecnología de estos últimos años es posible encontrar una solución a los factores humanos que son los principales causantes de accidentes hoy en día, gracias al uso de elementos electrónicos y programas informáticos. La seguridad activa integra todos aquellos elementos que incorporan los vehículos destinados a disminuir el riesgo de que se produzca un accidente. Entre ellos se pueden distinguir los sistemas de frenado, dirección y suspensión con control electrónico, la iluminación, los amortiguadores y los neumáticos. Los sistemas que los fabricantes han desarrollado en las últimas décadas para mejorar la seguridad activa son el antibloqueo de frenos y ruedas (ABS), la tracción total o los controles de estabilidad (ESP) y tracción.

El Reglamento General de Circulación informa del uso obligatorio de algunos de estos elementos que forman parte de la seguridad activa y pasiva del vehículo tales como los cinturones de seguridad. Estas infracciones son castigadas económicamente pues no sólo se pone en peligro la vida de los ocupantes del

automóvil sino que también se está jugando con la integridad de aquellos que se encuentran en la vía, ya sean pasajeros de otros vehículos o viandantes. Con la correcta utilización y conservación de estos elementos pueden reducirse en gran medida los accidentes y sus posibles consecuencias.

2.1.2. SISTEMAS ADAS

La utilización de vehículos es cada vez mayor. Esto es debido a que la utilización de éstos ofrece ante todo comodidad a la hora de desplazarse de un sitio a otro, lo que genera un aumento de la movilidad tanto de pasajeros como de mercancías. Las ventajas de estos medios de transporte son numerosas, sin embargo, también presentan importantes inconvenientes. Por ejemplo, se puede destacar lo contaminantes que son para el medio ambiente y el peligro que conlleva su uso, ya que además de ser muy pesados, circulan a velocidades muy elevadas, por lo que si se produce un impacto con otro vehículo o con un elemento cercano a la vía los daños que se puedan ocasionar pueden ser importantes. Los mayores perjudicados en una colisión en la que se vea implicado uno o más vehículos van a ser las personas integrantes de los vehículos, pero, sobre todo, como ya se ha mencionado anteriormente, los peatones. Es por ello, que se trata sin descanso de acondicionar al vehículo con piezas y sistemas que lo hagan más seguro para todos los usuarios de la vía.

Es el sector de la industria del automóvil el encargado de estudiar la manera de aumentar la seguridad de los vehículos para poder llegar al día en el que la cifra de accidentes de vehículos sea nula. Para ello surge lo que se conoce como Sistemas Avanzados de Asistencia al Conductor, ADAS (Advanced Driver Assistance System) [2]. Estos sistemas tratan de desarrollar tecnologías inteligentes aplicadas al automóvil con el propósito tanto de aumentar la seguridad en las carreteras consiguiendo reducir el número de accidentes, como de facilitar la conducción para que ésta sea más confortable. Los ADAS pueden cubrir un completo rango de sistemas que varían desde sistemas que proporcionan información o avisos, hasta sistemas que intervienen en el control y en las tareas

de maniobra del vehículo. Estos sistemas pueden llegar a prevenir hasta un 40% [4] los accidentes de tráfico, dependiendo del tipo de sistema y del tipo de escenario. A pesar de este dato, la introducción de los ADAS en el mercado está siendo lenta por el incremento que suponen en el coste del vehículo. En Europa, los países más implicados en este campo son Suecia, Holanda y Reino Unido. Sin embargo, España tiene margen de mejora en materia de seguridad vial para llegar a estar entre los mejores países europeos.

Entre los ADAS se pueden distinguir sistemas destinados a dar soporte a varios aspectos de la conducción proporcionando información, comúnmente conocidos como IVIS (In-Vehicle Information Systems). Ejemplos de IVIS son los sistemas de navegación y los sistemas que proporcionan información del tráfico o de las condiciones de la carretera, tales como los receptores RDS-TMC que se describirán más adelante. Por otro lado, se encuentran los sistemas de alerta o reacción, cuyo objetivo es reducir los errores humanos. Algunos ejemplos son el sistema de Adaptación Inteligente de la Velocidad, los sistemas de alerta de colisión longitudinal, los sistemas de alerta de abandono del carril y los asistentes de cambio de carril. Los medios de alertar al conductor pueden ser auditivos, visuales o por contacto, como por ejemplo mediante la vibración del asiento o del volante. Otro tipo de ADAS son aquellos que intervienen en el control del vehículo pero sin suplantar completamente al conductor, como lo harían los vehículos denominados “vehículos autónomos” [5], en el que el conductor deja de controlar el vehículo en su totalidad. A continuación se va a pasar a describir algunos de los sistemas de asistencia a la conducción existentes para entender mejor la utilidad que proporcionan. Algunos de estos son creados para mejorar la seguridad de las personas, mientras que otros, simplemente se encargan de mejorar la comodidad de los pasajeros.

ASISTENCIA AL APARCAMIENTO

Los sistemas de ayuda al aparcamiento llevan tiempo en el mercado, sin embargo, cada vez se van incorporando a más modelos. Este tipo de asistencia facilita la maniobra de aparcamiento que en numerosas ocasiones terminan siendo una tarea complicada, ahorrando tiempo al conductor y permitiendo estacionar el vehículo en huecos más reducidos en los que para el conductor sería complicado si no dispusiese de ayuda.

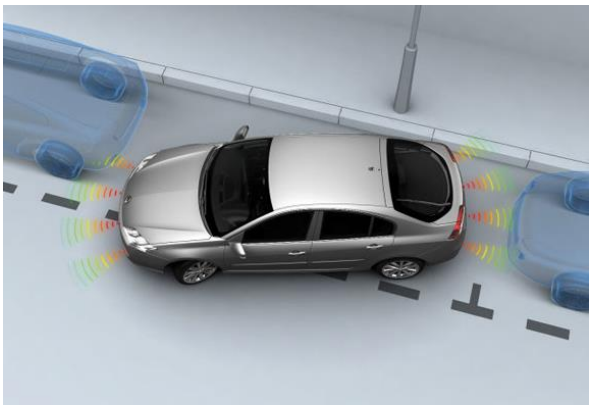


Figura 2.1: Sistema de asistencia al aparcamiento [7].

Los hay de distintos tipos, según la marca y la gama del vehículo. Los sistemas más sencillos, conocidos como "*parktronic*" [6] avisan mediante una señal acústica de la proximidad del vehículo trasero al que nos acercamos. Muchos ya incluyen la opción de detectar también los objetos que se tienen en la parte delantera del vehículo (Figura 2.1) [7]. El sistema consiste en usar señales de ultrasonidos que miden la distancia al objeto determinado en función del tiempo de vuelo. Los vehículos más sofisticados, como en el caso de Lexus, Audi o Mercedes, incluyen una cámara de visión trasera que ofrece, a diferencia del *parktronic*, una visión completa de la parte trasera del vehículo. Existe la opción de que el vehículo reduzca su velocidad o frene si existe la posibilidad de que se produzca un golpe con alguno de los elementos que delimitan la zona de aparcamiento. Lo más innovador en el estacionamiento asistido consiste en que mediante sensores de ultrasonido el mismo vehículo localiza una plaza de

aparcamiento y realiza las maniobras precisas, siendo únicamente necesaria la implicación del conductor para el accionamiento de los pedales[8].

SISTEMA DE NAVEGACIÓN MEDIANTE GPS Y TECNOLOGÍA TMC

El Canal de Mensajes de Tráfico (TMC, Traffic Message Channel) [9] consiste en un sistema estándar europeo para la recepción de tráfico en tiempo real. El sistema se basa en la transmisión de mensajes a través del sistema de radio FM. Para su utilización se requiere únicamente un receptor GPS dotado de sintonizador de radio y un procesador RDS encargado de recibir y descodificar los mensajes transmitidos a través de la señal de radio FM y enviarlos a un software que soporte esta tecnología. Mediante este sistema se obtiene información en tiempo real relativa a accidentes de tráfico, retenciones, estado del tráfico, posibles desvíos y cortes por obras. Esta información además de ser transmitida al conductor para que sea consciente de la situación, será interpretada por el propio dispositivo y actuará según el problema, por ejemplo modificando la ruta seguida.

SISTEMAS DE VISIÓN NOCTURNA

De noche o cuando las condiciones meteorológicas son adversas se reduce la visibilidad y, a veces, los faros no proporcionan la iluminación suficiente, por lo que la circulación se vuelve más compleja y peligrosa, sobre todo cuando se conduce por vías no urbanas. Estos sistemas de visión nocturna muestran a través de una pantalla situada en la consola (Figura 2.2) [10] lo que sucede en el campo de movimiento del vehículo cuando las condiciones de iluminación son reducidas, disminuyendo el número de atropellos y accidentes. En la actualidad, estos sistemas se ofrecen como

Figura 2.2: Sistema de Visión Nocturna [10].

equipamiento opcional en las gamas altas de determinadas marcas de vehículos, existiendo dos tipos: sistemas activos y sistemas pasivos. Por un lado, están los sistemas activos empleados por marcas como Mercedes-Benz o Toyota, que emplean dispositivos de luz infrarroja, instalados en los automóviles. Los sistemas pasivos, por otro lado, captan la radiación térmica emitida por los objetos mediante el empleo de una cámara termográfica. Las marcas que emplean estas cámaras en sus sistemas de visión nocturna son: BMW, Audi, General Motors y Honda. La ventaja de este último sistema frente al empleado por Mercedes es que transmite únicamente la información más importante, evitando distraer al conductor.

CONTROL ADAPTATIVO DE LUCES

El control de iluminación adaptativo ofrece una mejora del campo de visión de hasta un 70% [11] en comparación con los sistemas de faros convencionales ya que, como su propio nombre indica, las luces se van adaptando para proporcionar una mejor visibilidad al conductor. Intervienen dos tecnologías: las luces dinámicas en curva y las luces de giro estáticas. El sistema de iluminación dinámica en curva gira los faros en la dirección de la marcha según el grado de giro del volante de tal forma que va iluminando siempre dentro de las líneas de la carretera. Esto se consigue mediante el giro de los faros de xenón. Por otro lado, en el caso del segundo tipo mencionado, las luces se encienden automáticamente cuando se activa el indicador de dirección durante un cierto periodo de tiempo a velocidad reducida e iluminan en la nueva dirección de la marcha.

ADAPTACIÓN INTELIGENTE DE LA VELOCIDAD

Un gran número de accidentes de tráfico son debido a que los vehículos circulan por encima de los límites de velocidad establecidos, ignorando las señales de tráfico de la vía por la que circulan. Controlando la velocidad del vehículo se

podría reducir en gran medida el número de estos accidentes y la gravedad de los que se producen.

Los sistemas de adaptación de la velocidad son conocidos como ISA (Intelligent Speed Adaptation) [12]. Estos sistemas determinan la posición del vehículo a través del sistema GPS, obteniendo la información correspondiente a la vía por la que circula, determinando, entre otras cosas, los límites establecidos. Esta información es transmitida al conductor, advirtiéndole si este límite está siendo rebasado. También existe la posibilidad de que el propio vehículo sea el encargado de ajustar la velocidad por debajo de la permitida, bien actuando sobre los frenos, reduciendo el acelerador y/o ajustando la mezcla de combustible, dependiendo del sistema ISA.

CONTROL DE VELOCIDAD DE CRUCERO ADAPTATIVO (ACC)

No guardar una distancia de seguridad adecuada con el vehículo precedente también es causa de numerosos accidentes, ya que un frenazo o una maniobra incorrecta va a afectar al vehículo de detrás tanto más cuanto menor sea la distancia que los separa. Por este motivo surge la tecnología del Control de velocidad de crucero adaptativo (ACC, Adaptive Cruise Control) [13] cuyo objetivo es detectar automáticamente, con la ayuda de un sistema radar, la velocidad del vehículo y su distancia con respecto al vehículo que lo precede. Si en un momento dado se detecta que esta distancia no es suficiente para mantener las condiciones de seguridad se reduce la velocidad del vehículo, actuando sobre el sistema de frenos, pudiendo llegar a frenarlo por completo. De esta forma, se mantiene en todo momento la distancia de seguridad que haya sido programada. Una vez que el carril queda libre, el ACC acelera el vehículo hasta la velocidad prefijada.

ASISTENCIA DE MANTENIMIENTO DE TRAYECTORIA EN EL CARRIL (LKA)

Un informe publicado por ANFAC RESEARCH [14] estima que entre un 10% y un 18% de los accidentes que se producen hoy en día se podrían evitar

disponiendo de un sistema que sea capaz de detectar la desviación de la trayectoria de un vehículo.

El sistema LKA (Lane Keeping Assistance) detecta que el vehículo traspasa una línea, ya sea continua o discontinua, que delimita el carril. Esto puede deberse a una desviación de la trayectoria sin haber activado los intermitentes para indicar el cambio de carril o a una posible salida de la vía. Al producirse alguna de las acciones comentadas, el vehículo puede responder emitiendo una señal acústica y/o haciendo vibrar el volante o el asiento del conductor. También existe la opción de que el sistema actúe levemente sobre la dirección en sentido opuesto de tal manera que el vehículo quede fijado sobre el carril.

Figura 2.3: Sistema de Mantenimiento de Trayectoria en el Carril [15].

Existen varios tipos de sistemas LKA. Uno de ellos consiste en una cámara montada en la zona del retrovisor interior [15]. Las imágenes tomadas por dicha cámara son analizadas de manera continuada, detectando las marcas existentes sobre la vía e identificando aquellas que corresponden a líneas delimitantes de la vía. Una vez diferenciadas estas líneas, estudia la posición del vehículo respecto a las mismas (Figura 2.3).

Como algunos otros de los sistemas que se mencionan, este sistema puede ser activado o desactivado según desee el conductor en cada momento. La opción debe ser activada antes de iniciar el viaje y su funcionamiento se limita a velocidades superiores a 65 km/h, es decir, en vía interurbana y por autopista.

DETECCIÓN DE ÁNGULO MUERTO

Se trata de un sistema de control de vehículos que permite evitar situaciones de peligro o colisiones con otros vehículos durante un cambio de carril por culpa de la falta de visibilidad en lo que es conocido como ángulo muerto. Su precursor es el fabricante Volvo [16]. Existen dos tecnologías distintas para la detección de los vehículos localizados en los puntos críticos representados en la Figura 2.4, el procesado de imágenes proveniente de una cámara y tecnologías basadas en sensores radar de corto/medio alcance. Esta última tecnología consiste en un par de sensores de radar localizados en el parachoques trasero a cada uno de los lados del vehículo. Estos sensores de radar se encargan, a velocidades superiores a 60 km, de medir la distancia y la velocidad relativa de los vehículos que circulan en las proximidades. En el caso de que se detecte la presencia de un vehículo, sea visto o no por el conductor, el sistema va a avisar al conductor de este hecho inmediatamente mediante la iluminación de un piloto.

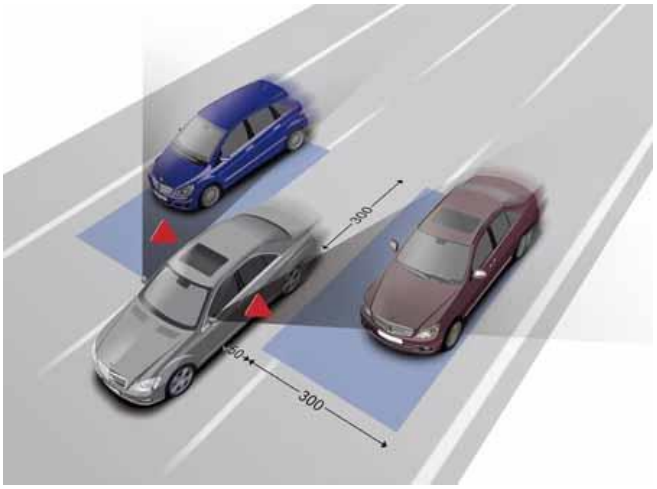


Figura 2.4: Sistema de Detección del Ángulo Muerto.

SISTEMAS DE IDENTIFICACIÓN DE SEÑALES

Esta tecnología consiste en detectar las señales de tráfico de la carretera e identificarlas para informar al conductor de la prohibición o advertencia que muestran las mismas [17]. Consiste en una cámara que capta las imágenes de la carretera y mediante un tratamiento digital realizado por un ordenador es capaz

de identificar la señal en la imagen e interpretarla para a continuación mostrarla al conductor a través de un interfaz situado en el salpicadero, como se muestra en la Figura 2.5. También puede tratarse de una aplicación más activa en la que el coche interviene dependiendo de la orden que muestre la señal.

Figura 2.5: *Display* con la información obtenida sobre las señales de tráfico de la vía, Sistema de Identificación de Señales.

SISTEMA DE PREVENCIÓN DE COLISIONES

El Pre-Crash System (PCS) [18] es un proyecto de Toyota que consiste en un sensor de radar de ondas milimétricas que realiza constantes envíos de información sobre los obstáculos localizados delante del vehículo. Este sistema determina la probabilidad de colisión según cuales sean las condiciones de conducción, la distancia al vehículo delantero y las velocidades de cada vehículo (Figura 2.6). En el caso de que exista un riesgo de colisión, se alerta al conductor mediante señales ópticas y/o sonoras o, incluso, se inicia la acción de frenado de forma automática.

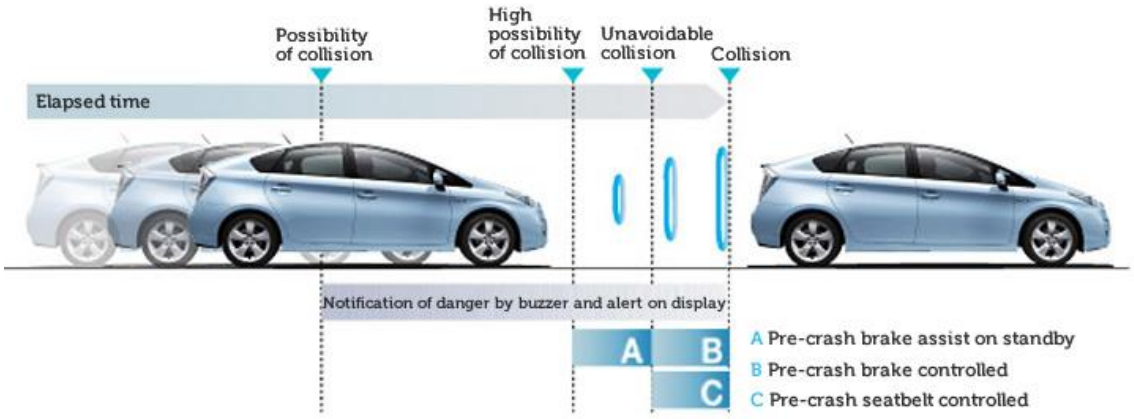


Figura 2.6: Pre-Crash System de Toyota (PCS) [18].

El Control de Crucero Adaptativo funciona en conjunto con el sistema PCS, dando lugar a un método eficaz para proteger a los pasajeros del vehículo.

SISTEMA DE PROTECCIÓN DE PEATONES

Hasta ahora se han visto tecnologías que ayudan, principalmente, a proteger a las personas que se encuentran en el interior del vehículo. Sin embargo, las personas que se ven más afectadas en el caso de verse implicadas en una colisión son las que no se encuentran protegidas por una carrocería, como pueden ser ciclistas, motoristas y peatones. En caso de que en un accidente se vean implicados peatones, estos son los que salen peor parados ya que no hay ningún tipo de estructura que los proteja. Un 15% [19] de las muertes que se producen por accidente en Europa son peatones.

Los fabricantes de vehículos tratan de desarrollar sistemas que eviten los atropellos o que reduzcan su efecto en caso de que sea inevitable. Las carrocerías, actualmente, son diseñadas para que en caso de colisión se deformen fácilmente, absorbiendo más cantidad de energía. Otra solución para la protección de peatones es un sistema actuador que eleva, entre otras cosas, el capó del motor de la forma que aparece en la Figura 2.7 [20]. Mediante sensores se detecta una colisión con un objeto y gracias a algoritmos inteligentes se diferencia el tipo de objeto,

distinguiendo si se trata de un peatón o no. Sólo en el caso de que el objeto que ha colisionado sea un peatón, el capó se elevará para mitigar el golpe. También han sido diseñados airbags externos que funcionarían de forma similar a los internos, reduciendo en gran medida el golpe recibido por el peatón ya que está impactando contra un elemento mucho más blando que lo que puede llegar a ser el frontal del vehículo.

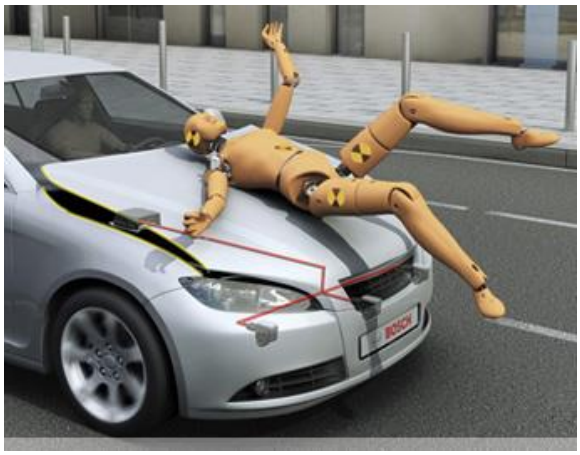


Figura 2.7: Capó elevable para la protección del peatón [20].

Sin embargo, con estos sistemas no se evita que el peatón sufra lesiones, sólo las reduce. Por ello, los estudios actuales se centran más en el desarrollo de sistemas que permitan evitar el golpe del peatón contra el vehículo o mitigar el golpe reduciendo su velocidad, ya que un impacto contra un obstáculo a 40 km/h en vez de a 60 km/h reduce en más del 50% la energía cinética [21]. Estos sistemas consisten en que mediante una cámara y algoritmos que procesen las imágenes captadas, el vehículo identifique la presencia de peatones en su trayectoria, alerte con tiempo al conductor y provoque una frenada de peligro automática en el caso de que el conductor no sea capaz de evitar la colisión, con la intención de quitar al vehículo toda la inercia posible y proteger al peatón. La cámara facilita al mismo tiempo otras funciones de asistencia al conductor que ya han sido tratadas, como pueden ser el reconocimiento de señales de tráfico, alerta de alcance y alerta al abandonar el carril. La seguridad de los peatones es el tema que se trata en este proyecto, por lo que este sistema será desarrollado con mayor precisión más adelante.

2.2. IMPLEMENTACIÓN DE LOS SISTEMAS ADAS

Cada vez son más las marcas que incorporan algunos de los sistemas ADAS comentados a sus modelos. Existen sistemas de adaptación al conductor exclusivos de las gamas consideradas de lujo, pero hay otros que se encuentran implementados en modelos más asequibles. A continuación, se comentan algunos de los sistemas de los que disponen las marcas de vehículos más innovadoras en este ámbito:

BMW:

La firma de automóviles BMW destaca sobre otras compañías líderes en materia de innovación y creatividad. Dispone de un sistema denominado BMW ConnectedDrive que consiste en una combinación de aplicaciones online, asistencia al conductor y soluciones de integración de dispositivos móviles por los cuales se mantendrá conectado el vehículo con el mundo exterior en todo momento. En cuanto a seguridad, BMW ofrece sistemas de asistencia visual como BMW Night Vision [22], ya mencionado, y el Asistente de luz de carretera que adapta la luz de los faros del vehículo según las necesidades. También existen los sistemas de asistencia al conductor que avisan de la salida del trayecto o de colisión.

FORD:

Ford presenta una tecnología basada en el Control de Velocidad Adaptable (ACC), que permite seleccionar la velocidad y la distancia que se quiere mantener con el vehículo de delante, de tal manera que si la velocidad del tráfico disminuye, el vehículo reduce la velocidad para conservar dicha distancia. La Advertencia de Colisión con Soporte de Frenado también está incluida con ACC, permitiendo alertar al conductor mediante un piloto si detecta una posible colisión con el vehículo delantero. En el caso de que la colisión sea inminente, aumenta la sensibilidad de la asistencia de frenado para que la respuesta sea más rápida. También dispone de un sistema de Asistencia Activa para Estacionar que busca lugares disponibles y a continuación estaciona el vehículo en paralelo de forma

automática. Por último mencionar que Ford proporciona a sus clientes un Sistema de Información de Punto Ciego denominado BLIS® [23], solucionando el problema presente en los vehículos convencionales conocido como punto muerto.

VOLVO:

Por su parte, Volvo ofrece la tecnología IntelliSafe [16] en la que destaca el sistema City Safety. El City Safety está diseñado para prevenir colisiones; procesa el tráfico que se encuentra delante pudiendo evitar una colisión cuando la velocidad sea igual o inferior a 15 km/h o reducir su severidad si ésta es mayor. Volvo también dispone de un Detector de Peatones que consiste en una unidad radar en la parrilla del coche que detecta los obstáculos localizados delante del vehículo y una cámara al lado del espejo retrovisor interior que determina de qué tipo de obstáculo se trata. Con este sistema se pueden detectar peatones de, como mínimo, 80 cm de altura.

2.3. VEHÍCULOS AUTÓNOMOS

La intención de fabricantes de vehículos como BMW o Mercedes-Benz es la de ir introduciendo gradualmente en el mercado distintas aplicaciones que doten al vehículo de autonomía. Sin embargo, otros fabricantes han ido más allá desarrollando un vehículo de conducción completamente automática.

Desde hace tiempo existen vehículos industriales útiles para el transporte de mercancía que no requiere de un conductor para su circulación, es decir, se mueve de manera automática. Estos vehículos reciben el nombre de AGV (en inglés Automatic Guided Vehicle) y mediante un sistema de guiado, ya sea por sensores o por cámaras, se garantiza el desplazamiento de materiales sin que interfiera un operario. Sin embargo, lo que se busca ahora es la adaptación de los coches con el fin de que se guíen de forma totalmente autónoma. Existen diversos proyectos con dicho objetivo, los más populares son el que está llevando a cabo Google y el de la universidad alemana Freie Universität de Berlín.

VEHÍCULO AUTÓNOMO DE GOOGLE

El *Google Driverless Car* [24] es un proyecto muy avanzado llevado a cabo por Google equipado con una tecnología que permite que un vehículo circule sin la necesidad de que lo controle una persona. Este proyecto lo dirige en la actualidad el ingeniero de Google Sebastian Thrun, director del Laboratorio de Inteligencia Artificial de Standford y co-inventor de Google Street View.

Estos coches han sido diseñados para evitar los errores humanos a partir de la idea de que un equipo informatizado reacciona mejor y más rápidamente que un ser humano, pues este último puede sufrir distracciones, situaciones de somnolencia o alguna otra adversidad durante la conducción. Además de aumentar la seguridad en las carreteras reduciendo el número de accidentes, estos vehículos realizan una conducción más eficiente desde el punto de vista energético. Actualmente, disponen de ocho vehículos con la tecnología automática: seis Toyota Prius, un Audi TT y un Lexus RX450.

En 2010 fue lanzado un prototipo capacitado para guiarse mediante los propios mapas de Google Maps. El automóvil recorrió el Estado de California, realizando un total de 225.000 kilómetros sin ser conducido por una persona, únicamente bajo supervisión. El Estado de Nevada permitió, en junio de 2011, circular por sus calles automóviles sin conductor. El Departamento del Estado de Vehículos a Motor (DMV) de Nevada ha aprobado, recientemente, la licencia de circulación del primer vehículo autónomo que corresponde a un Toyota Prius.

El funcionamiento del coche con piloto automático de Google se basa en cámaras de vídeo, sensores de radar, rayos láser, y una base de datos de la información que va recogiendo in situ sobre los vehículos que circulan por la carretera y guiándose mediante los mapas de Google Maps. En la Figura 2.8 [25] se muestran algunos de los componentes que hacen posible la conducción autónoma. Uno de los sensores se encuentra situado en el techo del vehículo y consiste en un sensor de rotación *lidar* con un alcance de 70 metros en todas direcciones (360 grados), proporcionando un preciso mapa tridimensional del entorno del vehículo.

El coche dispone de tres radares frontales y uno en la parte trasera encargados, también, de detectar los objetos cercanos al vehículo con una menor resolución, pero con mayor alcance. En una de las ruedas se localiza otro sensor que mide pequeños movimientos realizados por el vehículo y ayuda a localizar de forma precisa su posición en el mapa. Una cámara frontal delantera instalada cerca del retrovisor detecta las luces de los vehículos que se aproximan o se alejan y ayuda al ordenador de a bordo del coche a reconocer obstáculos en movimiento como peatones o ciclistas. La interpretación de los datos sensoriales está estrechamente ligado al problema de la localización del vehículo (o SLAM), que necesita conocer la localización del vehículo con una precisión mayor a la que ofrece un GPS hoy en día, por lo que para este fin se emplean sistemas que combinan GPS y RTK (Real Time Kinematic).

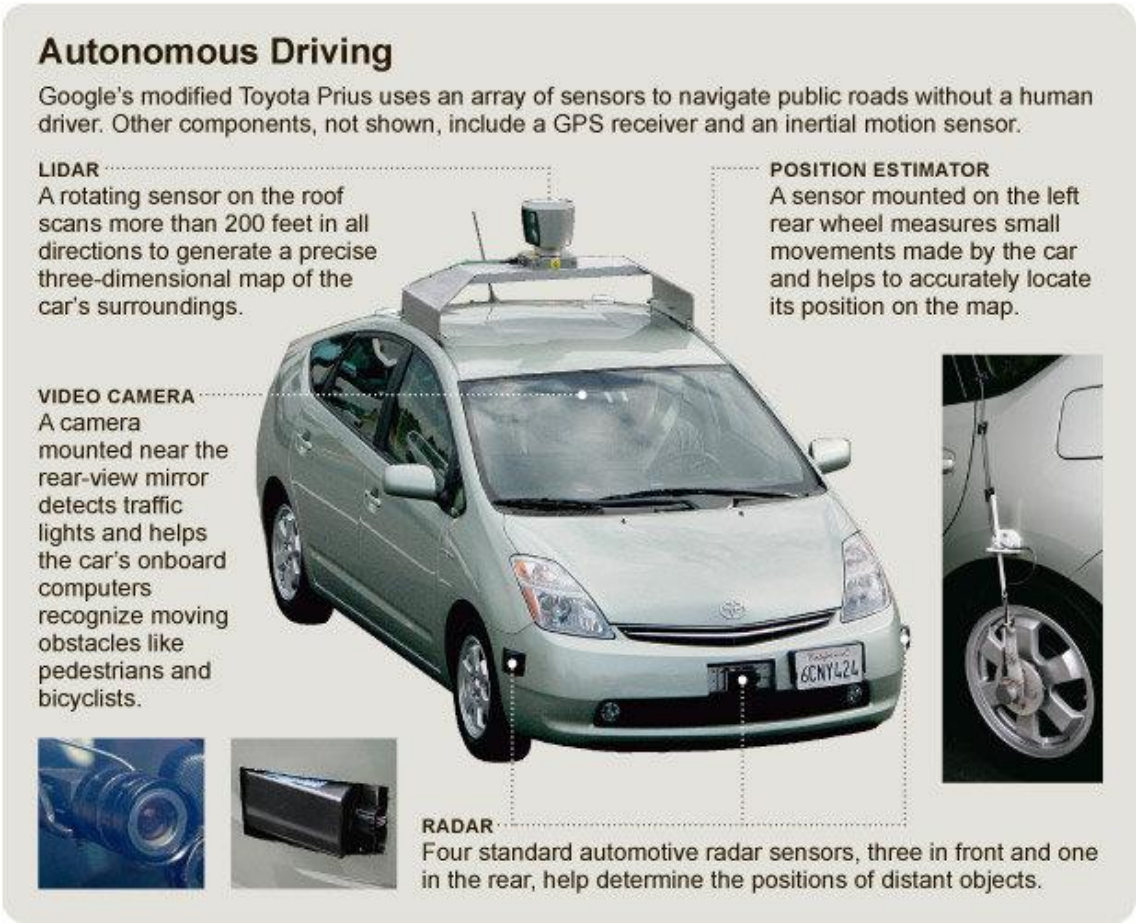


Figura 2.8: Componentes para el funcionamiento autónomo del *Google Driverless Car* [25].

VEHÍCULO AUTÓNOMO MIG

Científicos de la Freie Universität de Berlín (FU) llevan desarrollando desde el año 2006 un vehículo equipado con una avanzada tecnología que le permite circular de forma autónoma. Los investigadores afirman que en un futuro serán los coches sin conductor los que compongan el tráfico corriente de las ciudades, aumentando la seguridad humana, protegiendo el medioambiente y transformando el paisaje, en general.

El coche, apodado MIG (Made in Germany) [26], está equipado con cámaras, escáneres láser, sensores de calor y navegación por satélite que permiten que el vehículo detecte otros vehículos, peatones y otros elementos de la carretera, así como reconocer el estado de un semáforo, las señales de tráfico, ect. La información captada por estos instrumentos es enviada a un ordenador de abordo que calculará a partir de los datos un modelo tridimensional del entorno para evaluar la situación del tráfico.

Sus creadores afirman que un coche manejado de forma automática es más seguro que si lo conduce un ser humano, ya que, por ejemplo, estos vehículos pueden ver en todas direcciones con un alcance de 70 metros, cosa que no pueden hacer los humanos sin la ayuda de espejos adicionales. Por otro lado, el medioambiente se verá beneficiado con la aparición de este tipo de vehículos debido a que se reducirá el número de automóviles en las carreteras al poder ser éstos compartidos por diversos pasajeros.

Figura 2.9: coche autónomo de la Freie Universität de Berlín “Spirit of Berlin”.

En 2007, los investigadores de esta universidad crearon el modelo "Spirit of Berlin" (Figura 2.9) que formó parte, junto al vehículo de Google, en la competición organizada por la Agencia de Investigaciones Avanzadas del Ejército Norteamericano, DARPA [27] Urban Challenge donde sólo participan vehículos autónomos.

2.4. PLATAFORMAS DE INVESTIGACIÓN DE LA UNIVERSIDAD CARLOS III DE MADRID

La Universidad Carlos III de Madrid dispone de dos tipos de vehículos adaptados empleados para probar ante casos reales los algoritmos desarrollados por el Laboratorio de Sistemas Inteligentes (LSI) [28]. Estos vehículos son conocidos como IvvI y como iCab.

IvvI

El vehículo IvvI (acrónimo en inglés de Vehículo Inteligente basado en Información visual) es un vehículo real que dispone de una serie de cámaras ópticas, infrarrojas y láser que permiten recoger e interpretar la información en tiempo real del entorno por el que circula, percibiendo otros vehículos, obstáculos y la carretera. Se trata de una plataforma de experimentación para desarrollar Sistemas de Ayuda a la Conducción.

Se encuentra equipado con una serie de sistemas que permiten reaccionar ante una serie de situaciones de peligro. Los sistemas que se distinguen en este vehículo son:

- Sistema estéreo en blanco y negro de barrido progresivo que permite capturar imágenes en movimiento, evitando los problemas inherentes al vídeo entrelazado.
- Cámara a color para la percepción de cierta información, como puede ser la ofrecida por las señales de circulación.

- Cámara de infrarrojo lejano que capta el calor desprendido por ciertos obstáculos: peatones u otros vehículos.
- Cámara orientada enfocando al conductor para evaluar el grado de atención del mismo.
- GPS que proporciona información sobre la orientación y velocidad del vehículo al resto de sistemas de percepción.
- Iluminación infrarroja que permite trabajar de noche sin molestar al resto de ocupantes del vehículo.
- Dos ordenadores situados en el maletero del vehículo para el procesamiento de la información captada por los sistemas de visión por computador.
- Convertidor DC/AC conectado directamente a la batería del vehículo para obtener la alimentación eléctrica necesaria para el funcionamiento de los equipos y sensores.

Figura 2.10: Plataforma de investigación IvvI 2.0 de la Universidad Carlos III de Madrid.

El IvvI 2.0 (Figura 2.10) es el segundo vehículo adquirido por la universidad con el equipamiento necesario para desarrollar Sistemas de Ayuda a la Conducción. Se diferencia del primer vehículo en que además de emplear equipos más modernos, presenta tanto los ordenadores como los sensores o monitores integrados en el vehículo.

iCab

El iCab (Intelligent Campus Automobile) es un proyecto ideado para implementar un Sistema Inteligente de Transporte (SIT). Consiste en un vehículo eléctrico de la marca EZGO (Figura 2.11) al cual se le han realizado una serie de modificaciones tanto en la tracción como en la dirección para conseguir su control automático, por medio de un ordenador, o manual, mediante un joystick y unos pedales. Gracias a un sistema de visión estereoscópica, otro de visión infrarroja, un telémetro láser y un sistema GPS con sensores inerciales, este vehículo puede conocer e interactuar con entornos estructurados.

Figura 2.11: Plataforma de investigación iCab de la Universidad Carlos III de Madrid.

Capítulo 3

FUNDAMENTOS TEÓRICOS

En el presente capítulo se van a introducir los conceptos teóricos a los que se ha recurrido durante el desarrollo del algoritmo. En posteriores capítulos se profundizará más en alguno de estos conceptos.

3.1. TECNOLOGÍAS PARA LA OBTENCIÓN DE LA INFORMACIÓN

Como ya se vio en el apartado anterior, los sistemas de asistencia a la conducción pueden basarse en dos tipos de tecnología a la hora de obtener la información. Existen sistemas que se basan en tecnologías láser o radar y otros, en cambio, se basan en visión por computador. Algunos sistemas emplean ambas tecnologías para que la información extraída del medio sea lo más completa posible. Los láser, más concretamente los láser *lidar* [29], son muy empleados en los sistemas ADAS debido a su gran precisión y resolución de sus medidas. Sin embargo, la cantidad de información es reducida si la comparamos con la que es capaz de proporcionar un sistema de visión estéreo, ya que en estos últimos la información ofrecida es en 3D y los *lida*r, en cambio, sólo captan información en 2D lo que dificulta la tarea de clasificación de los objetos detectados.

En este proyecto se recurre a un sistema estéreo, ya que aporta mayor cantidad de información, trabaja bien ante oclusiones y son robustos a cambios de iluminación. Sin embargo, nos encontramos con la dificultad de que este algoritmo está destinado a funcionar en entornos urbanos, caracterizados por su gran variedad de elementos. A esto hay que añadirle la problemática de que en los entornos urbanos existe una importante falta de texturas y los patrones son muy repetitivos, por tanto la cantidad y complejidad de la información que debe procesar el algoritmo va a ser muy grande, dificultando que dicho procesamiento se realice en tiempo real.

3.2. ÓPTICA

Antes de pasar al tema de la visión estéreo se van a definir brevemente los dos modelos que permiten la captación de una imagen. Estos dos modelos son el modelo de lente fina y el modelo *pin-hole*.

3.2.1 MODELO DE LENTE FINA

Este modelo consiste en una lente de grosor despreciable y biconvexa que permite recoger la luz de una escena y proyectarla sobre una superficie llamada *plano de formación de la imagen*. Esta lente dispone de un eje de simetría denominado *eje óptico* (Figura 3.1). El *centro óptico* es el punto interior a la lente donde corta el eje óptico al plano de simetría de la lente. Sucede que todo haz de luz que pasa por el centro óptico continúa en línea recta sin cambiar de dirección. Todos aquellos haces paralelos que incidan perpendiculares al plano de simetría de la lente se acaban cortando en un punto llamado *foco*. La distancia que separa el centro óptico de la lente y el foco se denomina *distancia focal* (*f*).

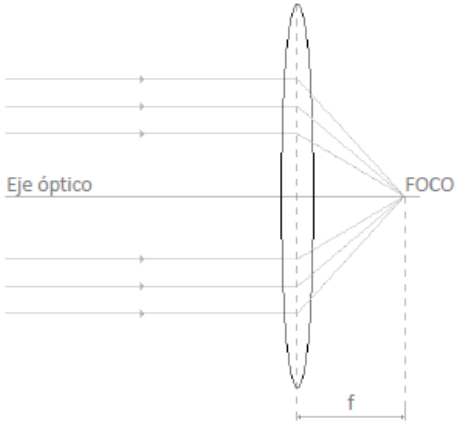


Figura 3.1: Trayectoria seguida por la luz al atravesar una lente fina (Modelo de Lente Fina).

3.2.2. MODELO *PIN-HOLE*

Una cámara *pin-hole* es una cámara muy simple que carece de lente. Consiste en una caja que protege una pieza de película fotográfica colocada en la parte trasera del interior de ésta. Esta caja dispone en la parte delantera de una pequeña apertura, de ahí su nombre. Este orificio es la única entrada de luz que permite la caja. Cuando la luz de una imagen pasa a través de este agujero se forma una imagen invertida en la película (Figura 3.2) [30]. El ojo humano trabaja de la misma forma. La distancia focal (f) en este caso será la distancia desde el punto pin-hole a la imagen.

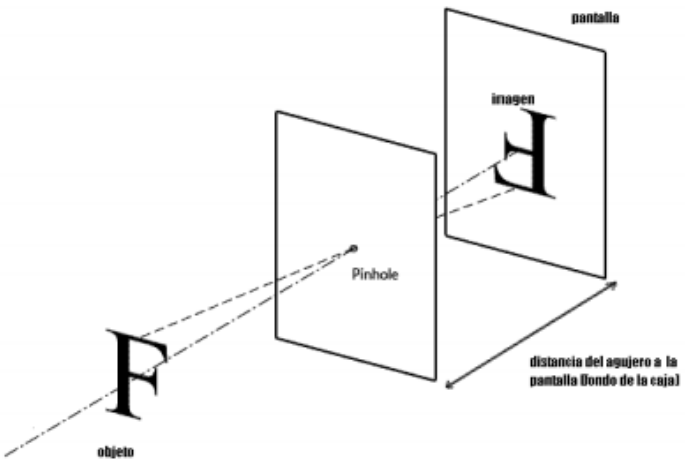


Figura 3.2: Funcionamiento del Modelo *pin-hole* [30].

Para la obtención de una imagen adecuada, interesa que el orificio sea igual o inferior al 1% de la distancia entre el orificio y la película fotográfica.

El modelo pin-hole describe la relación existente entre las coordenadas de un punto en 3D y su proyección en el plano de la imagen. Las ecuaciones (1) y (2) se obtienen por triangulación a partir de los parámetros que se indican en la Figura 3.3.

$$x = \frac{f}{Z} \cdot X \tag{1}$$

$$y = \frac{f}{Z} \cdot Y \tag{2}$$

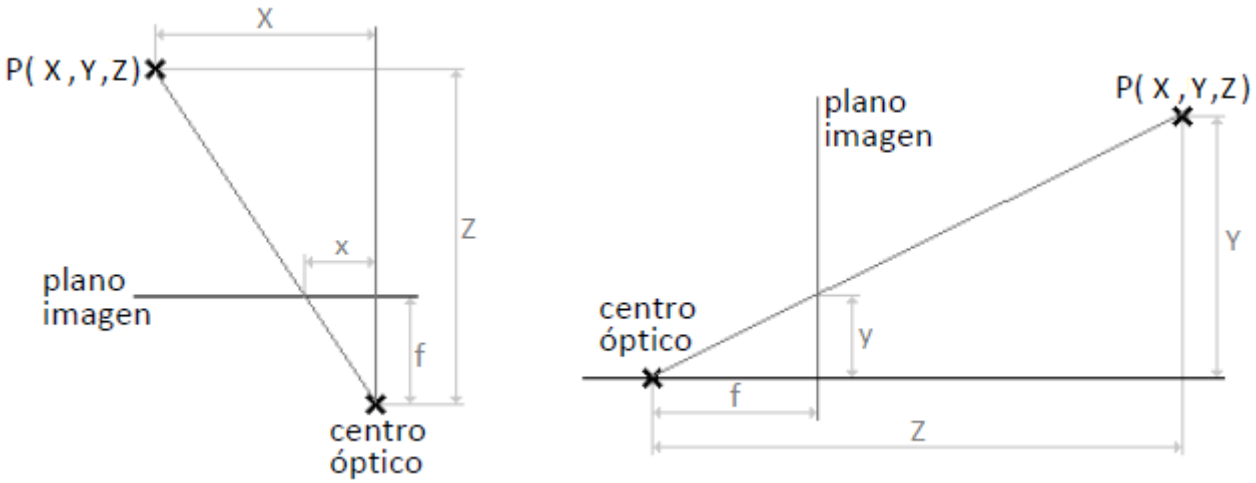


Figura 3.3: Parámetros del modelo *pin-hole*.

A continuación se pasa a detallar el funcionamiento de la visión estéreo en la que se basa este proyecto.

3.3. VISIÓN ESTÉREO

La visión estéreo se basa en el sistema de captar las imágenes que poseen los seres humanos, en el que, debido a la posición de los ojos y a la forma de moverlos, las imágenes que se reciben en cada ojo son prácticamente iguales;

únicamente se producen pequeñas diferencias en la posición relativa de los objetos, que es lo que se conoce como disparidad. El cerebro procesa las diferencias entre ambas imágenes y las interpreta de forma que se percibe la sensación de profundidad, lejanía o cercanía de los objetos que se encuentran a nuestro alrededor, reconstruyendo la estructura tridimensional de la escena. Lo mismo ocurre en el tratamiento de imágenes basado en la visión estéreo; en él se analizan las diferencias de la proyección de la escena en dos imágenes tomadas desde dos posiciones diferentes. A estas diferencias existentes entre ambas imágenes se les denomina también disparidad y su análisis permite obtener la dimensión perdida en la proyección de la escena tridimensional en la imagen bidimensional.

Para la captación de imágenes se emplea la cámara Bumblebee2 del fabricante Point Grey [31]. Esta cámara consiste en un sistema formado por un par de cámaras *pin-hole* alineadas horizontalmente, separadas entre sí una determinada distancia y con las que se obtiene las imágenes correspondientes al par estéreo.

El término visión estéreo se refiere a la habilidad de recuperar la estructura tridimensional a partir de dos imágenes de la misma escena. Un sistema estéreo convencional se caracteriza por estar formado por dos cámaras cuyos ejes ópticos se encuentren en paralelo y separados por una distancia horizontal denominado *baseline* (*B*) tal y como se representa en la Figura 3.4.

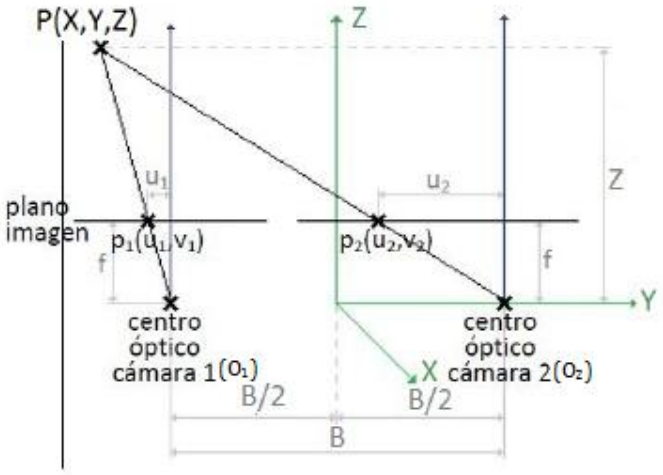


Figura 3.4: Modelo *pin-hole* para un sistema estéreo.

Desde un punto de vista computacional, un sistema de visión estéreo debe resolver dos tipos de problemas. El primero de ellos es conocido como *correspondencia*. Partiendo de dos imágenes bidimensionales, izquierda y derecha, el problema de la correspondencia trata de localizar qué dos puntos, $p_1(u_1,v_1)$ de la imagen izquierda y $p_2(u_2,v_2)$ de la imagen izquierda, corresponden a un mismo punto P(X,Y,Z). El segundo problema que un sistema estéreo debe resolver es la *reconstrucción*, cuyo objetivo consiste en encontrar las coordenadas del punto P. Para calcular tanto la profundidad Z como las coordenadas X, Y del punto P se recurre a las ecuaciones del modelo *pin-hole* (1) y (2) a partir de las cuales se obtienen las ecuaciones del sistema estéreo (3), (4), (5).

$$X = Z\,\frac{u1}{f} - \frac{B}{2} = Z\,\frac{u1}{f} + \frac{B}{2} \tag{3}$$

$$Y = \frac{B \cdot v1}{d} \tag{4}$$

$$Z = \frac{f \cdot B}{u1 - u2} = \frac{f \cdot B}{d} \tag{5}$$

En estas expresiones, f representa la distancia focal y d el correspondiente valor de disparidad dado por $d = u_1 - u_2$.

3.3.1. GEOMETRÍA DEL SISTEMA ESTÉREO

Un sistema estereoscópico se basa en la geometría epipolar, la cual permite relacionar objetos tridimensionales con sus correspondientes proyecciones en la imagen. La Figura 3.5 muestra la geometría de un par de cámaras estéreo. En dicha figura se representa el *plano epipolar* como aquel que forman los dos centros ópticos O_1 y O_2 de las cámaras con cualquier punto P del objeto. Este plano corta a las dos superficies *imagen 1* e *imagen 2* en las correspondientes *líneas epipolares* que se definen como las líneas que unen un mismo punto en la escena en las imágenes derecha e izquierda. Finalmente, la proyección del centro óptico de cada cámara sobre la otra cámara define el llamado *epipolo*. Los epipolos de cada una de las cámaras, e_1 y e_2, son los puntos por los que van a pasar cada una de las líneas epipolares.

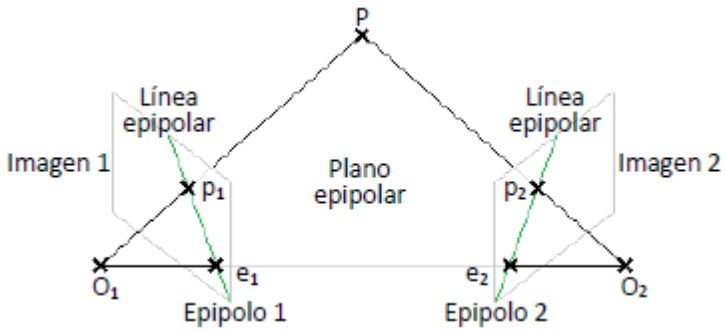


Figura 3.5: Representación de la geometría epipolar de un sistema estéreo.

3.4. TÉCNICAS PARA LA DETECCIÓN DE OBJETOS

El proceso de detección de obstáculos sigue dos objetivos: obtener la detección efectiva de los posibles obstáculos en la imagen y determinar la distancia hasta ellos para a partir de ahí tomar las correspondientes decisiones.

La detección de objetos es una tarea laboriosa y más cuando se tratan de escenarios tales como vías urbanas donde hay gran cantidad de objetos de

características muy variadas, los entornos son muy diversos y las condiciones de iluminación son cambiantes, por lo que la intensidad de las imágenes varía en función de la luz. Son muchas las técnicas para la detección de objetos en una imagen o en una sucesión de las mismas. A continuación se hace una breve introducción de cada una de ellas.

3.4.1. DETECTORES DE PUNTOS

Se emplean para localizar puntos de interés en la imagen, siendo estos puntos de interés aquellos que presentan texturas destacables en zonas específicas de la imagen. Estas técnicas destacan por ser invariantes a la iluminación y puntos de vista de la cámara.

3.4.2. SUSTRACCIÓN DEL FONDO

Consiste en encontrar las variaciones entre fotogramas a partir de una representación de la escena llamada modelo de fondo. Cualquier cambio en la imagen corresponde a un objeto en movimiento. Los píxeles que varían de un *frame* a otro son marcados para el procesamiento posterior.

3.4.3. SEGMENTACIÓN

Consiste en dividir la imagen en regiones con atributos similares, con el fin de poder aislar los distintos elementos que la componen y extraer aquellos que nos resulten de interés. Se destacan a continuación las técnicas de segmentación más relevantes para la detección de objetos:

3.4.3.1. Mean-Shift Clustering

Consiste en una técnica de clasificación iterativa, no paramétrica. Su principal ventaja es que se supone el conocimiento de toda la información necesaria y suficiente para la clasificación, por lo que no requiere a priori el conocimiento del número de clases (clusters). Se puede definir como la descomposición de una imagen en regiones homogéneas, las cuales

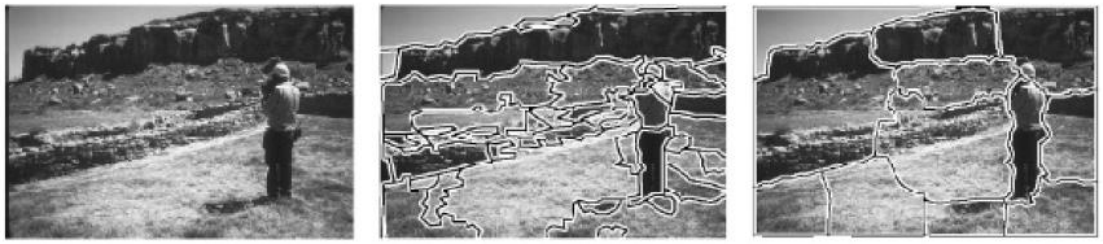
comparten características similares. Mediante esta segmentación se consigue minimizar el número de máximos locales en la imagen dejando únicamente aquellos más significativos. Aquellos píxeles que se encuentren en la vecindad de estos máximos locales serán asociados a ellos, dando lugar a las regiones homogéneas. En la Figura 3.6 (b) se muestra el resultado de haber aplicado a la Figura 3.6 (a) esta técnica de segmentación.

3.4.3.2. Graph-Cuts

En este caso la segmentación es formulada como un problema de partición de un grafo, siendo los vértices los píxeles y el grafo la imagen. Los vértices son particionados en un número de subgrafos (regiones), a través de la eliminación de los bordes ponderados del grafo. El peso se calcula a partir de la similitud existente en color, brillo o textura entre los nodos. Se puede ver un ejemplo del resultado de esta técnica en la Figura 3.6 (c).

3.4.3.3. Contornos Activos

Es una técnica de segmentación que consiste en evolucionar un contorno cerrado inicial adaptándolo al contorno del objeto que se desea detectar, es decir, trata de ajustar dicho contorno a la región del objeto. La evolución del contorno está gobernada por una función de energía que indica lo ajustado que se encuentra el contorno al borde del objeto. Es muy importante la inicialización del contorno, generalmente se sitúa fuera de la región del objeto y se comprime hasta localizar el borde del objeto.



(a) (b) (c)

Figura 3.6: Ejemplo de aplicación de dos de las técnicas de segmentación. (a) Imagen original. (b) Segmentación mediante la técnica mean-shift. (c) Segmentación mediante la técnica Graph-cuts.

3.4.4. APRENDIZAJE SUPERVISADO

La detección se puede realizar mediante el aprendizaje automático de las diferentes vistas de un objeto extraídas de un conjunto de ejemplos empleados como plantillas, es decir, a partir de un conjunto de ejemplos de aprendizaje se genera una función que relaciona las entradas con las salidas de interés. Se considera muy importante la correcta selección de las características o descriptores de los objetos que se van a tomar para la clasificación. Uno de los inconvenientes de este método es que para obtener resultados precisos, se requiere de un gran conjunto de ejemplos para la realización del aprendizaje. Cabe destacar las siguientes técnicas dentro de este ámbito:

3.4.4.1. Adaptive Boosting

Las técnicas de Boosting son métodos iterativos que obtienen clasificadores muy precisos mediante la combinación de muchos clasificadores base no tan precisos. Estos clasificadores base se distribuyen en grupos y, a su vez, estos grupos denominados “etapas” se enlazan formando una cascada y, actuando cada uno sobre las predicciones del anterior, dando lugar al clasificador final (Paul Viola y Michael Jones) [32].

3.4.4.2. Support Vector Machines (SVM)

Las Máquinas de Soporte Vectorial o Máquinas de Vectores de Soporte (SVM) son un conjunto de algoritmos de aprendizaje supervisado empleados para la clasificación y la regresión desarrollados por Vladimir Vapnik [33]. Dado un conjunto de ejemplos de entrenamiento (muestras) podemos etiquetar las clases y entrenar una SVM para construir un modelo que prediga la clase de una nueva muestra.

Tomando los datos de entrada como conjuntos de vectores en un espacio n-dimensional, una máquina de vectores soporte construirá un hiperplano de separación en ese espacio. Se considera que es mejor clasificador de datos aquel hiperplano que maximice la distancia (o margen) con los puntos que estén más cerca de él. Siendo los vectores de soporte los puntos que tocan el límite del margen. En el contexto que se está tratando en este proyecto de detección de objetos, las clases de datos corresponderán al objeto (muestras positivas), mientras que el resto de la imagen será tachada como muestras negativas.

Para separar linealmente los datos se procede a realizar un cambio de espacio mediante una función que transforme los datos de manera que se puedan separar linealmente. Esta función recibe el nombre de *Kernel*.

3.4.4.3. Redes Neuronales

Las redes neuronales son un método de reconocimiento de objetos basado en los sistemas neuronales biológicos a través de modelos matemáticos. Una red neuronal se compone de unidades que reciben el nombre de neuronas. Cada una de estas neuronas recibe una serie de entradas a través de interconexiones, proporcionando una salida.

3.5. RECONOCIMIENTO DE PEATONES

Una vez llevada a cabo la detección de los obstáculos se procede a la clasificación de los mismos según el objetivo buscado. En el caso de este proyecto, el interés reside en la detección de los peatones en la escena. A la hora de reconocer los objetos se puede recurrir a los *sistemas basados en detectores de formas* en los que se selecciona un conjunto de rasgos para discriminar los objetos que no son de interés con respecto de los que sí lo son, a pesar de que los fondos estén saturados o las condiciones de iluminación no sean óptimas. La otra opción es emplear *sistemas basados en descriptores*.

3.5.1. SISTEMAS BASADOS EN DESCRIPTORES DE FORMAS

El reconocimiento de formas consiste en el reconocimiento de patrones de los objetos, quedando éstos representados por una colección de descriptores. El punto esencial del reconocimiento de formas es la correcta clasificación. Mediante un mecanismo de extracción de características se extrae la información útil, eliminando la información redundante e irrelevante. Finalmente se lleva a cabo la etapa de toma de decisiones en la cual el sistema de reconocimiento asigna a cada objeto su categoría o clase.

3.5.2. SISTEMAS BASADOS EN DESCRIPTORES

Un descriptor representa una región de la imagen que ha sido previamente extraída. Cada región correspondiente a un objeto se caracteriza por un vector descriptor. Posteriormente, estos descriptores son evaluados por un clasificador que determina la presencia o ausencia del objeto buscado, en este caso, el peatón. El clasificador antes de su uso ha debido ser entrenado a partir de un conjunto de ejemplos, representados por el descriptor a clasificar. Entre los descriptores existentes destacan para la detección de peatones los basados en el análisis del componente principal (PCA), en wavelets, en histogramas de gradientes orientados (HOG) o los descriptores SIFT (Scale Invariant Feature Transform).

3.5.2.1. Descriptores Basados en el Análisis de Componentes Principales (PCA)

El análisis de componentes principales (o PCA, *Principal Component Analysis*) se caracteriza por reducir la dimensionalidad de un conjunto de rasgos hallando las causas de la variabilidad del conjunto de dicho conjunto y ordenándolas según su importancia, es decir, toma aquellos autovectores con los autovalores más altos, rechazando los de autovalores bajos pues los considera ruido. El objetivo será reducir el número de variables de la base de datos a un menor número perdiendo la menor cantidad de información posible.

3.5.2.2. Descriptores Wavelet de Haar

Estos descriptores permiten definir de manera robusta clases de objetos complejos, siendo invariantes a cambios de color y de textura. Entre los rasgos más empleados para la descripción de una persona, destacan estos descriptores. Presenta la capacidad de codificar rasgos tales como cambios de intensidades a diferentes escalas. La base de wavelet más sencilla es la de Haar que consiste en que se recorre la imagen con una ventana a la que se le aplican varios clasificadores en serie, cada uno más complejo que el anterior, los cuales usan las características para confirmar o descartar la hipótesis de que se trata del objeto buscado. Si la hipótesis se rechaza en cualquier nivel, el proceso no continúa pero si se confirma todos los filtros significará que se ha detectado el objeto deseado. Los patrones se consideran girados en varios posibles ángulos. Además, el algoritmo puede ejecutarse a varias escalas para obtener objetos de diferentes tamaños o de tamaño desconocido.

3.5.2.3. Descriptores SIFT

Los descriptores SIFT (Scale Invariant Feature Transform) consiste en un método para extraer características distintivas de una imagen en escala

de grises, de tal manera que pueda reconocer una misma característica entre diferentes vistas de un mismo objeto o escenas. El método fue diseñado por David G. Lowe [34]. SIFT proporciona un conjunto de características de un objeto. Se caracteriza de otros métodos de extracción de características en que las características extraídas son invariantes a la escala de la imagen y a la rotación. . También son muy resistentes a los efectos de "ruido" en la imagen pero poco robusto a cambios bruscos en la iluminación.

3.5.2.4. *Descriptores Basados en Histogramas de Gradientes Orientados*

Los descriptores HOG se basan en la orientación del gradiente en áreas locales de una imagen. Se obtienen dividiendo la imagen en celdas, calculando el histograma de la dirección del gradiente o de la orientación de los bordes en cada una de las celdas. Para mejorar la invarianza frente a cambios de iluminación o de sombras se realiza una normalización del contraste de cada uno de los histogramas. La combinación de estos histogramas representa el descriptor, el cual será entrenado para que sea capaz de distinguir a los peatones de entre todos los objetos encontrados. Más adelante se describen con más detenimiento estos descriptores, pues serán los empleados en el proceso de detección de peatones del presente proyecto.

3.6. SEGUIMIENTO DE OBJETOS

A través de la visión artificial, se ha logrado obtener métodos capaces de detectar elementos en movimiento así como saber qué dirección llevan mediante la detección de cambios en los distintos fotogramas que componen una secuencia de video. Movimientos perceptibles en la escena darán lugar a cambios en la secuencia de imágenes. Sin embargo, se trata de un proceso complejo debido a que cambios en la iluminación en los distintos fotogramas implicará cambios en los

niveles de intensidad de la imagen lo que puede dar lugar a considerarlo como un verdadero desplazamiento del objeto. También pueden surgir otros problemas tales como la aparición de ruidos en las distintas imágenes, movimientos complejos, naturaleza articulada de objetos, oclusión parcial y total del objeto de interés en un momento dado.

Para llevar a cabo el proceso de seguimiento se necesita tomar una característica determinada del elemento en cuestión. Se tienen las siguientes posibilidades:

- Color. Uso del espacio de color RGB o HSV.
- Bordes. La frontera de los objetos da lugar a cambios en la intensidad de la imagen. Son un buen recurso debido a que son menos sensibles a los cambios de iluminación.
- Flujo óptico. Se trata del campo de vectores de desplazamiento que define la traslación de cada pixel en una determinada región.
- Textura. Corresponde a la medida de la variación de intensidad de una superficie.

Existe la posibilidad de realizar el seguimiento del objeto mediante la detección inicial del objeto y a partir de ahí realizar el seguimiento, o bien, mediante su localización en cada uno de los *frames* del video. Son múltiples las técnicas que estiman la trayectoria de un objeto dentro del plano de una imagen en movimiento:

3.6.1. SEGUIMIENTO DE PUNTOS

El objeto es representado por un punto y su seguimiento se realiza asociando los puntos de un *frame* al siguiente. La asociación de estos puntos está basada en los estados previos de los objetos, proporcionando tanto la posición como el movimiento. Este método requiere de un mecanismo externo que detecte los objetos en cada uno de los *frames*. Dentro de esta técnica podemos distinguir entre seguimiento determinista o seguimiento probabilístico.

- *Método determinista por correspondencia*

Los métodos deterministas por correspondencia de puntos definen un coste de correlación del comportamiento del objeto en un *frame* con respecto al que presenta en el *frame* anterior. Este coste se define como una combinación de una serie de restricciones: proximidad, velocidad máxima, cambios de velocidad pequeños, movimiento común, rigidez y uniformidad por proximidad.

- *Métodos estadísticos por correspondencia*

Las medidas obtenidas de la secuencia de video contienen ruido. Además se puede dar que el movimiento del objeto se deba al movimiento de la cámara y no al cambio de posición de dicho objeto. Los métodos probabilísticos resuelven este problema considerando las observaciones y las incertidumbres del modelo para la valoración del estado del objeto en la imagen siguiente. Estos métodos utilizan el espacio de estados para modelar las propiedades del objeto tales como la posición, la velocidad y la aceleración. Las observaciones consisten normalmente en la posición del objeto dentro de la imagen, obteniéndola mediante mecanismos de detección. Entre los métodos estadísticos destacan el filtro de Kalman, filtro de partículas, filtro para la probabilidad conjunta de los datos y el seguimiento de múltiples hipótesis.

3.6.2. SEGUIMIENTO DEL *KERNEL*

El *kernel* hace referencia a la forma o apariencia con la que se representa el objeto de interés. Por ejemplo, el *kernel* puede ser una plantilla rectangular o una forma elíptica con un histograma asociado. Se realiza un seguimiento de los objetos mediante el cálculo del movimiento del *kernel* en *frames* consecutivos. Este movimiento se ve reflejado normalmente en forma de transformaciones paramétricas tales como translación, rotación y afines.

- *Seguimiento mediante el uso de modelos de apariencia multi-vista*

El modelado individual de objetos hace que no se tenga en consideración la interacción entre varios objetos y entre los objetos y el fondo en el transcurso del seguimiento. Un ejemplo de interacción puede ser la oclusión parcial o completa de un objeto por otro. Este método se implementa en la totalidad de la imagen y no sólo en las regiones de interés de la imagen.

- *Seguimiento basado en el uso de plantillas*

Los modelos basados en plantillas y densidad de la apariencia son muy utilizados por su relativa simplicidad y bajo coste computacional.

3.6.3. SEGUIMIENTO DE SILUETAS

El seguimiento se lleva a cabo por estimación de la región del objeto observado en cada *frame*. El método del seguimiento de la silueta usa la información recogida dentro de la región del objeto. Esta información puede encontrarse como densidad de la apariencia o de modelos de forma que son presentados normalmente como mapas de bordes. Dando los modelos de los objetos, el seguimiento de las siluetas se realiza bien por coincidencias en la forma o bien por la evolución del contorno. Existen dos métodos:

- *Evolución del contorno*

La evolución del contorno sigue un enfoque de minimización de energía. Al mover los puntos del contorno, la energía del modelo cambia, correspondiendo los mínimos a detecciones.

- *Correspondencia de forma* (*matching shape*)

Se trata de un método similar al método basado en el seguimiento mediante la correspondencia de plantillas, donde se busca la silueta de un objeto y su modelo asociado en cada *frame*.

Capítulo 4

DESARROLLO DEL ALGORITMO

A la hora de describir el algoritmo elaborado en este proyecto para la detección de peatones se van a considerar dos fases: una primera fase de determinación de las regiones que delimitan el área de cada uno de los obstáculos de interés de la imagen, conocidas como Regiones de Interés (ROI, Regions of Interest) y otra de análisis de dichas regiones para la detección de peatones.

4.1. DETECCIÓN DE OBSTÁCULOS

En esta primera fase se lleva a cabo la extracción de aquellos obstáculos que se encuentran en la vía urbana a estudio y que, por su cercanía al vehículo, resultan de interés por poder tratarse de peatones. Para la técnica de detección de los elementos localizados a una determinada distancia podría bastar el empleo de sensores con capacidad para medir estas distancias y que alerten al conductor en el caso de que esta distancia sea reducida. Sin embargo, la utilización de un sistema formado por cámaras estéreo proporciona una descripción del entorno muy completa. Ello da lugar a

que se pueda distinguir el tipo de elemento del que se trata y, sobre todo, la principal ventaja que presenta es que permite anticiparse a los movimientos, en este caso del peatón. No obstante, este método también presenta inconvenientes, ya que el procesamiento de toda la información resulta complejo, dando lugar a un alto coste computacional.

A continuación se detallan los pasos seguidos para la detección de los elementos presentes en la vía:

4.1.1. MAPA DE DISPARIDAD

Como ya se mencionó en el capítulo anterior de Fundamentos Teóricos, mediante en el tratamiento de imágenes basado en la visión estéreo se trabaja con un par de imágenes de la misma escena, tomadas desde dos posiciones distintas. Las diferencias existentes entre ambas imágenes va a dar lugar a lo que se conoce como disparidad.

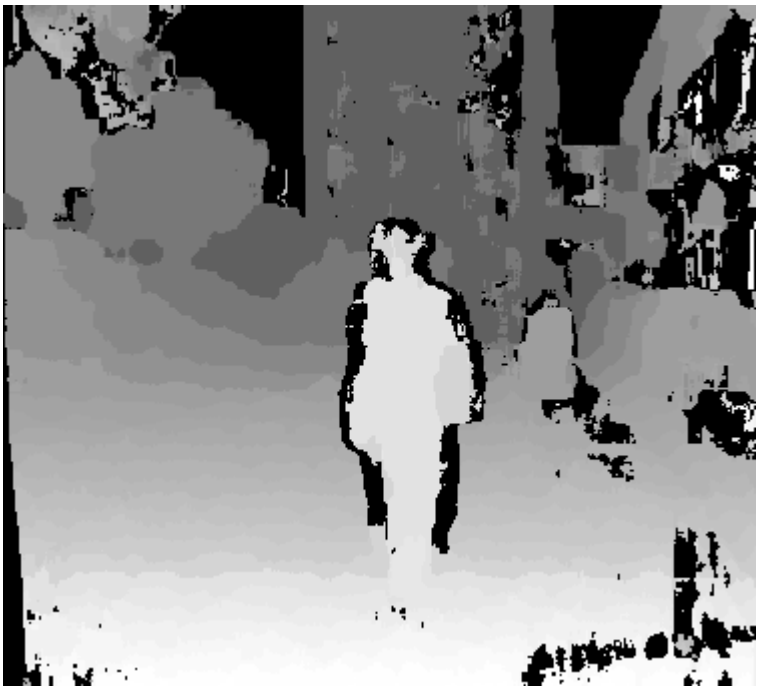


Figura 4.1: Mapa de disparidad.

El algoritmo diseñado en este proyecto irá procesando pares de imágenes (derecha e izquierda). Estas imágenes serán tomadas por dos cámaras iguales y paralelas, de tal forma que si se toma un punto P (X,Y,Z) en la escena real, su proyección sobre el plano de la imagen izquierda va a ser (u_1, v_1) y sobre el plano de la imagen derecha va a ser (u_2, v_2). Para el cálculo de la disparidad es necesario resolver el problema conocido como correspondencia estéreo, es decir, es necesario determinar qué elemento en la imagen izquierda se corresponde con otro en la imagen derecha (6). Es importante que los dos ejes de las cámaras del sistema estéreo sean paralelos entre sí para que se cumpla que $v_1 = v_2$ y así hablar de un problema 1D, en vez de 2D, por lo que va a ser necesaria una rectificación previa. Por otro lado, se realiza un preprocesamiento de las imágenes del par estéreo mediante la Laplaciana de la Gaussiana para conseguir igualar la ganancia de las dos cámaras.

$$d = u_1 - u_2 \tag{6}$$

La disparidad viene representada por los niveles de gris correspondientes a cada punto de la imagen; a medida que los elementos se encuentren más alejados de la cámara, el nivel de gris será menor. Las disparidades de todos los puntos de una imagen conforman el denominado *mapa de disparidad*, que puede ser representado como una imagen, tal y como se muestra en la Figura 4.1.

4.1.2. UV_DISPARITY

A partir del mapa de disparidad se obtiene el *uv_disparity* [35] [36], es decir, el *u_disparity* que contiene el histograma lateral correspondiente a la disparidad para cada una de las columnas del mapa de disparidad y el *v_disparity* que muestra el histograma para cada una de las filas. Como se observa en la Figura 4.2 en la que se representa el mapa de disparidad junto al *u-v disparity*, los obstáculos aparecen representados por líneas horizontales en el u_disparity, siendo su intensidad la altura de dichos obstáculos medida en píxeles. Por otro

lado, en el v_disparity se representan como líneas verticales cuya intensidad corresponde a la anchura medida, nuevamente, en píxeles.

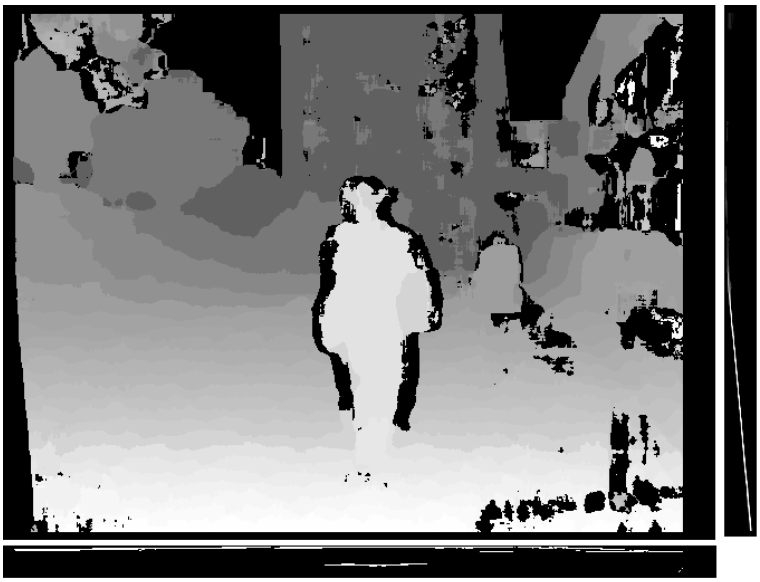
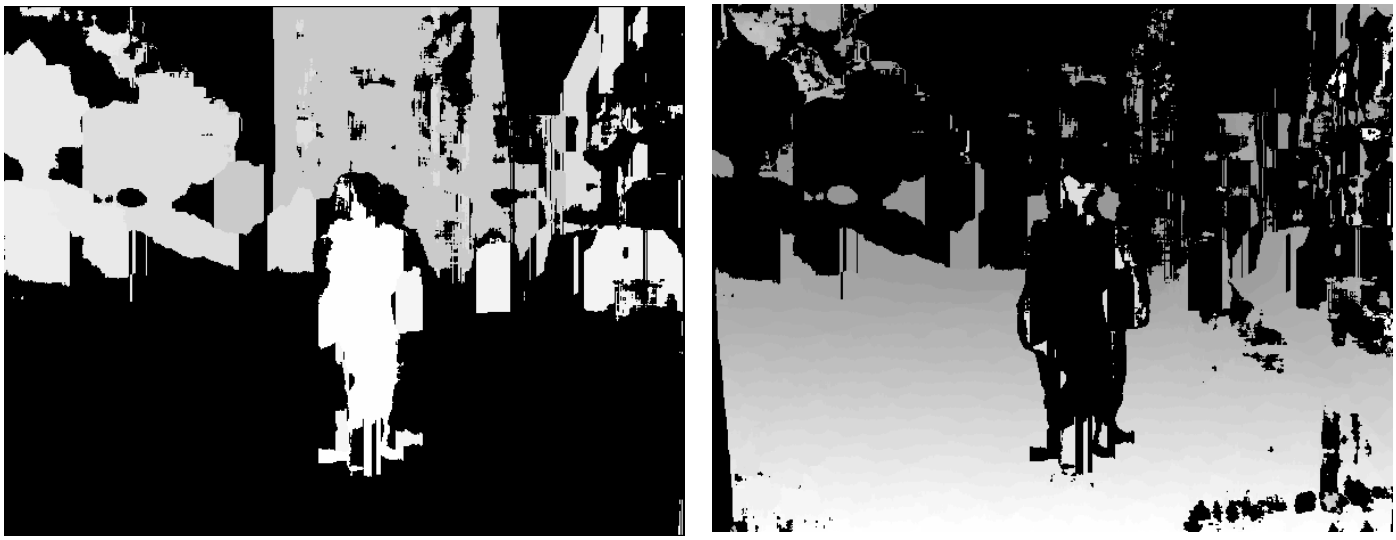


Figura 4.2: Representación de la v-disparity (vertical) y la u-disparity (horizontal).

4.1.2. MAPA DE OBSTÁCULOS Y MAPA LIBRE

A la hora de reconocer peatones se tiende primero a distinguir en la imagen los distintos objetos localizados a una determinada distancia y separarlos de lo que se considera fondo, para después diferenciar entre aquellos objetos que corresponden a personas y aquellos que no. Por tanto, para hacer más fácil la manipulación de toda la información contenida en el mapa de disparidad, se divide dicho mapa en dos: un mapa que únicamente represente las disparidades correspondientes a los obstáculos, *mapa de obstáculos*, y otro opuesto, que represente el espacio libre por el que puede circular el vehículo, denominado *mapa libre*. El mapa de obstáculos se obtiene a partir de la imagen *u_disparity*, previamente umbralizada, de tal forma que sólo se tendrán en cuenta aquellos obstáculos cuya altura en píxeles sea mayor que un valor umbral predefinido. Es importante elegir un valor umbral correcto para que se detecten todos aquellos elementos que obstaculicen el paso del

vehículo, pero que rechace como tales aquellos lo suficientemente pequeños como para permitir su paso, como es el caso de los badenes. Una vez umbralizado el *u_disparity*, se eliminan del mapa de disparidad todos los píxeles que no representen un obstáculo, dando lugar al *mapa de obstáculos* (Figura 4.3 (a)). Por el contrario, para la construcción del *mapa libre* se toman del mapa de disparidad aquellos píxeles que no representen un obstáculo en el *u_disparity* umbralizado. Como se aprecia en la Figura 4.3 (b), el mapa libre representa la calzada por la que puede circular el vehículo.



(a) (b)

Figura 4.3: (a) Mapa de Obstáculos, (b) Mapa Libre.

El *v_disparity* del mapa libre proporciona información de la calzada, ya que representa el perfil del suelo (*road profile*) como una línea oblicua, la cual permite calcular la posición real de los obstáculos con respecto a la cámara. Dicha línea se puede expresar como la ecuación de una recta (7), siendo **m** la pendiente y **b** la ordenada en el origen que corresponde al horizonte teórico de la imagen. Para calcularla se va a recurrir al método de la *Transformada de Hough para rectas.* Este método consiste en el cálculo de todas las rectas que pasan por cada punto de la imagen, dando como resultado aquella más votada, es decir, aquella que contenga un mayor número de píxeles. Hay que destacar que si se hubiera aplicado la

Transformada de Hough directamente en el mapa de disparidad, podrían haber surgido rectas con mayor número de píxeles correspondientes a otro tipo de obstáculos. En cambio, si se aplica al *v-disparity* obtenido del mapa libre, se obtiene que la recta más votada sea la correspondiente al perfil de la calzada.

$$v = m \cdot d + b \tag{7}$$

El mapa de obstáculos va a permitir obtener los objetos de interés presentes en la escena de una manera más rápida y sencilla.

4.2. DETERMINACIÓN DE LAS REGIONES DE INTERÉS

La detección de los peatones en las distintas imágenes se va a realizar, como ya se ha dicho anteriormente, mediante la implementación del algoritmo de Dalal y Triggs [37]. Sin embargo, dicho algoritmo presenta un alto coste computacional, por lo que se busca reducirlo, implementándolo únicamente en aquellas regiones de interés que puedan contener a uno o varios peatones. Se van a llevar a cabo una serie de descartes y clasificaciones, con el objetivo de que el número de píxeles procesados sea el menor posible, sin que ello afecte a la eficacia de la detección de los peatones.

En primer lugar, se van a descartar aquellos objetos localizados demasiado lejos del vehículo, pues no suponen un riesgo de accidente. Por tanto, no se van a tener en cuenta aquellos píxeles que presenten una disparidad inferior a una prefijada. Los obstáculos que se localicen en el mapa de obstáculos dentro del margen de búsqueda van a ser consideradas objetos de la escena. No obstante, el tamaño de algunas de ellas va a ser tan reducido que va a ser imposible que se trate de un peatón. Esto significa que se podrá hacer un nuevo descarte según el tamaño de dichos obstáculos. El resto de los blobs localizados van a ser englobadas por rectángulos, dando lugar a las conocidas Regiones de Interés (ROIs), con las que se va a trabajar durante el resto del proyecto. En la Figura 4.4 se muestra este descarte, pues en la Figura 4.4 (a) se observa como muchas de las regiones

representadas son demasiado pequeñas como para englobar a un peatón, por lo que se decide dejar de tenerlas en consideración (Figura 4.4 (b)).

(a) (b)

Figura 4.4: Representación de las ROIs obtenidas (a) sin descartar aquéllas de área más pequeñas, (b) después de descartar aquéllas ROIs que por su tamaño no pueden englobar a un peatón.

4.2.1. CLASIFICACIÓN DE LAS REGIONES DE INTERÉS

Con la intención de continuar reduciendo el tiempo de procesamiento del algoritmo, se va a llevar a cabo una clasificación previa antes de clasificar los obstáculos entre peatones y no peatones. Esta clasificación va a consistir en distinguir los obstáculos englobados en las regiones de interés como elevados o no elevados, con el objetivo de que únicamente las regiones que resulten como no elevadas pasarán a clasificarse entre peatones y no peatones mediante los descriptores HOG.

Para la clasificación entre obstáculos elevados y no elevados se va a estimar un valor teórico de disparidad. A este valor se le va a denominar *disparidad teórica* y corresponde a la disparidad que, en función de su posición en la imagen (coordenada v), presentaría un obstáculo si estuviera apoyado en el suelo. En su cálculo se requiere el empleo del valor de la coordenada vertical v inferior de la

región de interés, así como de los datos del *road profile*, **m** (pendiente de la recta) y **b** (horizonte teórico), ya que el *road profile* establece la relación entre la posición en la imagen y el valor de disparidad del suelo (8). Por otro lado, se va a considerar la *disparidad real* como aquel valor de disparidad que toma el suelo en el punto en el que se encuentra el objeto. Por tanto, para que un obstáculo pueda considerarse no elevado su disparidad real debe coincidir con su disparidad teórica. Si la disparidad real es mayor que la teórica significará que se trata de un obstáculo elevado. La disparidad real de la ROI en cuestión, corresponde al valor máximo del histograma de dicha región.

$$d_{teórica} = \frac{v-b}{m} \tag{8}$$

En el caso de algunos objetos, como pueden ser determinadas señales de tráfico se puede generar confusión, debido a que el elemento que las soporta es demasiado estrecho como para no ser considerado perteneciente a la región de interés, por ello se considera cierto margen.

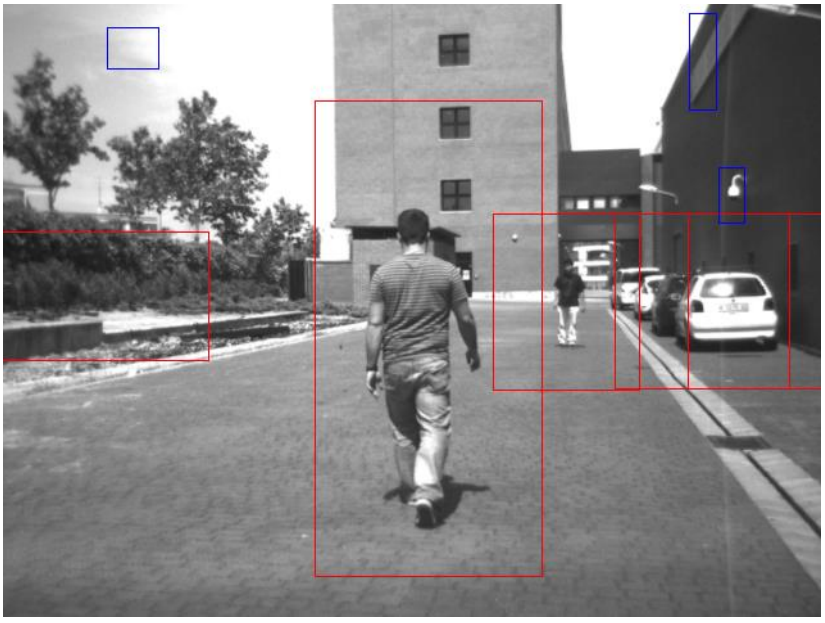


Figura 4.5: Representación de las ROIs, distinguiendo entre obstáculos elevados (rectángulos azules) y obstáculos no elevados (rectángulos rojos).

4.3. RECONOCIMIENTO DE PEATONES

4.3.1. HISTOGRAMAS DE GRADIENTES ORIENTADOS (HOG)

Una vez clasificadas las regiones de interés se descartan aquellas que engloben obstáculos determinados como elevados, para proceder al análisis de aquellas ROIs clasificadas como no elevadas y que implican la posibilidad de ser un obstáculo para el paso del vehículo. Para la clasificación de estos obstáculos como peatones o no peatones, siendo los primeros el objeto de interés de este proyecto, se emplea el método basado en el Histograma de Gradientes Orientados (HOG). En primer lugar, se detalla el funcionamiento de dicho método de reconocimiento de objetos, para posteriormente centrarse en el trabajo de Navneet Dalal y Bill Triggs [37] sobre el reconocimiento de peatones, en el que se fundamenta esta parte del proyecto.

El HOG es capaz de detectar la presencia de peatones presentes en una escena. Una vez detectados aquellos peatones que por su cercanía al vehículo corren peligro de ser atropellados, el conductor del vehículo puede ser alertado de la presencia de estos con la suficiente antelación como para poder reaccionar en el caso de que exista un riesgo. Es por ello que se insta a que el programa procese las imágenes tomadas con la mayor brevedad posible. El cálculo de los descriptores HOG presenta un coste de tiempo de computación bastante elevado por el hecho de calcular el HOG en cada una de las celdas. Por este motivo, como ya se ha comentado, se va a implementar este método únicamente en las regiones de interés no elevadas, pues así se reduce el número de celdas a analizar.

La elección de este método para llevar a cabo la detección de los peatones en la escena se basa en que éste concretamente destaca por su robustez frente a diferentes condiciones de iluminación, pequeños cambios en el contorno de la imagen y diferencias de fondos y de escalas. Los descriptores propuestos se basan en trabajos previos, tales como histogramas de bordes orientados, descriptores SIFT y reconocimiento de formas. Sin embargo, el método HOG presenta la diferencia de que los gradientes no se calculan uniformemente sobre un único

mallado denso, sino que divide la imagen en una serie de bloques distribuidos a lo largo y ancho de la misma y con cierto solape entre ellos. De esta forma, el avance de bloques se realiza eliminando la columna de las celdas de la izquierda y añadiendo la columna de la derecha para el desplazamiento horizontal, mientras que para el vertical se elimina la fila de las celdas de arriba, añadiendo la fila de celdas de abajo. A su vez, cada bloque se divide en subregiones o celdas, calculándose en cada uno de ellos el histograma de los gradientes orientados, de tal forma que se logra mejorar el rendimiento. En la Figura 4.6 se muestra una imagen donde se representan gráficamente los pasos seguidos por el método basado en los histogramas de gradientes orientados.

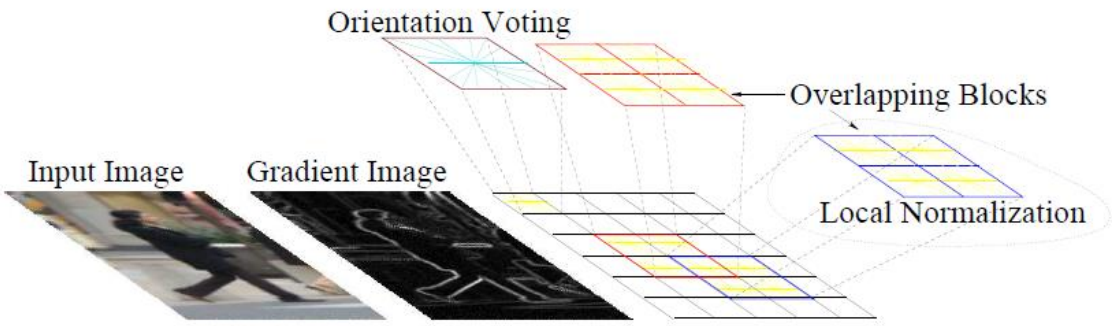


Figura 4.6: Desarrollo del método HOG.

El primer paso para aplicar el HOG a una determinada imagen o ROI es calcular las derivadas espaciales de dicha imagen a lo largo de los ejes x e y (I_x y I_y). En este caso, dichas derivadas se han obtenido calculando la convolución de las imágenes con filtros Gaussianos de diferente varianza. A continuación, se hallan tanto la magnitud (9) como la dirección de los valores del gradiente (10) en cada uno de los píxeles de la región:

$$|G| = \sqrt{I_x^2 + I_y^2} \tag{9}$$

$$\theta = \arctan\left(\frac{I_x}{I_y}\right) \tag{10}$$

El siguiente paso es dividir la imagen en celdas y calcular los histogramas de cada una de estas celdas. Cada píxel de la celda tiene un cierto peso en el histograma de orientación, basado en el valor calculado de la magnitud de su gradiente. Cada imagen se representa a través del histograma de cada una de las celdas. Estos histogramas quedan ordenados en un vector según su peso, dando lugar al vector de características de la imagen. Una imagen omnidireccional contiene los mismos píxeles en una fila aunque la imagen esté rotada, lo que significa que se va a obtener un vector de características invariante ante una rotación.

4.3.2. HISTOGRAMAS DE GRADIENTES ORIENTADOS PARA LA DETECCIÓN DE PEATONES

Para la implementación del algoritmo HOG se modifica el ejemplo "*people detect*" ofrecido por las librerías OpenCV de la versión 2.1. El método aplicado en este ejemplo se basa en el usado por Dalal y Triggs [37], el cual consiste en recoger información de una gran colección de imágenes tomadas como positivas por contener una o varias figuras humanas y otra gran cantidad de imágenes en las que no aparecen personas y que se tomarán como negativas. Este amplio grupo de imágenes se emplea para entrenar un detector mediante la técnica de aprendizaje denominada Máquina de Vectores de Soporte (*Support Vector Machine, SVM*). La información que se obtiene de cada una de las imágenes de muestra va a dar lugar a un descriptor de Histogramas de Gradientes Orientados (descriptor HOG). Una vez obtenido el detector, se emplea en la detección de peatones en las imágenes deseadas.

Se toman dos conjuntos de datos diferentes para el entrenamiento del SVM. El primero es el conjunto de datos de peatones MIT que contiene 509 entrenadores y 200 imágenes de prueba de peatones en distintas escenas de una ciudad. Éstas sólo contienen las vistas delantera y trasera del peatón, con un rango limitado de poses. A pesar de que se obtienen buenos resultados, se recurre a otro conjunto de datos, denominado "INRIA", que contiene 1805 de 64x128 imágenes de personas

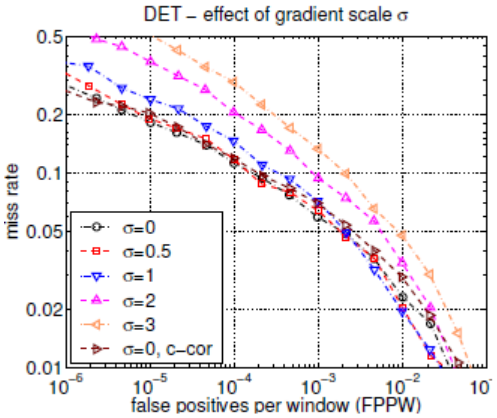
con poses y fondos aún más variados e, incluso, en algunas aparecen multitudes. El proceso de aprendizaje consiste en que, una vez obtenido el primer modelo de detección, se compara con el conjunto de imágenes de entrenamiento. Si en este punto una imagen entrenada como positiva no es detectada significa que se tiene un falso negativo y si, por el contrario, una imagen que no presenta peatón es detectada, se tiene un falso positivo. Lo que se hace a continuación es reentrenar el modelo añadiendo los falsos positivos o los falsos negativos detectados al conjunto correspondiente. Este proceso de reentrenamiento se repite tantas veces como sea necesario hasta obtener una precisión razonable del detector.

Una vez realizado el entrenamiento, se divide la región de interés en estudio en celdas, calculando en cada una de ellas el histograma de la dirección del gradiente o de la orientación de los bordes. Por otro lado, se sabe que el histograma de una imagen divide el rango de valores posibles de los píxeles de la imagen en una serie de subrangos o clases de igual o distinto tamaño. En cada clase se almacena la frecuencia de píxeles con un valor comprendido dentro de ese subrango, es decir, el número de píxeles en la imagen cuyo valor está dentro de esos valores. Del mismo modo, el histograma de gradientes orientados de una imagen es el histograma que tiene como rango de valores posibles las distintas orientaciones que pueden tomar los gradientes de los píxeles, es decir, los distintos grados que pueden tomar sus ángulos de gradiente. Más adelante se destacará que la mejor opción es tomar el rango [0º, 180º] y dividirlo en nueve subrangos: [0º,20º), [20º, 40º)…[160º, 180º], almacenando en cada uno la suma de las magnitudes de gradiente de los píxeles cuyo ángulo de gradiente se encuentre comprendido entre los valores de dicho subrango. Hay que destacar que para mejorar la invariancia a la iluminación, sombras, etc., Dalal y Triggs recurren a normalizar el contraste de los histogramas de cada una de las celdas en las que se divide la región de interés.

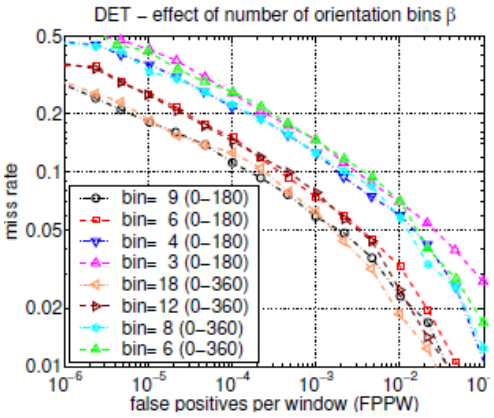
4.3.2.1. ESTUDIO DEL RENDIMIENTO DEL MÉTODO HOG

A continuación se va a pasar a comentar como los parámetros introducidos en el método HOG afectan al rendimiento. Para cuantificar el rendimiento del

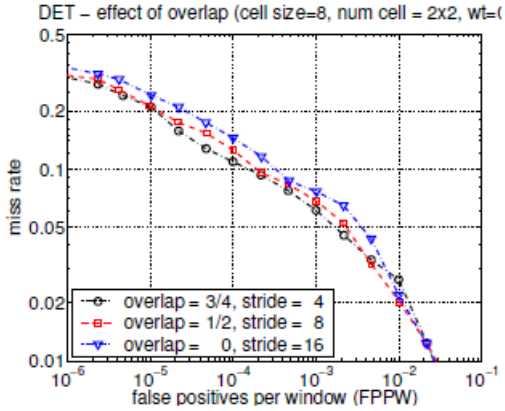
detector se trazan las curvas del Error de Equilibrio en la detección, es decir, Detection Error Tradeoff (DET) en una escala log-log contra FPPW (en inglés, Falsos Positivos Por Ventana). A menudo se utiliza el ratio 10^{-4} como punto de referencia para los resultados. Esto corresponde a una tasa de error de alrededor de 0.8 falsos positivos en las imágenes de 640x480 probadas. En las curvas DET, pequeñas mejoras en el ratio de pérdida dan lugar a importantes aumentos de FPPW en la constante tasa de fallos.



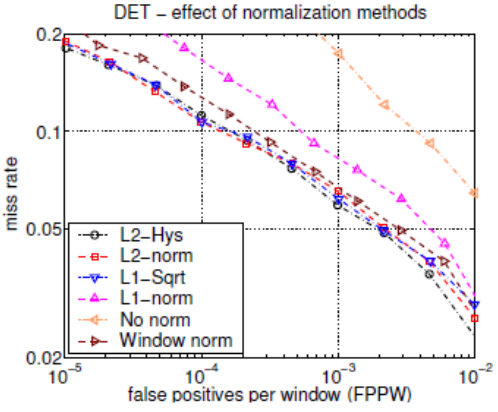
(a)



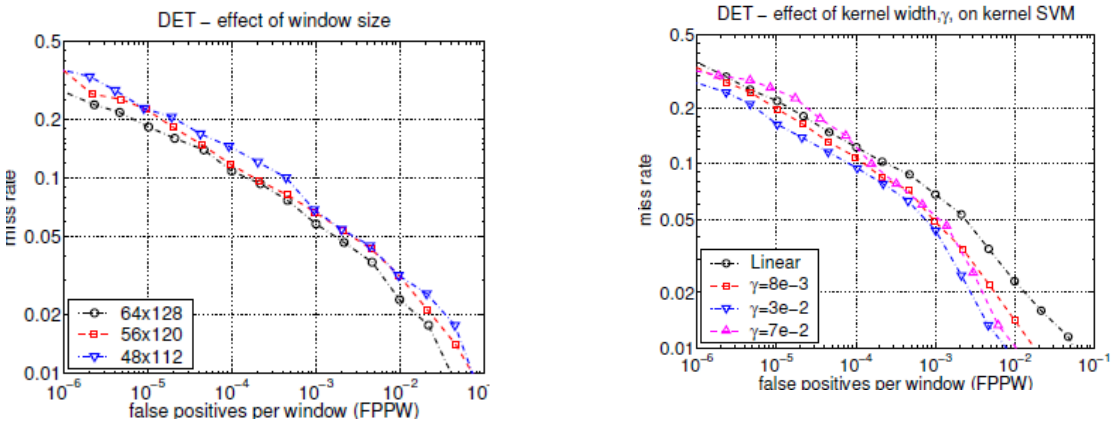
(b)



(c)



(d)



(e) (f)

Figura 4.7: Estudio de los efectos que tienen sobre el rendimiento la modificación (a) de la escala del gradiente, (b) del número de subrangos de orientación, (c) del número de celdas superpuestas, (d) del método de normalización empleado, (e) de las dimensiones de la ventana de detección, (f) del ancho del kernel, γ, en el kernel SVM.

EFECTO DE LA ESCALA DEL GRADIENTE

El rendimiento del detector es sensible al modo en el que se obtienen los gradientes, pero es el sistema más sencillo el que resulta ser el mejor al proporcionar mejores resultados. El uso de grandes máscaras disminuye el rendimiento, mientras que el *smoothing* lo daña significativamente. Dalal y Triggs recurren a calcular los gradientes utilizando un filtro Gaussiano seguido por una de las diversas máscaras de derivada discreta: centrada, descentrada, cúbica corregida, Sobel de 3x3 y diagonales de 2x2; llegando a la conclusión, como se demuestra en la Figura 4.7 (a), de que las máscaras centradas [-1, 0, 1] con σ=0 son las que proporcionan mejores rendimientos.

EFECTO DEL NÚMERO DE SUBRANGOS DE ORIENTACIÓN

A la hora de calcular el histograma de gradientes orientados de una imagen, lo que se hace, como ya se ha mencionado, es dividir el rango que contiene todos los posibles valores de orientación de los gradientes en varios subrangos. De esta forma, en el histograma se tendrán los distintos gradientes de los píxeles agrupados según pertenezcan a un subrango o a otro. En la Figura 4.7 (b) se muestra como incrementando el número de subrangos de orientación mejora el

rendimiento significativamente. Después de una serie de pruebas, representadas en la Figura 4.7 (b), se llega a la conclusión de que el número más apropiado de subregiones es 9, pues a partir de este número la diferencia experimentada es muy poca. Esto sería para el caso en el que el rango de orientación de los gradientes fuera (0º, 180º), es decir, en el caso de que el "signo" del gradiente fuese ignorado. En el caso de considerar el signo del gradiente, el rango de orientación sería (0º, 360º), al igual que en el descriptor SIFT en el que se basa el descriptor HOG. Sin embargo, si consideramos este signo, el rendimiento decrece incluso cuando el número de subrangos es doblado para preservar la resolución original de la orientación, ver también la Figura 4.7 (b). Esto se debe, a que en el caso del reconocimiento de personas, la amplia variedad de ropas y colores del fondo hacen que los signos de los contrastes no aporten información. Por tanto, el signo será despreciado, tomando el rango de orientación (0º, 180º).

EFECTO DEL NÚMERO DE CELDAS SUPERPUESTAS

El rendimiento mejora cerca de un 4% en 10^{-4} FPPW a medida que se aumenta la superposición desde 0 a 16, Figura 4.7 (c). En la gráfica se analiza el efecto de solape entre bloques para valores de solape 0 (*stride* 16), ½ (*stride* 8) y ¾ (*stride* 4). Se llega a la conclusión de que los mejores resultados dentro de que exista un solape entre bloques se obtienen para un solape de ¾. Esto es debido a que al estar más espaciados los bloques, es necesario un menor número de ellos para cubrir toda la imagen, lo que hace que el rendimiento mejore.

EFECTO DE TAMAÑO DE LOS BLOQUES Y DE LAS CELDAS

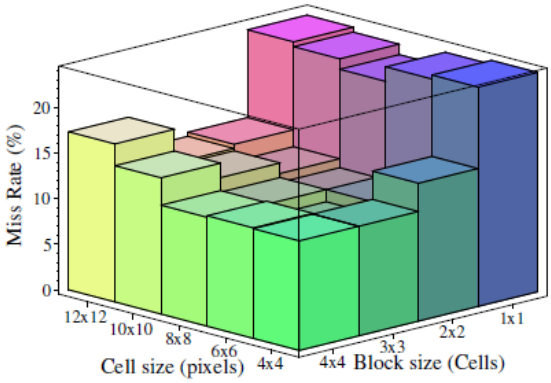


Figura 4.8: Gráfica que muestra la variación del rendimiento según el tamaño del bloque o la celda.

La Figura 4.8 representa el ratio de pérdidas de personas en función del tamaño del bloque y del tamaño de subbloque o celda. Se observa que para la detección de personas el óptimo se encuentra para un tamaño de bloque de 3x3 celdas y un tamaño de celda de 6x6 píxeles con un 10.4 % como ratio de pérdida en 10^{-4} FPPW. Los resultados empeoran cuando los bloques son demasiado grandes, pues aumenta el tiempo, y cuando son demasiado pequeños debido a que se suprime información importante.

EFECTO DEL MÉTODO DE NORMALIZACIÓN EMPLEADO

Ya se ha mencionado que para mejorar la invariancia a la iluminación y a los sombreados, se normaliza el contraste de los histogramas de cada una de las celdas en las que se divide la región a estudio. La Figura 4.7 (d) muestra como utilizando las normalizaciones L2-Hys, L2-norm y L1-sqrt se obtiene prácticamente el mismo rendimiento, mientras que si se normaliza según el sistema L1-norm el rendimiento del proceso se reduce un 27% en 10^{-4} FPPW.

EFECTO DE LAS DIMENSIONES DE LA VENTANA

Una ventana de detección de 64x128 incluye alrededor de 16 píxeles de margen alrededor de las personas en los 4 lados de la imagen. Esta frontera muestra una cantidad significante de información que permite la detección. En la Figura 4.7 (e), se observa que si se reduce este margen de 16 a 8 píxeles, es decir, se pasa a una ventana de detección de dimensiones 48x112 el rendimiento se reduce un 6% en 10^{-4} FPPW. Si manteniendo una ventana de 64x128 lo que se hace es aumentar el tamaño de la persona (en este caso también se está reduciendo la frontera) la pérdida de rendimiento es similar a la que se producía reduciendo el tamaño de la ventana, a pesar de que la resolución de la persona haya sido incrementada.

EFECTO DEL ANCHO DEL KERNEL, γ, EN EL KERNEL SVM

Por defecto se utiliza un suave entrenador SVM linean con SVMLight ligeramente modificado para reducir el uso de memoria pues genera vectores descriptores muy grandes. El empleo, por el contrario, de un kernel SVM Gaussiano $exp(-\gamma\|x_1 - x_2\|^2)$ aumenta el rendimiento alrededor de un 3% en 10^{-4} FPPW con el coste de un mayor tiempo de computo, como se justifica en la Figura 4.7 (f).

Por tanto, Dallal y Trigg tras estudiar la influencia de estos parámetros del descriptor HOG, concluyen que los valores definidos con anterioridad son los adecuados para obtener un buen rendimiento. De esta forma, demuestran que obteniendo los histogramas de gradientes orientados normalizados en una densa maya superpuesta se mejoran los resultados de otros detectores tales como los descriptores SIFT o los Haar Wavelet.

4.4. LOCALIZACIÓN DE LOS PEATONES

Una vez detectado un peatón, se puede conocer su posición real en el mundo, es decir, sus coordenadas (X,Z) donde Z indica la profundidad a la que se encuentra de la cámara. Para conocer la localización de los peatones con respecto a

la cámara se recurre a las expresiones (11) y (12), donde **m** es la pendiente del perfil de la calzada, **b** el horizonte teórico, **(u,v)** las coordenadas del peatón en la imagen, **f** la distancia focal de la cámara, **B** la distancia entre las cámaras (*baseline*) y $\mathbf{C_u}$ corresponde a la coordenada del centro óptico. Estos últimos tres datos son valores conocidos de la cámara. Estas expresiones proceden de la combinación de las expresiones propias de un sistema estéreo, ya vistas, (3) y (5) con (7), ya que al tratarse de obstáculos no elevados localizados delante del vehículo se cumple la relación dada por el Road *Profile* (7) entre la coordenada vertical de la imagen *v* y la disparidad *d*. Ambas ecuaciones se multiplican por $\mathbf{cos}\boldsymbol{\alpha}$ porque lo que interesa calcular es la distancia existente entre coche y peatón y no entre cámara y peatón. En la Figura 4.9 se representa dicha relación.

$$Z = \frac{m \cdot f \cdot B}{v - b} \cdot \cos\alpha \tag{11}$$

$$X = \frac{m \cdot B \cdot (u - C_u)}{v - b} \cdot \cos\alpha \tag{12}$$

El ángulo α es el denominado ángulo de *cabeceo* y corresponde al ángulo que forma el sistema estéreo con la calzada y se calcula mediante (13).

$$\alpha = arctan\frac{b - C_v}{f} \tag{13}$$

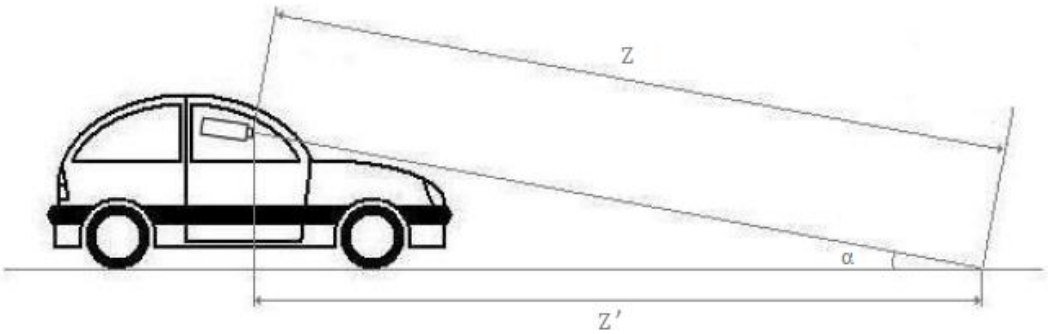


Figura 4.9: Relación entre la cámara y el peatón y entre el coche y el peatón.

Capítulo 5

IMPLEMENTACIÓN SOFTWARE

El presente algoritmo parte de otro algoritmo creado por la alumna de esta universidad, Violeta Jiménez Monje [38]. Su trabajo consistió en la creación de una serie de funciones que permitían la obtención de la Laplaciana de la Gaussiana, del mapa de disparidad, el u_disparity, el v_disparity, así como del mapa libre y de obstáculos. Por tanto, para conseguir el objetivo que se persigue en este proyecto se van a emplear tanto estas funciones como algunas propias de las librerías OpenCV.

5.1. DESCRIPCIÓN DEL CÓDIGO

En este apartado se van a describir aquellas líneas de código cuya interpretación puede resultar más compleja, así como algunas de las funciones correspondientes a las librerías OpenCV consideradas más importantes para el correcto desarrollo del algoritmo.

MAIN

Durante el desarrollo del programa, el análisis de las imágenes se lleva a cabo con imágenes que fueron tomadas por un par de cámaras estereoscópicas en su momento, pero que se encuentran almacenadas en una carpeta del ordenador. Para el acceso a estas imágenes se ha recurrido a la función *cvLoadImage*. Por un lado, se ha obtenido la imagen correspondiente a la proyección derecha de la escena y por el otro, a la proyección izquierda.

```
sprintf(imagender, "../data/der%d.tiff ",i);
IplImage* imgD = cvLoadImage(imagender,0);
sprintf(imagenizq, "../data/izq%d.tiff ",i);
IplImage* imgI = cvLoadImage(imagenizq,0);
```

Sin embargo, una vez finalizado el algoritmo, se adaptará para que las imágenes sean tomadas directamente de la cámara instalada dentro del vehículo.

Para poder trabajar con las imágenes de tipo IplImage en las funciones creadas por Violeta Jiménez Monje, es necesario que éstas sean transformadas a un tipo de datos *unsigned char*; para ello, se procede del siguiente modo:

```
// 8. Cambio las imágenes a un parámetro unsigned char para poder trabajar con ellas en las funciones creadas

 im_izda=((uchar *)(imgD->imageData));
 im_dcha=((uchar *)(imgI->imageData));
```

En este punto se accede a la función **laplaciana**, en la que se introducen las imágenes de datos *im_izda* e *im_dcha* a las que se les aplica la Laplaciana de la Gaussiana. Las matrices de datos *lp_izda* y *lp_dcha* se introducen como parámetros de entrada en otra de las funciones creada, la función **disparidad**. Esta función consiste en un proceso básico de comparación de las dos imágenes (proyección izquierda y proyección derecha), píxel a píxel, lo que se conoce como "*matching problem*" y de ella se obtienen las matrices de datos *disp_izda* y *disp_dcha*. Estas disparidades calculadas son sometidas a un post-procesamiento para desechar malas medidas tomadas por causa de oclusiones entre objetos. En este caso, se va a realizar un "*cross checking",* que consiste en comparar el mapa de disparidad

derecho con el izquierdo y viceversa; aquellos píxeles cuyo valor de disparidad sea distinto se marcarán como ocluidos. Estos píxeles ocluidos pueden ser completados asignándoles un valor de disparidad determinado. En este caso, cuando la disparidad entre ambos mapas sea mayor que 2, se le asignará a dicho píxel un valor nulo en uno de los mapas de disparidad, por ejemplo, el izquierdo.

```
for (int k=0; k<(tam_x*tam_y); k++)
{
    if (abs(disp_dcha[k]-disp_izda[k])>2)
        {
            disp_izda[k]=0;
        }
}
```

Una vez refinado el mapa de disparidad, se guardará la imagen de datos de disparidad en un parámetro IplImage para, posteriormente, trabajar con ella. Para ello, se definirá una cabecera que contenga la información de la imagen mediante la función *cvCreateImageHeader*. A continuación, se introducen los datos de disparidad en el campo *imageData* del IplImage, que contiene un puntero a la primera columna de datos de la imagen.

```
IplImage *imgDisp = cvCreateImageHeader(cvSize(width,height),depth,nchannels);
imgDisp->imageData = (char *)disp_izda;
```

La matriz de datos de disparidad *disp_izda*, se utiliza para obtener el *u_disparity*, que permite a su vez obtener tanto el *mapa de obstáculos* como el *mapa libre*. Para ello, se va a proceder de forma similar a la obtención del mapa de disparidad, empleando ahora las funciones **u_disparity** y **mapas**.

A partir de los datos del mapa libre se obtiene el *v_disparity* mediante la función **v_disparity**, permitiendo calcular el *road profile* mediante la Transformada de Hough para rectas, implementando la función correspondiente de las OpenCV, *cvHoughLines2*, y utilizando el método probabilístico que ofrece. Sin embargo, debido a que el ancho en el que se almacenan las imágenes cuando se

usan las librerías OpenCV es diferente al ancho con el que se almacenan en otras librerías empleadas por el algoritmo anterior, se requiere realizar la siguiente transformación:

```
// for(y) y for(x) recorren todos los píxeles de la matriz origen (v_disp), cargando los datos en el lugar
// indicado de la matriz destino (aux_v)

for(int y=0; y<tam_y; y++)
{
    for(int x=0; x<tam_coste; x++)
    {
        aux_v[imgvDisp->widthStep*y+x*imgvDisp->nChannels]=v_disp[y*tam_coste+x];

    }
}
// Se cargan los datos de la nueva matriz calculada a la imagen OpenCv
imgvDisp->imageData = aux_v;

// Se define una estructura para poder almacenar datos genéricos de forma dinámica en formato OpenCv
CvMemStorage* datos = cvCreateMemStorage(0);

// Se define una estructura para poder almacenar líneas de forma dinámica en formato OpenCv
CvSeq* lineas = 0;

// Se calcula la transformada de Hough de la imagen v, almacenando los resultados en lineas
lineas = cvHoughLines2(imgvDisp,datos,CV_HOUGH_PROBABILISTIC,1,CV_PI/180,50,75,20);

// Extracción de la primera línea de todas las obtenidas en la transformada de Hough
CvPoint* linea = (CvPoint*)cvGetSeqElem(lineas,0);
```

Para el empleo de la función *cvHoughLines2* es necesario definir una estructura para almacenar datos de forma dinámica (*datos*) y otra estructura para almacenar los resultados, también de forma dinámica (*líneas*). Una vez implementada la función de la Transformada de Hough para líneas, se procede a extraer las coordenadas de los puntos que definen la primera línea de todas las obtenidas, pues ésta es la que posee un número mayor de píxeles y, por tanto, la línea que representa el perfil de la carretera (*Road Profile*), representada en la Figura 5.1.

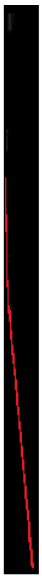
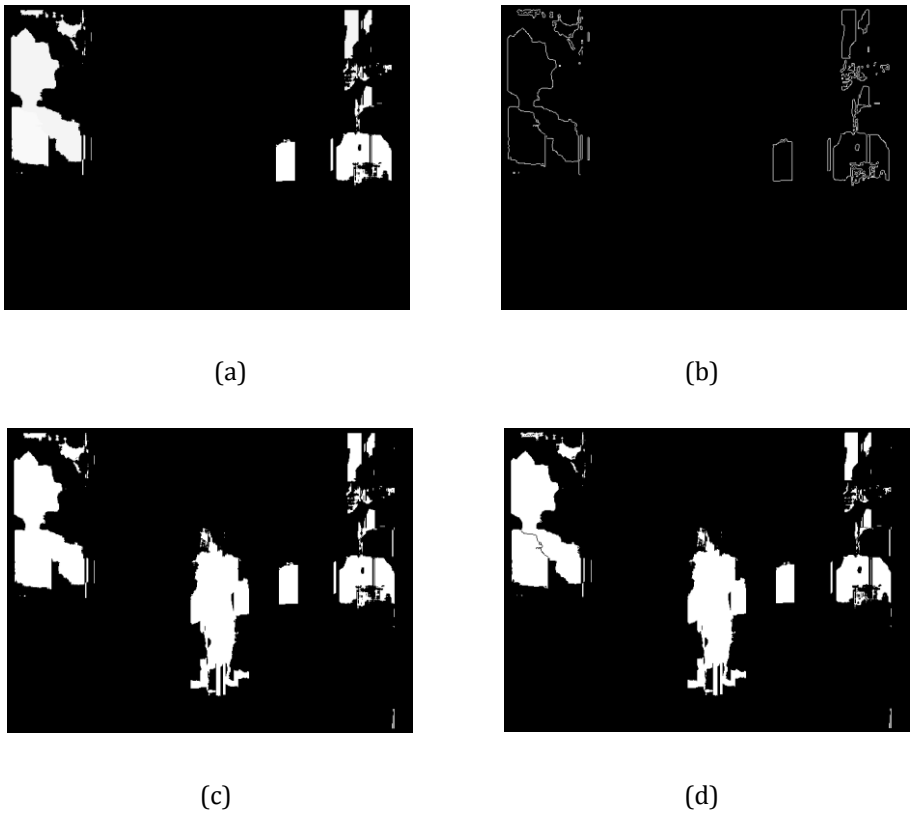


Figura 5.1: Representación de la recta resultante de la transformada de Hough (*Road Profile*).

A partir de este punto, el algoritmo se centra en la detección de los objetos de la imagen y sus clasificaciones posteriores. Interesa que los objetos queden bien definidos en el mapa de obstáculos. En primer lugar, se procede a determinar un campo de estudio para tomar sólo aquellos objetos que se encuentren a una determinada distancia del vehículo. Para ello, se lleva a cabo una umbralización, quedando fuera de las regiones de interés aquellos obstáculos que superen el valor umbral de disparidad inferior a 3. Los obstáculos pasan a tener 1 como nivel de gris, mientras que los elementos que no se consideren de interés pasan a valer 0, obteniéndose una imagen binaria.

Sin embargo, de esta forma, no se pueden diferenciar objetos que se encuentren próximos entre sí, siendo englobados como un único objeto y dificultando su clasificación. Por este motivo, se procede a aplicar el detector de bordes de Canny, mediante la función propia de las librerías OpenCV. Con esta función se obtiene una máscara con los bordes de la imagen, diferenciando los distintos obstáculos. Restando dicha máscara a la imagen binarizada del mapa de obstáculos umbralizado, se obtendrá otra imagen con los obstáculos bien diferenciados.



(a) (b)

(c) (d)

Figura 5.2: (a) Mapa de obstáculo umbralizado (obstáculos entre los valores de disparidad 3 y 8), (b) Aplicación del detector de Canny, (c) Mapa de obstáculos binarizado, (d) Imagen resultante de restar al mapa de obstáculos binarizado la máscara del detector de Canny.

Sucede que los objetos próximos al vehículo presentan distintas disparidades, pudiendo un mismo objeto quedar dividido como si fueran varios objetos. Una posible solución a este problema sería obtener la máscara de los bordes de Canny del mapa de obstáculos del cual se hayan eliminado por umbralización dichos elementos cercanos al vehículo. Por tanto, se aplica la función *cvCanny* al mapa de obstáculos que presenta únicamente los obstáculos cuyas disparidades están comprendidas entre 3 y 8 (Figura 5.2 (a)). De esta manera, los objetos más cercanos no se verán afectados por la máscara. A continuación, para conseguir un mapa de obstáculos con los objetos bien diferenciados, se resta la máscara (Figura 5.2 (b)) al mapa de obstáculos binarizado sin umbralizar (Figura 5.2 (c)), obteniendo como resultado la Figura 5.2 (d). Para una mejor comprensión de lo explicado, se adjuntan las líneas de código relativas a esta operación.

```
// imgbin1 es donde vamos a guardar la imagen del mapa de obstáculos quitándole los obstáculos más próximos a la cámara
IplImage *imgbin1 = cvCreateImageHeader(cvSize(width,height),depth,nchannels);
imgbin1->imageData = (char *) mapaobs_bin1;

// Imagen que va a contener los bordes obtenidos por el detector de bordes de Canny de la imagen imgbin1
IplImage* imgEdge = cvCreateImage( cvSize(width,height), 8, 1 );

// Aplicamos para detección de bordes la función cvcanny
cvSmooth( imgbin1, imgEdge, CV_BLUR, 3, 3, 0, 0 );
cvNot( imgbin1, imgEdge );
cvCanny(imgbin1, imgEdge, 1, 1, 3);

// En imgbin2 vamos a guardar la imagen binarizada del mapa de obstáculos de la imagen original, para poder proceder a obtener la resta
IplImage* imgbin2 = cvCreateImage( cvSize(width,height), 8, 1 );
cvCopy(imgMapaObs,imgbin2);

// Binarizamos con la función cvThreshold
cvThreshold( imgbin2,imgbin2, 1, 255, CV_THRESH_BINARY );

// Creamos la imagen resta que será resultado de restar el mapa de obstáculos binarizado y la imagen con los contornos de aquellos obstáculo
//que queremos dividir
IplImage* imgResta = cvCreateImage( cvSize(width,height), 8, 1 );

// Restamos ambas imágenes
cvSub(imgbin2,imgEdge,imgResta,0);
```

Para la obtención de las regiones de interés se lleva a cabo la detección de los contornos de los elementos localizados en *imgResta*. Para ello, se va a emplear la función *cvFindContours* de las librerías. El almacenamiento de dichos contornos va a ser dinámico, para facilitar el acceso a todos los contornos. Por ello, es necesario definir la memoria dinámica con la función *cvCreateMemStorage* y definir la variable que se emplea para almacenar los contornos como una secuencia dinámica (*CvSeq*). Por tanto, la función *cvFindContours* proporciona el número de contornos detectados y éstos quedan registrados en el tipo de datos *CvSeqcontours*, accediendo a cada uno de ellos mediante un simple bucle de la forma:

```
for( ; contours != 0; contours = contours->h_next )
    {

    }
```

La variable *h_next* es un puntero que apunta al siguiente contorno detectado. El bucle permite acceder a cada uno de los contornos almacenados, finalizando cuando el valor de *contours* sea nulo, es decir, se hayan considerado todos ellos. Sin embargo, analizar cada uno de los contornos supone un considerable tiempo de cómputo, por lo cual se ha decido rechazar todos aquellos

objetos cuyo tamaño sea inferior a 300 píxeles. El área de cada uno de los contornos va a ser calculada mediante la función *cvContourArea*. Todos aquellos contornos de un tamaño suficiente como para contener a una persona se recuadran mediante la función *cvBoundingRect*, dando lugar a lo que se designa como Regiones de Interés (ROIs). Éstas vienen definidas por las coordenadas del vértice superior izquierdo y del vértice inferior derecho, denominados *arriba* y *abajo* respectivamente. Llegado este punto, se crea una función que realiza la primera clasificación. A esta función se le llama ***tipo_obstaculo*** y va a ser la encargada de analizar las distintas regiones de interés, determinando si lo que contienen es un objeto elevado o no elevado.

FUNCIÓN *tipo_obstaculo*

Como ya se explicó anteriormente, el procedimiento que se lleva a cabo en este proyecto para considerar que un objeto es elevado o no lo es, consiste en comparar la disparidad real del objeto con la disparidad teórica, es decir, se compara la disparidad que debería tener el objeto, según su posición en la imagen, para estar apoyado sobre el suelo (disparidad teórica) con el valor de disparidad que, en realidad, presenta dicho objeto. El valor de la disparidad teórica se calcula mediante la expresión (14), introduciendo como valor de la coordenada v, el valor correspondiente a la coordenada *v* de la parte inferior de la región de interés. La expresión exacta pasaría a ser la siguiente:

$$d_{teórica} = \frac{v_{abajo} - b}{m} \qquad (14)$$

Por otra parte, la disparidad real se va a calcular recurriendo al histograma de disparidades de dicha región de interés, pues el valor que se busca es el correspondiente al máximo valor del histograma. Para calcular el histograma de cada una de las regiones de interés, es necesario, en primer lugar, extraer dichas regiones de la imagen, concretamente de la imagen correspondiente al mapa de

obstáculos. Las librerías OpenCV disponen de la función *cvSetImageROI* que permite extraer determinadas regiones de una imagen.

```
// Obtenemos las regiones de interés de la imagen
cvSetImageROI( imgROI, cvRect( arriba.x, arriba.y, widthR, heightR ));
```

A continuación, para obtener el histograma de cada ROI se ejecuta la función *cvCalcHist* de las librerías. Por último, la función *cvGetMinMaxHistValue* permite obtener el valor máximo del histograma y la posición en la que se encuentra dicho valor, la cual determina el valor de la disparidad real que se está buscando.

```
// Definimos el histograma mediante el tipo de datos correspondiente
CvHistogram *histograma;

// Creamos un histograma, por defecto un array
histograma = cvCreateHist(1,&hist_size, CV_HIST_ARRAY, ranges, 1);

// Calculamas el histograma de cada uno de las ROI
cvCalcHist( &imgROI, histograma, 0, NULL );

// Obtenemos el valor máximo del histograma y el índice de dicha celda
cvGetMinMaxHistValue( histograma, 0, &max_value, 0, &maxIdx );

// 19. El valor de disparidad real será el índice de la celda donde se
//encuentra el máximo del histograma
obstaculos->d_real=maxIdx;
```

Una vez se dispone de ambos valores de disparidad, real y teórica, se pasa a compararlos entre sí. En el capítulo anterior ya se vio que en el caso de que la diferencia entre la disparidad teórica y la real sea mayor de 2, el objeto que engloba la región de interés es catalogado como elevado; en el resto de los casos, se toma como no elevado.

Conocidos ya cuales de los obstáculos de la escena son elevados y cuales se encuentran apoyados sobre la vía, se procede a analizar los segundos con la intención de identificar si se trata de peatones o no. Por tanto, accediendo únicamente a aquellas ROIs clasificadas como no elevadas, se ejecuta la función ***peatones***, basada en el método HOG.

FUNCIÓN *peatones*

Debido a que la función *hog.detectMultiScale* empleada trabaja con imágenes de tipo *cv::Mat*, se realiza la conversión de la imagen original *imgD* de tipo *IplImage* a la imagen *imgDMat* de tipo *Mat*, para de ella extraer las regiones de interés.

```
// Transformación de una imagen de tipo IplImage a Mat
cv::Mat imgDMat(imgD);
```

En este caso, es necesario aumentar el tamaño de las ROIs, pues éstas sólo delimitan al obstáculo y, para la aplicación de la función *hog.detectMultiScale,* este espacio no es suficiente, pues se requiere un cierto margen. El tamaño idóneo de las regiones se ha obtenido tras realizar una serie de pruebas como ya se verá más adelante. La altura de todas ellas va a ser el valor equivalente en píxeles a lo que corresponderían 2.2 metros en el mundo real. Esta altura en píxeles (h_p) se puede obtener a partir del valor de profundidad Z, cuya expresión se veía en el capítulo anterior (11), realizándose la siguiente conversión (15). En este caso, no se toman las ROIs del mapa de disparidad, sino que para el correcto funcionamiento del HOG se debe trabajar con la imagen original en escala de grises. Por tanto, el modo de proceder en este caso es el siguiente:

```
//Guardamos las coordenadas en un cvRect para posteriormente guardar cada
//uno de los rois como imagen tipo Mat
roi= cvRect( arribaroi.x, arribaroi.y, widthR2, heightR2);

// Guardamos las rois como tipo Mat, obteniéndolas de la imagen original
//también de tipo Mat, gracias a la función cv::Mat Mat(img, cvRect)
img= imgD(roi);
```

$$h_p = 2.2 \cdot \frac{f}{Z} \tag{15}$$

Antes de aplicar la función *hog.detectMultiScale*, tiene que haber sido entrenado el HOG mediante el SVM, empleando la función *hog.setSVMDetector*. Este entrenamiento basta con realizarlo una única vez, por lo que se añade al **main**.

```
// Entrenamiento del HOG por el SVM
hog.setSVMDetector(cv::HOGDescriptor::getDefaultPeopleDetector());
```

Al llamar a la función *hog.detectMultiScale*, se obtiene como resultado un vector dinámico al cual se accede mediante un bucle en el caso de que se haya localizado en dicha región un peatón.

```
// Aplicamos el método HOG
hog.detectMultiScale(img, found, 0, cv::Size(8,8), cv::Size(24,16), 1.05, 2);

// Bucle que accede a cada una de las regiones donde se ha localizado uno o
//varios peatones
for( int c = 0; c < (int)found.size(); c++ )
{

    obstaculos->peaton=1;

}
```

Los peatones son representados en el *display* mediante un rectángulo verde, los objetos no elevados mediante uno rojo y los elevados mediante un rectángulo azul. Para representar un rectángulo en una imagen, las librerías OpenCV disponen de la función *cvRectangle*, a la cual únicamente hay que pasar las coordenadas que lo van a definir e indicar el valor de los canales RGB para que sea dibujado en el color deseado.

```
// Identificación del peatón o de los peatones mediante un rectángulo
//verde
cvRectangle(imgObs, arriba, abajo, CV_RGB(0, 255, 0),1);
```

Capítulo 6

RESULTADOS

En este capítulo se analizan los resultados obtenidos por el algoritmo desarrollado en el presente proyecto. Para realizar el estudio de los resultados se han seleccionado un par de secuencias de imágenes en las que se pueden observar dos situaciones diferentes que tienen en común la presencia de varios peatones en una vía urbana. Ambas secuencias de imágenes están compuestas por 129 *frames*. En primer lugar, se van a analizar los resultados obtenidos en cada una de las secuencias por separado, para luego poder alcanzar una conclusión común sobre el algoritmo desarrollado.

6.1. RESULTADOS OBTENIDOS

SECUENCIA 1

Esta secuencia de imágenes transcurre en una vía perteneciente a las instalaciones de la Universidad Carlos III de Madrid. En dicha vía se pueden observar dos peatones caminando a paso normal, tal y como se muestra en la Figura 6.1. Durante el transcurso de esta secuencia uno de los peatones se aleja de

la cámara, mientras que el otro se acerca a ella; en un punto determinado las trayectorias de ambos peatones se cruzan. Hay que señalar que a lo largo del video aparece un tercer peatón en la escena; sin embargo, éste no es detectado por el HOG, debido a que se encuentra demasiado lejos del vehículo y no corre el riesgo de ser atropellado. El área de búsqueda de peatones está limitada a una distancia de 32,5 metros con el objetivo de que el algoritmo sea más rápido. Se ha tomado esta distancia por ser la más adecuada para conseguir buenos resultados tanto en los tiempos de cómputo como de detección. Su obtención se verá más adelante con más detenimiento.

Figura 6.1: Primera imagen de le secuencia denominada Secuencia 1.

SECUENCIA 2

En esta segunda secuencia aparecen cinco peatones cruzando un paso de cebra de una vía interurbana, tal y como se puede ver en la Figura 6.2. Durante el transcurso de dicha secuencia se observa como existen peatones que quedan ocultos porque se cruzan con otro peatón y como algunos acaban desapareciendo

de la escena. En los últimos *frames* de la secuencia, se puede observar como aparece un sexto peatón. Este peatón es detectado únicamente en los últimos *frames*, ya que hasta entonces no entra en el campo de búsqueda fijado.

Figura 6.2: Primera imagen de le secuencia denominada Secuencia 2.

6.1.1. NÚMERO DE PÍXELES PROCESADOS Y TIEMPOS DE CÓMPUTO

El algoritmo que implementa el modelo del HOG presentado por Dalal y Triggs en [37] es capaz de localizar los peatones presentes en una escena, con un tiempo de procesamiento de la imagen del orden de segundos para imágenes de 640x480 píxeles. Sin embargo, como ya se ha dicho en repetidas ocasiones, interesa que el algoritmo trabaje en tiempo real. Con el objetivo de reducir dicho tiempo de cómputo, se opta por implementar el algoritmo únicamente en las regiones de la imagen consideradas como regiones de interés (ROIs), es decir, donde cabe la posibilidad de que se localice un peatón. Para reducir aún más este tiempo, también se lleva a cabo una clasificación previa de los elementos

detectados, distinguiendo entre elevados y no elevados. La idea de extraer las regiones de interés no elevadas de la imagen tiene como fin reducir la región de búsqueda de peatones, de tal forma que el número de píxeles analizados sea menor y, por tanto, el tiempo de procesamiento se reduzca.

Para estudiar la eficiencia de los resultados obtenidos con el algoritmo desarrollado en este proyecto, estos se van a comparar con los resultados alcanzados cuando se implementa el método basado en los histogramas de gradientes orientados (HOG) sobre toda la imagen. Además, se van a comparar estos resultados con los obtenidos en el caso de que se aplicara el HOG a todas las ROIs detectadas y no sólo a aquellas que han sido previamente clasificadas como no elevadas. De esta forma, se va a poder ver si la clasificación entre elementos elevados y no elevados, realizada antes de distinguir entre peatones y no peatones, permite mejorar la eficiencia de los resultados finales.

SECUENCIA 1

Si se lleva a cabo una comparación del número de píxeles procesados en cada uno de los *frames* de la secuencia (Figura 6.3), se comprueba cómo se consigue reducir en gran medida el número de píxeles a tratar al aplicar el HOG únicamente en aquellas áreas de la imagen que resultan de interés.

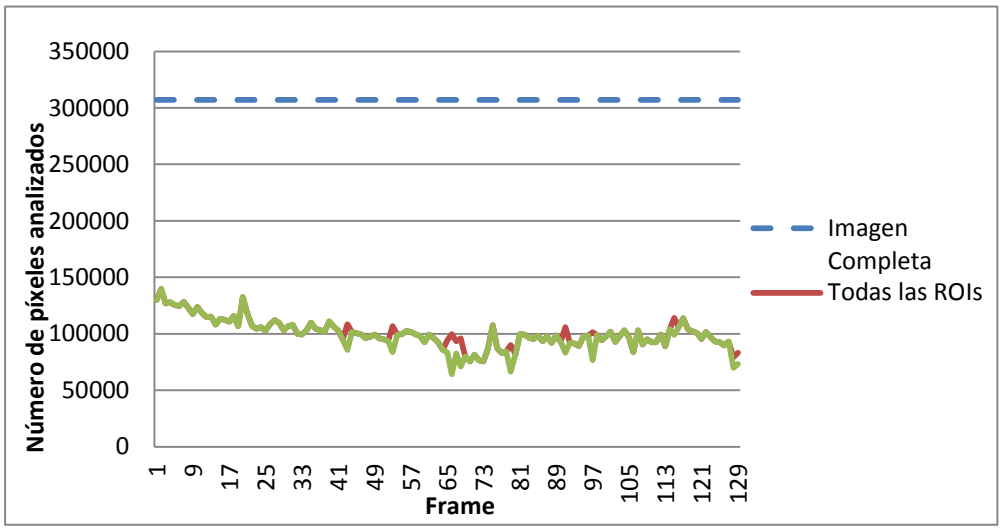


Figura 6.3: Número de píxeles procesados en cada *frame* de la Secuencia 1.

El promedio de píxeles analizados en el caso de tomar únicamente las ROIs es de alrededor de 101.000 píxeles y de 99.200 píxeles en el caso de descartar, además, las regiones elevadas, mientras que al analizar la imagen en su totalidad se requiere un procesamiento de 307.200 píxeles. En la Figura 6.3 se observa que la diferencia del número de píxeles que se analizan en cada uno de los *frames* en los dos primeros casos es muy pequeña. Esto se debe a que en esta escena en concreto son muy pocos los objetos detectados como elevados, teniendo en cuenta que únicamente se estudia aquella parte de la escena que se encuentra a cierta distancia del vehículo, pues es la que resulta de interés. Por el contrario, cuando el vehículo circule por vías urbanas más complejas, la diferencia entre estudiar todos y cada uno de los objetos hallados o sólo a aquellos que se encuentren sobre el suelo va a ser importante.

A continuación, se introduce la Figura 6.4 en la que se comparan los tiempos de cómputo obtenidos en los distintos *frames* de la secuencia. En el algoritmo que se desarrolla en este proyecto se realizan dos mediciones distintas. En una primera medición, se contabilizan los tiempos empleados para realizar todas y cada una de las operaciones que permiten la detección de los peatones. Sin embargo, las operaciones previas a la extracción de las regiones de interés de la escena forman parte de un sistema general de detección de obstáculos. El mapa libre también es utilizado para la obtención de toda clase de información que permita asistir al conductor, ya que proporciona el espacio libre que hay delante del vehículo. Por tanto, se realiza una segunda medición a partir de donde comienza la identificación de las regiones de interés, es decir, una vez que ya se dispone del mapa libre y del mapa de obstáculos. Realmente esta es la medida que resulta de interés y la que se va a tener en cuenta en los siguientes estudios. Aun así se comprueba en la Figura 6.4 (línea verde discontinua) como teniendo en cuenta todo el algoritmo, los tiempos resultantes son mejores que si se aplica el HOG a toda la imagen.

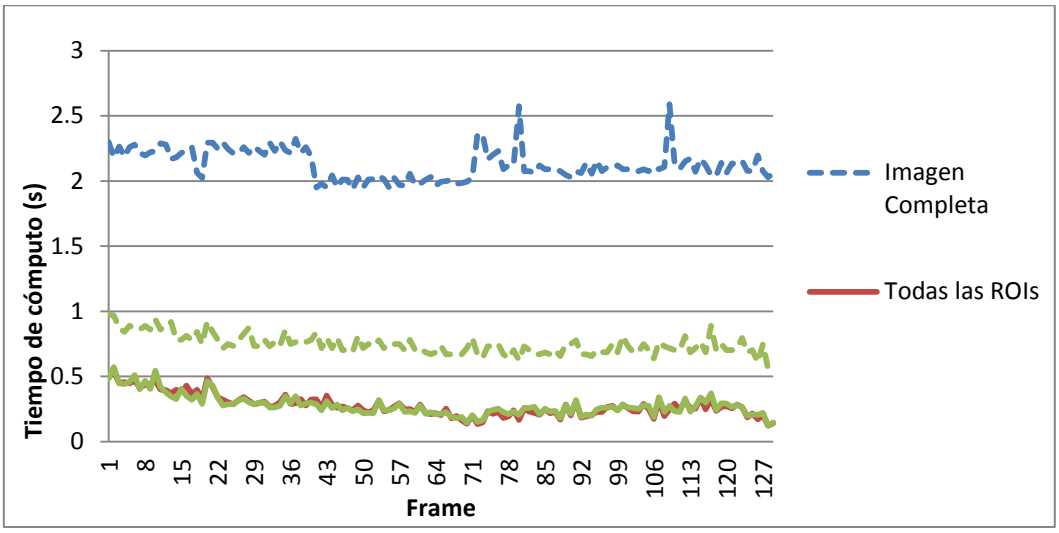


Figura 6.4: Tiempo de procesamiento de cada *frame* de la Secuencia 1.

En esta misma figura se comprueba como la reducción del número de píxeles permite disminuir de manera significativa el tiempo empleado para procesar cada uno de los *frames*. El tiempo medio de análisis de un *frame* cuando se usa el algoritmo que procesa únicamente las regiones no elevadas es de 0,279 segundos, mientras que si se analiza toda la imagen es de 2,128 segundos. Esto podría parecer lógico en un principio, ya que al tratarse de un menor número de píxeles a analizar, obviamente, el tiempo debe ser menor. Sin embargo, el conjunto de operaciones realizadas para la obtención de dichas ROIs supone también un coste computacional que ha de ser considerado. Se comprueba que el coste computacional derivado de la obtención de las ROIs así como del necesario para distinguir los obstáculos elevados y no elevados es inferior al obtenido al aplicar el HOG a toda la imagen.

SECUENCIA 2

En esta segunda secuencia aumenta el número de peatones en la vía, así como el número de elementos localizados en el entorno del vehículo. En estas circunstancias deberían ser más importantes las diferencias entre analizar todas las regiones de interés obtenidas o únicamente las clasificadas como no elevadas.

Para ello, se vuelve a representar en un gráfico (Figura 6.5) el número de píxeles que son analizados en cada uno de los *frames* que forman la nueva secuencia.

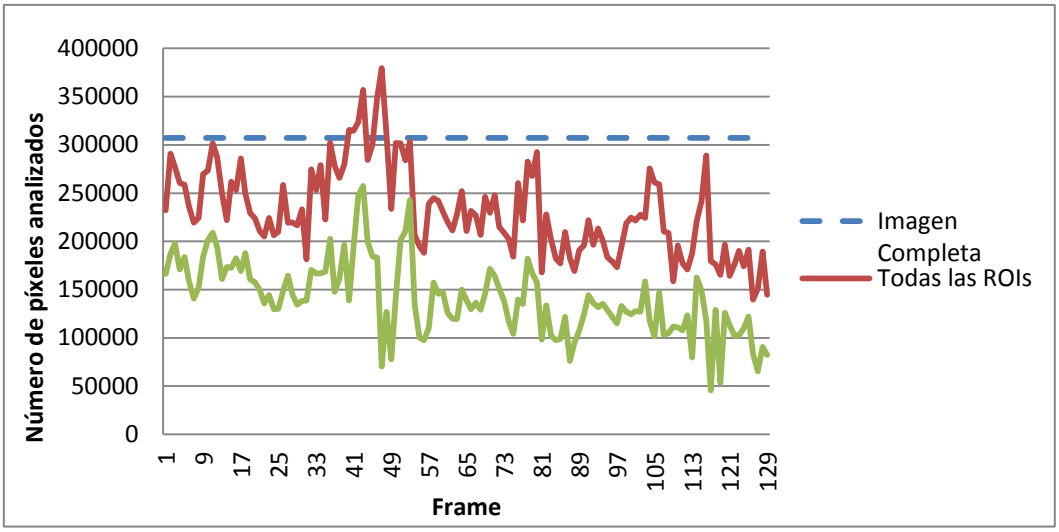


Figura 6.5: Número de píxeles procesados en cada *frame* de la Secuencia 2.

Se observa, como se preveía, una diferencia más destacada entre el número de píxeles analizados al aplicar el HOG a todas las regiones de interés o al aplicarlo sólo a las definidas como no elevadas; más concretamente, se diferencian en más de 90.000 píxeles de media. Esto se debe a que se detecta un mayor número de elementos elevados en comparación con la primera secuencia estudiada. También se aprecia como en algunos de los *frames* el número de píxeles procesados supera al total de píxeles que componen la imagen (307.200 píxeles). Esto es debido a que algunas de las ROIs se superponen. En algunos casos, también sucede que alguna ROI engloba a otra de ellas en su totalidad, lo cual se debe a que se están englobando objetos muy cercanos entre sí.

En la Figura 6.6, en la que se representan los tiempos de cómputo, también se pueden observar estos picos, pues se está procesando un mayor número de píxeles. De nuevo se introducen los tiempos de procesamiento resultantes de extraer las ROIs y analizarlas (línea verde continua) y los tiempos al considerar también los procesos previos a dicha extracción (línea verde punteada).

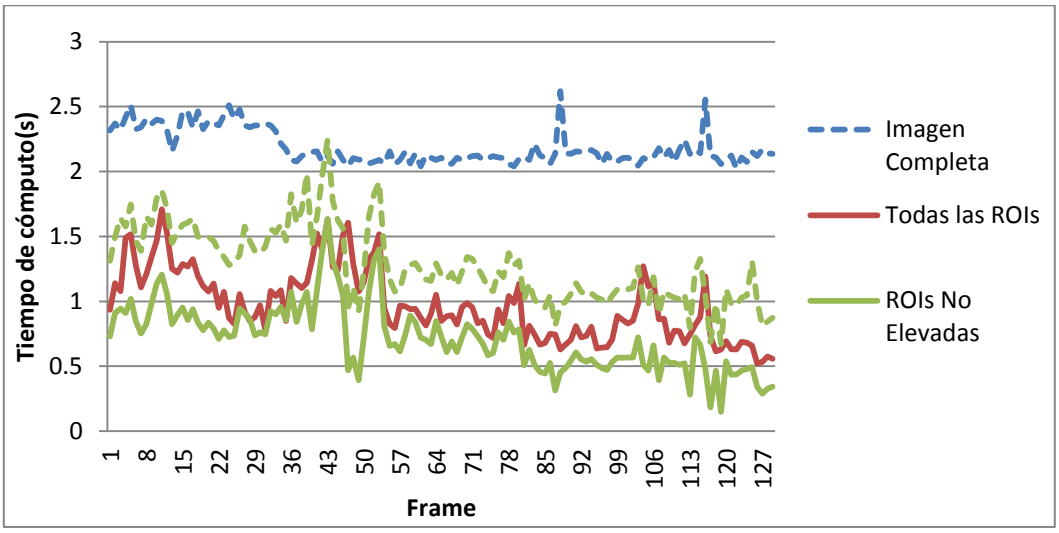


Figura 6.6: Tiempo de procesamiento de cada *frame* de la Secuencia 2.

Sin embargo, en el resto de los *frames* se puede apreciar como el tiempo de cómputo se reduce en gran medida al aplicar el método HOG sólo a las ROIs no elevadas. Atendiendo al tiempo de procesamiento, se comprueba que se obtienen mejores resultados al realizar la clasificación previa que separa los elementos no elevados de los que están elevados, los cuales no suponen un obstáculo para el vehículo. El tiempo medio que presenta este algoritmo es de 0,716 segundos, mientras que el tiempo de cómputo medio al procesar todas y cada una de las ROIs es de 0,971 segundos. Hay que mencionar que el tiempo medio de procesado de la imagen completa es de 2,186 segundos, similar al obtenido en la primera secuencia.

6.1.2. RECONOCIMIENTO DE LOS PEATONES COMO OBSTÁCULOS

Para que el HOG pueda identificar a un objeto como peatón es necesario que dicho peatón forme parte de una ROI. Pueden ser dos las causas de que un peatón no haya sido identificado como tal: que dicho peatón no haya sido reconocido como objeto de interés, es decir, no haya sido englobado por una ROI, o, por el contrario, sí que se detecte como objeto pero que el HOG no lo reconozca como peatón. Se ha podido comprobar visualmente que en la primera secuencia todos

los peatones que aparecen son detectados como obstáculos, mientras que en la segunda, el índice de acierto disminuye ligeramente debido a que se trata de una imagen más compleja, en la que los peatones son más difíciles de identificar por estar más cerca unos de otros. En este caso, el índice de éxito pasa a ser del 88%, es decir, de 462 detecciones posibles se han realizado 405. Si se calcula este mismo dato para el algoritmo que no hace distinción entre elementos elevados y no elevados, el número de peatones identificados como obstáculos pasa a ser 426. Esta diferencia se debe a que en algunas ocasiones el peatón no queda completamente dentro de la ROI y el algoritmo determina que se trata de un obstáculo elevado. La solución a este problema se encuentra en aumentar el tamaño de las regiones de interés, lo que implicaría a su vez un aumento del tiempo de cómputo. Como se trata de una diferencia pequeña, se decide mantener el tamaño fijado de las ROIs. Este hecho se va a demostrar más adelante, en el apartado 6.2, donde se va a realizar un estudio sobre la variación de los resultados al modificar las dimensiones de las ROIs.

6.1.3. FALSOS POSITIVOS Y FALSOS NEGATIVOS

Otros datos útiles para determinar el grado de eficacia del algoritmo desarrollado son los falsos positivos y negativos que se obtienen en cada una de las imágenes que componen la secuencia. Puede suceder que a la hora de tener en cuenta el tiempo de procesamiento de las imágenes resulte mejor dividir la escena en ROIs, pero esto puede empeorar la detección de peatones, haciendo ineficiente el algoritmo.

SECUENCIA 1

En este caso, va a existir un falso positivo cuando el algoritmo identifique como peatón a un objeto que realmente no lo es. Ya se ha comentado que en la escena estudiada sólo hay dos peatones que resultan de interés para el estudio, debido a su proximidad con el vehículo. En la Figura 6.7 se representa el número de falsos positivos obtenidos en cada uno de los *frames* que componen la primera

secuencia. Como se puede observar, no surge ningún falso positivo en los algoritmos que analizan sólo regiones de interés y su valor es mínimo cuando se aplica el HOG a toda la imagen.

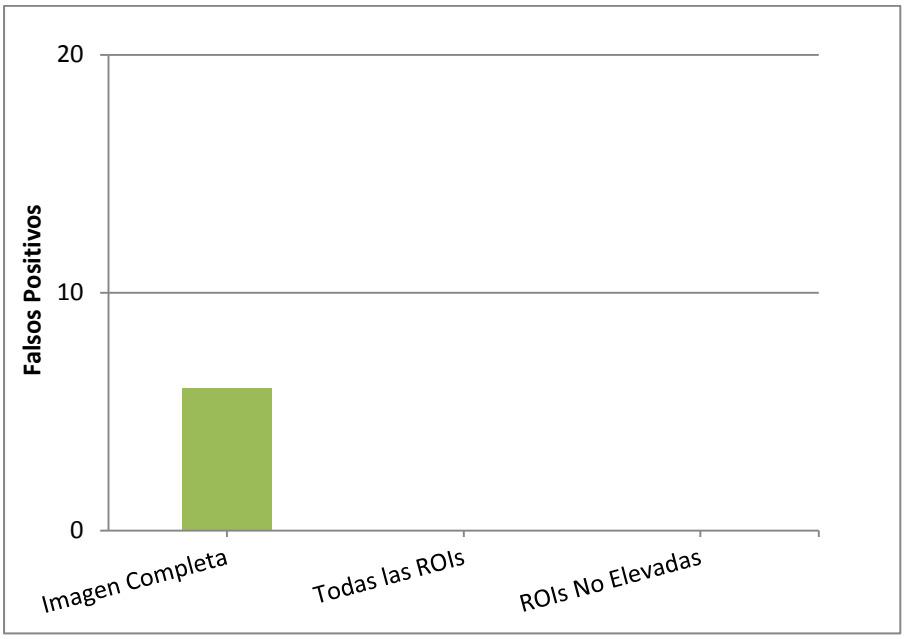


Figura 6.7: Número de Falsos Positivos obtenidos en cada uno de los *frames* de la Secuencia 1.

El problema surge con el número de falsos negativos obtenidos. Aparece un falso negativo cuando no se detecta un peatón que realmente se encuentra presente en la escena. En este caso se comprueba que donde se detecta un mayor número de falsos negativos es en los algoritmos en los que el HOG es implementado en las regiones de interés, tal y como se puede ver en la Figura 6.8.

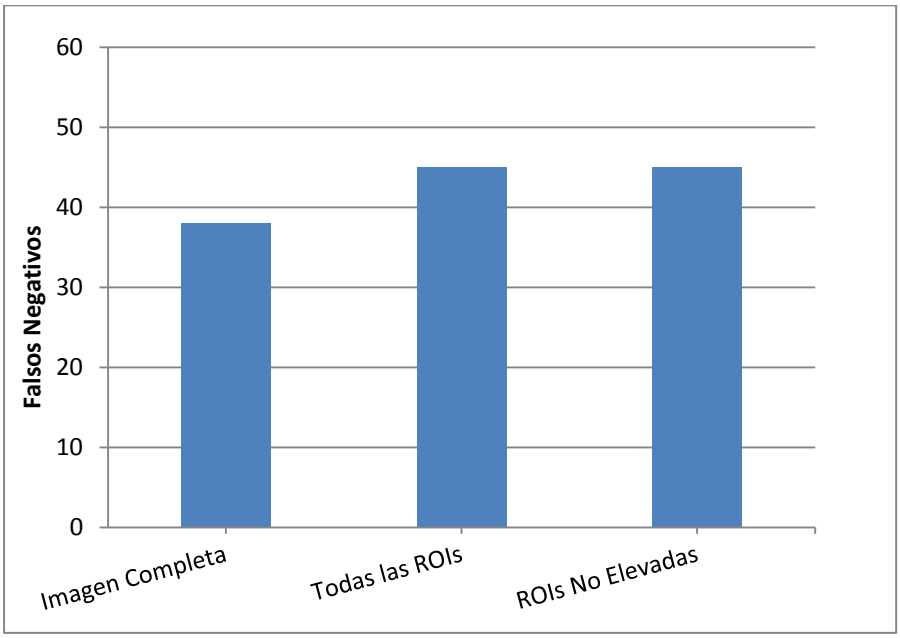


Figura 6.8: Número de Falsos Negativos obtenidos en cada uno de los *frames* de la Secuencia 1.

Si se obtiene el número total de peatones que deberían ser detectados, considerando los 129 *frames*, y se compara con los realmente identificados como peatones, se obtiene el índice de aciertos de cada uno de los algoritmos estudiados (Tabla 6.1). En esta secuencia, los datos de los algoritmos que extraen las regiones de interés coinciden, debido a que, como ya se ha dicho anteriormente, se detectan pocos obstáculos elevados. El índice de aciertos en ambos algoritmos es del 81%, es decir, únicamente se dejan de detecta el 19% de los peatones. Para la obtención de estos datos hay que tener en cuenta que durante la secuencia, uno de los peatones queda oculto tras el otro y que en un determinado momento uno de los peatones se aleja tanto del vehículo que deja de ser detectado por el HOG, pues la distancia que lo separa de las cámaras hace compleja su identificación como peatón.

En la Figura 6.8 se observa como no existe una gran diferencia entre el número de falsos negativos obtenido al analizar toda la imagen en comparación con los obtenidos al analizar únicamente las ROIs. Esto significa que los errores de detección no se pueden atribuir totalmente a la extracción de las regiones de

interés de la imagen, pues si se compara el índice de aciertos de ambos algoritmos, la diferencia no es significativa.

	ÍNDICE DE ACIERTOS		
	HOG_Imagen Completa	HOG_Todas las ROIs	HOG_ROIs No Elevadas
Peatones Detectados	196 peatones/234	189 peatones/234	189 peatones/234
% Aciertos	84%	81%	81%

Tabla 6.1: Datos relativos a los aciertos al aplicar el HOG a la imagen completa o a las ROIs.

Mientras que el porcentaje de aciertos al analizar la imagen completa es del 84%, al analizar sólo en las ROIs determinadas como no elevadas, este porcentaje se reduce al 81%. Esta diferencia no se puede considerar lo suficientemente significativa como para definir como inadecuado el proceso de reconocimiento de peatones en las regiones de interés.

A continuación se añade la Tabla 6.2 donde se muestran de forma detallada los índices de falsos negativos obtenidos en los tres algoritmos considerados para la realización del estudio de los resultados. Se comprueba que son pocos los *frames* en los que no se detecta ningún peatón y, como ya se ha dicho, se debe a la posición que toma uno de los peatones.

HOG_ IMAGEN COMPLETA			HOG_TODAS LAS ROIs			HOG_ROIs NO ELEVADAS		
FN = 0	FN = 1	FN = 2	FN = 0	FN = 1	FN = 2	FN = 0	FN = 1	FN = 2
97	26	6	90	33	6	90	33	6

Tabla 6.2: Relación de número de *frames* según el número de falsos negativos obtenidos para cada uno de los tres algoritmos a estudio (Secuencia 1).

SECUENCIA 2

En esta secuencia, el número de peatones presentes en la escena va variando conforme van pasando los *frames*. De nuevo, son pocos los falsos

positivos obtenidos. Hay que resaltar el caso del algoritmo en el que se buscan peatones en todas las regiones de interés, en el que se detecta un falso positivo en repetidas ocasiones al final de la secuencia, como se comprueba en la Figura 6.9. Esto es debido a que en estos últimos *frames* se está detectando un peatón cuando realmente no lo hay.

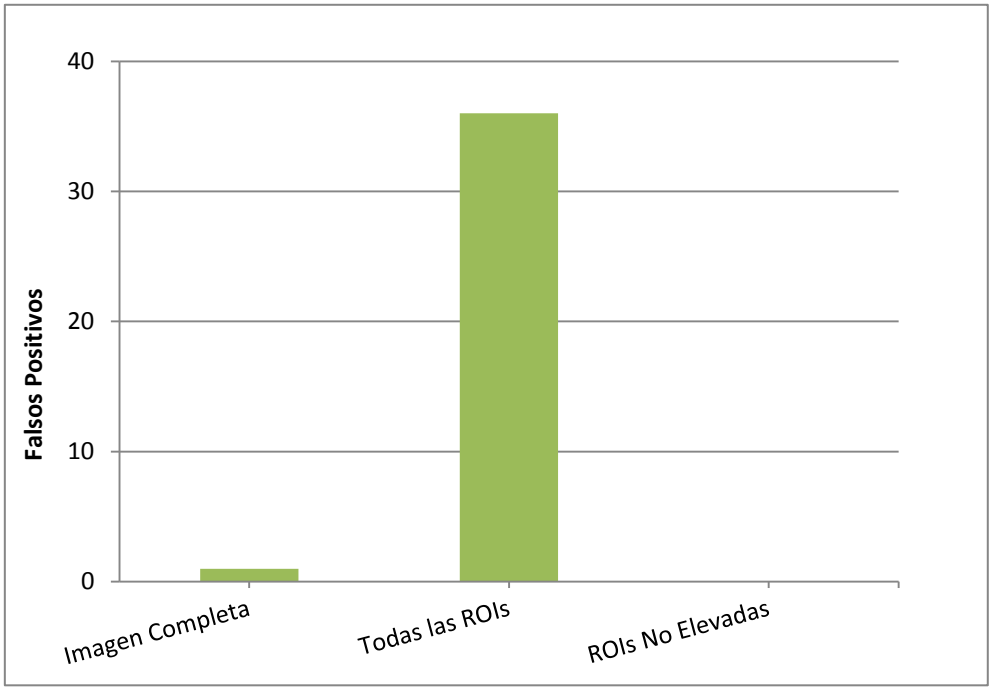


Figura 6.9: Número de Falsos Positivos obtenidos en cada uno de los frames de la Secuencia 2.

El número de falsos negativos obtenidos en esta secuencia (Figura 6.10) es mayor debido en parte a que al haber un mayor número de peatones en la escena, crece la probabilidad de que alguno de ellos no sea detectado. Al igual que sucedía a la hora de definir los peatones como objetos de interés, en este caso también se obtienen resultados ligeramente peores en el algoritmo en el que sólo se analizan las ROIs clasificadas como no elevadas si se compara con los resultados obtenidos en el algoritmo en el que se analizan todas las ROIs. Estos resultados tienen sentido, ya que al haber sido identificadas estas regiones como elevadas en la clasificación previa a la detección de los peatones, no son tomadas como posibles candidatas a contener peatones y por tanto, no son analizadas por el HOG.

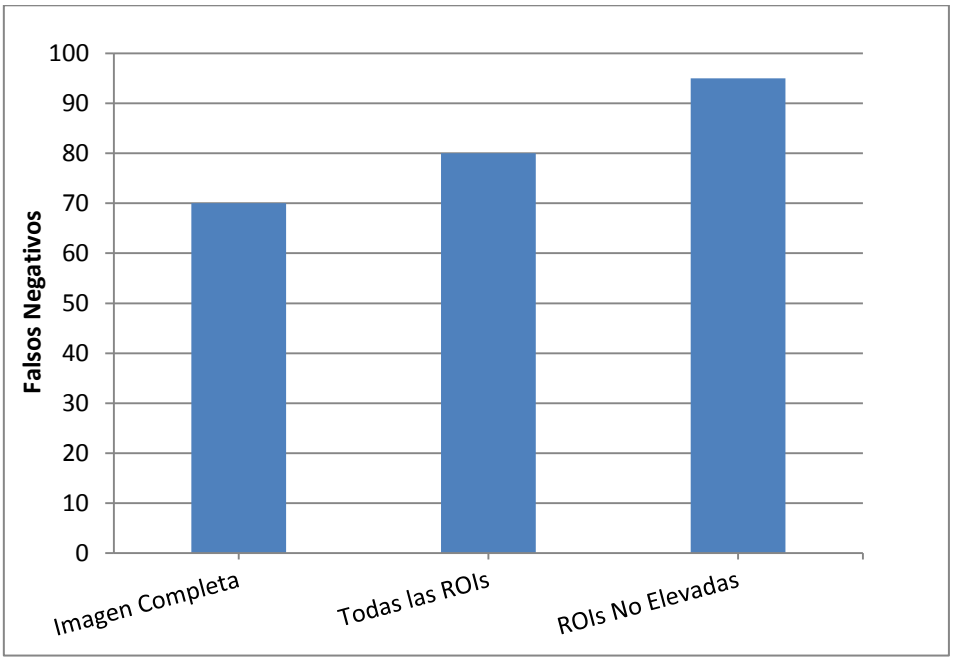


Figura 6.10: Número de Falsos Positivos obtenidos en cada uno de los frames de la Secuencia 2.

En esta secuencia, el índice de aciertos pasa a ser del 79% (Tabla 6.3) frente al 81% de la primera secuencia, pero como ya se ha dicho, se trata de una vía urbana con mayor cantidad tanto de peatones como de obstáculos en general, lo que hace más compleja la detección. A pesar de que la calidad de detección mejora discretamente si no se hace distinción entre obstáculos elevados y no elevados, la Figura 6.6 mostraba que este algoritmo presenta tiempos de procesamiento bastante más elevados que si sólo se aplica el HOG en las ROIs no elevadas, algo más de un 26%.

	ÍNDICE DE ACIERTOS		
	HOG_Imagen Completa	HOG_Todas las ROIs	HOG_ROIs No Elevadas
Peatones Detectados	392 peatones/462	382 peatones/462	367 peatones/462
% Aciertos	85%	83%	79%

Tabla 6.3: Datos relativos a los aciertos el HOG a la imagen completa o a las ROIs.

Al igual que sucedía en la primera secuencia, el índice de aciertos obtenidos en el algoritmo desarrollado en este proyecto, no difiere de forma significativa del obtenido al realizar la búsqueda de peatones en toda la imagen, como se observa en la Tabla 6.3.

En la Tabla 6.4 se introduce el número de *frames* en los que se detectan todos los peatones presentes, así como el número de *frames* en los que quedan uno o dos peatones sin identificar. Se puede ver con claridad como al haber más peatones en la vía, los resultados empeoran con respecto a los obtenidos en la primera secuencia; no obstante, siguen siendo muchos los *frames* en los que se detectan todos los peatones.

HOG_ IMAGEN COMPLETA			HOG_TODAS LAS ROIs			HOG_ROIs NO ELEVADAS		
FN = 0	FN = 1	FN = 2	FN = 0	FN = 1	FN = 2	FN = 0	FN = 1	FN = 2
73	42	14	65	48	16	59	45	25

Tabla 6.4: Relación de número de *frames* según el número de falsos negativos obtenidos para cada uno de los tres algoritmos a estudio (Secuencia 2).

6.2. DISCUSIÓN DE LOS RESULTADOS

A la vista de los resultados, se llega a la conclusión de que al realizar la extracción de las regiones de interés se pierde cierta información que se traduce en un aumento del índice de falsos negativos, es decir, la detección de los peatones empeora con respecto a la obtenida al realizar dicha búsqueda en toda la imagen. Sin embargo, se busca desarrollar un sistema que asista al conductor, por lo que es importante que se trabaje en tiempo real. Analizar todos los píxeles que forman la imagen conlleva un tiempo de procesamiento aproximado de 2 segundos, mientras que si sólo se realiza la búsqueda en las regiones de interés no elevadas, este tiempo es inferior a medio segundo en vías urbanas donde existe una gran cantidad de obstáculos. Teniendo en cuenta lo anterior, 2 segundos es un tiempo excesivo si lo que se quiere es prevenir accidentes. También se comprueba que,

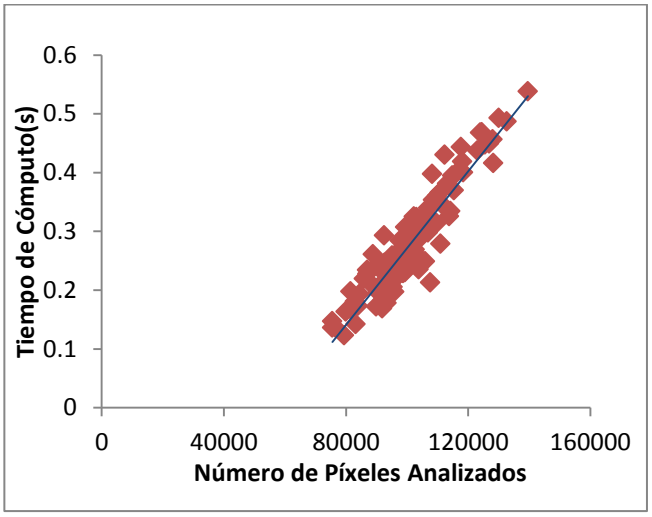
cuanto más próximo se encuentra el peatón con respecto a la cámara, mayor es la probabilidad de que se detecte dicho peatón; los problemas de detección surgen sobre todo cuando el peatón se encuentra alejado del vehículo pues se encuentra peor definido, tal y como se puede ver en la Figura 6.11.

Figura 6.11: ROIs extraídas de uno de los *frames* de la Secuencia 1.

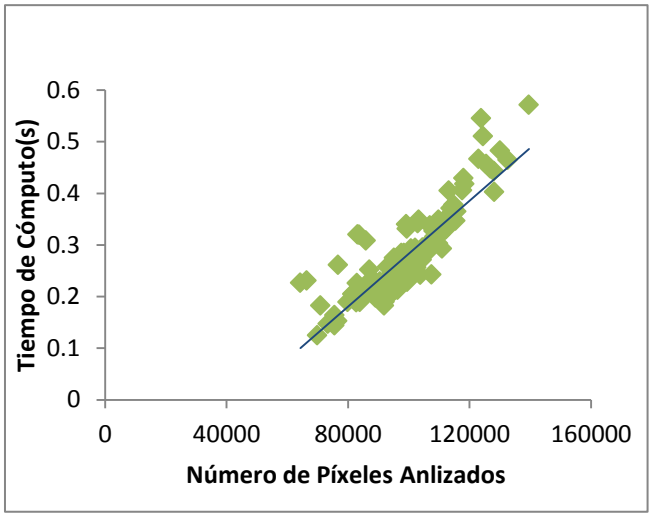
Además, hay que tener en cuenta que el algoritmo ha sido desarrollado para ser utilizado en vías interurbanas, en las cuales no se debe circular a una velocidad superior a 50 km/h. A esta velocidad, el algoritmo va a disponer de mayor margen de respuesta. En el caso de que un peatón captado por la cámara no sea detectado inicialmente por el algoritmo, si dicho peatón permanece en la escena será detectado en sucesivos *frames* o cuando su distancia con el vehículo se reduzca.

Si se lleva a cabo ahora una comparación entre los algoritmos que buscan peatones únicamente en las regiones de interés, se comprueba que la calidad de la detección es similar (Tablas 6.1 y 6.3). Sin embargo, la diferencia en los tiempos de cómputo es más pronunciada. Al realizar la búsqueda de peatones únicamente en aquellas regiones consideradas no elevadas se logra una reducción interesante del tiempo de cómputo medio. Esta diferencia va a aumentar en relación al número de elementos presentes en la vía, tal y como se ve en la Figura 6.6, correspondientes a la secuencia número dos.

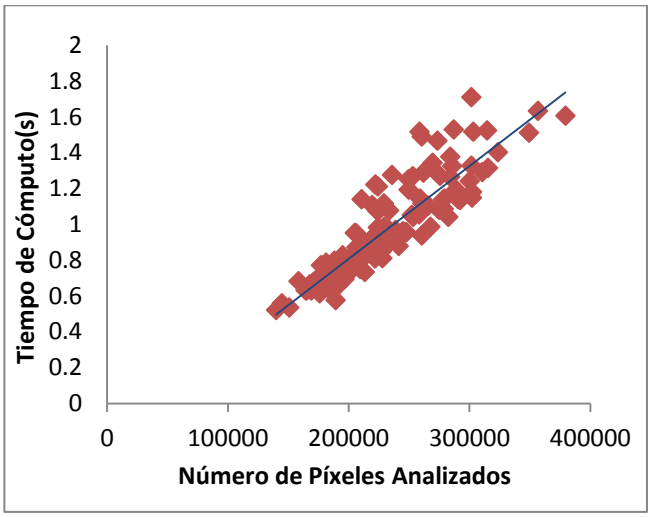
En la Figura 6.12 se observa como la nube de puntos que se obtiene al comparar el número de píxeles que se analiza en cada uno de los *frames* frente a los tiempos de procesamiento de los mismos presenta una tendencia lineal. Esto significa que al aumentar el número de píxeles aumenta el tiempo empleado para procesar el *frame*. Mientras que analizar la imagen completa, compuesta por 307.200 píxeles, lleva un tiempo de cómputo superior a los 2,5 segundos, se comprueba en la Figura 6.12 (a) y (c) que este tiempo se reduce de forma significativa al analizar sólo las regiones de interés extraídas y aún más si se descarta el análisis de los elementos considerados elevados (Figura 6.12 (b) y (d)). Por lo tanto, como ya se ha comentado, para acercarse al objetivo que es identificar los peatones presentes en la escena en tiempo real, interesa analizar el menor número de píxeles posible, consiguiendo una tasa de aciertos similar.



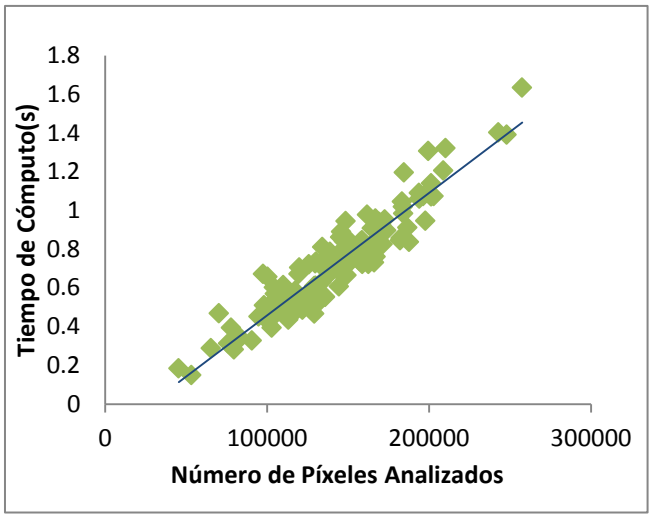
(a)



(b)



(c)



(d)

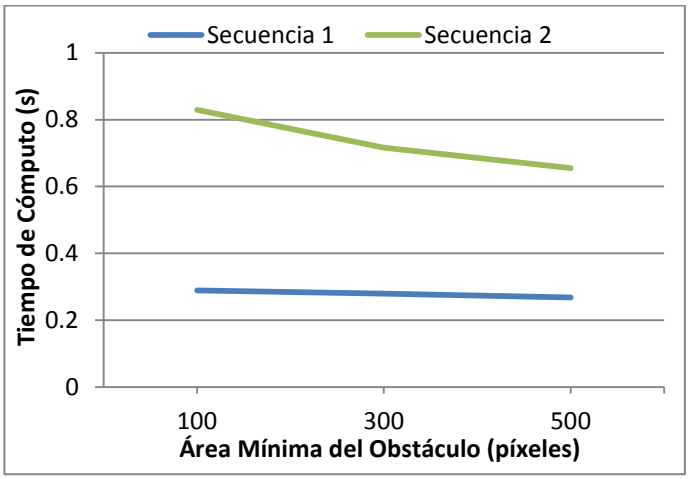
Figura 6.12: Tiempo de cómputo/ Número de píxeles procesados aplicando el HOG (a)en todas las ROIs en las imágenes de la Secuencia 1, (b) en las ROIs no elevadas en las imágenes de la Secuencia 1, (c) en todas las ROIs en las imágenes de la Secuencia 2, (d) en las imágenes de la Secuencia 2.

Alguno de los parámetros introducidos en el algoritmo desarrollado, influyen de forma directa en los resultados. Variando estos parámetros se consigue reducir los tiempos de cómputo de cada *frame* o mejorar el índice de aciertos obtenido. A continuación, se definen estos parámetros:

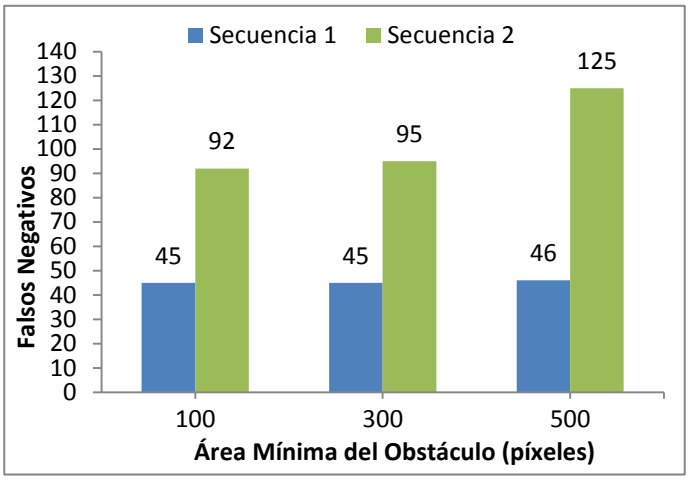
- Área mínima del obstáculo: se establece un valor mínimo del área del obstáculo que delimita cada objeto localizado en la imagen para rechazar aquellos que son demasiado pequeños para poder considerarse un peatón.
- Dimensión de la ROI: modificando las coordenadas que definen la región de interés se consigue analizar un mayor o menor número de píxeles alrededor del obstáculo.
- Umbral de detección: se prefija un valor de disparidad, de tal forma que aquellos objetos que lo superen, van a ser eliminados del mapa de obstáculos.

Sucede que, al conseguir una mejora en los tiempos, la detección de peatones pierde calidad, reduciéndose el índice de aciertos. Por ello, se ha

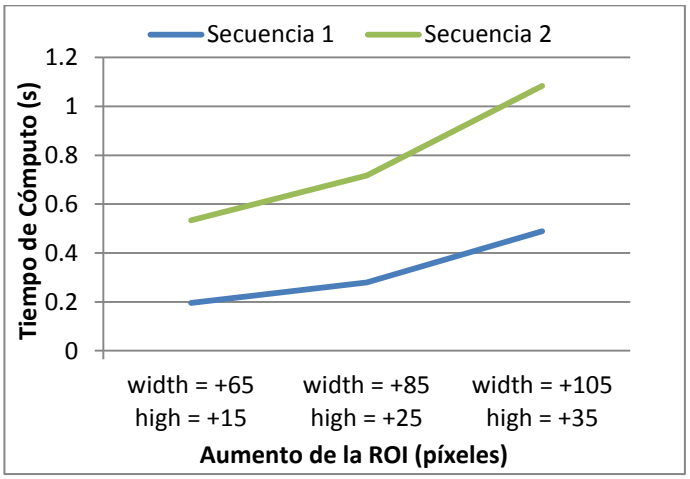
realizado un estudio para obtener los valores que hagan óptimos los resultados del algoritmo y que han sido utilizados en el análisis de las secuencias de estudio.



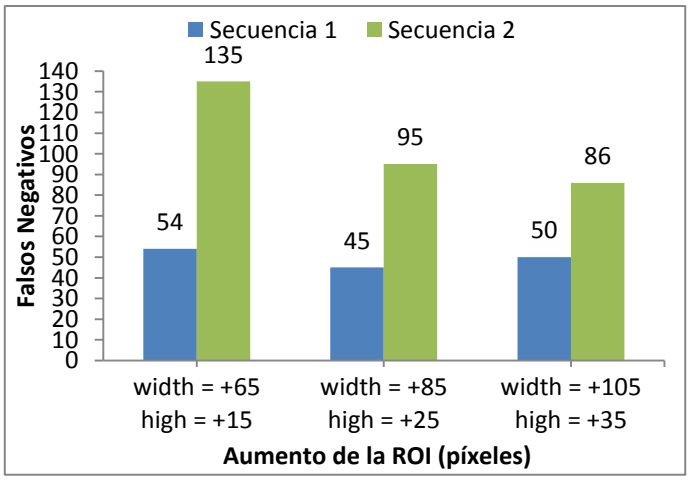
(a)



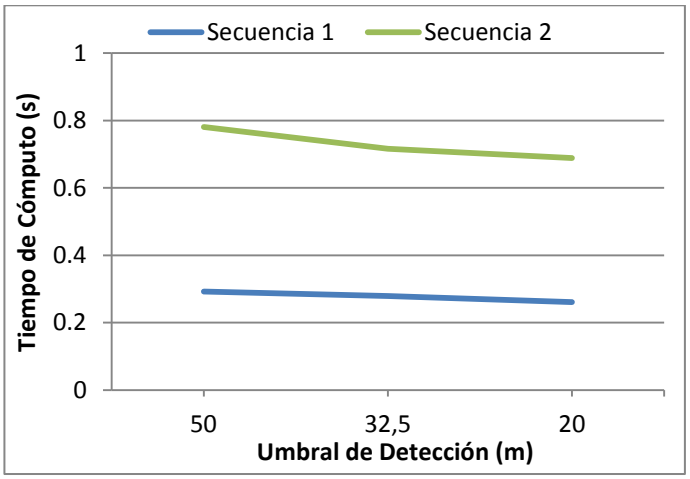
(b)



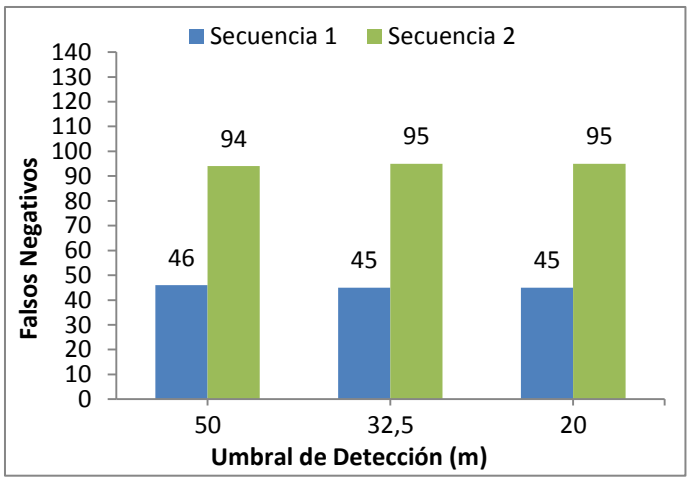
(c)



(d)



(e)



(f)

Figura 6.13: Resultados obtenidos (a) en el tiempo de cómputo al modificar el área mínima del objeto, (b) en el índice de aciertos al modificar el área mínima del objeto, (c) en el tiempo de cómputo al modificar la dimensión de la ROI, (d) en el índice de aciertos al modificar la dimensión de la ROI, (e) en el tiempo de cómputo al modificar el umbral de detección, (f) en el índice de aciertos al modificar el umbral de detección.

Las conclusiones que se han obtenido con la realización de este estudio han sido las siguientes:

- El área mínima que debe presentar un objeto para poder ser considerado peatón es de 300 píxeles. En la Figura 6.13 (a) y (b) se observa que al reducir más este valor se consigue mejorar el índice de aciertos en la detección de peatones (disminuye el número de falsos negativos), pero, a su vez, aumenta el tiempo de cómputo. Si por el contrario, se opta por aumentar este área mínima, aumenta de forma significativa el número de peatones que no son identificados, a cambio de reducirse mínimamente el tiempo de cómputo.

- Al aumentar el tamaño de las regiones de interés se consigue aumentar el margen existente entre el objeto y el borde de la ROI, resultando conveniente para aquellos casos en los que el peatón no haya sido totalmente englobado en la región de interés. Con el aumento fijado, se consigue reducir en gran medida el hecho de que un peatón sea

parcialmente encuadrado por el rectángulo que delimita la ROI. Al aumentar estas regiones, el número de píxeles a procesar en cada *frame* aumenta considerablemente, por lo que también lo hace el tiempo de dicho procesamiento (Figura 6.13 (c)), mientras que el índice de aciertos no se ve apenas incrementado (Figura 6.13 (d)). Si, por el contrario, se reduce el tamaño de las ROIs, aumenta de forma considerable el número de falsos negativos. Por tanto, se opta por tomar unos valores que ofrezcan resultados adecuados, sin que el tiempo de cómputo se vea muy alterado.

- El campo de detección de peatones se limita mediante el valor umbral mencionado anteriormente, de tal forma que se rechaza la búsqueda de aquellos peatones cuya distancia al vehículo sea tal que no haya riesgo inmediato de colisión. Sin embargo, como se puede observar en la Figura 6.13 (e) y (f) la modificación de este parámetro no altera en gran medida los resultados obtenidos, ya que cuanto más alejado se encuentra un peatón más se complica la tarea de identificación para el HOG, por lo que aunque sean detectados como objetos, siguen sin ser detectados como peatones.

Para concluir, se puede decir que con los valores definidos finalmente se ha logrado alcanzar un equilibrio, de manera que los resultados ofrecidos por este algoritmo cumplen con el objetivo que se persigue en este proyecto.

Capítulo 7

CONCLUSIONES Y TRABAJOS FUTUROS

Una vez analizados los distintos parámetros del algoritmo y obtenido los resultados correspondientes se puede proceder a exponer las conclusiones del proyecto. Además se mencionan posibles trabajos que podrían derivar del actual.

7.1. CONCLUSIONES

En el presente proyecto se ha buscado con éxito identificar los peatones presentes en la escena captada por un par de cámaras estéreo instaladas sobre un vehículo que circula por una vía urbana compleja. El estudio realizado sobre el efecto que tiene la variación de los distintos parámetros del sistema sobre el tiempo de cómputo y sobre la eficacia de la detección, ha permitido alcanzar los objetivos de este proyecto.

Durante el desarrollo de este trabajo, se ha estado resaltando la importancia de que el algoritmo funcione en tiempo real, ya que se pretende que el conductor sea alertado de una posible colisión con un peatón con tiempo suficiente para evitar dicha colisión o en caso de que se produzca, reducir sus consecuencias. A pesar de que los tiempos obtenidos varían según la complejidad de la vía, al ser más o menos los píxeles que se tienen que analizar, puede decirse que los resultados alcanzados, en cuanto a tiempo de cómputo se refiere, son positivos.

Por otro lado, cambios constantes en la iluminación y fallos en la detección de determinadas siluetas dan lugar a que se produzcan detecciones de falsos positivos y falsos negativos. No obstante, la tasa de error del algoritmo en este aspecto es lo suficientemente reducida como para no impedir lograr los objetivos finales. Buscar disminuir estos falsos resultados mediante el tratamiento de la imagen, implicaría un aumento del coste computacional. En la implementación del algoritmo de Dalal y Triggs [37], correspondiente a los descriptores HOG, aplicada a toda la imagen también se obtienen una serie de falsos negativos y falsos positivos, que dan lugar a que en determinados *frames* dejen de detectarse peatones o se definan como tales objetos que no lo son. Esto significa que se consigue mejorar la implementación del algoritmo de Dalal y Triggs, ya que a pesar de empeorar levemente el índice de aciertos, se logra reducir de manera considerable los tiempos de procesamiento de las imágenes captadas por el par de cámaras.

7.2. TRABAJOS FUTUROS

Otra posible vía para la mejora del mismo sería combinar el dominio visible con la utilización de cámaras de infrarrojo [39]. Estas cámaras no dependen en gran medida de la iluminación y ofrecen un menor nivel de ruido, haciendo más sencilla la etapa de detección de los peatones. Por el contrario, presentan la desventaja de que carecen de texturas y colores, dando lugar a pérdida de información. Por tanto, una combinación de ambos sistemas proporcionaría mejores resultados.

Una vez conocida las posiciones reales de los distintos peatones puede aplicarse una técnica de seguimiento para predecir las trayectorias de estos peatones. Esto permitiría estudiar si alguna de las posibles trayectorias coincide con la del vehículo y de ser así, alertar al conductor con la antelación adecuada para que reaccione ante esa situación. Para este fin se puede recurrir a la aplicación del *filtro Kalman* que permite obtener las posibles trayectorias de los peatones localizados en la imagen mediante estimaciones de los estados de un sistema lineal. En este caso mediante la estimación que proporciona el filtro de Kalman se pretende generar el seguimiento de cada uno de los peatones localizados en procedimientos anteriores.

Capítulo 8

COSTES DEL PROYECTO

En este capítulo se pretende hacer una breve estimación del coste del proyecto. Por un lado, se calcula el coste de las horas empleadas por el ingeniero en la realización del proyecto y, por otro, se calcula el coste del material necesario para la realización del mismo.

En la Tabla 8.1 se expresan el número de horas aproximadas de cada una de las fases en las que se ha dividido el proyecto:

ETAPA	TIEMPO
Estudio y comprensión del problema a resolver	40 horas
Recopilación de la documentación	50 horas
Redacción de la memoria (parte teórica)	100 horas
Implementación del código	200 horas
Pruebas del algoritmo	20 horas
Redacción de la memoria (parte práctica)	50 horas
TOTAL	**460 horas**

Tabla 8.1: Número de horas empleadas en la realización del proyecto.

Si se supone que el salario base del ingeniero es de 35€/h (impuestos incluidos), se obtiene que el coste de la mano de obra es:

$$460\ horas \cdot 30 \frac{€}{hora} = 13.800\ €$$

Los materiales utilizados para llevar a cabo este proyecto y sus correspondientes costes son los mostrados en la Tabla 8.2:

MATERIAL	**COSTE**
PC estándar	1.200 €
Paquete de librerías OpenCV	Gratuito
Cámara estéreo (Modelo *Bumbleblee* del fabricante *Pointgrey*)	3.000 €
Interfaz salpicadero (Pantalla *Xenarc* 8 pulgadas)	400 €
Otro material (cables, soportes, etc.)	100 €
TOTAL	**4.700 €**

Tabla 8.2: Coste de los materiales empleados en la realización del proyecto.

Esto supone un **coste total del proyecto** de:

$$13.800\ € + 4.700\ € = \mathbf{18.500}\ €$$

ANEXO 1

SOFTWARE

1. VISUAL STUDIO C++

Para la realización de este proyecto se ha empleado el entorno de desarrollo integrado para Windows, Microsoft Visual Studio, más concretamente Microsoft Visual C++ 2010 Express. Este entorno presenta varios lenguajes de programación. En este caso, se va a utilizar Visual C++. El lenguaje de programación C++ es un lenguaje orientado a objetos.

2. LIBRERÍAS OPENCV

OpenCV [2] es un conjunto de librerías de software que contiene una gran cantidad de algoritmos para la visión por computador en tiempo real, desarrollada originalmente por Intel. Su publicación se da bajo licencia BSD, que permite que estas librerías sean utilizadas de forma gratuita, tanto para propósitos comerciales como de investigación. Su programación se realiza en código C y C++ optimizados y, además, aprovechan las capacidades de procesadores con múltiples núcleos.

OpenCV también puede utilizar el sistema de primitivas de rendimiento integradas de Intel, IPP (Intel Performance Primitives), que consisten en un conjunto de rutinas de bajo nivel, específicas para los procesadores de esta compañía. Estas librerías son multiplataforma, es decir, están disponible para GNU/Linux, Mac OS X y Windows.

Las librerías OpenCV se caracterizan por contener más de 500 funciones que abarcan una gran gama de áreas en el procesamiento de imágenes, como reconocimiento de objetos, calibración de cámaras, seguridad, visión estéreo y visión robótica. OpenCV incluye cuatro componentes diferentes:

- **CxCore:** Contiene las estructuras de datos básicas, el soporte XML y las funciones de dibujo.
- **CvReference:** Incluye todas las funciones para el procesamiento y análisis de imágenes, captura de movimiento, reconocimiento de patrones, calibración de cámaras y análisis de estructura de imágenes.
- **CvAux:** Contiene funciones para correspondencia estéreo, cambio de forma del punto de vista de las cámaras, seguimiento 3D en estéreo, funciones para el reconocimiento de objetos PCA y modelos de ocultos de Markov embebidos.
- **HighGui:** Permite crear y abrir ventanas para mostrar las imágenes, leer y escribir gráficos, tanto de imágenes como de vídeo, y manejar de forma simple el ratón, el puntero y otros elementos del teclado.
- **ml (Machine Learning):** Se emplea para convertir los datos en información mediante la extracción de reglas o patrones de los datos.
- **Cv:** Contiene las funciones necesarias para las operaciones de procesamiento de imágenes.

Las funciones correspondientes a estas librerías se distinguen porque son nombradas como “cv” más el nombre de la función, mientras que si se trata de tipos de datos se le antepone “Cv” al nombre del tipo de dato.

De los tipos de datos empleados hay que destacar aquellos utilizados para definir imágenes. En la mayoría de los casos, se han definido las imágenes como un tipo de dato *IplImage*. Es el tipo de datos básico y permite representar las imágenes sean del tipo que sean. Sin embargo, para la implementación del método HOG ha sido necesario convertir las regiones de interés al tipo Mat, ya que éste almacena los elementos como cualquier matriz y ofrece la posibilidad de acceder a información adicional, que en ocasiones puede resultar útil. Otro tipo de datos muy utilizado para definir las coordenadas de un determinado punto mediante números enteros ha sido *CvPoint*.

ANEXO 2

CÓDIGO DEL ALGORITMO

A continuación, se introduce el código del algoritmo desarrollado con los comentarios pertinentes para hacer más sencilla su comprensión.

PARÁMETROS

```
//#define OFF_LINE
#define WIDTH_IMAGEN 640
#define HEIGHT_IMAGEN 480
////////////////////////////
#define ROWSperTHREAD 40   // the number of rows a thread will process
#define BLOCK_W 128         // the thread block width for the disparity
#define ANCHO_BLOQUE 64    //el valor del ancho del bloque para el cálculo de la
laplaciana de la gausiana
#define RADIUS_H 8          // Kernel Radius 5V & 5H = 11x11 kernel
#define RADIUS_V 8
#define MIN_SSD 5737500//((RADIUS_H*2+1)*(RADIUS_V*2+1)*255)   // The mimium
acceptable SSD value (por ejemplo el mayor posible en ventana de 7x7)
#define STEREO_MIND 0.0f   // The minimum d range to check
#define STEREO_MAXD 30.0f   // the maximum d range to check
#define SHARED_MEM_SIZE ((BLOCK_W + 2*RADIUS_H)*sizeof(int) ) // amount of shared
memory used
#define VALOR_INICIAL 0 //valor inicial para el array de salida, será el que se
usará si disp no ha sido menor que MIN_SSD
#define U_DISP_UMBRALIZADO
#define UMBRAL_AREA_ESTUDIO 1
#define B 0.119915
#define f 811.9104
#define cu (0.503221*640)
```

```
//////////////////////
```

DETECCIÓN OBSTÁCULOS

```
#include "parametros.h"

#ifndef RECTANGULO
	#define RECTANGULO

struct punto{
	int x;
	int y;
};

struct rectangulo{
	punto izq_sup;
	punto der_inf;
};

struct obstaculo{
	struct rectangulo;
	int elevado;
	int d_teorica;
	int d_real;
	int peaton;
	//CvPoint posicion_real;
	//int ancho;
	//int alto;
	//int arribaroi;
	//int abajoroi;

	//int obsel;
	//int obsnoel;
	float z;
	float x;
};
#endif
////////////////////////////////CLASE OBSTACULO/////////////////////////////////

#ifndef OBSTACULO
	#define OBSTACULO

class Obstaculo{
	//Variables miembro
	public:
		rectangulo rect;
		int area;
	//Constructor
		Obstaculo(int unArea = 0):  area(unArea)
		{
			rect.izq_sup.x = 0;
			rect.izq_sup.y = 0;
			rect.der_inf.x = 0;
			rect.der_inf.y = 0;
		}
		void SetArea(int unArea)
```

```
		{
			area = unArea;
		}
		void SetRectangulo(double un_x_min,double un_y_min,double
un_x_max,double un_y_max)
		{
			rect.izq_sup.x = un_x_min;
			rect.izq_sup.y = un_y_min;
			rect.der_inf.x = un_x_max;
			rect.der_inf.y = un_y_max;
		}
};
#endif
```

MAIN

```
//1. Definición de las librerías necesarias para la ejecución del programa.

#include <iostream>
#include <cstdlib>
#include <windows.h>

#include "StdAfx.h"

using namespace std;

#include <cv.h>
#include <highgui.h>
#include <cvaux.h>
#include <cxcore.h>

#include <stdio.h>
#include <math.h>
#include <time.h>

#include <cstdlib>
#include <iostream>
#include <vector>

#include <conio.h>


#include "parametros.h"
#include "deteccion_obstaculos.h"

using namespace cv;

////////////////////////////////////////////////////////////////////////////////
/////////


// 2. Definición de las variables fijas.

#define tam_x		640		// Ancho de las imagenes estéreo.
#define tam_y		480		// Alto de las imagenes estéreo.
```

```
#define tam_coste          30            // Disparidad máxima (=tamaño vector
coste).
#define rad_vent           8             // Radio de la ventana de búsqueda (lado
de la ventana
                                                    //de búsqueda=2*rad_vent+1)
#define umbral_u           48            // Valor umbral para la binarización de
la u_disparity.
#define umbral_obs         1             // Valor umbral para la binarización del
mapa obstáculo.
#define umbral_v           30            // Valor umbral para la umbralización de
la v_disparity.
#define umbral_blob        300L   // Área mínima de cada obstáculo.
#define umbralobs_max      8             // Valor umbral para la binarización del
mapa obstáculo.
#define umbralobs_min      3             // Valor umbral para la binarización del
mapa obstáculo.

////////////////////////////////////////////////////////////////////////////////
///////////

// 3. Declaración de las funciones.

void laplaciana(unsigned char*, int*, unsigned char*);
void disparidad(unsigned char*, unsigned char*, unsigned char*, unsigned char*);
void u_disparity(unsigned char*, unsigned char*);
void mapas(unsigned char*, unsigned char*, unsigned char*, unsigned char*);
void v_disparity(unsigned char*, unsigned char*);
void tipo_obstaculo(float b, float m, int j, obstaculo* obstaculos, CvPoint
arriba, CvPoint abajo, int width, int height, int widthR, int heightR, IplImage*
imgDisp);
void peatones(IplImage* imgObs, CvPoint arriba, CvPoint abajo, int i, obstaculo*
obstaculos, CvPoint posicion, int h, cv::HOGDescriptor hog, IplImage* imgD);



int main(int argc, char** argv)
{


// Comienza el proceso.
cout << "COMIENZA EL CALCULO" << endl;

// Inicialización de la variable que determinará la imagen de la secuencia a
procesar.
int i=1;

// Se introduce un bucle for para trabajar con todas las imágenes almacenadas en la
carpeta data.
while (true)
{


// Definición e inicialización de variables necesarias.

float m,b, Z, theta, a, X;
int k, h, w, z, afija;
unsigned int size = WIDTH_IMAGEN*HEIGHT_IMAGEN*sizeof(unsigned char);
unsigned char* suma =  (unsigned char*)malloc(size);
int s=0;
```

```
char imagen[50];
char imagender[25], imagenizq[25];

////////////////////////////////////////////////////////////////////////////////
///////////

// Inicialización de las matrices de datos de imágenes.

unsigned char* im_izda=  (unsigned char*)calloc((tam_x*tam_y),    sizeof(unsigned
char));
unsigned char* im_dcha=  (unsigned char*)calloc((tam_x*tam_y),    sizeof(unsigned
char));
unsigned char* lp_izda=  (unsigned char*)calloc((tam_x*tam_y),    sizeof(unsigned
char));
unsigned char* lp_dcha=  (unsigned char*)calloc((tam_x*tam_y),    sizeof(unsigned
char));
unsigned char* disp_izda=(unsigned char*)calloc((tam_x*tam_y),    sizeof(unsigned
char));
unsigned char* disp_dcha=(unsigned char*)calloc((tam_x*tam_y),    sizeof(unsigned
char));
unsigned char* u_disp=   (unsigned char*)calloc((tam_x*tam_coste),sizeof(unsigned
char));
unsigned char* mapa_obs =(unsigned char*)calloc((tam_x*tam_y),    sizeof(unsigned
char));
unsigned char* mapa_lib =(unsigned char*)calloc((tam_x*tam_y),    sizeof(unsigned
char));
unsigned char* v_disp =  (unsigned char*)calloc((tam_coste*tam_y),sizeof(unsigned
char));
unsigned char* mapaobs_bin1 =  (unsigned char*)calloc((tam_x*tam_y),sizeof(unsigned
char));
unsigned char* char_obs=  (unsigned char*)calloc((tam_x*tam_y),    sizeof(unsigned
char));

////////////////////////////////////////////////////////////////////////////////
///////////

// Carga de ficheros.

sprintf(imagender, "../dataOr/der%d.tiff ",i);
IplImage* imgD = cvLoadImage(imagender,0);
sprintf(imagenizq, "../dataOr/izq%d.tiff ",i);
IplImage* imgI = cvLoadImage(imagenizq,0);

// Creación de un bucle para cerrar el algoritmo cuando ya no existan más fotogramas
en la
//secuencia de imágenes o no se consiga cargar la imagen.

if ((!imgD)||(!imgI))
{
   printf("Ya no se dispone de mas imagenes para analizar");
   break;
}

////////////////////////////////////////////////////////////////////////////////
////////

// Cambio de las imágenes a un parámetro unsigned char para poder trabajar con ellas
en las
//funciones creadas.
```

```
 im_dcha=((uchar *)(imgI->imageData));
 im_izda=((uchar *)(imgD->imageData));


 // LAPLACIANA DE LA GAUSSIANA //

// Se define la máscara usada para el cálculo de la Laplaciana de la Gaussiana.

int lg [25] = {  0, -1, -4, -1,  0,
                       -1, -4,  2, -4, -1,
                       -4,  2, 32,  2, -4,
                       -1, -4,  2, -4, -1,
                        0, -1, -4, -1,  0};

// Se aplica la función laplaciana tanto a la imagen izquierda como a la derecha,
grabándolas,
//respectivamente, en las matrices de datos lp_izda y lp_dcha.

laplaciana(im_izda, lg, lp_izda);
laplaciana(im_dcha, lg, lp_dcha);

// Almacenamiento en variables de los parámetros de la imagen oringinal (proyección
derecha).

int width     = imgD->width;
int height    = imgD->height;
int depth     = imgD->depth;
int nchannels = imgD->nChannels;

///

IplImage *imglapI = cvCreateImageHeader(cvSize(width,height),depth,nchannels);
imglapI->imageData = (char *)lp_izda;
IplImage *imglapD = cvCreateImageHeader(cvSize(width,height),depth,nchannels);
imglapD->imageData = (char *)lp_dcha;

////////////////////////////////////////////////////////////////////////////////
///////////////

// A continuación, se obtiene el mapa de disparidad, la u_disparity, el mapa de
obstáculos,
//el mapa libre y la v_disparity para utilizarlas durante el resto de la ejecución
del algoritmo.

// DISPARIDAD //

// Se ejecuta la función disparidad creada para obtener la disparidad existente
entre las imagenes
//izquierda y derecha (una vez aplicadas sobre ellas la laplaciana de la gaussiana)
y las guardamos
//en las matrices de datos disp_izda y disp_dcha.

disparidad(lp_izda, lp_dcha, disp_izda, disp_dcha);


// CROSS-CHECKING //
// Se comparan pixel por pixel ambas imágenes para obtener la disparidad.
```

```
for (int k=0; k<(tam_x*tam_y); k++)
{
   if (abs(disp_dcha[k]-disp_izda[k])>2)
          {
                 disp_izda[k]=0;
          }
}

////

IplImage *imgDisp = cvCreateImageHeader(cvSize(width,height),depth,nchannels);
imgDisp->imageData = (char *)disp_izda;

////////////////////////////////////////////////////////////////////////////////
///////////////


// U_DISPARITY //     // El resultado se almacena en u_disp

u_disparity(disp_izda, u_disp);

// MAPA OBSTÁCULO Y MAPA LIBRE //   // Los resultados se almacenan en mapa_obs y
mapa_lib, respectivamente

mapas(disp_izda, u_disp, mapa_obs, mapa_lib);

////

IplImage *imgMapaObs = cvCreateImageHeader(cvSize(width,height),depth,nchannels);
imgMapaObs->imageData = (char *) mapa_obs;
IplImage *imgMapaLib = cvCreateImageHeader(cvSize(width,height),depth,nchannels);
imgMapaLib->imageData = (char *) mapa_lib;


// V_DISPARITY //       // El resultado se almacena en v_disp

v_disparity(mapa_lib, v_disp);

////

IplImage *imgvDisp = cvCreateImageHeader(cvSize(tam_coste,tam_y),depth,nchannels);
imgvDisp->imageData = (char *) v_disp;

////////////////////////////////////////////////////////////////////////////////
///////////////////////////////////

// TRANSFORMADA DE HOUGH //


// Se define la matriz donde se van a volcar los datos en formato OpenCV.
char* aux_v =  (char*)calloc(((tam_coste+2)*tam_y),sizeof(char));

// El siguiente paso se realiza para realizar la conversión entre el formato de
imagen de las MIL
//(formato que ya no se utiliza, pero que se tuvieron en cuenta al crear las
funciones) y el de las OpenCV.

// for(y) y for(x) recorren todos los píxeles de la matriz origen (v_disp), cargando
los datos en el lugar
```

```
//indicado de la matriz destino (aux_v).

for(int y=0; y<tam_y; y++)
{
   for(int x=0; x<tam_coste; x++)
   {
          aux_v[imgvDisp->widthStep*y+x*imgvDisp->nChannels]=v_disp[y*tam_coste+x];

   }
}

// Se cargan los datos de la nueva matriz calculada.
imgvDisp->imageData = aux_v;

// Se define una estructura para poder almacenar datos genéricos de forma dinámica.
CvMemStorage* datos = cvCreateMemStorage(0);

// Se define una estructura para poder almacenar líneas de forma dinámica.
CvSeq* lineas = 0;

// Se calcula la transformada de Hough de la imagen v, almacenando los resultados en
lineas.
lineas = cvHoughLines2(imgvDisp,datos,CV_HOUGH_PROBABILISTIC,1,CV_PI/180,50,75,20);

// Extracción de la primera línea de todas las obtenidas en la transformada de
Hough.
CvPoint* linea = (CvPoint*)cvGetSeqElem(lineas,0);


////////////////////////////////////////////////////////////////////////////////
//////////////////////

// Conocidas las coordenadas de los puntos extremos de la línea que definen el road
profile (carretera)
//se pueden obtener tanto la pendiente (m) como el horizonte teórico (b).
// Estos dos datos se emplearán por ejemplo para obtener la disparidad teórica, ya
que: d_teorica= (y1-b)/m
// y1 la se obtendrá del rectángulo que engloba a cada obstáculo, pues se trata de
la coordenada y del
//punto inferior derecha que define dicho triángulo

m=(float)(((float)linea[1].y -(float)linea[0].y)/((float)linea[1].x -
(float)linea[0].x));
b= (float)((float)linea[0].y -m*(float)(linea[0].x));

// Se calcula el ángulo para calcular la distancia real existente entre la cámara y
un objeto determinado
theta = atan((b-cu)/f);




   //     DETECCIÓN DE OBSTÁCULOS     //
   // Umbralización para no tener en consideración aquellos elementos que se
   encuentren alejados del vehículo.

   for(int k=0; k<(tam_x*tam_y); k++)
```

```
{
	if (mapa_obs[k]<umbralobs_min)
	{

		mapa_obs[k]=0;
	}
	else
	{
		mapa_obs[k]=mapa_obs[k];
	}

}

// Ubralización para considerar sólo aquellos obstáculos con varias disparidades
y que no estén muy próximos
//a la cámara.

for(int k=0; k<(tam_x*tam_y); k++)
{
	if (mapa_obs[k]>umbralobs_max)
	{

		mapaobs_bin1[k]=0;
	}
	else
	{
		mapaobs_bin1[k]=mapa_obs[k];
	}

}

// imgbin1 es donde se va a guardar la imagen del mapa de obstáculos sin los
obstáculos más próximos
//a la cámara.

IplImage *imgbin1 = cvCreateImageHeader(cvSize(width,height),depth,nchannels);
imgbin1->imageData = (char *) mapaobs_bin1;

// Imagen que va a contener los bordes obtenidos por el detector de bordes de
Canny de la imagen imgbin1.
IplImage* imgEdge = cvCreateImage( cvSize(width,height), 8, 1 );

// Se ejecuta la función cvcanny para la detección de bordes.
cvSmooth( imgbin1, imgEdge, CV_BLUR, 3, 3, 0, 0 );
cvNot( imgbin1, imgEdge );
cvCanny(imgbin1, imgEdge, 1, 1, 3);

// En imgbin2 se va a guardar la imagen binarizada del mapa de obstáculos de la
imagen original,
//para poder proceder a obtener la resta.
IplImage* imgbin2 = cvCreateImage( cvSize(width,height), 8, 1 );
cvCopy(imgMapaObs,imgbin2);

// La binarización se realiza empleando la función cvThreshold.
cvThreshold( imgbin2,imgbin2, 1, 255, CV_THRESH_BINARY );

// Creación de la imagen resta que será resultado de restar el mapa de obstáculos
binarizado y la imagen
//con los contornos de aquellos obstáculos que se quieren dividir.
```

```
IplImage* imgResta = cvCreateImage( cvSize(width,height), 8, 1 );

// Se restan ambas imágenes.
cvSub(imgbin2,imgEdge,imgResta,0);


////////////////////////////////  OBTENCIÓN DE LOS CONTORNOS
////////////////////////////////////

// Declaración de variables.

CvSeq* contours = 0;
CvMemStorage* storage = cvCreateMemStorage(0);
int contour_num;
CvBox2D box;
double area;

CvRect r;
int j=0;
CvPoint arriba;
CvPoint abajo;

int widthR=0;
int heightR=0;
int obsel=0,obsnoel=0;
int elevado=0;

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////

// Función para detectar los contornos de los obstáculos.
contour_num=cvFindContours( imgResta, storage, &contours,
sizeof(CvContour),CV_RETR_LIST, CV_CHAIN_APPROX_SIMPLE, cvPoint(0,0) );

// Creación de una imagen a color para representar los rectángulos de colores
según los obstáculos estén
//elevados o no.
IplImage* imgObs = cvCreateImage(cvSize(width,height), 8, 3);
cvCvtColor(imgD, imgObs, CV_GRAY2BGR);

// Bucle para trabajar con cada uno de los obstáculos.
for( ; contours != 0; contours = contours->h_next )
    {

// Se va a obtener gran cantidad de elementos por lo que se descartarán áquellos
que tengan un tamaño suficiente como para no ser considerados un peligro para el
vehículo.

    // Obtención de las áreas de cada uno de los contornos.
       area=cvContourArea(contours,CV_WHOLE_SEQ );

       //// Llamamiento a la estructura obstaculo.
       struct obstaculo* obstaculos= (struct obstaculo*)malloc(j*sizeof(struct
       obstaculo));

       // Se descartan aquellas áreas de menor tamaño, menores a un umbral
       definido (umbral_blob).
       if (area>umbral_blob)
       {
```

```
 // SE DIBUJAN LOS RECTÁNGULOS

 // Se obtiene un rectangulo que contiene a todos los puntos del contorno.

 r = cvBoundingRect( contours, 0 );
 CvPoint arriba = cvPoint(r.x, r.y);
 CvPoint abajo = cvPoint((r.x + r.width), (r.y + r.height));

// Se calcula la posición de la ROI en pixeles.
CvPoint posicion = cvPoint ((abajo.x-((abajo.x - arriba.x)/2)),abajo.y);


///  DISTINCIÓN ENTRE OBSTÁCULOS ELEVADOS Y OBSTÁCULOS NO ELEVADOS  ///

// Distinguimos los tipos de obstáculos (entre elevados y no elevados).
Llamamos a la función creada
tipo_obstaculo(b, m,j, obstaculos+j, arriba, abajo, width, height, widthR,
heightR, imgDisp);


// Se pintan los rectángulos para distinguir según estén elevados o no en
imgObs
       //Arriba y abajo son las coordenadas de los rectángulos que
       delimitan las ROIs obtenidas en la función
       //tipo_obstáculo. Estos rectángulos varían su color según sean
       elevados o no elevados (elevados verdes
       //y no elevados rojos)

       if(obstaculos[j].elevado==0)  //Obstáculos no elevados (rojo)
       {

              //Cálculo de la coordenada Z en el mundo real (distancia que
              hay desde la cámara hasta el obstáculo).
              Z = f*B*(m/(posicion.y-b))* cos (theta);
              // en pixeles...
              z = Z/0.0428;

              // Cálculo de la altura en pixeles (h) que a una distancia Z
              de la cámara equivale a 2m (altura del rectángulo
              //que delimita a la ROI)
              h = (2.2*f)/Z;

              // Cálculo de la coordenada x en el mundo real del obstáculo.
              X = ((m*B*(posicion.x-cu))/((posicion.y)-b)) * cos (theta);
              // en pixeles...
                     w = X / 0.0428;

                     // Cálculo de la anchura en metros (a).
                     a = Z*(abajo.x-arriba.x)/f;

                     // Cálculo de una anchura fija (afija) en píxeles
                     afija = (0.6*f)/Z;

              // Se descartan aquellos elementos cuya anchura en metros sea
              inferior a 10 cm.
              if (a>0.1)
              {
```

```
                        // Declaración del HOG y del entrenador SVM.
                        cv::HOGDescriptor hog;

      hog.setSVMDetector(cv::HOGDescriptor::getDefaultPeopleDetector());


                        // Ejecución de la función peatones que determinará
                        cuales de los obstáculos NO elevados son PEATONES.


peatones(imgObs,arriba,abajo,i,obstaculos+j,posicion,h, hog, imgD);


                         if (obstaculos[j].peaton==1)
                               {

                               //Modificación de las coordenadas que
                               delimitarán a los peatones para que queden
                               representados
                               //por un rectángulo de una altura de 1.7 m en
                               el mundo real.

                                     h = (1.7*f)/Z;
                                     arriba.y = abajo.y - h;
                                     arriba.x = abajo.x - afija;


                                      // Identificación de los peatones mediante un
                                     rectángulo en verde.
                        cvRectangle(imgObs, arriba, abajo, CV_RGB(0, 255, 0),1);


                                     }
                         }
                        }
                        else //Obstáculos elevados (azul)
                        {
                         cvRectangle(imgObs, arriba, abajo, CV_RGB(0, 0, 255),1);

                        }

                  // Contador para saber cuántos obstáculos se obtienen en cada
                  frame.
                  j++;

                  }

    }

// Se muestra la imagen original con los correspondientes rectángulos.

cvNamedWindow("Imagen Obstaculos",CV_WINDOW_AUTOSIZE);
cvShowImage("Imagen Obstaculos",imgObs);
cvWaitKey(10);

// Se libera el recurso de memoria donde están las imágenes almacenadas.

cvReleaseImage(&imgD);
cvReleaseImage(&imglapD);
```

```
    cvReleaseImage(&imgI);
    cvReleaseImage(&imglapI);
    cvReleaseImage(&imgDisp);
    cvReleaseImage(&imgMapaObs);
    cvReleaseImage(&imgMapaLib);
    cvReleaseImage(&imgvDisp);
    cvReleaseMemStorage( &storage );
    cvReleaseImage(&imgObs);
    cvReleaseImage(&imgbin1);
    cvReleaseImage(&imgbin2);
    cvReleaseImage(&imgEdge);
    cvReleaseImage(&imgResta);


    // Se incrementa el valor de i para pasar a la siguiente imagen de la secuencia.
           i++;

    } // Cierre del bucle que maneja la secuencia de imágenes


    return 0;
}
```

FUNCIÓN *tipo obstáculo*

```
    void tipo_obstaculo(float b, float m, int j, obstaculo* obstaculos, CvPoint
    arriba, CvPoint abajo, int width, int height, int widthR, int heightR, IplImage*
    imgDisp)
    {

           // DISPARIDAD TEÓRICA

           // Cálculo de la disparidad teórica.
           obstaculos->d_teorica=((abajo.y-b)/(m));

           // DISPARIDAD REAL

           // Para  obtener la disparidad real de cada uno de los obstáculos
    obtenidos se van a guardar las ROIs en imágenes.

           widthR=abajo.x-arriba.x;
           heightR=abajo.y-arriba.y;

           ////////////////////////////////////////////////////////////////////////
    ////////////////////////////////////////////////
           //  Regiones de Interés (para calcular la disparidad real)


           // Se crea una imagen para almacenar las ROIs.
           IplImage* imgROI = cvCreateImage(cvSize(width,height), 8, 1);

           cvZero( imgROI );
           cvCopy(imgDisp,imgROI);
```

```
	// Se obtienen las regiones de interés de la imagen según las coordenadas
de los obstáculos obtenidos.
	cvSetImageROI( imgROI, cvRect( arriba.x, arriba.y, widthR, heightR ));

	////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////

	// Una vez se tienen las ROIs se calculan sus histogramas para conocer el
valor máximo de éstos.

	// Obtención del histograma para la disparidad real.

	int hist_size = 256;
	float range_0[]={0,255};
	float* ranges[] = { range_0 };
	float max_value=0;
	int maxIdx=0;

	// Definición del histograma mediante el tipo de datos correspondiente.
	CvHistogram *histograma;

	// Creación de un histograma (por defecto un array).
	histograma = cvCreateHist(1,&hist_size, CV_HIST_ARRAY, ranges, 1);

	// Se calcula el histograma de cada una de las ROIs.
	cvCalcHist( &imgROI, histograma, 0, NULL );

	// Obtención del valor máximo del histograma y el índice de dicha celda.
	cvGetMinMaxHistValue( histograma, 0, &max_value, 0, &maxIdx );

	// El valor de disparidad real será el índice de la celda donde se
	//encuentra el máximo del histograma.
	obstaculos->d_real=maxIdx;

	// Clasificación entre obstáculos elevados y no elevados.

	if (abs(obstaculos->d_teorica-obstaculos->d_real)>2)
		{
			obstaculos->elevado=1; //obstáculo elevado
		}

	else
		{
			obstaculos->elevado=0; //obstáculo sobre el suelo
		}
	// Se hacen los resets necesarios.
	cvResetImageROI(imgROI);
	cvReleaseHist(&histograma);
}
```

FUNCIÓN *peatones*

```
void peatones(IplImage* imgObs, CvPoint arriba, CvPoint abajo, int i, obstaculo*
	obstaculos, CvPoint posicion, int h, cv::HOGDescriptor hog, IplImage* imgD)
{
	// Definición de las variables locales necesarias.

	cv::Mat img;
	CvRect roi;
	int widthR2=0;
	int heightR2=0;
	CvPoint arribaroi;
	CvPoint abajoroi;
	char imagender[50];


	// Declaración del vector found empleada en la función HOG.
	cv::vector<cv::Rect> found;

	// Carga de la imagen original (imgD), para almacenarla con un tipo Mat para
	obtener de ahí las rois
	//aumentadas y poder ser procesadas por el HOG.
	cv::Mat imgDMat(imgD);


	// Incremento del tamaño de los rectángulos que abarcan las regiones de interés
	para que los obstáculos
	//que abarcan puedan ser mejor procesados por el HOG.

	arribaroi.y = abajo.y - h;
	abajoroi.y = abajo.y + 25;

	arribaroi.x = arriba.x - 45;
	abajoroi.x = abajo.x + 40;


	// Estos bucles son necesarios debido a los problemas que aparecen cuando alguno
	de los obstáculos
	//se encuentra muy próximo a alguno de los extremos de la imagen y a la hora de
	aumentar el recuadro
	//se exceden los límites de la imagen original.

		if (arribaroi.x>640)
	{
		arribaroi.x=640;
	}
		if (arribaroi.x<0)
	{
		arribaroi.x=0;
	}
		if (arribaroi.y>480)
	{
		arribaroi.y=480;
	}
		if(arribaroi.y<0)
	{
		arribaroi.y=0;
```

```
    }
            if (abajoroi.x>640)
    {
            abajoroi.x=640;
    }
            if (abajoroi.x<0)
    {
            abajoroi.x=0;
    }
            if (abajoroi.y>480)
    {
            abajoroi.y=480;
    }
            if (abajoroi.y<0)
    {
            abajoroi.y=0;
    }


    // Obtención del ancho y del alto de estas nuevas regiones que abarcan al
    obstáculo.

    widthR2=abajoroi.x-arribaroi.x;
    heightR2=abajoroi.y-arribaroi.y;

    // Para resolver el problema de que falle en determinados frames (porque la
    altura es demasiado pequeña
    //para el HOG).

    if (heightR2<100)
    {
            heightR2=100;
    }

    // Se pinta el rectángulo que englobará aquellas regiones clasificadas como no
    elevadas.
    cvRectangle(imgObs, arribaroi, abajoroi, CV_RGB(255, 0, 0),1);

    // Almacenamiento delas coordenadas en un cvRect para posteriormente guardar cada
    uno de los ROIs como
    //imagen tipo Mat.
    roi= cvRect( arribaroi.x, arribaroi.y, widthR2, heightR2);


    // Almacenamiento de las ROIs como tipo Mat, obteniéndolas de la imagen original
    también de tipo Mat,
    //gracias a la función cv::Mat Mat(img, cvRect)
    img= imgDMat(roi);

    // Ejecución del HOG.
  hog.detectMultiScale(img, found, 0, cv::Size(8,8), cv::Size(24,16), 1.05, 2);

  for( int c = 0; c < (int)found.size(); c++ )
  {
          obstaculos->peaton=1;
  }

}
```

BIBLIOGRAFÍA

[1] DGT, "Anuario Estadístico de Accidentes 2010". <http://www.dgt.es/ was6/portal/contenidos/es/seguridad_vial/estadistica/ publicaciones /anuario_estadistico/anuario_estadistico014.pdf> [Consulta: Septiembre 2012].

[2] CARSTEN, O. M. J.; NILSSON, L. Safety assessment of driver assistance systems. *European Journal of Transport and Infrastructure Research*, 2001, vol. 1, no 3, p. 225-243.

[3] BRADSKI, Gary; KAEHLER, Adrian. *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Incorporated, 2008.

[4] GIETELINK, Olaf, et al. Development of advanced driver assistance systems with vehicle hardware-in-the-loop simulations. *Vehicle System Dynamics*, 2006, vol. 44, no 7, p. 569-590.

[5] MILANÉS MONTERO, Vicente, et al. Sistema de posicionamiento para vehículos autónomos. *Revista Iberoamericana de Automatica e Informatica Industrial. RIAI*, 2008, vol. 5, no 4, p. 36-41.

[6] MERCEDES-BENZ. Sistemas de Asistencia a la Conducción.<http://www.mercedes-benz.es/content/spain/mpc/mpc_spain_website/es/home_mpc/passengercars/home/new_cars/models/c-class/_w204/facts_/comfort/assistancesystems.html> [Consulta: Septiembre 2012].

[7] RENAULT. Sistema de Asistencia a la Conducción.< http://www.renault.es/gama-renault/renault-vehiculos-turismos/gama-koleos/koleos-old/acabados-y-equipamientos/lista-equipamientos-opciones/index.jsp>[Consulta: Septiembre 2012].

[8] MUÑOZ, Javier. <http://www.carnetonline.es/2009/01/16/dispositivos-de-asistencia-al-estacionamiento/>[Consulta: Octubre 2012].

[9] MARKO, Paul D. *System And Method For Improved Traffic Flow Reporting Using Satellite Digital Audio Radio Service (SDARS) And Vehicle Communications, Navigation And Tracking System*. U.S. Patent No 20,120,316,765, 13 Dic. 2012.

[10] ELMUNDO.com. Sistema de Visión Nocturna. <http://www.elmundo.es/elmundomotor/2005/07/13/tecnica/1121243673.html>[Consulta: Octubre 2012].

[11] EURO NCAP. Sistemas de mejora de la visibilidad. <http://es.euroncap.com/es/rewards/technologies/vision.aspx>[Consulta: Octubre 2012].

[12] CISNEROS, Oscar. Los sistemas d Adaptación Inteligente de la Velocidad. <http://www.centrozaragoza.com:8080/web/sala_prensa/revista_tecnica/hemeroteca/articulos/R38_A7.pdf>[Consulta: Octubre 2012].

[13] EUROPEAN COMMISSION. Control de Velocidad de Crucero Adaptativo. <http://ec.europa.eu/information_society/activities/intelligentcar/smart_cars_technologies/driverassit/adapcruisesontrol/index_es.htm>[Consulta: Octubre 2012].

[14] TECMOVIA. La asistencia y mantenimiento de carril podría evitar hasta un 18% de los accidentes.<http://www.tecmovia.com/2013/02/26/la-asistencia-y-mantenimiento-de-carril-podria-evitar-hasta-un-18-de-los-accidentes/>[Consulta: Octubre 2012].

[15] VOLKSWAGEN. Lane Assist.<http://www.volkswagen.es/es/innovación/>[Consulta: Octubre 2012].

[16] VOLVO. Tecnologías VOLVO <http://www.volvocars.com/es/explore/pages/volvo-technologies.aspx> [Consulta: Octubre 2012].

[17] WIKIPEDIA. Reconocimiento de Señales de Tráfico. <http://es.wikipedia.org/wiki/Reconocimiento_de_se%C3%B1ales_de_tr%C3%A1fico>[Consulta: Octubre 2012].

[18] TOYOTA. Pre-crash Safety System (Millimeter wave radar type). <http://www.toyota-global.com/innovation/safety_technology/safety_technology/pre-crash_safety/>[Consulta: Octubre 2012].

[19] REPSOL. Mecánica. El coche. PBDS, El capó que protege a los peatones. <hhttp://.repsol.es/se/motor/elcoche/ficha.aspx?idReportaje=1296>[Consulta: Octubre 2012]

[20] BOSCH. Protección de peatones.<http://rb-kwin.bosch.com/es/es/safety_comfort/occupant_protection/pedestrianprotection.html>[Consulta: Octubre 2012].

[21] FITSA, Fundación Instituto Tecnológico para la Seguridad del Automóvil. Tecnologías vehiculares para la mejora de la protección de peatones y ciclistas. <https://espacioseguro.com/fundacionfitsa0/admin/_fitsa/archivos/publicaciones/0000018/12-Peatones.pdf>[Consulta: Octubre 2012]

[22] BMW. BMW ConnectedDrive. Seguridad. Sistemas de asistencia a la visibilidad < http://www.bmw.es/es/es/insights/technology/connecteddrive/2010/safety/vision_assistance/night_vision.html >[Consulta: Octubre 2012].

[23] FORD. Tecnología. Detector de ángulo muerto (BLIS). <http://www.ford.es/AcercadeFord/Tecnologia>[Consulta: Octubre 2012].

[24] WIKIPEDIA. Google driverless car. <http://en.wikipedia.org/wiki/Google_driverless_car>[Consulta: Noviembre 2012].

[25] IHS GlobalSpec CR4. The Civil Engineering Section.<http://cr4.globalspec.com/blog/15/Building-Design-Blog>[Consulta: Noviembre 2012].

[26] TENDENCIAS TECNOLÓGICAS. Crean un coche autónomo que podría cambiar las ciudades.< http://www.tendencias21.net/Crean-un-coche-autonomo-que-podria-cambiar-las-ciudades_a4987.html>[Consulta: Noviembre 2012].

[27] DARPA, Defense Advanced Research Projects Agency.< http://www.darpa.mil/> [Consulta: Noviembre 2012].

[28] LSI, Laboratorio de sistemas inteligentes <http://www.uc3m.es/portal/page/portal/investigacion/nuestros_investigadores/grupos_investigacion/laboratorio_sistemas_inteligentes>[Consulta: Febrero 2013].

[29] MUSLEH, B.; DE LA ESCALERA, A.; ARMINGOL, J. M. 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*, 2012, vol. 9, no 4, p. 462-473.

[30] Generación, Representación y Principios de Procesamiento Digital de Imágenes. <http://catarina.udlap.mx/u_dl_a/tales/documentos/lem/jimenez_c_e/capitulo2.pdf>[Consulta: Febrero 2013].

[31] POINT GREY, Bumblebee 2 <http://ww2.ptgrey.com/stereo-vision/bumblebee-2>[Consulta: Febrero 2013].

[32] VIOLA, Paul; JONES, Michael. Fast and robust classification using asymmetric adaboost and a detector cascade. *Advances in Neural Information Processing System*, 2001, vol. 14.

[33] CHAPELLE, Olivier; HAFFNER, Patrick; VAPNIK, Vladimir N. Support vector machines for histogram-based image classification. *Neural Networks, IEEE Transactions on*, 1999, vol. 10, no 5, p. 1055-1064.

[34] LOWE, David G. Object recognition from local scale-invariant features. En*Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*. Ieee, 1999. p. 1150-1157.

[35] LABAYRADE, Raphael; AUBERT, Didier; TAREL, J.-P. Real time obstacle detection in stereovision on non flat road geometry through. En *Intelligent Vehicle Symposium, 2002. IEEE*. IEEE, 2002. p. 646-651.

[36] HU, Zhencheng; LAMOSA, Francisco; UCHIMURA, Keiichi. A complete uv-disparity study for stereovision based 3d driving environment analysis. En *3-D Digital Imaging and Modeling, 2005. 3DIM 2005. Fifth International Conference on*. IEEE, 2005. p. 204-211.

[37] DALAL, Navneet; TRIGGS, Bill. Histograms of oriented gradients for human detection. En *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. IEEE, 2005. p. 886-893.

[38] JIMÉNEZ MONJE, Violeta. Detección y localización de obstáculos en entornos urbanos mediante visión estéreo. 2011.

[39] OLMEDA, Daniel; DE LA ESCALERA, Arturo; ARMINGOL, Jose Maria. Contrast invariant features for human detection in far infrared images. En*Intelligent Vehicles Symposium (IV), 2012 IEEE*. IEEE, 2012. p. 117-122.