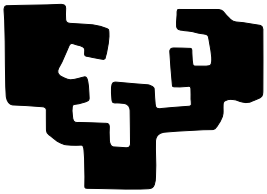
eman ta zabal zazu



Universidad del País Vasco
Euskal Herriko Unibertsitatea

DEPARTAMENTO DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES

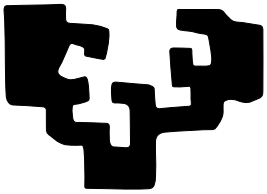
VISIÓN ARTIFICIAL APLICADA A LOS SISTEMAS DE TRANSPORTE INTELIGENTES: APLICACIONES PRÁCTICAS

por:
Andoni Cortés Vidal

Supervisado por:
Dr. Marcos Nieto Doncel
&
Prof. Clemente Rodríguez Lafuente

Donostia – San Sebastian, Viernes 30 de Diciembre, 2020

eman ta zabal zazu



Universidad del País Vasco
Euskal Herriko Unibertsitatea

DEPARTAMENTO DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES

VISIÓN ARTIFICIAL APLICADA A LOS SISTEMAS DE TRANSPORTE INTELIGENTES: APLICACIONES PRÁCTICAS

por:
Andoni Cortés Vidal

Supervisado por:
Dr. Marcos Nieto Doncel
&
Prof. Clemente Rodríguez Lafuente

Donostia – San Sebastian, Viernes 30 de Diciembre, 2020

A mis dos personas preferidas,
y a todas las personitas que me rodean.
Por todos los caminos que nos quedan por recorrer.

a mi tovarich... a mi hermano...

"Un lugar es tan especial como las personas que te acompañan"

Resumen

La visión artificial se ha utilizado a lo largo de las últimas décadas para resolver problemas en ámbitos muy diferentes del mundo real. Esto es así porque permite la adquisición no intrusiva de información de diferentes escenarios, imitando precisamente uno de los principales órganos sensoriales del ser humano. Estos sistemas adquieren gran cantidad de información, tanto específica como general, del entorno físico y real del problema que se pretende resolver. Por otro lado, los problemas en el contexto de los sistemas de transporte han ido adquiriendo importancia desde la aparición de los vehículos a motor, la creación de grandes infraestructuras viarias que son transitadas por millones de usuarios anualmente y el aumento de las velocidades de circulación en dichos canales de transporte. Esto nos llevará inevitablemente a la búsqueda de sistemas más seguros adaptados a los requisitos de los futuros esquemas urbanos e interurbanos que irán adecuando y definiendo las diferentes estructuras de comunicación viaria. Esta tesis se focaliza en diferentes investigaciones y desarrollos llevados a cabo en el ámbito de los sistemas de transporte inteligente a diferentes niveles. Se han enfrentado problemas como la segmentación de vehículos, la detección y reconocimiento de elementos intrínsecamente relacionados con la infraestructura vial contemplando la posibilidad de extender esta detección a elementos ajenos a la infraestructura que puedieran generar situaciones de peligro si irrumpesen de manera fortuita en las vías de transporte y se han realizado también, estudios teóricos sobre temas concretos que pueden actuar de guía para investigaciones realizadas en las líneas analizadas. En lo referente a los campos de aplicación, se proponen soluciones en diferentes áreas relacionadas con los sistemas de transporte inteligente. En concreto,

soluciones para el peaje en sombra, donde se perseguían los objetivos de detección, clasificación y estimación de velocidad de los vehículos que transitaban una vía, y soluciones para sistemas de asistencia avanzada a la conducción como el reconocimiento de señales de tráfico para ajustar la velocidad del vehículo e informar al conductor.

Agradecimientos

¿Por donde empezar? Hay muchas personas que han hecho posible esta tesis, que me han animado, me han dado su apoyo y soporte en muchos aspectos. Lejos quedan ya los primeros pasos de esta larga travesía. Mucho debería agradecer, por tanto, y a muchas personas diferentes. Empezaré, no obstante, por agredecer la confianza y apoyo de mis directores de tesis a quienes aprecio sobremanera. Tanto a Clemente que se embarcó conmigo en esta empresa sin pensárselo ni un momento y cuyo apoyo he notado en cada palabra que he podido escribir, como a Marcos cuyos consejos han iluminado el camino en muchos de los serpenteantes vericuetos que hemos transitado y con cuyas conversaciones he crecido y abarcado tanto. Alguien dijo una vez "Si veo más lejos es porque he subido a hombros de gigantes", ellos sin duda son los gigantes que me han sustentado todo el camino. Gracias a Julián Florez, una fuerza de la naturaleza por su optimismo desbordante y su tracción abrumadora, la persona que alojó en mi cabeza la idea inicial de retomar la tesis que aparqué tiempo atrás. Gracias también a mi tutor de tesis por su diligencia y apoyo en las tareas más administrativas, arduas sin duda. Continuaré agradeciendo a Oihana, inquebrantable y cercana, toda la facilidad que me ha proporcionado para gestionar los tiempos y objetivos y la confianza que siempre ha demostrado tener en mí. A mis compañeros de Vicomtech, un grupo de ese tipo de personas que convierten la cercanía en amistad, gente buena y muy grande. También me gustaría dejar un hueco para mis amigos, para los de siempre, para los nuevos. Amigos que me han demostrado su empatía en este proceso, moldeando los momentos a mi situación. Sería una lista interminable mencionarlos a todos, pero he notado su apoyo en cada conversación, en cada palabra.

A mi familia, en general, que me acompaña y conforta y en particular a mi hermano, una de las personas que me definen, gracias a las horas de tertulia incombustible que me regala y a quien aún en la distancia tan presente tengo. Y me gustaría acabar con la persona que me ofrece un apoyo incondicional independientemente de las circunstancias que me rodeen, mi querida Amaia, cuantos ratos te he robado para poder centrar mi atención en la ciencia, cuanta frustración y abatimiento ahogados en tu sonrisa.

Gracias

Andoni Cortés Vidal

enero 2021

Índice general

Índice de figuras

Índice de tablas

La gente viaja a lugares remotos para mirar, con fascinación, el tipo de personas que ignoran en casa

Dagobert D. Runes

CAPÍTULO

1

Introducción

1.1 Contexto

En una época en la que la inquietud y el tedio nos invitan a movernos de un lugar para otro, y en la que el transporte y la movilidad han impregnado tan profundamente la forma en la que vivimos, es comprensible la importancia, cada vez mayor, que adquiere la aplicación de la tecnología a estos ámbitos. Esta proliferación de sistemas tecnológicos en el mundo del transporte persigue diferentes objetivos, entre ellos, aumentar la seguridad y la comodidad en los desplazamientos que se producen en las diferentes redes de transporte, reducir los tiempos y los costes de los desplazamientos, y reforzar de manera efectiva las leyes relacionadas con el tráfico.

Y es que las malas prácticas y los errores en los procesos vinculados con el transporte, tanto en los desplazamientos como en la utilización de sus diferentes sistemas de regulación, se traducen en pérdidas vitales y/o económicas. Todo esto ha llevado a las sociedades modernas a invertir gran cantidad de recursos en la mejora de las vías, de los sistemas de transporte y de los elementos y sistemas aplicados a sus infraestructuras.

Y es en este contexto, en el que se desarrollan los dos trabajos a partir de los cuales ha nacido y se ha fraguado este trabajo de tesis, uno de ellos vinculado a los sistemas de gestión avanzada del tráfico (ATMS) que se introducirá posteriormente, y el otro más

ligado a los sistemas de ayuda a la conducción avanzada (ADAS) de los que también se hablará más adelante.

El principal desafío a nivel científico dentro de la aplicación de la tecnología al mundo del transporte está en incorporar tecnologías punteras y desarrollos innovadores, probados principalmente en entornos académicos, dentro de soluciones reales que puedan interaccionar con los agentes presentes en los sistemas de transportes: vehículos, señalética, infraestructuras, etc. Y es en este punto, donde encontramos al primero de los protagonistas nucleares de este trabajo: la visión artificial. Un campo científico que día a día se va volviendo más maduro y fiable y que ha venido siendo durante estas últimas décadas un candidato perfecto que incorpora a dichas infraestructuras dado su carácter generalista y su relativamente ajustado coste económico, estando presente a día de hoy en la mayor parte de ellas, si bien en multitud de instalaciones, meramente, a modo de sistema de supervisión. Esta incorporación se realiza de muy diferentes formas y niveles y ha llevado a la visión artificial a convertirse en un elemento casi indispensable dentro de la evolución que están sufriendo las vías de transporte.

Sin embargo, esta disciplina no actúa sola y de hecho la mayor parte del éxito alcanzado, y de la madurez y fiabilidad que a día de hoy ofrece, es gracias a su combinación con otras áreas de la inteligencia artificial (véase la figura 1.1). Y así, como uno de los compañeros más íntimos de la mayoría de aplicaciones de visión artificial aplicada presentes en el mercado, nos encontramos al campo del *machine learning*. Esta disciplina de la inteligencia artificial lleva en auge durante las últimas décadas con aplicaciones inverosímiles sobre infinidad de problemas del mundo cotidiano, dando soluciones sorprendentes y fiables.

La combinación de ambas disciplinas nos permite y nos permitirá en el futuro venidero, dar los pasos adecuados para llegar a un nuevo paradigma de utilización de las redes y sistemas de transporte, que se volverán más seguros y más confortables para los usuarios, donde los eventos inesperados se podrán prevenir con la suficiente antelación para que no supongan un trastorno significativo en los desplazamientos y la movilidad cotidianos.

El papel de la visión artificial o visión por computadora en la mayor parte de los desarrollos en los que se aplica consiste en la interpretación y análisis del entorno capturado por el sensor instalado en una cámara. Son muchas las tareas que se suelen

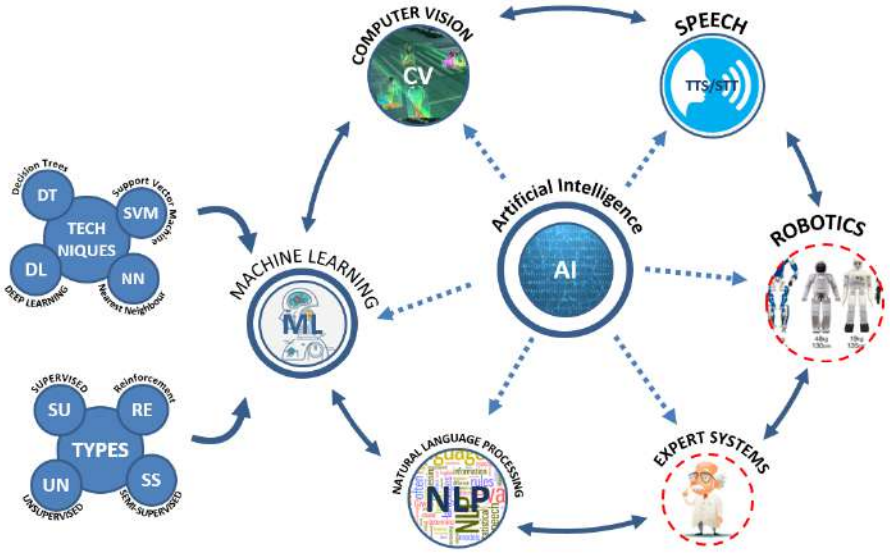


Figura 1.1: Subcampos dentro de la IA.

realizar en dichos análisis de imágenes y en diferentes dominios (espacial o temporal) pero, principalmente, se realizan análisis básicos que no ofrecen una interpretación semántica de la escena hasta que no se añaden técnicas de inteligencia artificial. Así, entre las diferentes tareas que se llevan a cabo, son muy comunes las de detección de elementos de interés, análisis del movimiento presente en la imagen, reconstrucciones tridimensionales, clasificación de elementos, seguimiento de elementos detectados, etc. Con esta amalgama de algoritmos y técnicas de visión artificial que además ofrecen la ventaja de utilizar muchas de las instalaciones de cámaras ya presentes para supervisión visual de un escenario, las posibilidades de mejora son amplias y aplicables a la mayor parte de escenarios relacionados con transporte. Y con su aplicación vamos acercandonos a lo que se han venido a llamar sistemas de transporte inteligente. Pero ¿qué entendemos por sistemas de transporte inteligente (ITS de ahora en adelante)?

Si tuviésemos que dar una definición general podríamos recurrir a la que se proporciona en la wikipedia:

> *"Los sistemas de transporte inteligente son soluciones tecnológicas diseñadas para mejorar la operación y seguridad del transporte, ubicándose dentro del ámbito del Internet de las cosas [Gershenfeld et al.04]."*

Las ITS tienen muchos ámbitos de aplicación y, aunque principalmente se aplican a transporte terrestre (por carretera o ferroviario), también pueden aplicarse a otros tipos de transporte como aéreo o marítimo (véase la figura 1.2).

Figura 1.2: Diferentes agentes dentro de los sistemas de transporte inteligentes (ITS).

Dentro de los sistemas de transporte inteligente podemos establecer diferentes categorías, entre ellas, los sistemas relacionados con la infraestructura de las redes viarias o **sistemas de gestión avanzada de tráfico** (en inglés Advanced Traffic Management Systems o ATMS) y los sistemas empotrados en el interior de los vehículos que transitan dichas infraestructuras, conocidos como **sistemas automáticos de asistencia a la conducción** (en inglés Advanced Driver Assistance Systems o ADAS). Tanto unos como otros realizarían funciones de advertencia, de prevención, de alerta, informativas, de gestión, de mantenimiento, de supervisión y de inspección de los diferentes elementos y agentes de las redes viarias.

A nivel funcional, entre los diferentes sistemas que nos encontramos para gestionar la infraestructura de la red viaria (véase figura 1.3), podemos citar: conteo y clasificación de vehículos, estimación de velocidad, control de intersecciones,

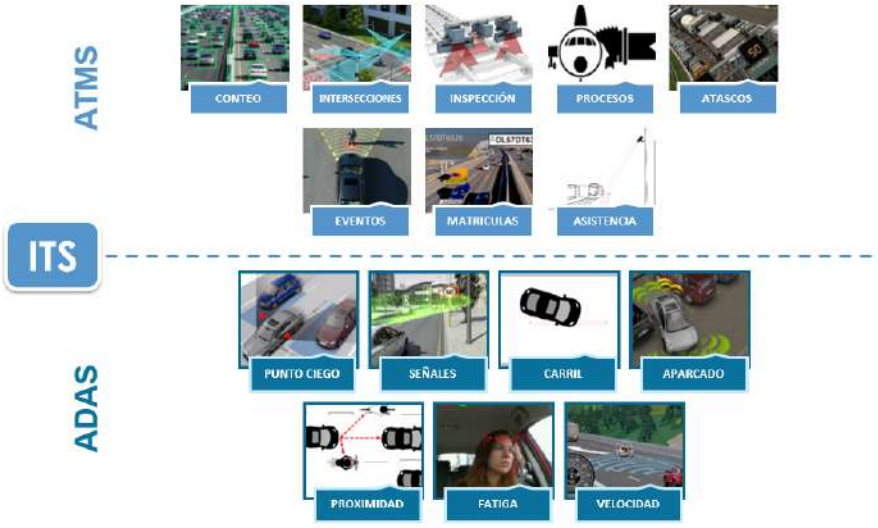


Figura 1.3: Sistemas de transporte inteligente : ADAS y ATMS.

inspección de elementos, gestión de procesos, gestión de atascos, detección de anomalías, asistencia automática, análisis del flujo de circulación.

Por otro lado, entre los sistemas enmarcados dentro de los ADAS, podemos mencionar: Análisis del punto ciego, Detección de señales, Salida de carril, Asistencia al aparcamiento, Sistema de prevención de colisiones, Detector de fatiga, Estimación de la ego-velocidad. Muchos de los cuales o ya se implementan o se comienzan a implementar en vehículos de transporte por los diferentes fabricantes de automóviles.

Particularmente, además, los ADAS presentan una línea de evolución cada vez más destacada en la comunidad científica actual: su aplicación en la creación de **vehículos autónomos**. Estos sistemas en lugar de realizar tareas meramente informativas o asistenciales irán un paso más allá y en última instancia, acabarán traduciendo la información obtenida del análisis de la escena a acciones concretas en el vehículo interaccionando con sus sistemas para modificar su comportamiento. A día de hoy existen diferentes niveles de automatización definidos por la SAE (Sociedad de Ingenieros de la Automoción o, en inglés, Society of Automotive Engineers) véase la figura 1.4. Estos niveles establecen el grado de desarrollo en el que se encuentra un vehículo en función del nivel de autonomía que presenta.

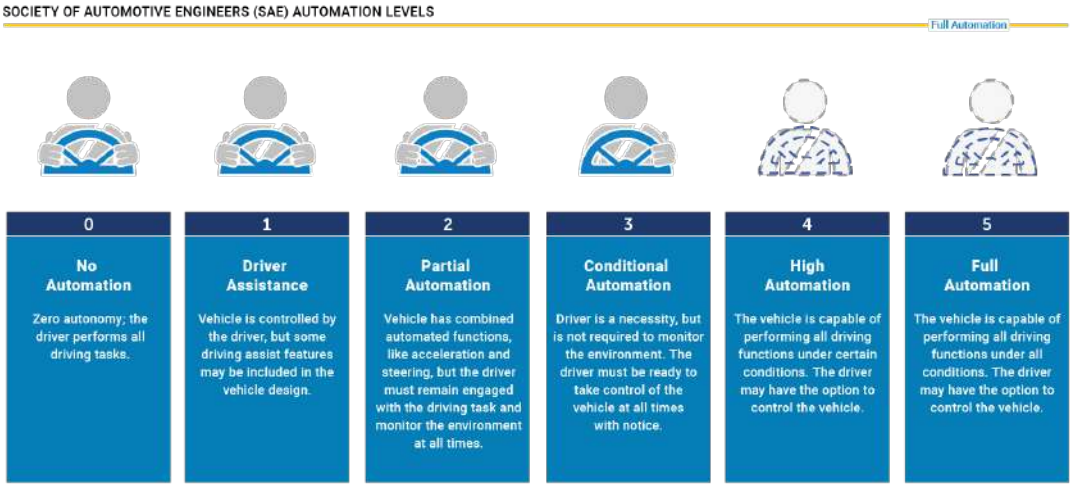


Figura 1.4: Niveles de automatización definidos por la Society of Automotive Engineers (SAE [1]).

Van desde la ausencia de automatización (nivel 0), en el que entrarían principalmente los vehículos hasta finales del siglo XX y en el que todas las decisiones se centran en un conductor humano, hasta una automatización total (nivel 5), en la que un sistema se encargaría de realizar las acciones sobre el vehículo en función del análisis de situación del entorno en el que se encuentra y en función del objetivo que se le haya definido, bajo cualquier condición climática posible. Todavía no existen vehículos comerciales que puedan ser considerados de nivel 4 o 5. Por ejemplo, el Tesla Autopilot, de Tesla Motors, se encontraría en el nivel 2, nivel en el que el vehículo puede realizar varias tareas autónomas a la vez pero en el que el conductor debe de estar en control durante el desplazamiento. El Audi A8 podría encajar dentro del nivel 3, en el cual el vehículo proporciona “autonomía condicional", esto es, únicamente es autónomo en ciertas condiciones.

Sin embargo, las aplicaciones de estas tecnologías en el mundo del transporte no se reducen únicamente a las enunciadas previamente. Este dueto superlativo que forman la visión artificial y el *machine learning* se emplea también en el ámbito del ***mobile mapping***, una serie de sistemas que se nutren de la información recolectada por un

[1] https://www.sae.org/standards/content/j3016_201806/

vehículo pertrechado con diferentes tipologías de sensores. Esta información se utiliza posteriormente para tareas de gestión de la infraestructura de la red de carreteras, comprobando que los elementos instalados continúan en su lugar sin presentar ningún tipo de deterioro o para realizar tareas de *road mapping*, esto es, obtener un mapa de una vía que estamos transitando con sus elementos estructurales posicionados geoespacialmente. También hay lugar para las aplicaciones de navegación de internet que utilizan la información de los sensores para generar una vista 360 de la carretera en cada punto.

Es en este marco de sistemas y desarrollos relacionados con las ITS en el que, como ya se mencionó, nace esta tesis, derivada de la investigación que he llevado a cabo en Vicomtech durante los últimos años y relacionada precisamente con proyectos en los que se ha introducido la visión artíficial y *machine learning* como repuesta a las necesidades que presentaba la sociedad en el campo de las ITS. En concreto, los proyectos que han motivado la realización de esta disertación han sido dos colaboraciones con diferentes consorcios de empresas, uno a nivel europeo y otro a nivel nacional. Uno de ellos, cuyo nombre ha ido evolucionando en el tiempo y que empezó conociéndose como iToll pasando por INTELVIA[2] para acabar convirtiéndose en EagleTD, surgió en 2008 con el título de "Desarrollo experimental de un Nuevo Sistema de Peaje Free Flow basado en la Gestión Integral, Inteligente e interoperable de la Información"(véase la figura 1.5).

Este proyecto se fraguó a partir de la colaboración y puesta en común de ideas de diferentes empresas privadas y centros tecnológicos, entre ellos: Ikusi, mondragón sistemas, Lks, NTS, til-its euskadi, Ikerlan y Vicomtech, con el apoyo de Kutxa e Infraestructuras viarias. Se presentaba como la irrupción de las empresas vascas en el mercado de las ITS y en concreto en el sector del peaje, por el objetivo que perseguía: el diseño e implementación de un sistema de peaje multicarril en *Free Flow*, basada en visión artificial e interoperable con los futuros sistemas de peaje basados en posicionamiento por satélite (Galileo). El objetivo, en relación a los usuarios, era claro: ahorrar a los conductores tiempo y momentos de frustración permitiéndoles atravesar las zonas de peaje sin detenerse ni reducir la velocidad como ocurre con alguna de las tecnologías utilizadas actualmente (ViaT). De cara a los constructores y entidades de

[2]https://intelvia.wordpress.com/alcance/

Figura 1.5: Descripción visual del proyecto iTOLL.

mantenimiento de vía, se proporcionaba la posibilidad de registrar los tipos de vehículos y el número que circulaban por la vía y sacar estadísticos descriptivos del tráfico al que se veía sometida para luego justificar presupuestos, inversiones y ayudas públicas, lo que se conoce como peaje en sombra.

Entre los objetivos tecnológicos que presentaba teníamos por un lado el desarrollo de nuevos algoritmos de detección y clasificación de vehículos con independencia de las condiciones meteorológicas presentes además del almacenamiento del número de matrícula, la detección del tipo de vehículo y la estimación del tonelaje mediante el sistema de visión. Por otro lado, existía el objetivo del desarrollo de receptores híbridos (GPS/Galileo/EGNOS) para mejorar la precisión del posicionamiento con los receptores existentes. En esta disertación se recogen las contribuciones realizadas al proyecto iToll en el ámbito de este proyecto de tesis, y que se presentan en el apartado 2.

El segundo proyecto, se gestó a nivel europeo en el marco H2020, véase la figura 1.6. El consorcio en torno a este proyecto implicó a diferentes entidades internacionales : Vicomtech (España),ERTICO (Belgica), Hoda Research Institute (Alemania), Intel Corporation (Bélgica), TeleConsult Austria GmbH (Austria), TomTom international (Holanda), Eindhoven University of Technology (Holanda), ACASA (España), IFSTTAR (Francia), IMI (BCN - España).

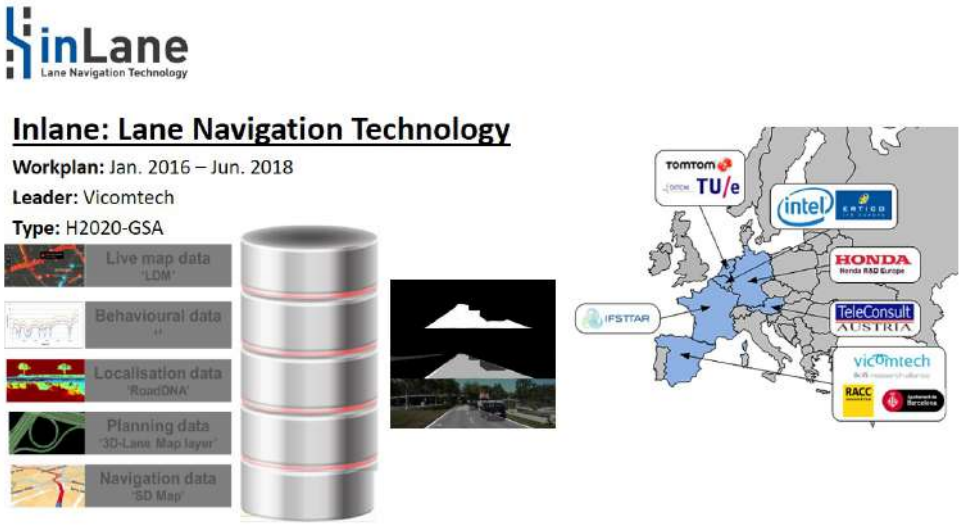


Figura 1.6: Descripción visual del proyecto Inlane.

InLane [3] se planteaba en el contexto del *lane navigation*, esto es, la geolocalización (coordenadas geográficas) del vehículo a nivel de carril, percibiendo la necesidad de realizar con más precisión dicha localización para futuras aplicaciones como la aplicación a sistemas ADAS, a servicios basados en localización hyper-específica y a coches autónomos, en las que se requeriría este aumento de precisión.

En InLane se proponía la utilización de un modelo de bajo coste, y por tanto accesible, que proporcionase a los conductores una nueva generación de sistemas de asistencia de navegación, que incluyera características tales como la precisión e integridad de los cálculos de posición gracias al uso de señales EGNSS, detección automática de elementos en la carretera usando técnicas de visión y *machine learning* avanzadas, desarrollo de estándares para codificar nuevas clases de contenido de datos de carretera, generación automática de datos de cartografía y localización con precisión alta e integridad, un nivel de mapa de información dinámico y local para asistir al conductor, el posicionamiento del vehículo en el carril, la posición relativa lateral del vehículo en la calzada situando al vehículo en relación a la calzada, un nivel de confianza en la localización que no se base únicamente en medidas GNSS sino en la

[3] https://inlane.eu/

combinación de todas las fuentes de posicionamiento y en los mapas aumentados y un nivel de fiabilidad en términos de carril.

Como puede verse, este proyecto era más ambicioso y perseguía un número mayor de objetivos que los que entraron dentro del marco del proyecto de esta tesis. Más específicamente, las contribuciones en las que se enfoca este trabajo se extienden al desarrollo del sistema de reconocimiento y localización de elementos en la carretera (TSR de aquí en adelante) que se acotó, en principio, a las señales de tráfico , entendiendo que esto era extensible a cualquier tipología de elementos adicionales.

1.2 Objetivos y Aportaciones

En el apartado anterior se ha esbozado una visión general de la motivación de este trabajo de tesis. En este apartado se describirán de manera sucinta los objetivos específicos que se perseguían a nivel técnico. En lo referente al desarrollo en el ámbito de los ATMs donde se buscaba la creación de una solución que permitiese el conteo y clasificación de vehículos en tiempo real utilizando reglas heurísticas y visión artificial, se persiguió el siguiente objetivo:

1. La elaboración de un algoritmo de sustracción de fondo capaz de adaptarse a las diferentes condiciones lumínicas y meteorológicas presentes en la escena y de lidiar con las sombras generadas por los vehículos en días soleados que generaban multitud de falsos positivos.

En lo referente a los sistemas ADAS, el objetivo era el desarrollo de un sistema de reconocimiento de elementos de la vía de transporte. Esto se particularizó en un sistema de reconocimiento de señales de tráfico de limitación de velocidad. Los retos a enfrentar también fueron de índole diversa:

1. Realización de una detección precisa y rápida de objetos de interés.

2. Reconocimiento fiable añadido a la capacidad de descartar detecciones incorrectas.

3. Elaboración de una herramienta de anotación versátil y que redujese el esfuerzo invertido en el etiquetado.

4. Estudio de la utilización de un clasificador entrenado únicamente, a partir de imágenes sintéticas. Esto, a su vez, implicaba dos puntos adicionales:

   a) Selección de *datasets* públicos de imágenes reales para generar clasificadores entrenados con dichos *datasets*

   b) Selección y desarrollo de un conjunto de procesos de aumentación de datos adecuado para el análisis que se pretende realizar.

   c) Métrica de comparación entre modelos a partir de *datasets* de test de cardinalidad y taxonomía heterogenea.

5. Elaboración y diseño de una metodología que permitiese una incorporación ágil no sólo de elementos diferentes, sino también de un mayor número de elementos del mismo tipo pero en diferentes condiciones. Con la idea de conseguir un modelo entrenado de tal forma que presentase una alta capacidad de generalización.

6. Incorporación al sistema de un proceso que permitiese una localización espacial de la señal de mayor o igual precisión que la proporcionada por los sistemas actuales, pero basada en visión artificial.

Además de estos objetivos, de este trabajo de tesis se han generado diferentes resultados, algunos directamente de la consecución de dichos objetivos y otros de manera indirecta, en sus procesos derivados.

1. Solución software de conteo y clasificación de vehículos basado en visión artificial como resultado además del proyecto EagleTD.

2. Dos publicaciones en revistas indexadas de primer cuartil.

3. Publicaciones en revistas indexadas de cuartil menor y conferencias.

4. Software de aumentación de imágenes basada en lenguaje de script.

5. Software de anotación avanzada para escenarios de transporte.

6. Sistema de reconocimiento de señales de tráfico de velocidad basado en *deep learning* como parte del resultado del proyecto europeo InLane.

7. Base de datos pública con señales generadas a partir de señales europeas.

1.3 Estructura del documento

Esta tesis se ha estructurado de la siguiente manera:

Capítulos 1 En este capítulo se ha realizado un esbozo contextual de las circunstancias que rodean a este trabajo. Se ofrece una breve introducción al mundo de los sistemas de transporte inteligentes y de las posibles aplicaciones a día de hoy de las soluciones basadas en visión artificial y *machine learning* que se ofrecen a nivel de mercado y académico.

Capítulo 2 Se definen y describen brevemente diferentes conceptos y tecnologías que ayudarán al lector a experimentar una mejor comprensión de la disertación. También se describen ciertas aportaciones realizadas en la práctica de aumentación de datos explicada entre dichos conceptos. Este capítulo se limita a tecnologías utilizadas dentro de los algoritmos y recursos empleados en la experimentación y el desarrollo asociado a esta tesis.

Capítulo 3 Se describen las contribuciones realizadas dentro de uno de los casos prácticos explicados. En concreto, se explican con mayor detalle la línea de investigación recorrida en el ámbito de los ATMS.

Capítulos 4 En este capítulo se definen los aspectos relacionados con el segundo caso práctico descrito, vinculado profundamente con los ADAS. Se define una visión global del sistema que se obtuvo como resultado y se pormenoriza el análisis al que dió lugar dicho desarrollo. Finalmente se aportan ciertas conclusiones sobre el estudio realizado.

Capítulo 5 El objetivo de este capítulo es esbozar un conjunto de conclusiones sobre el desarrollo en general y conclusiones particulares sobre cada uno de los temas tratados en esta disertación. Asimismo, se enumera una serie de líneas futuras que quedan pendientes y que resultarían de gran interés para una continuación de las líneas marcadas en esta tesis.

Capítulo 6 Se presenta una relación de las publicaciones a las que ha dado lugar esta tesis y de otras publicaciones relacionadas indirectamente de los resultados y desarrollos.

Si quieres construir un barco, no animes a tus hombres a reunir madera, dándoles órdenes y distribuyendo el trabajo. En lugar de eso, enséñales a anhelar la infinita inmensidad del mar.

Antoine de Saint-Exupéry

CAPÍTULO

2

Conceptos generales de visión artificial y *machine learning*

2.1 Introducción

Desde la aparición de la visión artificial o por computadora en la década de los 70 aproximadamente, ha prevalecido una tendencia clara a aplicar esta tecnología a diferentes aspectos de nuestra vida cotidiana, entre ellos el transporte. En este ámbito, la visión artificial se utiliza principalmente en dos frentes, como ya se ha comentado previamente. Por una lado, en favor de los usuarios de las vías de transporte, aportando sistemas que doten de mayor seguridad y confort a la experiencia de la conducción y por otro, proporcionando a los gestores de las infraestructuras sistemas para hacer más seguro el tránsito de vehículos o la gestión de servicios, monitorizando y controlando las vías de circulación. Existe mucha literatura relacionada con la aplicación de la visión artificial al mundo de las ITS [Loce et al.17] [Mine et al.19] [Ghosh and Lee00] [Javadi18], donde los autores describen de manera más o menos pormenorizada diferentes sistemas basados en visión artificial aplicada al mundo del transporte, clasificando la información en diferentes categorías en relación al tipo y a la finalidad de la aplicación. En general, estos sistemas se ejecutan a través de una secuencia de etapas (o *p*ipeline) como la que se propone a continuación:

Figura 2.1: Pipeline habitual en soluciones basadas en visión artificial y *machine learning*.

El objetivo de esta secuencia de pasos, que en ocasiones puede estar más desarrollado o menos en función de las necesidades, es interpretar el escenario en relación a la funcionalidad de la aplicación y considerar los elementos que tienen sentido dentro de esa funcionalidad, a diferentes niveles de abstracción semántica. Tras el análisis realizado por el sistema se produce una salida que podría consistir en desatar un conjunto de acciones que intervengan en los mecanismos físicos del entorno, en ofrecer información visual como una suerte de monitorización asistencial para el usuario objetivo, o en proporcionar ambos tipos de salida: informativa y activa.

La amalgama de técnicas utilizadas a lo largo del tiempo en los proyectos que aquí se presentan, aún compartiendo unas bases profundamente ligadas a los campos de la visión por computador y el *machine learning*, son de lo más heterogéneas y arbitrarias. Estas técnicas se encuentran dentro de alguna de las etapas de la secuencia de etapas explicada previamente (véase la figura 2.1).

En la etapa de adquisición se produciría la captura de las imágenes o de los datos por parte de los sensores, que podrían ser cámaras convencionales en cualquier rango del espectro electromagnético, dispositivos LIDAR, etc... Estos datos pasarían a una etapa de procesamiento donde se mejorarían mediante técnicas de filtrado o procesos de mejora de calidad. Con las imágenes mejoradas se procedería a la detección de los elementos de interés presentes en ellas. Tradicionalmente, esta etapa de detección se conformaba de dos fases: una etapa de muestreo, una de generación de descriptores y

una final de clasificación. La finalidad de la etapa de muestreo era precisamente la de generar muestras con mayor o menor inteligencia con las que alimentar la etapa de clasificación. Esta etapa, la de clasificación, sería la encargada de determinar finalmente si esa muestra generada pertenecía o no al tipo de clase de elementos que pretendemos detectar. Tras la etapa de muestreo y antes de la etapa de clasificación, a partir de la muestra se generaría un vector descriptor. Este vector era utilizado por el clasificador que habrá sido entrenado para tipificar elementos dentro de ese espacio de características al que pertenece el vector descriptor. Una vez detectados los elementos se procedería con otra etapa de clasificación, si fuera necesario. En esta nueva etapa de se asignarían a las detecciones unas tipologías específica que pertenecerían al conjunto de clases aprendidas por el nuevo clasificador. Paralelamente si fuese necesario se podría acompañar dicha etapa con una de seguimiento de elementos, para incorporar el carácter temporal en las propiedades que queremos analizar. Posteriormente, entraríamos a una etapa de comprensión de la escena, donde se establecerían criterios semánticos que nos proporcionarían un conocimiento más abstracto de la información obtenida. Para concluir, se informaría al usuario, a un operario o a un centro del control sobre el resultado del análisis realizado o se actuaría directamente sobre los sistemas físicos para realizar alguna acción como respuesta a situaciones detectadas. De igual modo, si la información tridimensional de la localización resultase necesaria, paralelamente a la etapa de análisis y tras un proceso de calibración que podría ser independiente, se podría realizar una etapa de localización tridimensional de los elementos detectados para nutrir de ese componente espacial 3d al sistema.

En aras de dotar de claridad al documento, a continuación describiré las herramientas y tecnologías utilizadas en los proyectos, subdivididas precisamente en estos dos campos de aplicación. Pero antes de continuar, no está de más aportar una breve definición de dichas disciplinas: la visión artificial y el *machine learning*.

1. **Visión artificial**, **visión por computador** o ***computer vision***: es una disciplina científica dentro de la cuál tienen cabida los algoritmos que tienen por objetivo, literalmente desde wikipedia, *"adquirir, procesar, analizar y comprender las imágenes del mundo real"* generando información que pueda ser manejada por un ordenador. Desde un punto de vista más profano, se podría considerar como

un intento o el camino para dotar de percepción visual, a semejanza de la humana, a una computadora y de comprensión de esa percepción.

2. ***Machine learning***, **aprendizaje máquina** o **aprendizaje automático**: es una categoría o rama del campo de la inteligencia artificial (y en este punto inteligencia podría entenderse como considere apropiado el lector) que trata de trasladar la capacidad de aprendizaje del ser humano a las máquinas. Aquí aprendizaje se podría entender como la capacidad de procesar información nueva para transformarla en conocimiento y la capacidad de generalización de dicho conocimiento, esto es, la capacidad de aplicar conocimiento aprendido a partir de experiencias previas a nuevas experiencias sin ser programado explícitamente para ello.

Sobre la visión artificial poco más añadiremos en este apartado. Sin embargo, en lo referente al *machine learning* añadiremos que a día de hoy existen tres paradigmas principales (véase la figura 2.2) en los que se pueden dividir los tipos de aprendizaje: aprendizaje supervisado, no supervisado y reforzado.

Figura 2.2: Paradigmas de aprendizaje.

En toscas pinceladas, el aprendizaje supervisado trata de establecer la relación existente entre unas variables de entrada y otras de salida que el investigador le proporciona al algoritmo de entrenamiento. El conjunto de datos que se utilizará para el entrenamiento debería estar, por tanto, formado por observaciones anotadas, como

se muestra en la figura 2.3. Esto quiere decir que para cada observación habría una etiqueta asociada que la identificaría como parte de un tipo de observaciones. Así, el algoritmo de aprendizaje es capaz de evaluar la corrección de la respuesta del modelo comparándola con la etiqueta asociada a esa entrada. Por ejemplo, si fuese una base de datos de imágenes de frutas a cada imagen habría que asociarle una etiqueta (manzana, pera, plátano...) de tal forma que si se le introduce una imagen de una manzana al modelo el algoritmo de entrenamiento pueda corroborar que el modelo acertó con su predicción, devolviendo una probabilidad alta de pertenencia a la clase manzana como salida, o no acertó. A su vez, estos algoritmos de aprendizaje supervisado se pueden agrupar en casos de **regresión** cuando la variable de salida es un valor continuo real y de **clasificación**, cuando la variable de salida es una categoría.

En un contexto de aprendizaje no supervisado el tipo de conjunto de datos que se usa no necesita una etiqueta que las describa. Estos algoritmos suelen agruparse en métodos de **clusterización** cuando quieres revelar las agrupaciones o relaciones subyacentes en los datos y **asociación**, cuando quieres descubrir reglas de asociación entre los datos.

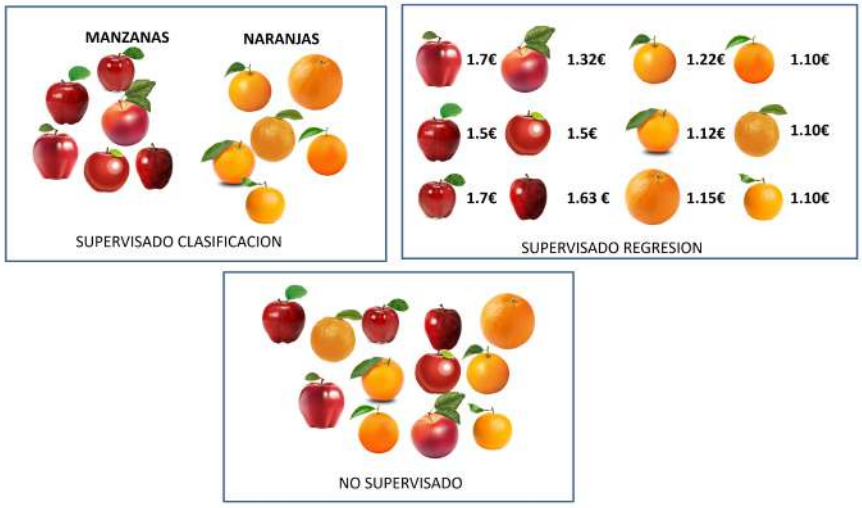


Figura 2.3: Estructuración de los datos en paradigmas supervisados y no supervisados.

El aprendizaje reforzado suele conllevar la presencia de un agente, una entidad que es capaz de interactuar con su entorno y generar una serie de acciones para resolver una tarea. Se utilizan mecanismos de recompensas o penalizaciones para reforzar un

comportamiento y que el agente vaya aprendiendo a realizar mejor la tarea. De esta manera el agente podrá identificar secuencias de acciones mejores o peores para conseguir su objetivo.

2.2 Técnicas en el campo del *computer vision*

En los siguientes párrafos se detallarán diferentes técnicas utilizadas o sobre la que se han basado posteriores ocurrencias e ideas para mejorar los resultados de los diferentes algoritmos, con relación al mundo de la visión por computador.

2.2.1 Conversión del espacio de color

Dentro de las técnicas utilizadas para evitar los cambios de iluminación en las imágenes una muy habitual es cambiar el espacio de color en el que se trabaja. Habitualmente las imágenes se encuentran dentro del espacio de color BGR. Este espacio presenta en cada valor de pixel una mezcla entre el matiz del color y su iluminación. Así encontramos colores similares al ojo humano como pueden ser el cyan y el azul en zonas opuestas de este espacio. Para minimizar el impacto que tienen los cambios de iluminación y para distribuir los colores por dicho espacio con una mayor coherencia perceptiva, es habitual cambiarlo a uno que distribuya iluminación y matiz en canales diferentes. Así, tendríamos diferentes espacios de color (véase la figura 2.4) que servirían a este propósito, como por ejemplo HSV, HSI, Cielab, IHSL[Hanbury and Serra02].

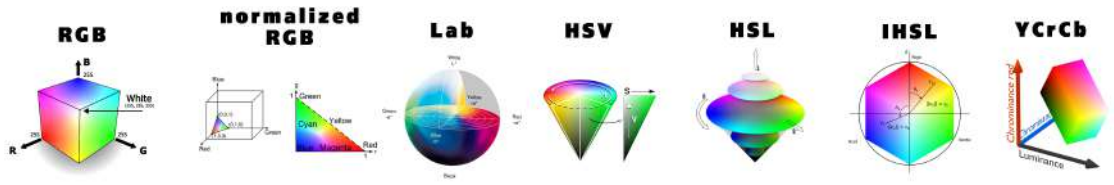


Figura 2.4: Diferentes espacios de color.

La principal ventaja del espacio de color IHSL frente a los demás modelos de color de coordenadas polares (como el HSV o el HSI) es que resuelve el problema que

presenta el canal del matiz (H) frente a colores con poca saturación. Por este motivo se utiliza posteriormente en uno de los sistemas que se describen, aunque computacionalmente sea algo más costoso 2.1.

$$
\begin{aligned}
s &= max(RGB) - min(RGB), \\
l &= 0{,}2125R + 0{,}7154G + 0{,}0721B, \\
cr_x &= R - \frac{G+B}{2}, cr_y = \frac{\sqrt{3}}{2}(B-G), \\
cr &= \sqrt{cr_x^2 + cr_y^2}, \\
H &= \begin{cases} undefined & \text{if } cr = 0 \\ \arccos(\frac{cr_x}{cr}) & \text{elseif } cr_y \leq 0 \\ 360° - \arccos(\frac{cr_x}{cr}) & \text{otherwise} \end{cases}
\end{aligned}
\tag{2.1}
$$

2.2.2 Generación de gradientes

Uno de los procedimientos más habituales para trabajar con imágenes es obtener para cada píxel el vector de gradiente en relación a la intensidad de sus vecinos. Este vector nos indica en qué dirección se encuentra la mayor variación de intensidades, figura 2.5.

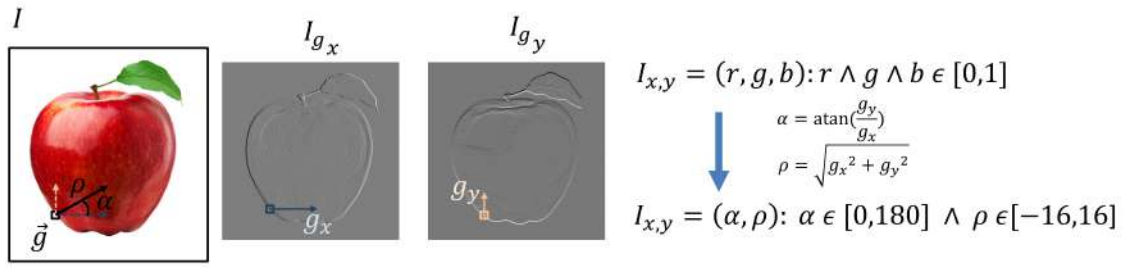


Figura 2.5: Conversión de un pixel del espacio de color al espacio de gradientes.

Para calcular estos gradientes se utilizan habitualmente operadores matriciales convolucionales (véase la figura 2.6) que aproximan la primera derivada aplicándolos en cada una de las dimensiones de manera separada para luego combinarlas y obtener un vector de gradiente bi-dimensional. Estos operadores se aplican a la imagen por

cada pixel, centrando el operador en el pixel y calculando la operación de convolución entre sus elementos adyacentes y los correspondientes valores del kernel.

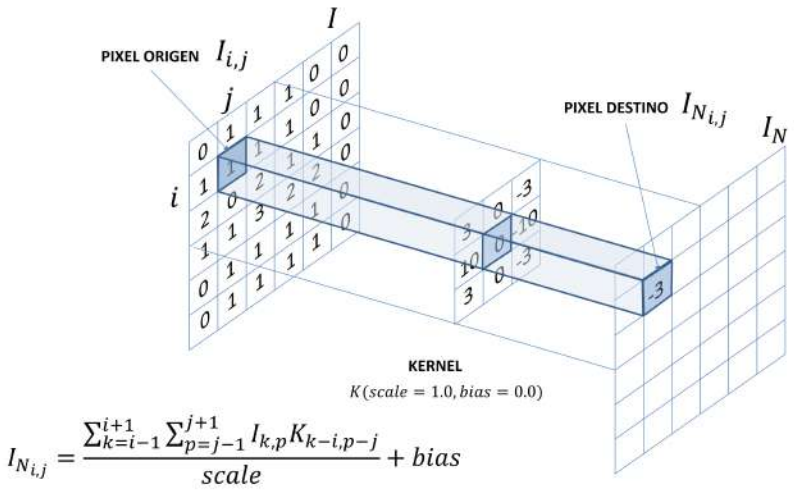


Figura 2.6: Convolución entre un kernel 3x3 y un píxel de la imagen.

Existen multitud de operadores para el cálculo de estos vectores gradiente, en el caso que nos ocupa se ha utilizado principalmente el operador Scharr, cuyos kerneles están definidos en la ecuaciones 2.2 y 2.3.

$$\begin{bmatrix} +3 & 0 & -3 \\ +10 & 0 & -10 \\ +3 & 0 & -3 \end{bmatrix} \tag{2.2}$$

$$\begin{bmatrix} +3 & +10 & +3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{bmatrix} \tag{2.3}$$

Pasar a trabajar con la imagen de gradientes proporciona ciertas ventajas frente a trabajar directamente con color. Por un lado, ofrece la posibilidad de eliminar gran cantidad de información que podría resultar espuria o insustancial y por otro lado reduce la dimensionalidad de los datos de entrada que pasan a ser tuplas bi-dimensionales en lugar de tripletas de color BGR.

2.2.3 **Watershed**

Este proceso se utiliza para rellenar zonas de imagen con el contenido de los píxeles vecinos. Este proceso en concreto se suele aplicar sobre la imagen en escala de grises. Su objetivo es segmentarla en N regiones de interés, donde N vendría indicado por el número de semillas. Este proceso interpreta la imagen como si fuera un mapa topográfico donde los diferentes valores de intensidad indican alturas. Así zonas claras se considerarían cumbres y zonas oscuras valles. Existen diferentes algoritmos para aplicar watershed, siendo el más común el que rellena las áreas por inundación (algoritmo de inundación por prioridad). Se va rellenando iterativamente a partir de las cuencas expandiendo las etiquetas para cada una de las cuencas hasta que ese relleno se encuentra con una zona de relleno que viene de otra cuenca. En ese punto se establece un límite. Estos límites actúan como particiones de la imagen que se considera rellenada al no tener más que propagar en una iteración.

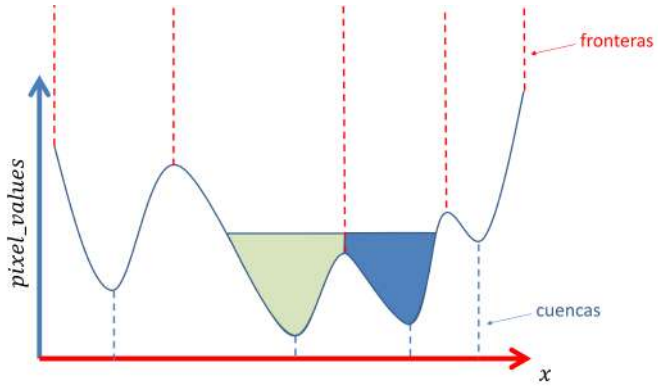


Figura 2.7: Segmentación mediante el algoritmo watershed.

Cómo se vaya segmentando la imagen va a depender en gran medida de lo que representen los valores asociados a los píxeles y de las semillas de inicio seleccionadas. Una buena estrategia suele ser partir de los centroides de las componentes conexas detectadas en la imagen e ir iterando sobre la transformada de distancia de la imagen dilatada.

2.2.4 Transformación de imágenes

Existen multitud de procesos que se le pueden aplicar a las imágenes para modificarlas y generar nuevas muestras. Estos procesos de transformación tienen sentido en contextos en los que es necesario aumentar un conjunto de datos para alimentar los sistemas con información diferente. Una notable aplicación de estas transformaciones se puede observar en los procesos de aumentación de datos aplicados principalmente para, como su propio nombre indica, aumentar la cantidad de datos con los que nutrir el aprendizaje de un modelo en un escenario con escasez de muestras.

Podemos clasificar las transformaciones más comunes en los siguientes grupos, véase la figura 2.8.

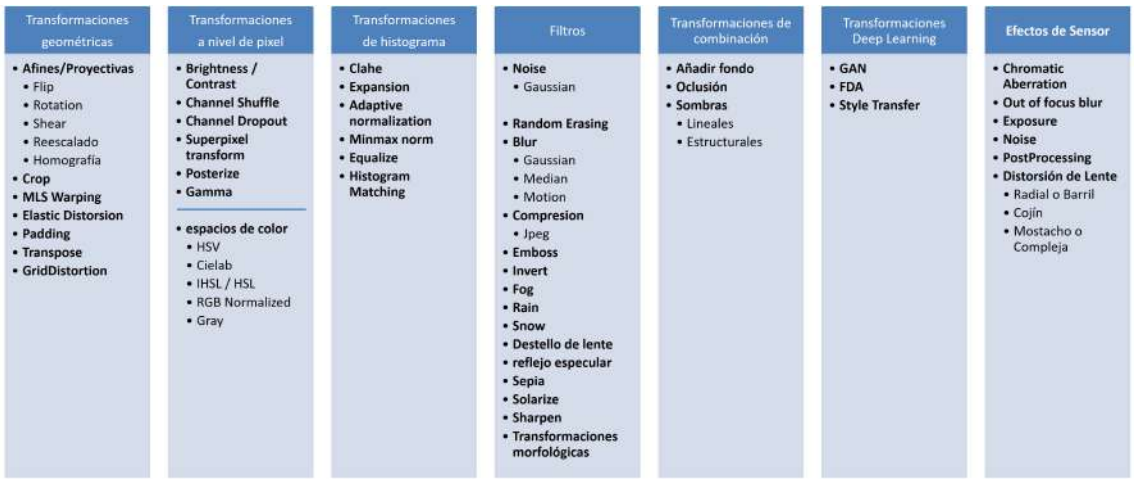


Figura 2.8: Lista de transformaciones sobre imágenes.

De entre ellas, sólo se han utilizado las siguientes en este trabajo, dejando como línea abierta la implementación de algunas de estas transformaciones para desarrollos futuros.

2.2.4.1 *Transformaciones Geométricas (figura 2.9)*

Transformaciones afines (2D y 3D): Consiste en cualquier transformación que represente una relación entre dos imágenes y que puede ser expresada en la forma de una multiplicación de una matriz (transformación lineal) seguida por un vector que se

Figura 2.9: Transformaciones geométricas aplicadas.

suma (traslación). Se utilizan las transformaciones afines para expresar: **rotaciones** (transformación lineal), **traslaciones** (vector suma) y **escalas** (transformación lineal). Dentro de las transformaciones geométricas, las afines son aquellas que mantienen el paralelismo, las líneas se mantienen como líneas, se conservan las proporciones. Estas transformaciones tienen 6 o 12 grados de libertad dependiendo de las dimensiones (véanse las ecuaciones 2.4, 2.5).

$$\begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} = R \left(\begin{array}{ccc|c} s_x & sh_x^y & sh_x^z & t_x \\ sh_y^x & s_y & sh_y^z & t_y \\ sh_z^x & sh_z^y & s_z & t_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} \tag{2.4}$$

$$R = \left(\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x & 0 \\ 0 & \sin\theta_x & \cos\theta_x & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \left(\begin{array}{ccc|c} \cos\theta_y & 0 & \sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \left(\begin{array}{ccc|c} \cos\theta_z & -\sin\theta_z & 0 & 0 \\ \sin\theta_z & \cos\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \tag{2.5}$$

Transformaciones proyectivas: Estas transformaciones modelan la proyección de un objeto en un espacio 3d sobre un plano. Se tratan de transformaciones lineales no singulares (invertibles) de coordenadas homogéneas. Este tipo de transformaciones mantendría la concurrencia, la colinealidad, las discontinuidades de la tangente, las proporciones cruzadas de cuatro puntos alineados (proporciones de proporciones de longitud). Para la aplicación de estas transformaciones proyectivas se ha utilizado una homografía entre planos.

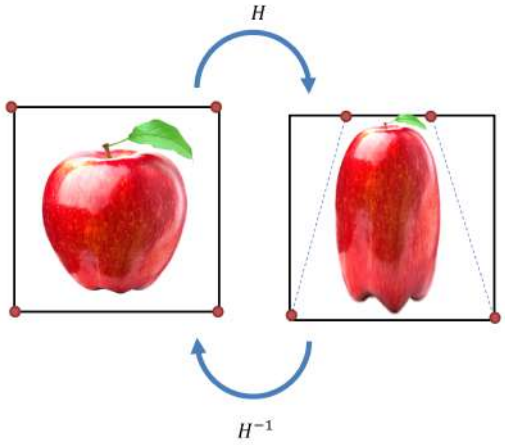


Figura 2.10: Transformación basada en homografía.

Se aplica una transformación entre el plano de la imagen y un plano imaginario de un escenario tridimensional generado de manera aleatoria. Cada punto del plano destino $p_w = (x_w, y_w)$ se corresponderá con un punto en el plano de imagen origen $p_i = (x_i, y_i)$. La matriz de la homografía, en la ecuación 2.6, tiene nueve elementos pero solo ocho son independientes, por ello presentan 8 grados de libertad.

$$\begin{pmatrix} \frac{x_i}{w} \\ \frac{y_i}{w} \\ w \end{pmatrix} = H \begin{pmatrix} x_w \\ y_w \\ 1 \end{pmatrix} = \begin{pmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{pmatrix} \begin{pmatrix} x_w \\ y_w \\ 1 \end{pmatrix} \tag{2.6}$$

Crop: Realiza un recorte aleatorio de una zona de la imagen

Moving Least Squares (MLS) Wrapping [Schaefer et al.06]: Aplica una deformación sobre la imagen basándose en una colección de puntos de control en la imagen origen p y una colección de puntos q que denotarían la posición de los puntos originales tras la deformación. Dado un punto v en la imagen, se resuelve la mejor transformación afín $l_v(x)$ que minimiza :

$$\sum_i w_i |l_v(p_i) - q_i|^2 \tag{2.7}$$

donde p_i y q_i son vectores fila y los pesos w_i tienen la forma

$$w_i = \frac{1}{|p_i - v|^{2\alpha}} \tag{2.8}$$

Como los pesos en este problema de mínimos cuadrados dependen del punto de evaluación v, esta técnica recibe el nombre de minimización de mínimos cuadrados que se mueven. Así se obtiene una transformación diferente para cada v. Como la función l_v es una transformación afín, está formada por dos partes: una matriz de transformación lineal M y una traslación T

$$l_v(x) = xM + T = (x - p_*)M + q_* \tag{2.9}$$

donde p_* y q_* son los centroides ponderados:

$$p_* = \frac{\sum_i w_i p_i}{\sum_i w_i} \tag{2.10}$$

$$q_* = \frac{\sum_i w_i q_i}{\sum_i w_i} \tag{2.11}$$

Así, la función de deformación afín aplicable a cada punto quedaría así

$$f_a(v) = (v - p_*)\left(\sum_i \hat{p}_i^T w_i p_i\right)^{-1} \sum_j w_j \hat{p}_j^T \hat{q}_j + q_* \tag{2.12}$$

donde $\hat{p}_i = p_i - P_*$ y $\hat{q}_i = q_i - q_*$

Elastic [Simard et al.03]: Esta operación genera para cada pixel un vector de movimiento aleatorio, tanto en dirección como en modulo. Para homogeneizar resultados se aplica posteriormente una media para cada vector en relación a los vectores de desplazamiento de sus pixeles vecinos.

Padding: Añade un marco de n pixeles blancos (transparentes en el caso de imágenes con canal alpha) a la imagen.

Transpose: Realiza una transposición de la imagen.

Grid: La transformación en este caso se produce dividiendo la imagen en un grid de MxN celdas de igual tamaño, transformando las dimensiones de dichas celdas posteriormente de manera aleatoria y recomponiendo la imagen.

Flip: Se realiza un cambio del sistema de coordenadas de la imagen.

2.2.4.2 *Transformaciones a nivel de píxel (figura 2.11)*

Figura 2.11: Transformaciones a nivel de pixel.

Brillo/Contraste: Mediante la transformación lineal de uno de los canales de la imagen se generan efectos que aumentan o disminuyen el brillo y el contraste, aplicándole un coeficiente y un sesgo a cada valor de pixel $I_{c_{i,j}} = \alpha I_{c_{i,j}} + \beta$

Shuffle: Con esta operación se intercambian de manera aleatoria los canales de una imagen.

Dropout: Esta operación descarta un canal al azar.

Posterize: La posterización es una operación que reduce el número de posibles elementos del espacio de color utilizado.

Gamma: La corrección gamma (figura 2.12) es una técnica que permite corregir una imagen usando una operación no lineal (véase la ecuación 2.13).

$$I_{i,j} = \left(\frac{I_{i,j}}{255}\right)^{\frac{1}{\gamma}} 255 \tag{2.13}$$

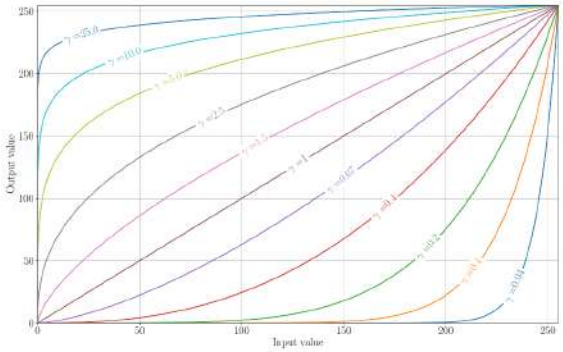


Figura 2.12: Corrección gamma para diferentes valores de gamma.

Log: Se aplica la función logaritmo a los valores de intensidad para aumentar el contraste en las zonas oscuras y disminuirlo en las zonas claras. Este operador se aplica mediante una tabla de consulta o Look-up table (LUT), figura 2.13.

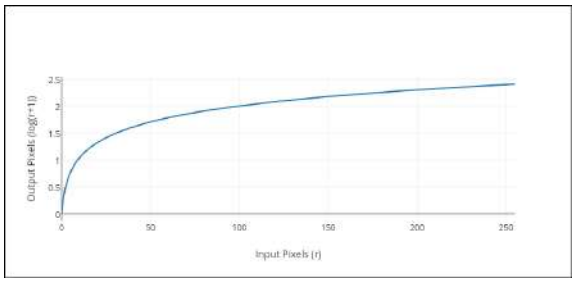


Figura 2.13: LUT del operador log.

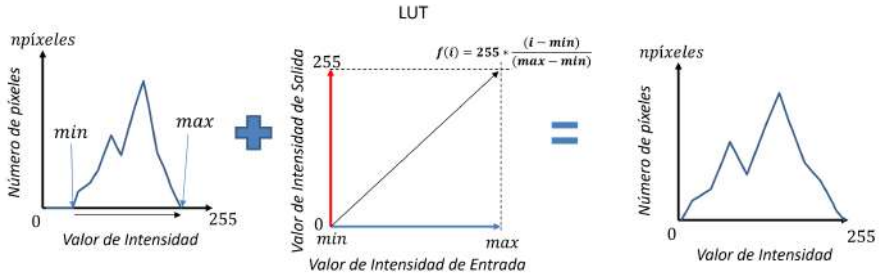
2.2.4.3 *Transformaciones de histograma (figura 2.14)*

Figura 2.14: Transformaciones de histograma.

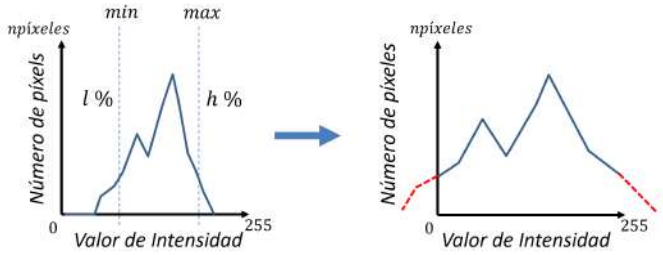
Expansión: Consiste en aumentar el rango de niveles de gris usados de un histograma expandiendo sus límites superior e inferior. El límite inferior pasaría a ser cero y el superior 255, repartiendo el resto de valores en el eje de abscisas de manera proporcional. Se puede utilizar una función 2.14 implementada en una tabla de consulta o Look Up Table (LUT) para hacer este cambio, figura 2.15.

$$f(I_{i,j}) = 255{,}0\frac{I_{i,j} - valor\_minimo}{valor\_maximo - valor\_minimo} \tag{2.14}$$

Una variante de esta expansión (prune) consistiría en seleccionar como mínimo y como máximo valores de intensidad que dejasen fuera un tanto por ciento de los píxeles de la imagen que caerían sobre sobre las zonas extremas del histograma (véase la figura 2.15 (b)).



(a) Normalización Min-Max



(b) Prune

Figura 2.15: Expansión del histograma.

Equalize: Se trata de una transformación sobre la imagen que pretende modificarla de tal manera que su histograma siga una distribución uniforme, dicho de otro modo, que exista el mismo número de píxeles para cada valor de intensidad. La función de distribución acumulada del histograma de una imagen con el histograma ecualizado debería de ser aproximadamente lineal. La ecualización no se aplica a cada uno de los canales RGB por separado ya que esto daría lugar a resultados no deseados. Lo habitual de tratarse de una imagen RGB es pasarla a HSV y aplicar la ecualización al canal V, la luminancia. Es una técnica que se utiliza principalmente para mejorar el contraste en las imágenes, figura 2.16.

El proceso de ecualización puede verse como la aplicación de una transformación a los niveles de gris. Esto es, un mapeo o asociación entre niveles de intensidad utilizando como LUT la función de distribución acumulada, donde el nivel de gris del pixel (x, y) de la imagen original $I_{x,y}$ pasa a ser $I'_{x,y} = 255\frac{cdf(I_{x,y})}{I_w I_h}$

Adaptive: La ecualización adaptativa del histograma utiliza métodos adaptativos para calcular diferentes histogramas, cada uno correspondiéndose a una sección diferente de la imagen. Funciona mejor que la ecualización del histograma si lo que se

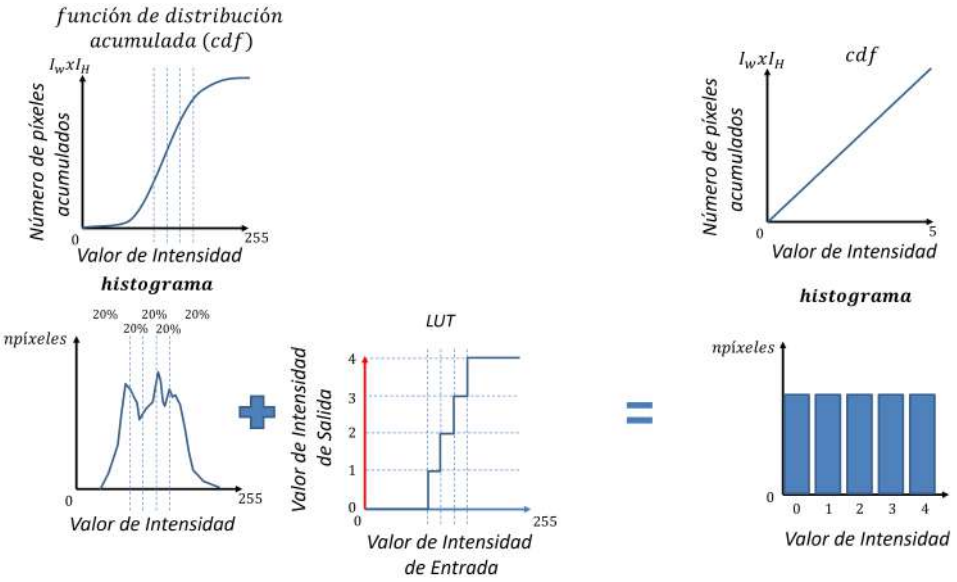


Figura 2.16: Ecualización del histograma (5 niveles de intensidad).

busca es mejorar el contraste local en la imagen y enfatizar bordes en regiones específicas. Entre los efectos no deseados que produce este tipo de ecualización hay que destacar el resaltado del ruido que se produce en zonas homogéneas a consecuencia de este aumento de contraste.

Clahe: o ecualización de histograma adaptativo local, así se denomina la técnica de procesamiento de imagen que aplica ecualización del histograma en diferentes partes de la imagen. Su principal objetivo es la mejora del contraste. Es similar al adaptativo pero añadiendo una limitación al contraste para que el ruido no se amplifique debido a la ecualización local.

2.2.4.4 *Filtros (figura 2.17)*

Figura 2.17: Filtros.

Adición Ruido: En este proceso se añaden píxeles con valores espurios a la imagen de manera aleatoria. La ubicación, frecuencia y valor de intensidad de estos píxeles vienen definidos según el tipo de ruido que se desee añadir:

1. Ruido gaussiano: Se trata de un proceso que genera un ruido en la imagen que sigue una distribución normal o gaussiana.

2. Ruido sal y pimienta: ruido que se presenta diseminado de manera dispersa por la imagen con una serie de píxeles blancos y negros.

Difuminado: El difuminado consiste en la atenuación general de bordes (de las altas frecuencias) de la imagen. Existen diferentes tipos de difuminado que se pueden aplicar:

1. Difuminado gaussiano : Difuminado que se obtiene al aplicar un kernel gaussiano en una operación de convolución sobre una imagen.

2. Difuminado basado en mediana: Este tipo de difuminado calcula para el entorno cercano de cada pixel su mediana, sustituyendo su valor por dicha mediana.

3. Difuminado de movimiento: este proceso trata de emular el desenfoque producido en las imágenes de cámaras que tratan de capturar elementos en movimiento. Para realizar este efecto se aplica un operador convolucional sobre la imagen utilizando un kernel gaussiano direccional y con orientación aleatoria.

Compresión JPEG: Este proceso pretende simular los objetos que aparecen en las imágenes proporcionadas por algunas cámaras en las que se aplica el algoritmo de compresión jpeg en el envío de las imágenes.

Inversión: Inversión del valor de los diferentes canales.

Pixelización: Reducción brusca de la resolución de una imagen convirtiéndola en un mosaico de teselas de gran tamaño a modo de píxeles.

Sharpen: acentuación de los bordes de la imagen haciendo una suma ponderada entre una imagen difuminada de la imagen original $I^{'}$, la propia imagen original I y un valor de sesgo:

$$I_{i,j} = \alpha I_{i,j} + \beta I^{'}_{i,j} + \gamma \tag{2.15}$$

Operaciones morfológicas: Las operaciones morfológicas aplican un elemento estructural sobre cada uno de los píxeles ya sea de erosión o de dilatación.

2.2.4.5 *Transformaciones de combinación (figura 2.18)*

Figura 2.18: Transformaciones de combinacion de imágenes.

Fondo: la aplicación de un fondo tras una imagen se realiza únicamente cuando la imagen tiene un canal de transparencia que define las zonas de la imagen en las que se puede ver dicho fondo. Simplemente se sustituyen los pixeles que presenten un grado de transparencia μ, multiplicando dicho coeficiente por el valor de intensidad del pixel y sumándolo con el contenido del fondo, ponderado por $1-\mu$. Esto nos devuelve una imagen en la que se aprecia el fondo en las zonas que presentaban transparencia.

Sombras: Se trata este de un proceso mediante el cual se añade una sombra sobre la imagen. Esta sombra puede ser lineal (separando la imagen en dos zonas) o estructurada. Las sombras lineales se generan calculando dos puntos aleatorios en dos laterales aleatorios de la imagen y ensombreciendo una de las zonas divididas por dicha línea. Las sombras estructuradas se generan a partir de imágenes externas. Estas imágenes se binarizan siguiendo un proceso de binarización conocido como Otsu que calcula el valor del umbral de binarización de manera que la dispersión dentro de cada clase (intra clase) sea lo más pequeña posible pero la distancia entre las diferentes clases (inter clase) sea lo más alta posible, entendiendo por clase cada uno de los rangos de intensidad después de dividir la imagen usando un umbral. Una vez binarizada y redimensionada la imagen de sombra al tamaño de la imagen destino, se utiliza esta imagen de sombra a modo de máscara para oscurecer las zonas de la imagen que coinciden con zonas claras en esta máscara (véase la figura 2.19). Las imágenes en las experimentaciones del capítulo 4 se obtienen de una base de datos pública de texturas [Cimpoi et al.14].

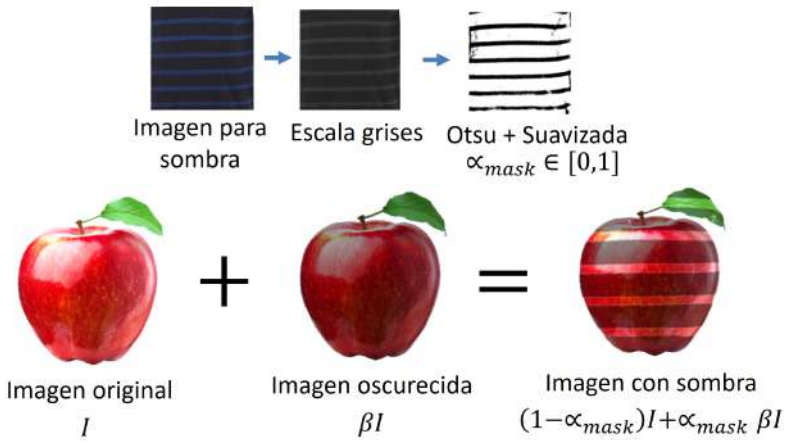


Figura 2.19: Algoritmo de generación de sombra estructurada.

Reflejo Especular: Este proceso genera un reflejo especular gaussiano en un punto aleatorio de la imagen. Este reflejo especular se genera utilizando la representación gráfica de un filtro gaussiano bidimensional y luego se superpone sobre el contenido de la imagen.

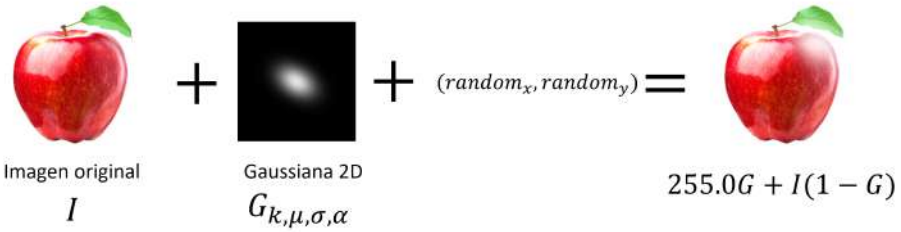


Figura 2.20: Algoritmo de generación del resaltado o reflejo especular.

En la figura 2.20, se puede ver el proceso de generación del resaltado especular. La gaussiana vendrá definida por la media μ, la desviación estándar tanto en x como en y σ_x, σ_y, el ángulo con el que se desea rotar el filtro α y el tamaño del kernel k. Finalmente se proporciona una ubicación aleatoria para aplicar dicho filtro utilizando la función indicada en la imagen. La intensidad del reflejo vendrá determinada por la media del filtro cuyo mayor rango es de 0 a 1.

2.2.5 Generación de vectores descriptores

Otro de los procedimientos habituales, y que se utilizará a lo largo de este texto, es convertir la imagen en un vector descriptor de características con información más compacta y relevante, de este modo pasamos la imagen al espacio de los datos donde estará representada por este vector descriptor multi-dimensional. Este tipo de conversiones suelen denominarse *hand-crafted features* ya que es el investigador el que decide las dimensiones y generación de dichas características.

Uno de los descriptores más conocidos y usados a nivel de visión artificial es el **histograma de gradientes orientados o HOG**. Esta representación de la imagen reduce la información que se va a utilizar en procesos posteriores conservando la información más útil. Un debate interesante surge a la hora de definir qué es información útil o que información queremos preservar y cuál no. Muy habitualmente por ejemplo el color presente en una imagen no nos sirve para llevar a cabo un análisis eficaz, ya que en ocasiones, por ejemplo, se requiere que el sistema funcione en escenarios nocturnos o de poca luz, donde la señal de matiz es débil. En estos casos la información de color no sería siquiera accesible y habría que trabajar en algoritmos que pudiesen recibir como entrada imágenes en grises. Por supuesto, la definición de utilidad viene marcada también por el problema que se pretenda resolver. Si por ejemplo quisiésemos inferir el color de una prenda a partir del análisis de su imagen, el color, en este caso, sería una característica determinante y muy útil.

Habitualmente un algoritmo de generación de un vector característico para una imagen, convierte la información presente en la imagen (Anchura x Altura x canales) en un vector unidimensional de n posiciones. En este caso HOG genera un vector de histogramas de direcciones de gradientes como se ilustra en la figura 2.21. En realidad se genera un vector de histogramas por bloque y luego se concatenan dichos vectores para obtener el descriptor final de la muestra. A su vez cada bloque es una concatenación de los histogramas de las celdas que lo componen.

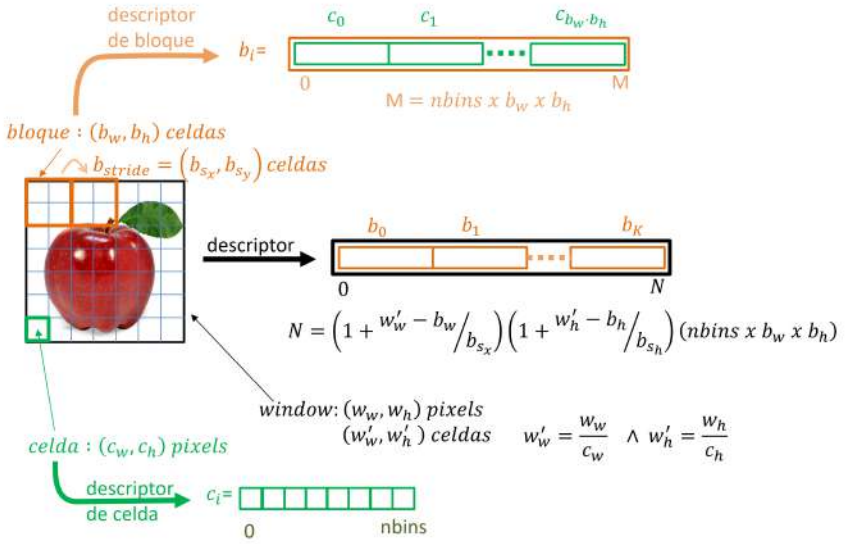


Figura 2.21: Descriptor HOG (en negro). Formado por la concatenación de descriptores de bloque (en naranja) que a su vez están formados por los descriptores de celda (en verde).

2.3 Técnicas en el campo del *machine learning*

Para empezar introduciré al lector a las técnicas más intrínsecamente ligadas al ámbito del *machine learning* utilizadas. Dichas técnicas se han abordado en todos los casos mediante el paradigma de aprendizaje supervisado, que como ya se ha explicado anteriormente trataría de modelar la relación entre la entrada y la salida proporcionada por un supervisor, adaptando el conocimiento, esto es, ajustando sus parámetros internos mediante el análisis de la salida generada en relación a la salida esperada para una entrada, con el objetivo de realizar tareas predictivas o de inferencia una vez el modelo haya aprendido dicha relación.

Principalmente a lo largo de los trabajos aquí presentados se han utilizado dos técnicas concretas de *machine learning*: las máquinas de soporte vectorial y técnicas basadas en redes neuronales que de un tiempo a esta parte vienen recibiendo el nombre de *deep learning* por motivos que explicaremos más adelante.

2.3.1 **Máquinas de soporte vectorial**

Las máquinas de soporte vectorial, tratan de dividir el espacio de datos en dos zonas mediante un hiperplano, un subespacio plano afín de codimensión 1 (en dos dimensiones el hiperplano se correspondería con una recta). Para ello parten de un conjunto de elementos representados en el espacio de los datos con sus respectivos vectores de características y sus etiquetas. En un ejemplo como en el que vemos en la figura 2.22 pueden existir múltiples líneas para separar esos datos bidimensionales. Para obtener una línea óptima se busca la que tenga mayor distancia contra todas las muestras presentadas. Tiene que pasar lo más lejos posibles de todos los puntos, es decir, la mínima distancia a un punto debería de ser la máxima posible. Esta menor distancia a un punto se conocería como el margen, que puede ser suave o duro en función de si permitimos que algún punto se interne en la región entre márgenes o no. Las muestras más cercanas al hiperplano reciben el nombre de vectores soporte. Como estamos trabajando con un clasificador en el que se intenta maximizar este margen se dice que el clasificador es un clasificador de margen máximo; este tipo de clasificadores suelen ser muy sensibles a la presencia de outliers en los datos de entrenamiento. Para resolver precisamente esta sensibilidad a los outliers se utilizan los márgenes suaves, de manera que la ubicación del hiperplano se realiza permitiendo algunas clasificaciones erróneas. Para determinar el número de clasificaciones erróneas permitidas en favor de una mejora en la clasificación, es decir, para ubicar el hiperplano, se utiliza la validación cruzada. Este tipo de clasificadores donde permitimos cierto grado de error al clasificador recibe el nombre de Clasificador de vectores de soporte o clasificador de margen suave, y a las muestras que se encuentran en el límite de los márgenes o entre los márgenes se les denomina vectores de soporte.

Cuando tenemos el problema de una amplia intersección entre clases, es cuando entran en juego las máquinas de soporte vectorial o SVM. Lo que tratan de hacer es ir aumentando la dimensionalidad de los datos buscando una dimensión mayor en los que estos datos sean separables. Pero, ¿cómo aumentamos las dimensiones de estos datos? ¿Qué nuevo valor añadimos al vector de dimensión n con el que ya vienen los datos? Para ello, las máquinas de soporte vectorial manejan un concepto llamado funciones Kernel para, de manera sistemática, encontrar los clasificadores de vectores de soporte en dimensiones más altas. Estos kernels se aplican a los datos, observando

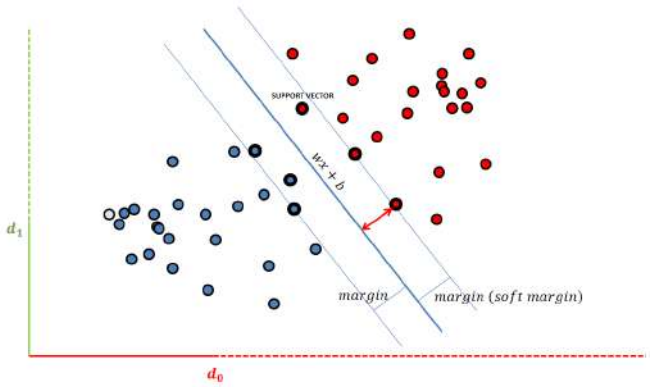


Figura 2.22: Clasificador de soporte vectorial.

las muestras dos a dos. Tras la aplicación el objetivo es la obtención de un clasificador de soporte vectorial adecuado para el nuevo vector con las dimensiones adicionales resultado de haber aplicado el kernel. Mediante *cross-validation* encontramos el valor para d que denota el número de dimensiones aumentadas, con el mejor clasificador de soporte vectorial.

Para el entrenamiento de los SVMs utilizamos, en nuestro caso, un sistema de *cross-validation* como el que se explica en la figura 2.23. El método de validación cruzada es un método estadístico que realiza diferentes particiones de los conjuntos de datos y sobre ellas, diversas iteraciones de entrenamiento y test, alternando entre combinaciones de estas particiones con el objetivo de encontrar un parámetro óptimo. Para alcanzar ese valor óptimo para el parámetro, este se va modificando en las diferentes iteraciones y nos vamos quedando con el mejor resultado de entrenamiento. Finalmente cuando ya hemos establecido cual es el valor optimo del parámetro se reentrena aplicando ese valor al parámetro y utilizando todo el *dataset* para obtener el modelo definitivo.

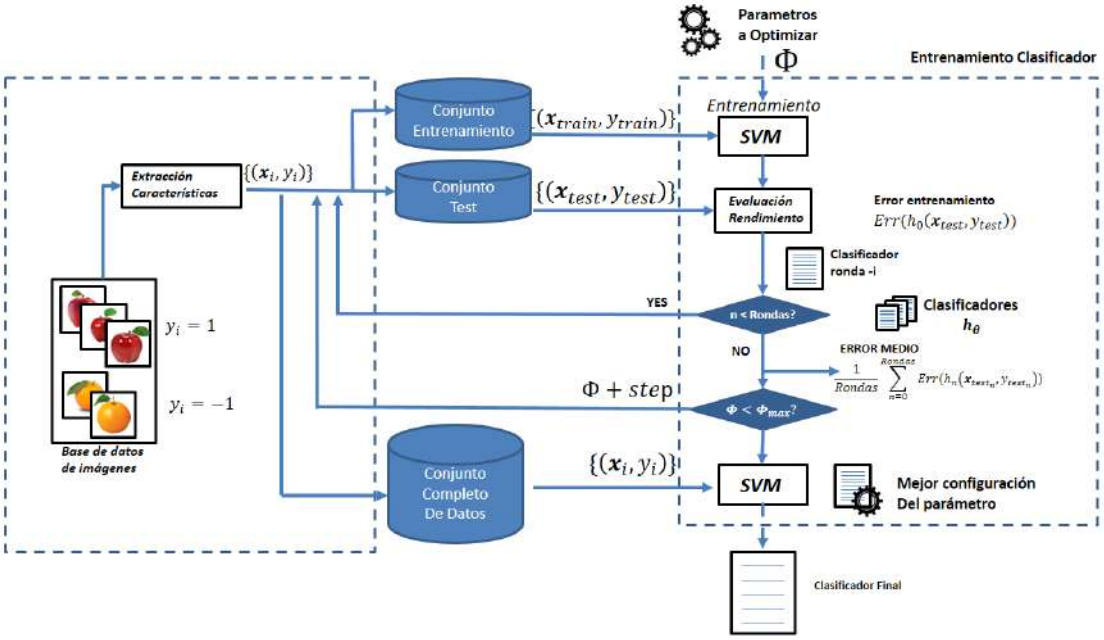
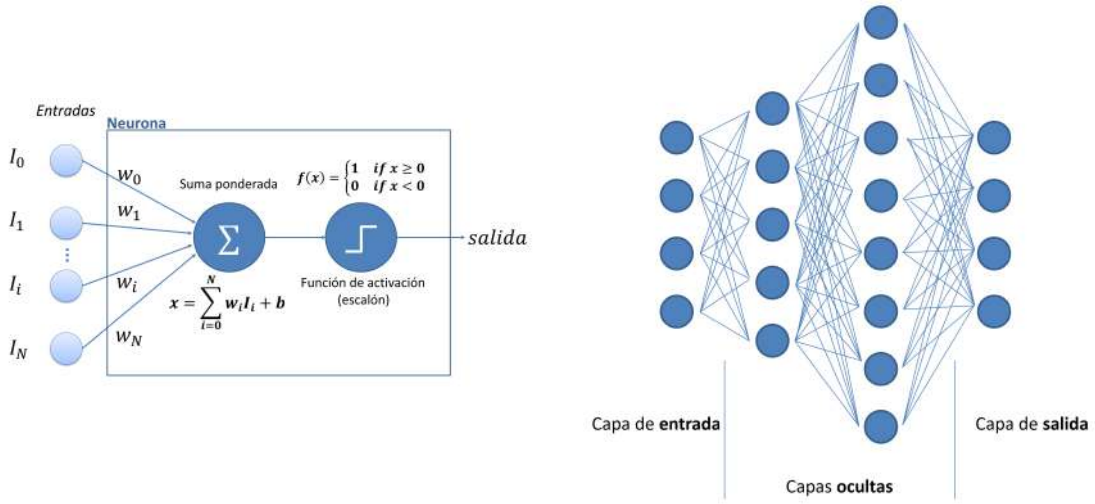


Figura 2.23: Algoritmo de *cross-validation* empleado.

2.3.2 Redes Neuronales Convolucionales

Antes de introducir la familia de modelos utilizados en este trabajo vamos a realizar una breve introducción al siempre fascinante mundo de las redes neuronales. Empecemos por el principio: ¿qué son las redes neuronales? Las redes neuronales son una familia de algoritmos dentro del *machine learning* que lleva con nosotros desde mediados del siglo pasado. El concepto de red neuronal, surge de entender el proceso de aprendizaje que queremos implementar para una máquina como una analogía del proceso que se produce en nuestro cerebro, donde un sistema interconectado de neuronas que pueden activarse y apagarse, modelan el conocimiento. No deja de ser una suerte de inspiración ya que todavía no sabemos cómo funciona exactamente el cerebro, pero la idea general es que el conocimiento se almacenaría en las conexiones entre estas neuronas artificiales. Desde un punto de vista más esquemático, si tuviésemos que definir una red neuronal, se podría decir que es un conjunto de neuronas agrupados en niveles o capas que se interconectan entre sí, figura 2.24.

Para entender una red neuronal hay que entender cómo funciona su unidad básica, la neurona. Una neurona es un elemento que recibe n entradas numéricas y devuelve



(a) Perceptrón **(b)** Diagrama

Figura 2.24: Disección de una red neuronal.

una salida, como puede observarse en la figura 2.24. Hasta aquí no se diferencia demasiado de lo que sería una función matemática que modelara una dependencia entre los valores de entrada y el valor de salida. También se podría ver como una función desde el ámbito de la programación (que de hecho es como se implementa), con unos valores de entrada y un valor de salida. La operación que realiza una neurona implica a un conjunto de valores llamados pesos, a un valor de sesgo o bias y a una función de activación. En realidad se trata de una suma ponderada de las entradas, a cuyo resultado se le aplica ese bias y que finalmente se introduce en una función de activación que devolverá una salida entre cero y uno, aunque esto último dependerá de esa función de activación vinculada a la neurona. En lo que se refiere a la arquitectura en la organización de estas neuronas, es práctica habitual que se distribuyan en diferentes capas conectadas entre sí. De este modo a partir de las entradas las neuronas de que se activen en una capa harán a su vez que otras neuronas en capas posteriores se activen o desactiven, propagándose así una secuencia de activaciones y desactivaciones hasta la salida, que se corresponderá con el resultado de la red neuronal. Y es aquí donde aparece la nueva designación por la que se le conoce hoy día. Cuando el número de capas ocultas (capas internas) es alto es cuando el asunto comienza a volverse profundo y se empieza a hablar de *Deep Learning*.

Este campo del *machine learning*, lleva en el panorama científico con el nombre de redes neuronales artificiales, aproximadamente desde que, en 1943, Warren McCullock propuso el primer modelo formal de una neurona, que a su vez dio pie en 1958 a que Frank Rosenblatt, creara el perceptrón, una neurona artificial o unidad básica de inferencia a partir de la cual se puede formar una red neuronal artificial más compleja mediante interconexiones de este elemento. Las redes neuronales han evolucionado mucho desde entonces pero no ha sido hasta hace unas pocas décadas cuando se ha producido un salto cuantitativo en su aplicación en todo tipo de sistemas de visión artificial aplicados a la resolución de problemas reales: reconocimiento de caracteres, de texto, reconocimiento facial, de voz, generación de texto, clonación de voz, mejora de imagen, traducción de idiomas, conducción autónoma, inspección de piezas, análisis genético, pronóstico de enfermedades, predicción bursátil, clasificación de elementos, conteo de personas, detección de incendios, detección de comportamientos irregulares, aplicaciones artísticas ... la cantidad de aplicaciones es interminable. Este incremento sustancial en el uso de estas técnicas se ha debido principalmente a tres razones: mejoras en el hardware aparición y desarrollo de las GPUs, mejoras científicas como el desarrollo y la aplicación de la backpropagación al proceso de entrenamiento o como la resolución del problema del desvanecimiento del gradiente y abundancia masiva de datos (*big data*) en las nuevas sociedades.

Diferentes hitos, mostrados en la figura 2.25, nos han traído hasta la situación que vivimos actualmente en el *deep learning*, pero dejando de un lado el camino recorrido y con esta contextualización en la mente del lector, continuemos con un viaje conceptual a través del *Deep Learning*.

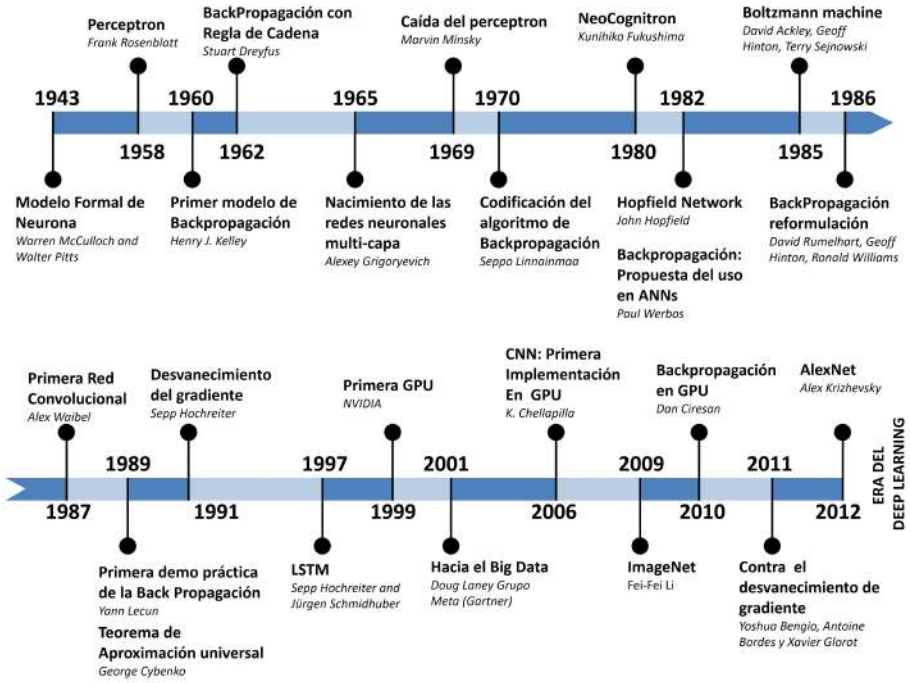


Figura 2.25: Hitos en la historia de las redes neuronales.

2.3.2.1 **Clasificadores Lineales**

Los clasificadores lineales son un ejemplo de clasificadores paramétricos donde todo el conocimiento de los datos de entrada se resume en una matriz de pesos que se configura durante el proceso de entrenamiento.

Partimos de una situación en la que tenemos acceso a un espacio de características n-dimensional donde cada punto, definido por un vector de n valores, representa un elemento de interés. En este caso cruces azules o guiones naranjas (caso a en la figura 2.26). Imagine el lector cualquier tipo de elementos (naranjas o manzanas, perros y gatos...). Cada dimensión representaría una característica del elemento, que coge valores numéricos y describe alguna característica que lo define y que a la postre nos permitirá diferenciarlo (altura, número de patas, diámetro,...).

Los clasificadores lineales tratan de crear diferentes particiones en un espacio de características n-dimensional utilizando hiperplanos y estableciendo regiones disjuntas donde caen las diferentes clases de los elementos que nos interesa tipificar.

Desde una perspectiva de red neuronal, se puede obtener un clasificador lineal

utilizando una capa de entrada (variables independientes), una capa de salida (variables dependientes) y una capa intermedia, formada por neuronas conectadas a la entrada: la Full Connected (FC) o totalmente conectada. Esta capa conecta cada una de sus neuronas con todas las neuronas de la capa precedente. La capa de salida, en caso de clasificación lineal, realiza una función de activación lineal; en los ejemplos es una función escalón. Esta función, es práctica habitual representarla como una capa en los esquemas aunque no tienen parámetros que aprender y simplemente definen el tipo de función de activación que utilizarán las neuronas de una capa.

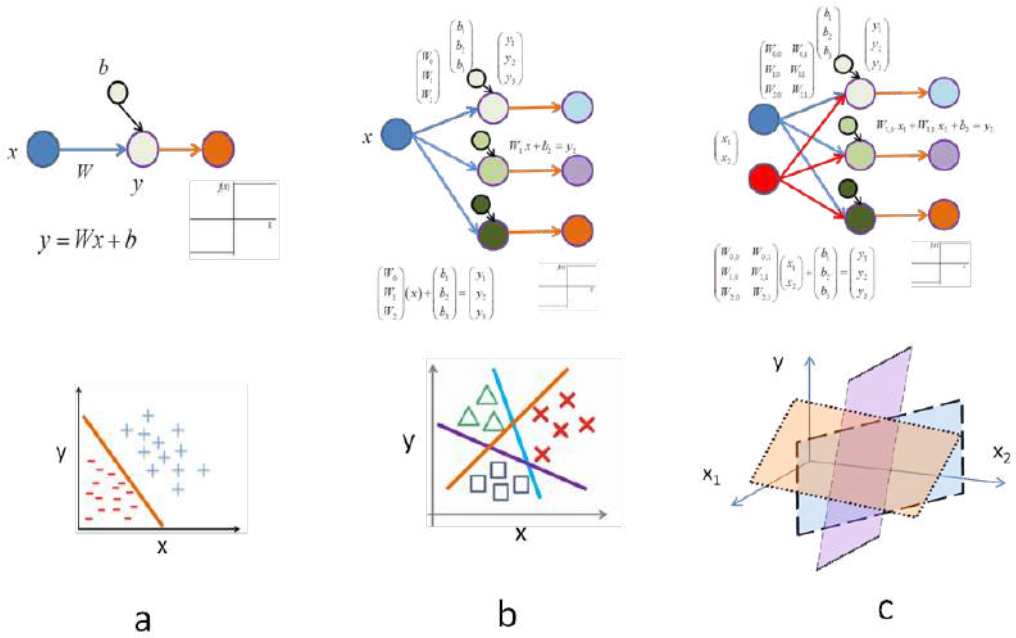


Figura 2.26: Esquema simplificado de clasificadores lineales.

En la figura 2.26, el caso a representa un clasificador lineal de una variable independiente y dos clases, clasificador binario. Lo que se debe aprender del modelo es el valor de W y bias, b, que clasifica mejor los datos: dos parámetros. Al ser una recta, W es la pendiente de la recta y bias la ordenada en el origen. En el caso b, tenemos un clasificador lineal multiclase de tres clases y una variable independiente. Los parámetros que se deben ajustar son: tres W de la capa FC, y 3 bias, uno por cada salida. En el caso c, el modelo b es ampliado a dos variables de entradas. Los parámetros que se deben aprender son la matriz W de la FC (2(entradas)x3(salidas)=6), y 3 bias, uno por

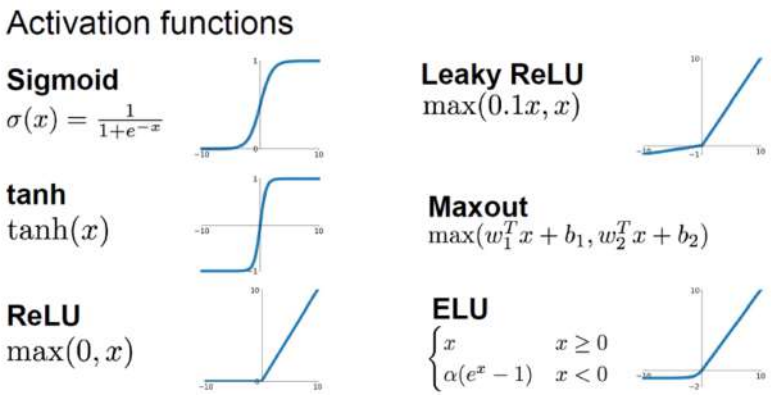


Figura 2.27: Expresiones algebraicas y representación gráfica de diferentes funciones de activación.

cada salida. El separador de clases en este caso es un plano; en general serán hiperplanos.

2.3.2.2 **Clasificadores no Lineales**

Si las neuronas utilizasen funciones de activación lineales la salida total de la red sería una combinación lineal de estas funciones lineales, es decir, una función lineal y se comportaría como un modelo de regresión lineal, eliminando la posibilidad de manejar distribuciones de datos no-lineales. De ahí surgió la necesidad de desarrollar nuevas funciones de activación (vease la figura 2.27) que añadiesen un componente de no-linealidad a la neurona.

Estas funciones, es práctica habitual representarlas como capas en los esquemas aunque no tienen parámetros que aprender y simplemente definen el tipo de función de activación que utilizarán las neuronas de una capa.

Por tanto el objetivo principal de la función de activación de una neurona es determinar la salida de la neurona (si esta se activará o no y cuantificar esta activación) añadiendo, además, el componente no lineal que ayuda a la neurona a aprender datos complejos o que no sigan una distribución lineal.

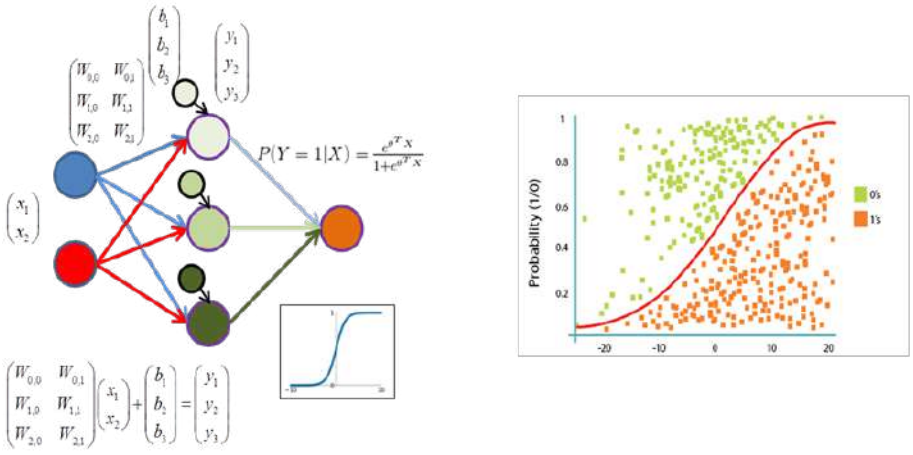


Figura 2.28: Representación gráfica de una red neuronal con clasificación no lineal (sigmoide).

Entre las funciones de activación no-lineales tenemos la sigmoide/logística, tangente hiperbólica, la unidad de rectificación lineal (ReLU), la Leaky ReLU, MaxOut y ELU entre otras. La evolución y aparición de nuevas funciones de activación obedece habitualmente a la solución de alguno de los inconvenientes que presentaban las que ya existían. Por ejemplo, la función sigmoide tenía el problema del desvanecimiento del gradiente al realizar la *backpropagation* en la etapa de entrenamiento. Los gradientes se volvían tan pequeños que no servían para guiar el descenso del gradiente. La red dejaba de aprender o le suponía un coste temporal tremendo alcanzar una configuración de pesos útil. Además es una función con un coste computacional alto al igual que la tangente hiperbólica. Así surgió la función ReLU que seguía siendo no-lineal y era computacionalmente eficiente. Aunque presentaba un problema (*Dying ReLU problem*) cuando las entradas se aproximaban a cero o eran negativas y el gradiente se volvía cero evitando su propagación y la capacidad de seguir aprendiendo. Para solucionar esto surgió la función *leakyReLU* que presentaba una pendiente positiva en la zona negativa y así posibilitaba la propagación hacia atrás del gradiente. Aunque no proporcionaba predicciones consistentes para valores negativos de entrada. Existen otras aproximaciones como las unidades *maxout* que tratan de aproximar una función convexa con un conjunto de funciones lineales. Una unidad *maxout* selecciona el máximo de las entradas como salida.

Y finalmente la unidad lineal exponencial o ELU que es similar a la función ReLU con alguna diferencia. No sufre del problema de desvanecimiento del gradiente ni de la explosión del gradiente y no sufre tampoco del problema de neuronas que se mueren. ELU ofrece menores tiempos de entrenamiento y una mayor precisión que el uso de ReLU y sus variantes. Aunque computacionalmente hablando es algo más costosa que ReLU y sus variantes debido precisamente a la no-linealidad asociada a las entradas negativas.

2.3.2.3 **Capas en las redes neuronales**

Entendiendo las redes neuronales como una secuencia de capas interconectadas, resulta de gran interés establecer el tipo de capas que se pueden utilizar de manera más habitual, si bien es cierto que innovar en el desarrollo de capas diferentes o en combinaciones de capas distintas es una línea abierta dentro del mundo del DL.

Continuando en un orden arbitrario de relevancia, entre las más utilizadas nos encontramos al buque insignia de las capas dentro de las redes profundas: las capas convolucionales, un tipo de capas muy importantes sobre todo para trabajar con imágenes y que da nombre a una nueva tipología de red donde solo aparecen este tipo de capas (redes totalmente convolucionales). Estas capas presentan la particularidad de que a cada una de sus neuronas se le asocia un campo receptivo. Esto quiere decir que la operación de convolución no se realiza contra todas las neuronas del nivel precedente sino que únicamente con una ventana en entorno inmediato a la posición en la que se encuentra la neurona. Además la distribución de las neuronas en esta capa es algo diferente, se distribuyen a lo largo de tres dimensiones. La tercera dimensión, la profundidad, denota el número de filtros de la capa o canales. Cada canal tendrá asociado un kernel de convolución de tamaño (K_w, K_h, I_c) de pesos y un bias (véase la figura 2.29. Este kernel y bias, se comparten por todas las neuronas pertenecientes a ese canal. Así una capa convolucional de M x N neuronas con 64 filtros, tendrá como salida un cubo de $M_{out}xN_{out}x64$ valores. En cada canal, cada neurona aplicaría la convolución entre el kernel de pesos asociado a su canal y su campo receptivo de la entrada.

El tamaño de salida de la capa convolucional no tiene por qué coincidir con el de entrada. Aplicando un kernel de convolución se puede reducir la dimensionalidad del

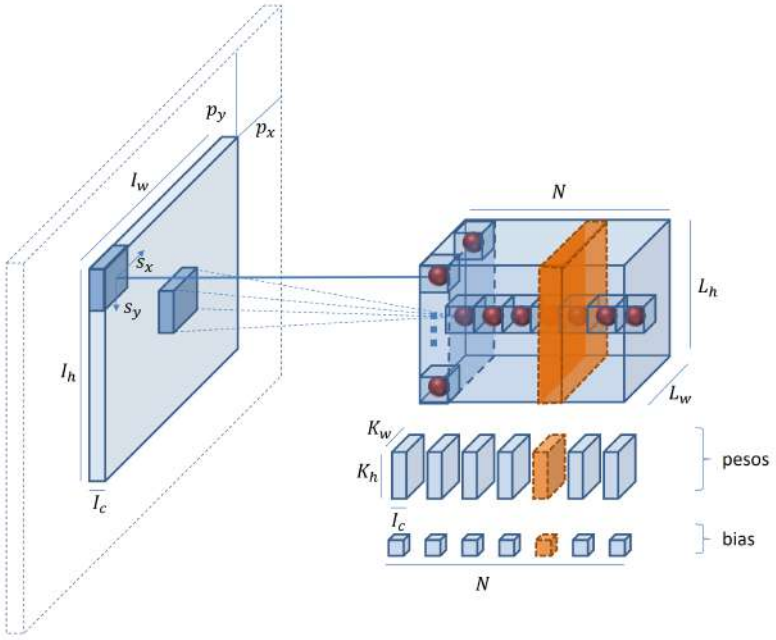


Figura 2.29: Estructura de una capa convolucional.

input de entrada, aumentarla o dejarla igual, todo depende de los valores asignados a los hiperparámetros $stride$ (paso en píxeles de una neurona a otra) y $padding$ (número de píxeles adicionales que se añaden alrededor de la imagen).

En general el cálculo del volumen de salida que se corresponde a una capa convolucional con los parámetros (s, p, k), se realizaría utilizando la siguiente formula.

$$(O_w, O_h) = (\frac{I_w + 2p_x - K_w}{s_x} + 1, \frac{I_h + 2p_y - K_h}{s_y} + 1) \tag{2.16}$$

Y su número de parámetros se correspondería con el resultado de la siguiente ecuación:

$$Par = W_c + B_c = Input_c K^2 N + N \tag{2.17}$$

Existe otro tipo de capa derivada de la convolucional que agrega un nuevo parámetro para su configuración: la capa de convolución dilatada [Yu and Koltun16]. Esta capa es idéntica a la anterior con la salvedad del parámetro d de dilatación que ensancha la zona de aplicación del kernel, en lugar de aplicar el kernel de convolución

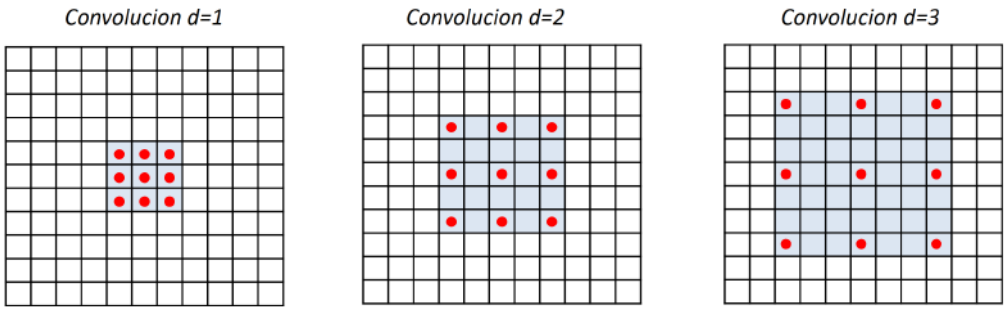


Figura 2.30: Expansión del kernel en la capa de convolución dilatada.

a los píxeles más inmediatos al pixel procesado se aplica a pixeles a cierta distancia definida por dicho parámetro, con lo que los espacios entre los elementos del kernel original aumentan (véase la figura 2.30). El objetivo que persigue esta dilatación es obtener un campo receptivo mayor manteniendo el número de parámetros ya que no aumenta el tamaño del kernel (solo aumentan los espacios entre sus elementos).

Otro de los grandes iconos dentro del conjunto de capas utilizadas en redes neuronales profundas son las capas *pool*, de las que Geoffrey Hinton dijo:

> *The pooling operation used in convolutional neuronal networks is a big mistake, and the fact that it works so well is a disaster.*

Se trata de una capa que reduce la dimensionalidad espacial (ancho y alto) de su predecesora, manteniendo la misma profundidad, es decir el mismo número de filtros. Existen diferentes versiones de capas *pool*, en función de la decisión que toman para realizar esa reducción. En general, consiste en elegir un valor de salida por cada neurona para todo el campo receptivo que tiene. La elección definirá, como hemos dicho, el tipo de capa que estamos utilizando: máximo, media... La reducción de dimensionalidad se decide a través del parámetro *stride* que se corresponde con el desplazamiento que se produce en píxeles entre neuronas.

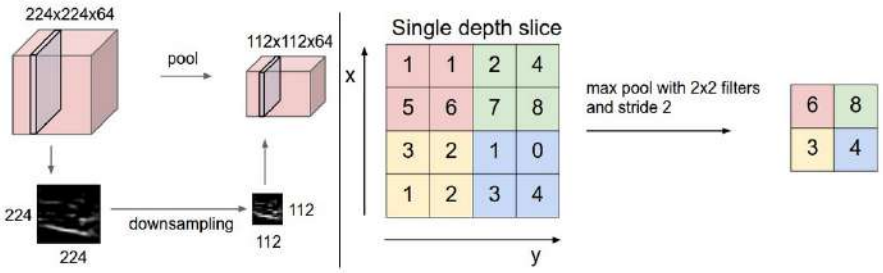


Figura 2.31: Capa *max-pool*, cada neurona elegirá el máximo de su campo receptivo.

Habiendo mencionado ya la capa *pool* y la capa convolucional, introduciremos ahora sus contrapartidas *unpool* y la convolución traspuesta. Aclararemos que utilizamos el concepto de contrapartida con la siguiente connotación: hasta el momento hemos descrito capas que tratan de reducir las dimensiones del espacio de características, y estas dos últimas capas tratan de aumentarlas.

La capa de convolución traspuesta aplicaría la convolución sobre la entrada, habiendo aumentado previamente la resolución de la misma con padding, como puede verse en la figura 2.32, y la capa *unpool* ampliaría la resolución ampliando la salida y ubicando el valor para cada neurona en la posición en la que se encontraba el máximo en su zona de ampliación local (véase la figura 2.33).

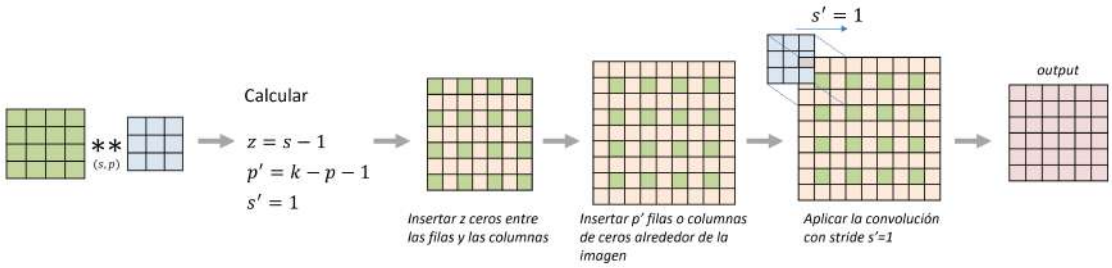


Figura 2.32: Esquema simplificado del funcionamiento de la convolución traspuesta.

Las dimensiones de salida de una capa convolucional traspuesta se pueden calcular con la ecuación

$$(O_w, O_h) = (s_x(I_w - 1) + K_w - 2p_x, s_y(I_h - 1) + K_h - 2p_y) \tag{2.18}$$

No hay que confundir las convoluciones traspuestas con las capas de deconvolución que realizan la operación inversa a la convolución. Las convoluciones traspuestas son

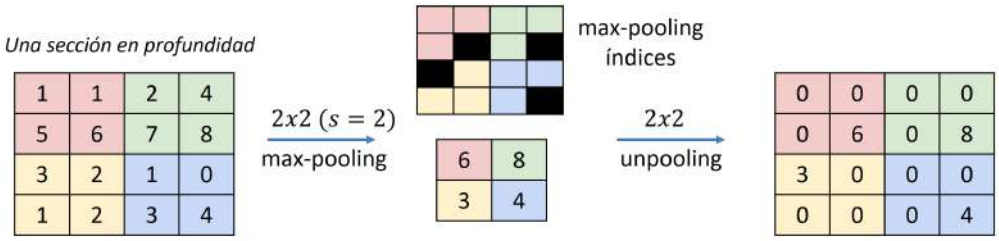


Figura 2.33: Capa *unpool* para aumentar resolución.

convoluciones convencionales pero en las que se han modificado el mapa de características de la entrada utilizando para ello sus parámetros de $stride$ y $padding$. Es importante advertir que estos parámetros ($stride$ y el $padding$) no se corresponden con el número de ceros añadidos alrededor de la imagen y la cantidad de desplazamiento que sufre el kernel cuando lo desplazamos por los datos de entrada como en las capas convolucionales convencionales.

A continuación veremos un tipo de capas usadas para evitar un problema que se produce a la hora de entrenar los modelos llamado *overfitting*. Para evitarlo, entre otros mecanismos distintos, se desarrollaron las capas de regularización, entre ellas la capa de *batch normalization* y la capa *dropout*.

2.3.2.4 **Capas de regularización**

Las capas de regularización se utilizan para evitar la situación de sobreajuste u *overfitting* a los datos de entrada en la fase de entrenamiento.

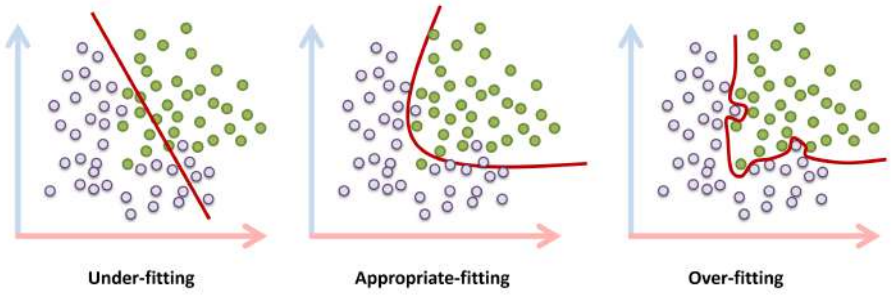


Figura 2.34: Situaciones producidas en el entrenamiento.

Dentro de las capas que se han utilizado en esta disertación para regularizar la red

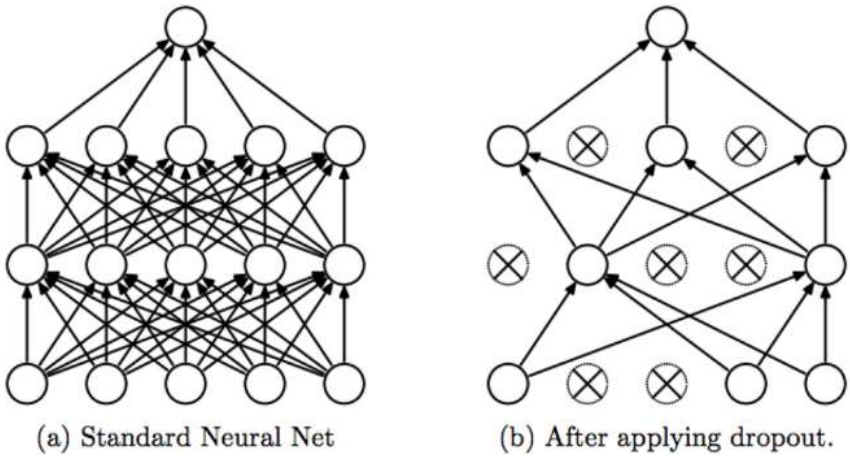


Figura 2.35: Diagrama esquemático del proceso de DropOut.

nos encontramos dos principalmente: las capas dropout tienen la peculiaridad de permitir que algunas unidades (neuronas) se ignoren durante la fase de entrenamiento y con ignorar nos referimos a no tenerlas en cuenta a la hora de computar el gradiente hacia atrás (en la etapa de *backpropagation*) que a la postre establecerá cual es la actualización que se aplicará a los pesos. Las neuronas seleccionadas para ser ignoradas son aleatorias, teniendo una probabilidad de ser seleccionada p y una probabilidad de ser no seleccionada (1-p) de entre las neuronas pertenecientes a la capa precedente a la capa dropout.

Las capas *dropout* fuerzan a la red a aprender características más robustas dentro de muchos y diferentes subconjuntos aleatorios de neuronas, y aunque el tiempo de convergencia en el entrenamiento en teoría por época se reduce, el número de iteraciones requeridos para converger se duplica.

Por otro lado, nos encontramos las capas de *batch-normalization* las cuales persiguen el objetivo de normalizar las activaciones de una capa previa a lo largo del mini-batch. Para ello la capa de *batch-normalization* calcula la media y la varianza de las características en un mini-batch y posteriormente, sustrae dicha media y divide las características por la desviación estándar en el mini-batch. Esto restringe las activaciones que pasan a tener media 0 y una desviación estándar de uno.

2.3.2.5 **Funciones de pérdida**

Una vez contamos con los componentes básicos para definir y estructurar una red, lo siguiente de lo que deberíamos ocuparnos es de entrenarla o ajustarla a los datos de entrada. Este ajuste o entrenamiento no es más que un proceso en el que la red a partir unos datos de entrada, comprueba como de bien se ajusta su salida en relación a lo que debería de salir y va cambiando sus pesos internos para hacerlo mejor cada vez. Este "hacerlo mejor cada vez" consiste en reducir la diferencia que hay entre lo que devuelve y lo que se supone que debería devolver y, así, el proceso de modificación de pesos se convierte así en un proceso de minimización de esa diferencia a la que empezaremos a designar con el nombre más técnico de función objetivo, que en un proceso de optimización se correspondería con la función a ser optimizada. En el caso concreto de un valor de entrada, esta función objetivo recibe el nombre de función de pérdida. Esta función de pérdida nos permite cuantificar cuánto difiere el resultado de una red en relación al resultado correcto. Es decir, con la función de pérdida se estaría evaluando como de bien funciona la red (la configuración de pesos de la red) para una única entrada, y es la función de coste la que resultaría de sumar las funciones de pérdida para varias muestras de elementos (*batch*) del conjunto de datos de entrenamiento, o para todo el conjunto de entrenamiento. Esto es, el coste sobre el *dataset* es una suma de las pérdidas sobre los ejemplos. Aunque la definición de estos conceptos depende mucho de cada autor y, en ocasiones, se consideran equivalentes e incluso se les da algún nombre adicional como función de error.

> *The function we want to minimize or maximize is called the objective function or criterion. When we are minimizing it, we may also call it the cost function, loss function, or error function.*
>
> Ian Goodfellow [Goodfellow et al.16] Page 82

A modo de resumen una función de pérdida es una parte de una función de coste que es un tipo de función objetivo a optimizar. Así si consideramos la red como una función f que pretende mapear, o encontrar la relación entre la entrada x_i y con la salida que le proporcionamos, la función de coste a minimizar sería el sumatorio de las diferentes funciones de pérdida (véase ecuación 2.19) más un término de regularización para evitar el *overfitting*, como puede verse en la ecuación 2.19.

$$\min_{f} \frac{1}{N} \sum_{i=1}^{N} L_\theta(y, f(x_i)) + \lambda R(f) \tag{2.19}$$

Donde N es el número de muestras del conjunto de entrada, y donde podemos adivinar dos términos separados por el símbolo de suma. Uno conocido como riesgo empírico que está formado por $L()$ que es la función de pérdida, θ el vector de parámetros, x_i los datos de entrada y $f(x_i)$ la respuesta del modelo y el otro, conocido como término de regularización, del que hablaremos más adelante, que representa la complejidad del modelo, y en el que λ se corresponde con equilibrio para balancear ambos términos.

Esta función de pérdida establecerá por tanto cómo de bien lo está haciendo el entrenamiento y no existe únicamente una función de pérdida (véase la figura 2.36), que suele depender del problema a resolver.

El término de regularización se utilizaría para penalizar el ajuste de modo que empeore ligeramente con el objetivo de evitar el overfitting y que el modelo no se ajuste a la información en tus datos que no represente sus propiedades reales. Hay diferentes funciones de regularización, entre las más conocidas: *Ridge Regression* o L2, *Lasso Regression* o L1 o *Elastic-Net Regression*.

Si imaginamos gráficamente la función de coste como una superficie irregular de valles y cumbres, en general, es preferible una superficie suave por el que el algoritmo de optimización puede desplazarse razonablemente mediante modificaciones iterativas de los pesos del modelo.

Esto nos llevaría al procedimiento de optimización [Lydia and Francis19] que definirá cómo modificar los parámetros para ir minimizando la función de coste que hayamos definido. La estrategia de optimización más popular utilizada en este proceso de minimización es el descenso del gradiente. Esta estrategia modifica iterativamente los parámetros de la red (sus pesos y bias) para minimizar la función de coste diferenciable a un mínimo local. Se comienza definiendo unos valores iniciales semi-aleatorias normalmente para dichos parámetros y desde allí iterativamente, se actualizan utilizando para ello el gradiente de la función.

Esta inicialización podría partir de una configuración inicial (de pesos y bias) aprendida con anterioridad, proceso que se conoce como fine-tuning. Concepto este que hay que diferenciar del transfer-learning, que es un proceso similar en el que antes

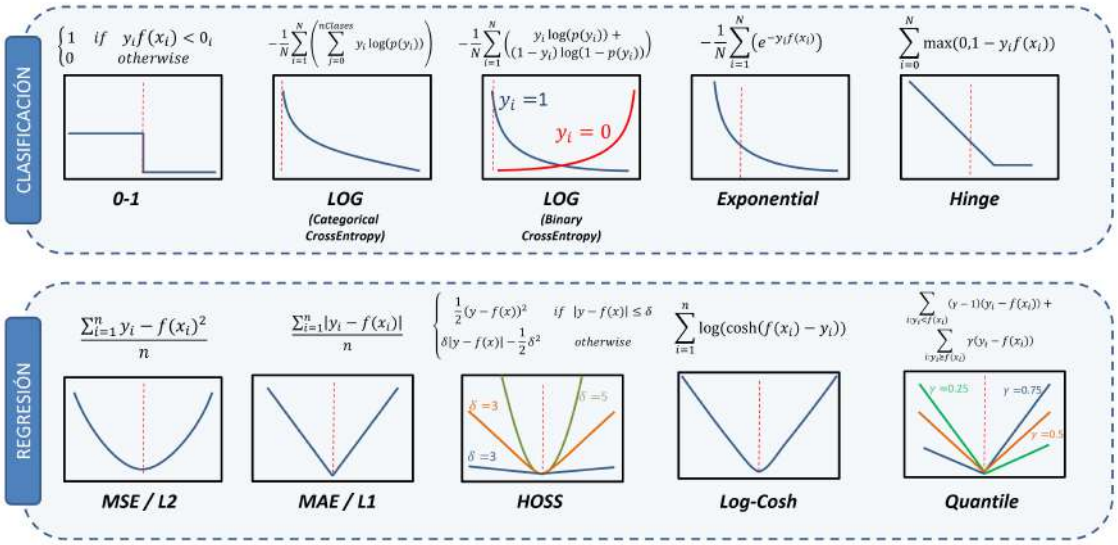


Figura 2.36: Funciones de coste más típicas en el dominio de la clasificación y la regresión.

de entrenar se copian los pesos aprendidos en entrenamientos previos a las capas respectivas del nuevo modelo, pero en la que habitualmente se mantienen esos pesos fijos y no se les permite modificarse o aprender (los mantiene congelados), permitiendo que se actualicen únicamente las últimas capas de la red destino que definirían el dominio objetivo al que se quiere aplicar el conocimiento adquirido en el entrenamiento del dominio de origen.

> *A gradient measures how much the output of a function changes if you change the inputs a little bit.*
>
> Lex Fridman (MIT)

El gradiente le indica al algoritmo el cambio en el error en relación al cambio en los pesos. En términos matemáticos se correspondería con la derivada parcial de la función de coste con respecto a los parámetros. La actualización se realizaría dando un paso en la dirección del gradiente, un paso definido por el *learning rate*. Para alcanzar el mínimo local hay que asignarle un valor apropiado a este *learning rate* (véase la figura 2.37). Si es muy grande podría comenzar a alejarse del mínimo y si es demasiado pequeño puede tardar mucho tiempo en alcanzarlo.

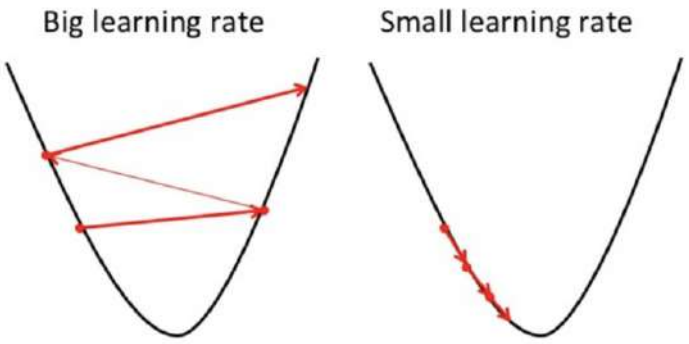


Figura 2.37: Elecciones incorrectas del *learning rate* en el descenso del gradiente.

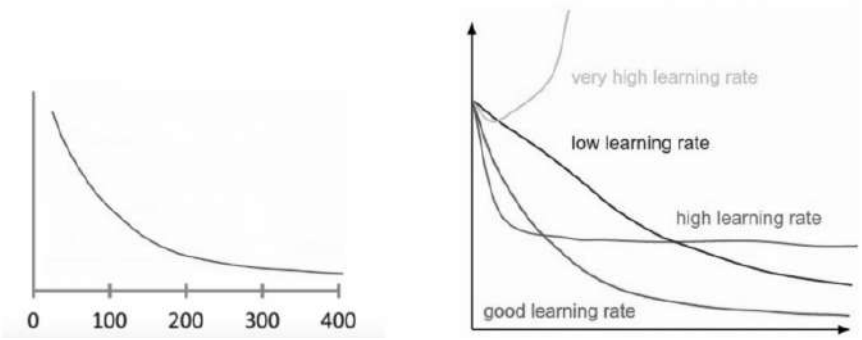


Figura 2.38: Evolución de la función de pérdida por épocas usando diferentes tasas de aprendizaje (*learning rates*).

Por supuesto, para poder avanzar en la dirección del gradiente, es un paso totalmente ineludible y determinante el cálculo de dicho gradiente. Esta tarea nada baladí es la parte que le corresponde al algoritmo de back-propagación, que es capaz de calcular este gradiente utilizando la regla de la cadena que en el ámbito del cálculo matemático se trata de una fórmula para obtener las derivadas de una función compuesta de una manera eficiente.

Para comprobar como de bueno es el *learning rate* una práctica habitual es comprobar la evolución del valor de la función de coste entre iteraciones. Dicha evolución nos puede proporcionar una intuición clara de si vamos por buen camino o no (véase la figura 2.38).

Existen diferentes tipos de descenso de gradiente: batch gradient descent (también llamado vanilla gradient descent), stochastic gradient descent o mini-batch gradient

descent. El primero calcula el error para cada muestra en el *dataset* de entrenamiento pero realiza la actualización cuando todas las muestras han sido evaluadas (en cada época). El estocástico actualiza los parámetros para cada muestra del conjunto de entrenamiento. Y por último, el mini-batch es una combinación entre los dos anteriores. Divide el conjunto de entrenamiento en subconjuntos (o *batches*) más pequeños y realiza la actualización tras evaluar cada uno de estos *batches*.

Y así iteración tras iteración, se va reduciendo el error ejecutando pasos hacia adelante para la estimación del error de un mini-batch de muestras, volviendo hacia atrás para recalcular el gradiente y actualizando los pesos para descender por la función de coste hacia un mínimo local en pasos definidos por el learning rate. Iterando de esta manera finalmente accedemos, como hemos dicho, a un mínimo local que puede ser suficientemente bueno para resolver el problema que estamos enfrentando.

2.3.2.6 **Modelos de detección y clasificación**

En el caso que nos ocupa, las redes neuronales que se han utilizado en el marco de esta disertación se engloban principalmente dentro de las redes de clasificación y detección de objetos. En estos campos, también los modelos han sufrido una evolución vertiginosa y así tenemos para el ámbito de la clasificación desde modelo AlexNet en 2012 al modelo MobileNetv3 en 2019 y, en lo referente al mundo de la detección, desde el modelo OverFeat en 2013 al modelo Yolov5 en 2020, figura 2.39.

En este trabajo se optó por utilizar un par de modelos para las tareas de clasificación. Por un lado el modelo AlexNet, un modelo sencillo, maduro y que lleva tiempo siendo utilizado en tareas de clasificación lo que le aportaba ciertas ventajas para los análisis realizados. Y, por otro lado, el modelo ResNet 101 para las tareas de clasificación del TSR desarrollado. Este era un modelo más complejo y que presentaba una mejor precisión en tareas de clasificación frente al AlexNet. Finalmente, para las tareas de detección de objetos, se ha utilizado el modelo yolo v3, un modelo que si bien no es el más actual, presentaba un compromiso adecuado entre coste computacional y precisión, convirtiendolo en el modelo idoneo para su aplicación en el proyecto.

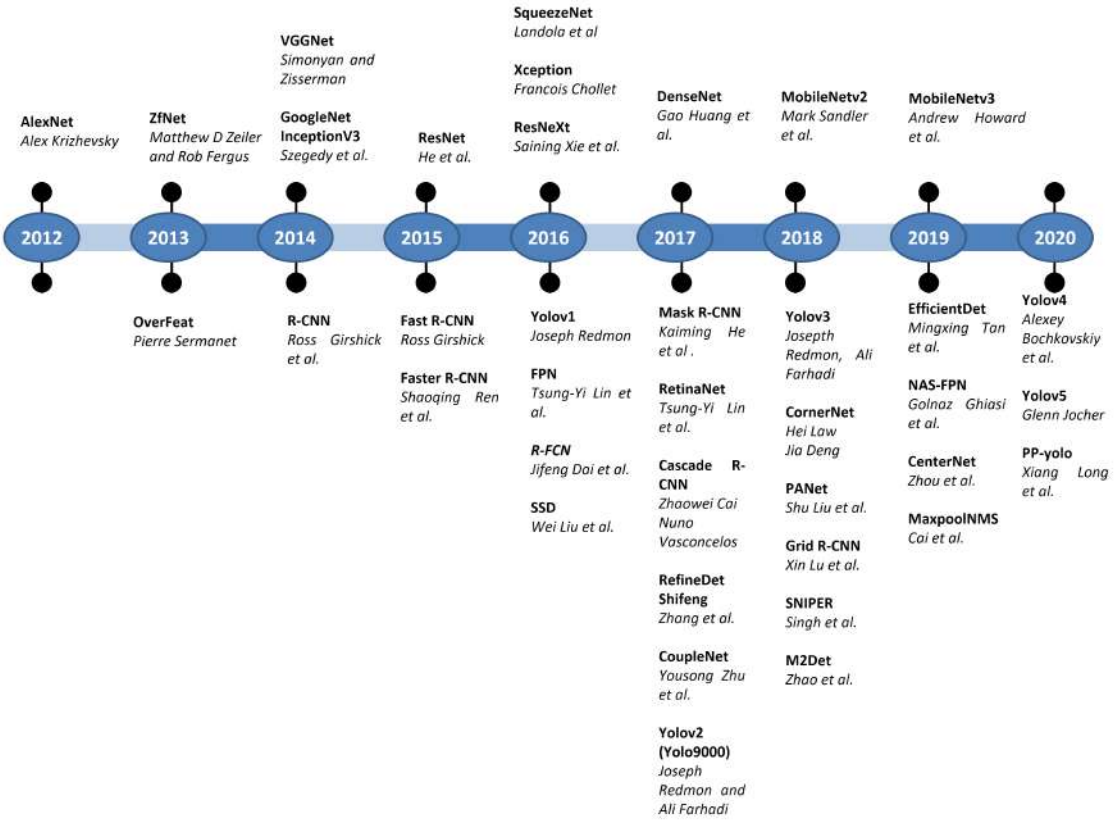


Figura 2.39: Parte superior: evolución de los modelos en el ámbito de la clasificación. Parte inferior: línea temporal de diferentes modelos en el ámbito de la detección.

2.3.2.7 **AlexNet**

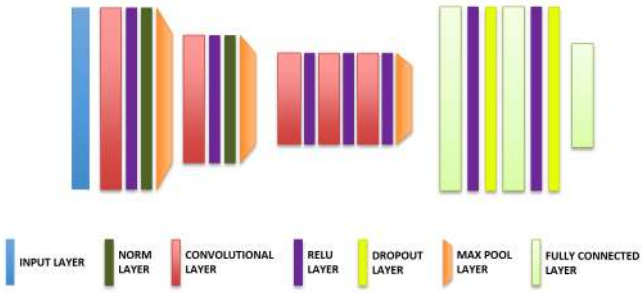


Figura 2.40: Arquitectura AlexNet.

AlexNet, en la figura 2.40, es un modelo de red neuronal convolucional que fue desarrollado por Alex Krizhevsky para el reto de ImageNet (ILSVRC), reto que ganó en 2012, donde consiguió una tasa de error del 15.3Este modelo se nutre de una capa de entrada de 227x227x3 y está compuesto de 5 capas convolucionales, 3 capas de Max pooling y 3 capas totalmente conectadas.

La función de activación de las capas convolucionales es una **ReLU** (Unidad Lineal Rectificada). Esto le permite realizar el entrenamiento más rápido que utilizando otro tipo de funciones.

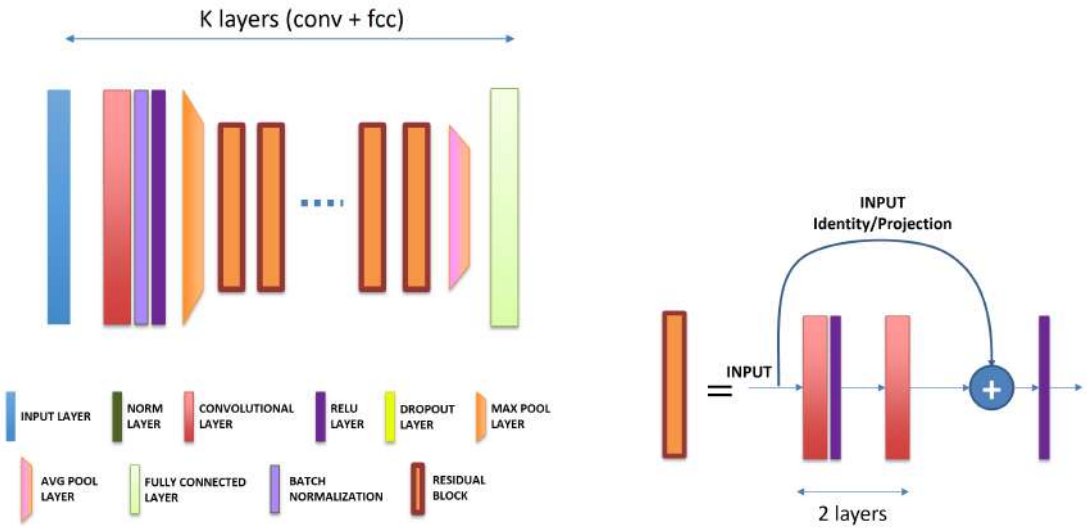
El número de operaciones por capa se puede observar en la tabla 2.1. El modelo AlexNet presenta 62 M de parámetros que hay que calcular para realizar el ajuste del modelo.

AlexNet introduce también capas de normalización de respuesta local y de dropout. Las capas de normalización estan pensadas para realizar una normalización en el vecindario del pixel amplificando la neurona excitada mientras atenuas las neuronas circundantes al mismo tiempo. Por otro lado, se intercalan ciertas capas **dropout** durante el entrenamiento para reducir el ***overfitting***. Estas capas invalidan aleatoriamente la salida de algunas de las neuronas de esa capa obligando al modelo a reajustarse nuevamente. Hay que tener en cuenta que aunque ayuda a la red a evitar situaciones de *over-fitting*, permitiéndole escapar de un mínimo local, el numero de iteraciones para la convergencia se duplica.

AlexNet Network (Details)								
Input	**Output**	**Capa**	**Stride**	**Pad**	**Kernel**	**in**	**out**	**# of Param**
227 x 227 x 3	55 x 55 x 96	conv1	4	0	11 x 11	3	96	34.944
55 x 55 x 96	27 x 27 x 96	maxpool1	2	0	3 x 3	96	96	0
27 x 27 x 96	27 x 27 x 256	conv2	1	2	5 x 5	96	256	614.656
27 x 27 x 256	13 x 13 x 256	maxpool2	2	0	3 x 3	256	256	0
13 x 13 x 256	13 x 13 x 384	conv3	1	1	3 x 3	256	384	885.120
13 x 13 x 384	13 x 13 x 384	conv4	1	1	3 x 3	384	384	1.327.488
13 x 13 x 384	13 x 13 x 256	conv5	1	1	3 x 3	384	256	884.992
13 x 13 x 256	6 x 6 x 256	maxpool5	2	0	3 x 3	256	256	0
		fc6			1 x 1	9216	4096	37.752.832
		fc7			1 x 1	4096	4096	16.781.312
		fc8			1 x 1	4096	1000	4.097.000
Total								**62.378.344**

Tabla 2.1: Parámetros del modelo Alexnet.

2.3.2.8 **ResNet**



(a) Modelo ResNet **(b)** Bloque residual

Figura 2.41: Modelo ResNet.

ResNet, representado en la figura 2.41, es un modelo de clasificación en el ámbito de las redes neuronales profundas que pretende resolver dos problemas que presentan o presentaban en su momento los modelos existentes en el momento de su aparición.

Por un lado, el hecho de que las redes neuronales no siempre devuelvan el modelo óptimo tras el aprendizaje. Por ejemplo, imaginemos una red que tiene como entrada una imagen y que queremos que devuelva en su salida la misma imagen. la solución más simple sería igualar todos los pesos a uno y todos los bias a cero para todos las capas ocultas. Pero cuando se entrena una red así, el algoritmo de propagación inversa y de descenso de gradiente acaban devolviendo pesos y bias en un rango de valores mayor.

Por otro lado, la intuición nos diría que si una red neuronal alcanza cierto grado de precisión, si aumentásemos el número de capas, esta precisión no debería empeorar. Es decir, que un modelo más profundo no debería producir más error que sus contrapartes más superficiales. Sin embargo, en [He et al.16] argumentan que esto no es así y que la precisión baja añadiendo nuevas capas a la red. Esto ocurre porque al propagar la derivada hacia atrás (en el proceso de backpropagación), a las capas iniciales llega un valor casi insignificante, lo que se conoce como desvanecimiento del gradiente.

ResNet para solventar estos problemas introduce dos nuevos tipos de conexiones: los atajos de identidad y los atajos de proyección.

Si las dimensiones de la salida del segundo layer de convolución son iguales a la entrada del bloque entonces se utiliza una conexión de identidad sumando a dicha salida la señal de entrada. Si son dimensiones diferentes se utiliza una conexión de proyección en la cual se cambia el tamaño de la entrada al tamaño de la salida del segundo layer de convolución. Este cambio se hace, habitualmente, de una de estas dos maneras: añadiendo padding a la señal de entrada o mediante un layer de convolución 1x1, ver tabla 2.2.

Como nota adicional mencionar que el número en la versión de ResNet indica el número de layers de la red (convolucionales y completamente conectados).

Como se puede observar en la tabla 2.3, el número de parámetros del ResNet está por debajo del AlexNet aunque computacionalmente el numero de operaciones en coma flotante es mayor.

ResNet18 Network (Details)									
#	**Input**	**Output**	**Capa**	**Stride**	**Pad**	**Kernel**	**in**	**out**	**# of Param**
1	227 x 227 x 3	112 x 112 x 64	conv1	2	1	7 x 7	3	64	9.472
	112 x 112 x 64	56 x 56 x 64	maxpool	2	0.5	3 x 3	64	64	0
2	56 x 56 x 64	56 x 56 x 64	conv2-1	1	1	3 x 3	64	64	36.928
3	56 x 56 x 64	56 x 56 x 64	conv2-2	1	1	3 x 3	64	64	36.928
4	56 x 56 x 64	56 x 56 x 64	conv2-3	1	1	3 x 3	64	64	36.928
5	56 x 56 x 64	56 x 56 x 64	conv2-4	1	1	3 x 3	64	64	36.928
6	56 x 56 x 64	28 x 28 x 128	conv3-1	2	0.5	3 x 3	64	128	73.856
7	28 x 28 x 128	28 x 28 x 128	conv3-2	1	1	3 x 3	128	128	147.584
8	28 x 28 x 128	28 x 28 x 128	conv3-3	1	1	3 x 3	128	128	147.584
9	28 x 28 x 128	28 x 28 x 128	conv3-4	1	1	3 x 3	128	128	147.584
10	28 x 28 x 128	14 x 14 x 256	conv4-1	2	0.5	3 x 3	128	256	295.168
11	14 x 14 x 256	14 x 14 x 256	conv4-2	1	1	3 x 3	256	256	590.080
12	14 x 14 x 256	14 x 14 x 256	conv4-3	1	1	3 x 3	256	256	590.080
13	14 x 14 x 256	14 x 14 x 256	conv4-4	1	1	3 x 3	256	256	590.080
14	14 x 14 x 256	7 x 7 x 512	conv5-1	2	0.5	3 x 3	256	512	1.180.160
15	7 x 7 x 512	7 x 7 x 512	conv5-2	1	1	3 x 3	512	512	2.359.808
16	7 x 7 x 512	7 x 7 x 512	conv5-3	1	1	3 x 3	512	512	2.359.808
17	7 x 7 x 512	7 x 7 x 512	conv5-4	1	1	3 x 3	512	512	2.359.808
	7 x 7 x 512	1 x 1 x 512	avg pool	7	0	7 x 7	512	512	0
18	1 x 1 x 512	1 x 1 x 1000	fc				512	1000	513.000
Total									**11.511.784**

Tabla 2.2: Parámetros del modelo ResNet.

Model	*Parameters*	FLOPS	Top5 accuracy*
ResNet-18	11 million	$1{,}8x10^9$	81,3 %
ResNet-34	21 million	$3{,}6x10^9$	84,1 %
ResNet-50	23 million	$3{,}8x10^9$	93,29 %
ResNet-101	42 million	$7{,}6x10^9$	93,95 %
AlexNet	62 million	$1{,}5x10^9$	84,70 %

Tabla 2.3: Comparación entre modelos.

*Top5 accuracy from [paperswithcode].

2.3.2.9 **Yolov3**

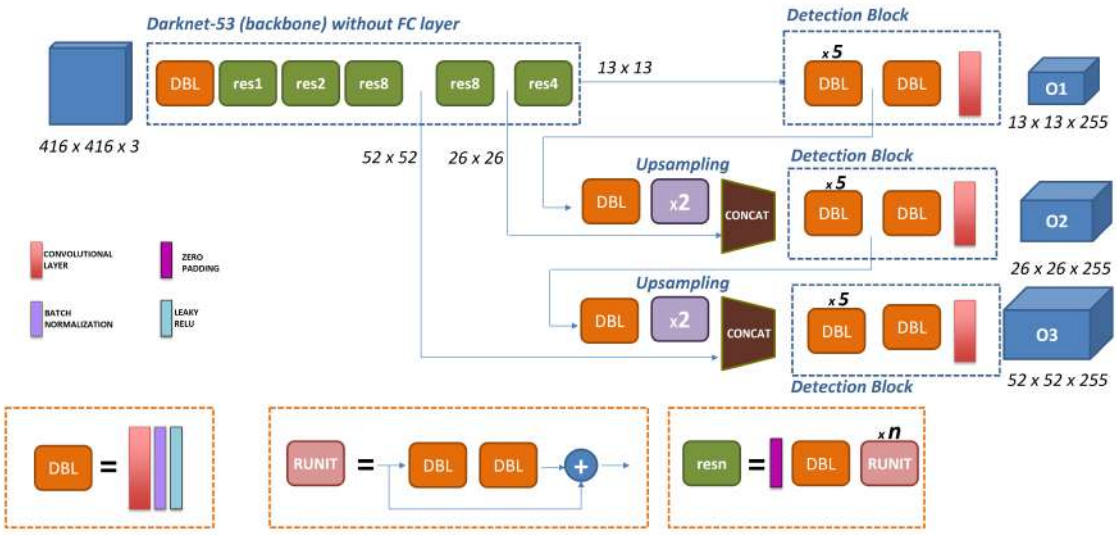


Figura 2.42: Red neuronal convolucional yolov3.

Yolo, figura 2.42, es un modelo de red convolucional diseñado para la detección de elementos en una imagen. Existen diferentes versiones de este modelo que han ido aportando mejoras a lo largo de la última década, en ocasiones estructurales y de eficiencia, en ocasiones de precisión de resultados.

Dentro de los modelos de detección basados en deep learning existen dos tendencias principales, una en la que se le proporciona al modelo las zonas que debe clasificar mediante un generador de zonas y una segunda en la que al modelo se le proporciona una serie de cajas de referencia (o anclajes) con cierta relación de aspecto y el propio modelo encuentra las detecciones mediante un proceso de regresión. Yolo se enmarcaría dentro de esta segunda tendencia de detectores.

En líneas generales, el esquema de funcionamiento de yolo se divide en dos etapas: una de extracción de características y otra de detección. Tras pasar por estas dos etapas el modelo retorna como salida un conjunto de cajas y de probabilidades de pertenencia a cada una de las clases para las diferentes cajas. Aunque esta salida se definirá con más detalle un poco más adelante.

Para la generación de características, yolo v3 usa como backbone (la red que se utiliza para la generación de características en un modelo) una variante de la red

convolucional Darknet-53 (véase la figura 2.43) que presenta 53 niveles entrenados en ImageNet. Esta versión de Darknet incluye el uso de conexiones residuales a diferencia de su predecesora.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Table 1. **Darknet-53.**

Figura 2.43: Red neuronal convolucional DarkNet-53.

De este *backbone* yolo extrae tres vectores de características de diferentes dimensiones. Es a partir de aquí cuando entramos en la fase de detección. Para una entrada de 416x416 los tres vectores serían de 13x13, 26x26 y 52x52. Esto es así porque Yolov3 realiza detecciones a tres escalas diferentes, característica esta que lo diferencia de sus predecesores. Cada uno de los vectores iría a su etapa de detección y además desde su etapa de detección generaría una salida que se concatenaría a la entrada del nivel de detección inmediatamente inferior (haciendo previamente un redimensionado). Esto convierte el número de capas de la red en 75 layers convolucionales (52 capas convolucionales de DarkNet-53 más las (15+8) capas convolucionales de los bloques de detección). Estas detecciones se ejecutan aplicando un kernel 1x1 en mapas de características de tres tamaños distintos en tres lugares diferentes de la red. Por este motivo, aunque se gana en precisión a la hora de detectar

objetos pequeños, el coste computacional de este modelo es algo mayor que el de su predecesor.

Cada uno de los vectores de características de cada nivel de detección define una matriz de celdas que si los aplicamos sobre la resolución original dividen la imagen en regiones. Y por cada una de esas celdas se obtiene una salida tras la detección, estructurada de la siguiente manera (figura 2.44):

1. Desviación de la localización de la caja de anclaje: bb_x, bb_y, bb_w, bb_h, desplazamiento de los centros (en x e y, relativos al centro del a celda) y tamaños de caja (ancho y alto, relativos a las dimensiones del anclaje)

2. Probabilidad de que la celda contenga un objeto, lo que implica que el centroide del *bounding box* del objeto cae en dicha celda. Esto define el valor de confianza de presencia de objeto. Este nivel de confianza de presencia del objeto en la celda es un valor entre 0 y 1.

3. Para cada clase, la probabilidad de que esa caja pertenezca a esa clase (entre 0 y 1)

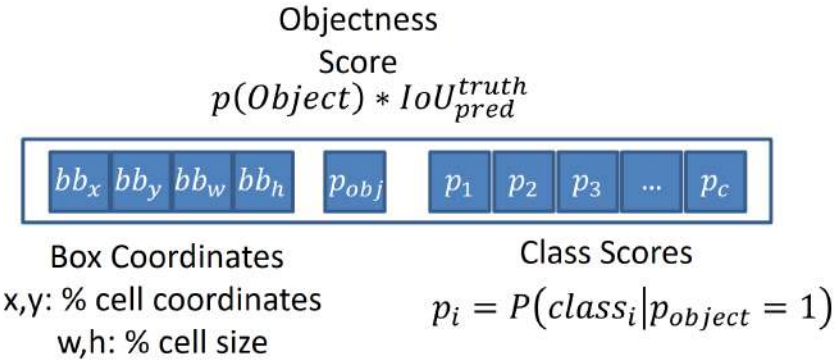


Figura 2.44: Salida del modelo para un anclaje Yolo.

Sin embargo, para facilitar la convergencia Yolov3 utiliza un recurso denominado cajas ancla o *anchor boxes*. Así, a la hora de buscar la caja que mejor se adapta a una caja etiquetada en el groundtruth, se partiría del *anchor box* que mejor IoU presenta con la caja del groundtruth y se ajustaría mediante la aplicación de sutiles cambios en sus dimensiones y posición. La caja del ground truth debería parecerse al anclaje

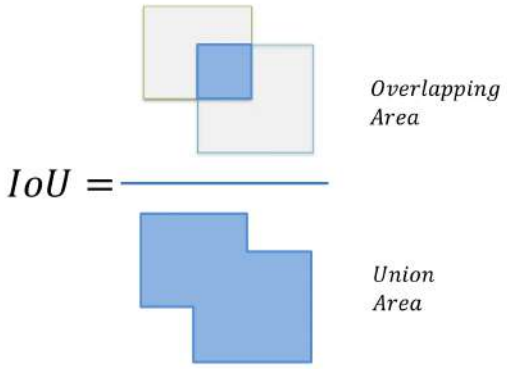


Figura 2.45: Representación visual del operador intersección sobre unión.

seleccionado, lo que derivaría en un proceso de convergencia rápido. En total se usan tres *anchor boxes* por cada celda de cada nivel de detección. Estas cajas definidas a priori tendrán diferentes relaciones de aspecto. La relación de aspecto de estas *anchor boxes* se obtienen de aplicar un algoritmo de clusterización K-Means a las cajas etiquetadas en la base de datos de entrenamiento para los diferentes elementos a detectar.

Para poder detectar diversos objetos en la misma celda se puede disminuir el tamaño de la celda, aún así dos objetos podrían caer en la misma celda, es por ello que para cada celda la salida es de $(5+C)K$, donde K es el número de *anchor boxes*, de modo que los objetos pueden caer en diferentes anclas dentro de la misma celda. En el caso de detectar el mismo objeto múltiples veces se aplica un proceso adicional después de la detección, de supresión de no-máximos, que elimina las detecciones con baja probabilidad que están muy cercanas (IoU>threshold, figura 2.45) a detecciones de la misma clase con probabilidad mayor.

El tamaño total de la salida de un nivel de detección comprenderá un vector como el descrito en la figura 2.46 para cada celda y dentro de cada celda, para cada anclaje. Esto es, continuando con el ejemplo iniciado anteriormente, para el nivel de detección que parte de una entrada de 13x13, la salida será $13x13xKx((4+1)+C)$ siendo K el número de *anchor boxes* para las celdas de ese nivel y siendo C el número de clases que se quieren detectar.

El total de la salida conjunta del modelo Yolov3 será la suma de los tres niveles de salida. Suponiendo una imagen de entrada de 416 x 416 con 3 canales, la salida del

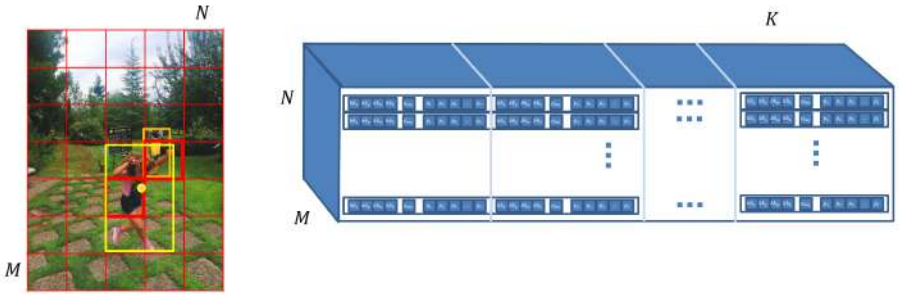


Figura 2.46: Salida del modelo Yolo para un nivel de detección.

detector será un vector de 3 matrices de $[52x52x3x(4+1+num\_classes),(26x26x3x(4+1+num\_classes)),(13x13x3x(4+1+num\_classes))]$ que se corresponderían con las detecciones en las tres escalas con 3 anclajes por escala y por celda.

Una vez realizada la detección se puede calcular la función de pérdida contra las etiquetas del *groundtruth* y de este modo ir iterando en busca de un error menor en la función de coste. Esta función de coste se calcula de la siguiente manera:

$$\begin{aligned}
&\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}\left[\left(x_i-x_i^{'}\right)^2+\left(y_i-y_i^{'}\right)^2\right]\\
+\lambda_{coord}&\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}\left[\left(\sqrt{w_i}-\sqrt{w_i^{'}}\right)^2+\left(\sqrt{h_i}-\sqrt{h_i^{'}}\right)^2\right]\\
&+\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}(C_i-C_i^{'})^2\\
&+\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{noobj}(C_i-C_i^{'})^2\\
+\sum_{i=0}^{S^2}1_{i}^{noobj}&\sum_{c\in classes}(p_i(c)-p_i^{'}(c))^2
\end{aligned}\tag{2.20}$$

Como se puede observar, la función de coste (véase la figura 2.20) se divide en varias partes:

1. Perdida del centroide (x,y) : función de pérdida para el centroide del *bounding*

box. Cuanto menor es más cerca están los centroides de la predicción y del *groundtruth*. Como se trata de una regresión se utiliza el error medio cuadrático.

2. Perdida del ancho y el alto (w,h)

3. Perdida de probabilidad de objeto: indica como de probable es que haya un objeto en la celda actual. Se utiliza *binary cross entropy* como función de coste. Esta probabilidad es 1 para la celda que contiene un objeto y 0 para la celda que no.

4. Perdida de clasificación: se aplica *binary cross-entropy* para cada clase y se suman ya que no son mutuamente excluyentes, puede haber más de una clase que de 1.

La calidad de una predición de caja se mide por su IoU con el objeto que trata de predecir (el groundtruth de la caja del objeto) y va de 0 a 1. Para cada celda espacial y para cada predicción de caja centrada en esa celda, la función de pérdida encuentra la caja con el mejor IoU con el objeto centrado en esa celda. Este mecanismo de distinción entre las mejores cajas y el resto yace en el nucleo de la función de pérdida de yolo. Las mejores cajas y únicamente ellas incurren en la función de pérdida de coordenadas y en la función de pérdida de clasificación. Esto empuja a los parámetros de la red asociados con esas cajas, a mejorar la escala de la caja y su localización, así como la clasificación. El resto de cajas únicamente incurren en la pérdida de confianza.

El término de función de pérdida de probabilidad de objeto le enseña a la red a predecir un IoU correcto, mientras que la función de pérdida de coordenadas le enseña a la red a predecir una mejor caja. Todas las predicciones de caja contribuyen a la función de perdida de probabilidad de objeto pero solo las cajas con mejor ajuste en cada celda espacial contribuyen también a las pérdidas de coordenadas y de la función de pérdida de clasificación.

The only way to make roads safer is removing the human factor from the equation.

CAPÍTULO

3

Conteo y Reconocimiento de Vehículos

3.1 Contexto

Dentro del ámbito del transporte, un factor importante para que la conducción sea segura, es mantener el buen estado de las carreteras. Para ello los diferentes gobiernos suelen dedicar una partida del presupuesto general del estado a este cometido. Las distintas entidades que se dedican al mantenimiento de estas infraestructuras en ocasiones se ven en la necesidad de justificar los costes que suponen las reparaciones y mejoras. Para ello un buen indicador suele ser el uso que se realiza de la vía de transporte. Se puede cuantificar ese uso analizando de manera más o menos pormenorizada la cantidad de vehículos que transitan la carretera por unidad de tiempo y su tipología. Además, este mismo sistema se puede extender a la aplicación de sanciones con fines punitivos, como por ejemplo, por exceso de velocidad o al cobro free-flow de peajes. Este tipo de cobro no obliga al vehículo a detenerse en un puesto de peaje para pagar sino que a partir de la información relacionada con el vehículo, recogida por los algoritmos y, del cruce de esta información con bases de datos de vehículos de la DGT, es posible realizar el cobro automático repercutiéndolo a una cuenta bancaria especificada con anterioridad y asociada con el vehículo, evitando

colas y demoras innecesarias generadas por los sistemas de peaje actuales en las carreteras. Es en este contexto en el que nos embarcamos en una colaboración con una empresa vasca para el desarrollo de una solución que permitiese realizar el conteo y clasificación de los vehículos que atravesaban un tramo de vía concreta, utilizando técnicas de bajo coste e instalación poco intrusiva en la propia vía de transporte.

La solución, llamada Eagle TD, tenía como objetivo una gestión más fluida del tráfico para prevenir accidentes, detectar congestiones, vehículos en sentido contrario, intensidad de la circulación... recogiendo la información obtenida tras el análisis de las imágenes proporcionadas por una serie de cámaras, generaba estadísticos útiles que se interpretaban y permitían una toma de decisiones adecuada. Este sistema se desarrolló a partir del proyecto Intelvia subvencionado por el plan Avanza del Ministerio de Industria y el Fondo Europeo de Desarrollo Regional, del que ya se ha hablado brevemente en la introducción.

En aquel momento, las aproximaciones más sofisticadas basadas en visión para vigilancia del flujo de tráfico combinaban información proveniente de cámaras con otras tecnologías tales como tags instaladas en vehículos, escáneres laser que reconstruyen la forma 3D de los vehículos, o GPS para estimar la dirección de las sombras que arrojan [NREL00]. Sin embargo, la mayoría de vehículos no llevaban incorporados tags y estos podían ser hackeados, los escáneres laser incrementaban, y a día de hoy incrementan, el coste de los sistemas y son sensibles a condiciones meteorológicas, análogamente al GPS, cuya complejidad de calibración hace que la obtención de resultados satisfactorios sea más costosa de obtener. Existían diferentes motivos por los que las tecnologías basadas en visión por computador podían resultar interesantes para los operadores de carreteras en general y para estas aplicaciones de peaje en particular. Por mencionar algunos de ellos: la visión por computador proporciona información contextual presente en el escenario (color, luces, número de matrícula) además de geometría (volumen del vehículo), costes reducidos de hardware, es levemente intrusiva, permite sistemas en tiempo real debido a la mejora computacional del hardware, y habría que añadir a esto que muchos operadores ya tienen instalaciones de cámaras previas a cualquier aplicativo de visión artificial con meros propósitos de supervisión humana.

Existen un gran número de trabajos en la literatura relacionados con la clasificación de vehículos usando visión por computador y a día de hoy la irrupción del *deep*

learning ha aumentado de una manera significativa este número de trabajos.

3.2 Descripción del problema

El problema que se pretendía resolver con este desarrollo se enfocaba principalmente en la estimación de la ocupación de la carretera, usando para ello un sistema instalado en un pórtico y compuesto de un ordenador a pie de pórtico y una cámara instalada en su parte superior. También interesaba además de la ocupación, la tipología de los vehículos que transitaban dicha vía, su velocidad y el sentido de su desplazamiento. A nivel tecnológico eran numerosos los retos que había que atacar, habida cuenta de que se trataba de un sistema basado en visión artificial. Los principales retos se encontraban en el campo del seguimiento, la detección y la clasificación de objetos. Otro de los principales escollos encontrados lo establecía uno de los criterios de funcionamiento, ya que como todo el proceso se realizaba en la propia infraestructura y debido a que la conexión no era demasiado buena por motivos de cobertura y tecnología de comunicación inalámbrica, había que procesar en tiempo real la cantidad ingente de datos que generaba la captura. Hay que recordar que para evitar que un objeto pasase por la zona sin ser capturado por el sistema se requería una tasa alta de imágenes por segundo. Para que el lector se haga una idea, para calcular el framerate necesario podemos utilizar la siguiente ecuación:

$$fps = \frac{10000 v_t}{36 d_m} \tag{3.1}$$

Donde v_t es la velocidad en km/h a la que circulará el vehículo más rápido y d_m se correspondería con la distancia máxima que le queremos permitir avanzar antes de realizar otra captura en milímetros. Si por ejemplo, la cámara abarca una zona de 10 metros de ancho por 20 de largo, si un coche viniese lo suficientemente rápido como para avanzar esos 20m de largo entre capturas, evitaría someterse al análisis del algoritmo. Siendo esto así, si le permitimos a un vehículo a 120 km/h avanzar 1 metro entre capturas, el framerate debería de ser de 33 frames por segundo, que se considera tiempo real. Esto limitaría el tiempo de proceso a 30 milisegundos por frame.

El sistema también tenía que hacer frente a las diferentes condiciones lumínicas y atmosféricas que se produjesen. Este es un problema común y bien conocido en la

mayoría de los desarrollos basados en visión artificial en exteriores. Cuanto mayor es el abanico de situaciones que un sistema es capaz de registrar mayor complejidad entrañan los algoritmos aplicados y un mayor nivel de desafío plantea el problema.

Un problema derivado de esto, particularmente importante, fue discriminar las sombras arrojadas por los vehículos en días soleados.

Así pues, diseñamos un sistema que apostaba por una aproximación tradicional para resolver el problema de la detección de vehículos de una manera robusta y precisa. También era capaz de operar en tiempo real, y funcionar en escenarios realmente desafiantes (cámaras de bajo coste, iluminación pobre y presencia de sombras, perspectiva desconocida. La figura 3.1 muestra una imagen de una instalación real donde el sistema está funcionando actualmente).Finalmente, el sistema inferiría una clasificación 3D del vehículo asignándole una categoría usando las secuencias de imágenes proporcionadas por la cámara.

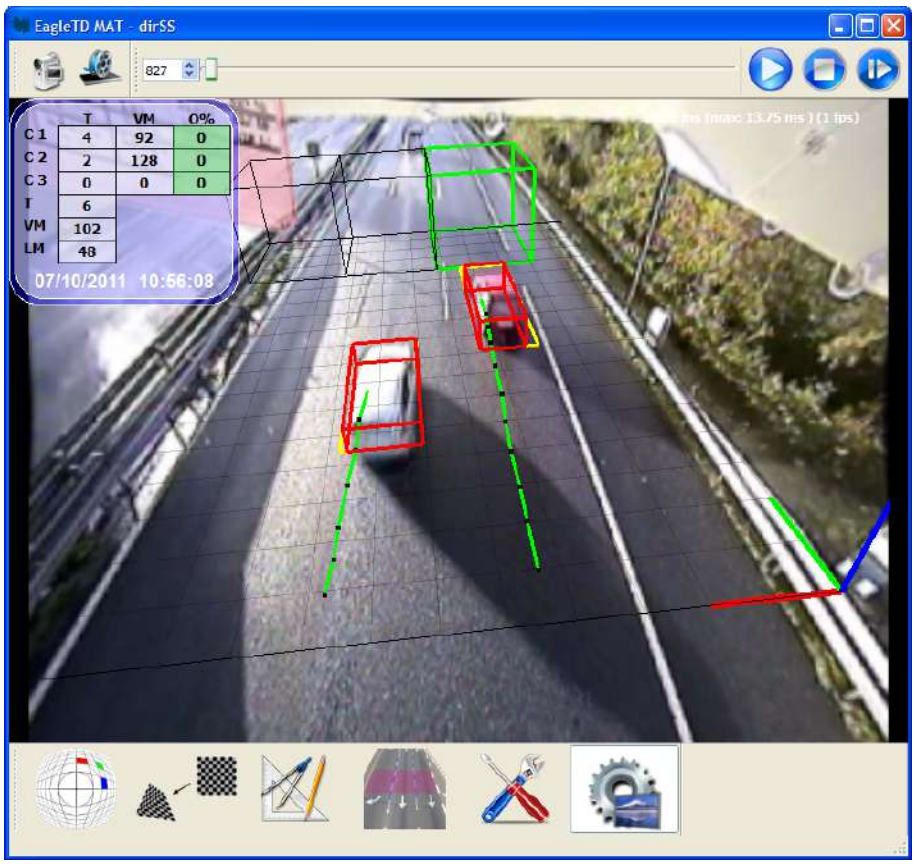


Figura 3.1: Vista desde un pórtico de la solución EagleTD.

La novedad de nuestra aproximación radicaba en la utilización de un procedimiento de sustracción de fondo multi-pista en el que los umbrales de segmentación podían adaptarse de manera robusta a cambios de iluminación, manteniendo un alto nivel de sensibilidad a nuevos objetos pertenecientes al primer-plano, eliminando además de manera efectiva las sombras proyectadas y los reflejos de las luces en la carretera, y en situaciones de congestión de tráfico, a diferencia de las aproximaciones existentes que no cubrían todas estas estas funcionalidades al mismo tiempo [Mayo and Tapamo09]. Un módulo de seguimiento proporcionaba la coherencia espacial y temporal necesaria para la clasificación de vehículos, que primero generaba estimaciones 2D de la silueta de los vehículos, y luego aumentaba las observaciones a volúmenes 3d por medio de un método basado en Cadenas de Markov Monte Carlo (MCMC). Su ventaja sobre aproximaciones previas, tales como [Pang et al.07, Buch et al.10, Haag and Nagel00, Johansson et al.09], era que podía directamente aplicarse sobre tracks 2D existentes para inferir la dimensión perdida debido a la proyección de la cámara. Lidiar con oclusiones severas estaba fuera del ámbito de este trabajo, pero nuestra aproximación podía ser utilizado en práctica aplicándolo a puntos de vista diferentes de la carretera, tales como esos que se consiguen de cámaras instaladas en pórticos de autopista, con un rendimiento significativamente mejor que las aproximaciones que en aquel entonces se encontraban en el estado del arte.

Para probar el algoritmo se realizó una batería de pruebas que se analizan posteriormente, con escenarios que presentan variaciones en las condiciones meteorológicas, en días soleados con sombras direccionales en movimiento, reflejos en la carretera de los focos del coche, días de lluvia y situaciones de congestión de tráfico. Obtuvimos resultados del conteo de vehículos y clasificación comparables a los de los sistemas ILD, que eran los sistemas más usados para estos tipos de mediciones de tráfico, mientras manteníamos las principales ventajas de los sistemas basados en visión, es decir, no requerir complicadas operaciones de instalación de equipo en la propia carretera o la necesidad de tecnología adicional como escáneres laser, tags o GPS.

3.3 Estado del Arte

Las contribuciones científicas principales de este trabajo en el campo de la visión por computador se encontraban en la sustracción de fondo. Las siguientes secciones revisan los trabajos relacionados con esta técnica de visión artificial.

Sustracción de Fondo

Una de las técnicas principales que se utilizó en el desarrollo del sistema de conteo y clasificación fue la sustracción de fondo, cuya salida alimentaba un sistema de seguimiento y posteriormente de estimación de volumen tridimensional. Las técnicas de sustracción de fondo tratan de modelar la variación de intensidad de los píxeles de fondo para que, de esta manera, la imagen se pueda segmentar en dos grupos de píxeles, los pertencientes al fondo y los pertenecientes al primer-plano.

La sustracción de fondo es una técnica ampliamente utilizada y conocida, sin embargo presenta una serie de problemas de no fácil solución. Entre los más conocidos están:

1. **Efecto camuflaje**: se produce cuando el objeto de interés es del mismo color que el fondo. En ese caso se habrá aprendido y no se mostrará como elemento de primer-plano.

2. **artefactos de compresión**: la presencia de los efectos producidos por la compresión de las imágenes pueden generar distorsiones en los valores de intensidad que presentan los píxeles aún sin producirse cambios en el escenario. Esto produciría ruido a la hora de separar el fondo del primer-plano. Se puede solventar parcialmente si se usa un modelo multimodal capaz de representar diferentes estados del fondo aunque, sin duda, la mejor manera de resolverlo es utilizar imágenes de buena calidad.

3. **Captura ruidosa**: La relación señal ruido que presente la imagen puede ocasionar la aparición de un moteado de píxeles de primer-plano debido al ruido que no se consigue incorporar al modelo de fondo. Un preprocesado de la imagen, para normalizarla, podría atenuar este efecto. El ruido también pueden producirlo diversas condiciones meteorológicas que pueden afectar

sensiblemente la precisión del modelo de fondo generado, como la lluvia o la niebla.

4. **Cambios de iluminación**: ocurre que al producirse cambios de iluminación producidos por elementos del escenario o por las condiciones climatológicas , tanto del área analizada en general (soleado-nublado), como de zonas más localizadas (focos de vehículos), este cambio puede no ser asumido por el modelo de fondo haciendo aparecer esas zonas como primer-plano. Se puede atenuar este efecto utilizando un espacio de color que permita un modelado de los cambios de iluminación independiente del matiz del color como HSV, HSI, CieLab, IHLS.

5. **Fondo dinámico**: Se considera un fondo dinámico aquel que presenta movimiento y cambia con el tiempo de manera reiterada, por ejemplo, el movimiento de las ramas de un árbol agitado por el viento. Este efecto se puede resolver utilizando un modelo multimodal para el fondo, de tal manera, que capture los diferentes estados en los que incurre el fondo a lo largo del tiempo.

6. **Fantasmas**: Este tipo de efectos se producen cuando al iniciar el modelo de fondo hay presente algún objeto perteneciente al primer-plano que se aprende o cuando un objeto se detiene lo suficiente y acaba incorporándose al modelo de fondo, de tal forma que al volver a moverse dicho objeto aparece una zona de primer-plano incorrecta formada por los píxeles que el desplazamiento de dicho objeto revela. Este problema no es trivial y se suele enfrentar incorporando inteligencia adicional a la política de actualización del modelo de fondo.

7. **Sombras**: Las sombras que proyectan los elementos de la escena llegan a generar un nivel de iluminación muy bajo en los píxeles que se corresponden con la zona de la sombra proyectada. Estos píxeles oscurecidos pueden generar una región de primer-plano y con ello una alteración de la región del elemento detectado. Habitualmente se trata de minimizar este efecto recurriendo a técnicas heurísticas y metodologías complejas que aporten más información sobre las diferentes regiones. De esta manera se pueden intentar descartar las regiones de sombra en relación por ejemplo a la presencia de un elemento en una zona anexa, o mediante procesos que traten de atenuar o eliminar dichas sombras.

Esto por contrapartida, puede dar lugar a la detección como sombra de un elemento con un nivel de intensidad bajo, un objeto negro, por ejemplo.

8. **Actualización prematura**: Otro de los puntos que hay que valorar y analizar es la actualización del fondo ya que el propio fondo puede ir cambiando con el tiempo. Para ello la política de actualización debe de evitar la incorporación de elementos de primer-plano cuando estos dejan de moverse. Esto se podría atenuar alimentando el modelo fondo con las detecciones de los elementos identificados hasta el momento, de tal modo que esas regiones desactiven la actualización, políticas nada triviales para resolver un problema complejo. Esto nos puede llevar a una situación conocida como *deadlock*, producida cuando un objeto se para, el modelo de fondo lo incorpora y cuando empieza a moverse se genera un fantasma, repitiéndose esta secuencia en el tiempo y generando muestras espurias a lo largo de la trayectoria de dicho objeto.

9. **Tiempo real**: Este proceso de segmentación del fondo puede resultar en ocasiones pesado y costoso y poco adecuado para soluciones en las que se requiere una respuesta en tiempo real. Por ello, este es otro de los puntos a tener en cuenta al desarrollar soluciones que utilicen la sustracción de fondo.

10. **Escenarios nocturnos**: La imagen recibida durante la noche puede variar considerablemente con respecto a la imagen diurna. Esto hay que reflejarlo en el modelo de fondo, o bien modelando dos fondos diferentes o bien utilizando un modelo multimodal que sea capaz de establecer rangos adecuados para el día y para la noche.

Cuando nos embarcamos en el proyecto ya existían multitud de métodos en la literatura científica que trataban de resolver estos problemas [Bouwmans14] [Benezeth et al.10] [Parks and Fels08], algunos de manera más específica , algunos atacándolos todos manera general, a los que se han añadido muchos otros hasta nuestros días [Garcia-Garcia et al.20] [Akilan et al.18]. La gran proliferación de algoritmos se debía también en parte a que se trataba de una técnica presente durante mucho tiempo en la disciplina de la visión por computador. En aquel momento ya estaban presentes métodos que modelaban la variación de los valores de intensidad de los píxeles del fondo utilizando distribuciones unimodales [Wren et al.97]

[Horprasert et al.99], o que los modelaban usando métodos multimodales como las mezclas de gaussianas [Stauffer and Grimson99, Hu et al.06], algunos lo hacían con kernels de densidad no-paramétrico [Elgammal et al.00] o había quienes lo basaban en particiones dentro del espacio de color que se utilizara como los codebooks [T. Chalidabhongse03] [Kim et al.05]. Sin embargo, en estas últimas temporadas y como no podía ser de otra forma, han comenzado a proliferar los métodos de segmentación de fondo basados en redes neuronales [Tezcan et al.20].

Los modelos unimodales eran rápidos y simples pero no eran capaces de adaptarse a múltiples fondos, por ejemplo, cuando en el fondo se podían observar árboles movidos por el viento. Las aproximaciones basadas en mezclas de gaussianas podían lidiar con estos movimientos en el fondo, pero no podían manejar variaciones rápidas con precisión usando unas pocas gaussianas, y por lo tanto estos métodos tenían problemas para una detección precisa de regiones de primer-plano. La estimación de kernels de densidad no paramétrico descrita en [Elgammal et al.00] permitía una rápida adaptación a los cambios de fondo. Sin embargo, los métodos basados en codebook eran los métodos de sustracción de fondo basados en color más sensibles, aplicados tanto a interior como a escenarios exteriores, incluso con algo de movimiento en el fondo [T. Chalidabhongse03]. La aproximación codebook, a diferencia de otros métodos mencionados, explícitamente modelaba los cambios de iluminación de los píxeles.

Hay que tener en cuenta que, en exteriores, era necesario actualizar el modelo de fondo de una manera rápida y efectiva para adaptarlo a las nuevas condiciones de la escena (iluminación, condiciones atmosféricas, sombras). Las aproximaciones más sofisticadas eran aquellas que aprendían el fondo de manera selectiva en función de los píxeles catalogados como vehículos potenciales y también en función del flujo óptico [Cucchiara et al.00]. En [Gupte et al.02] los vehículos se clasificaban de acuerdo a la altura y anchura del blob extraído, pero sin tener en cuenta la supresión de las sombras proyectadas. Alternativamente, otras aproximaciones usaban pistas de gradiente en lugar de valores de intensidad, mejorando la robustez frente a cambios de iluminación [Aubert et al.04]. Sin embargo, las regiones planas que podrían estar presentes en algunos vehículos no se extraían y necesitaban todavía un procesado adicional para eliminar las sombras. La mayoría de estos trabajos usaban los modelos de sombra descritos en [Mech and Ostermann99]. Este modelo distinguía entre penumbra y

umbra. Umbra es la sombra que recibe iluminación que viene únicamente de una luz de ambiente difusa, mientras que penumbra recibe iluminación de luz ambiente difusa y parte de luz directa.

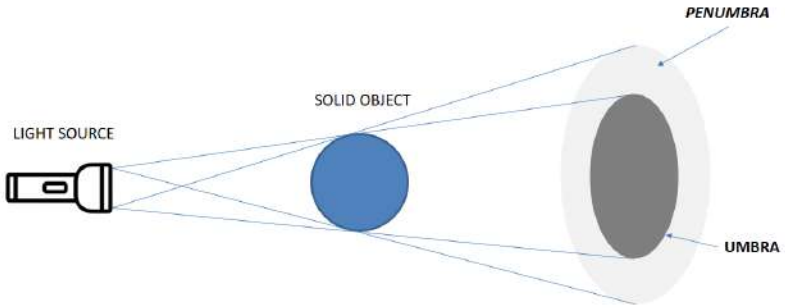


Figura 3.2: Descripción de umbra y penumbra.

Por lo tanto, la penumbra tiene más similaridad cromática con respecto a su color original que el caso de la umbra. De acuerdo a la taxonomía propuesta en [Prati et al.03], los métodos de supresión de sombras podían ser clasificados en deterministas y estadísticos. Los primeros usaban procesos de decisión on/off, mientras que los últimos usaban funciones probabilísticas que definían diferentes clases. Los métodos deterministas podían subdividirse a su vez en basados en modelos y no basados en modelos. Los basados en modelos usaban modelos explícitos de vehículos y fuentes lumínicas para realizar el seguimiento [Roller et al.93], mientras que los no basados en modelos, no [Cucchiara et al.01]. Por otro lado, los modelos estadísticos podían subdividirse en paramétricos y no paramétricos. Las aproximaciones paramétricas usaban una serie de parámetros que determinaban las características de las funciones estadísticas del modelo [Mikic et al.00], mientras que los no paramétricos automatizaban la selección de los parámetros del modelo como una función de los datos observados durante el entrenamiento [Horprasert et al.99]. De acuerdo a [Prati et al.03], las técnicas deterministas basadas en modelos podían obtener mejores resultados en la eliminación de sombras, pero hay que apuntar que

estos eran tremendamente complejos para una implementación práctica en vigilancia de tráfico en exteriores. Por lo tanto, las aproximaciones deterministas no basadas en modelos eran más adecuadas para aplicaciones en exteriores, mientras que los métodos estadísticos no paramétricos son mejores para interiores, ya que la escena era más constante y así su descripción estadística era más efectiva. En [Prati et al.03] también se manifestaba que el espacio de color HSV podía distinguir sombras con mayor precisión, ya que estas no cambiaban significativamente el tono de color y tendían a disminuir la saturación en el caso de penumbra aunque no en umbra. El sistema Sakbot [Cucchiara et al.01] usaba todas estas conclusiones para el conteo de vehículos desde vídeos. En [Blauensteiner et al.06] se mostraba que el espacio de color IHLS estaba mejor adaptado para detección de cambios y supresión de sombras que el HSV y el RGB normalizado, ya que permitía manejar el problema del canal hue inestable en colores débilmente saturados.

Sin embargo, en aquel momento surgieron métodos tanto para sustracción de fondo como supresión de sombras que mezclaban múltiples pistas, tales como bordes y color, para obtener segmentaciones más precisas. Por ejemplo, [Huerta et al.09] aplicaban reglas heurísticas combinando un modelo cónico de brillo y cromacidad en el espacio de color RGB junto con sustracción de fondo basado en bordes, obteniendo mejores resultados de segmentación que otras aproximaciones previas del estado del arte. También apuntaban que añadir modelos de vehículos de nivel más alto podía permitir obtener mejores resultados ya que ayudaba con situaciones de segmentación incorrecta. Esto es lo que hicieron en [Johansson et al.09], donde el tamaño, la posición y la orientación de un caja 3D alrededor del vehículo, que incluía la simulación de sombra obtenida de la información del GPS, se optimizaba con respecto a las imágenes segmentadas. Además, se mostraba en algunos ejemplos que esta aproximación podía mejorar el rendimiento comparado al uso aislado de la detección de sombra o simulación de sombra. Su mejora era más evidente en casos donde la detección de sombra o la simulación de la misma era imprecisa. Sin embargo, un inconveniente a tener en cuenta para esta aproximación era la inicialización de la caja, que puede llevar a errores significativos.

En nuestro caso, necesitábamos un método de sustracción de fondo que resolviera una serie de problemas que se identificaron en nuestro escenario objetivo:

- Sombras arrojadas por los vehículos
- Reflejos de los focos delanteros
- Escenarios nocturnos
- Condiciones climatológicas: días lluviosos, soleados, niebla
- Atascos o ocupación elevada
- Problemas inherentes a las técnicas empleadas para la segmentación y detección de objetos
- Mala calidad de imagen, por compresión de la señal o por limitaciones de la cáptura

Es por esto que se optó por avanzar en el camino de un sistema adaptativo y robusto guiado por una estrategia de segmentación multi-pista que detectase píxeles no pertenecientes al fondo que se correspondían con vehículos en movimiento o detenidos. Así, esta aproximación de manera adaptativa umbralizaba unos mapas de disparidad de luminancia y cromacidad entre el fondo aprendido y el fondo actual. Luego añadía características adicionales derivadas de las diferencias de gradientes para mejorar la segmentación de coches oscuros que arrojasen sombra, y eliminaba los reflejos de luz de la carretera.

Finalmente la segmentación era utilizada también por un módulo de seguimiento de dos pasos que combinaba la simplicidad de un filtro de kalman 2d lineal y la complejidad de una estimación tridimensional del volumen usando métodos de cadenas de Markov de Monte Carlo. El objetivo de este desarrollo era superar el estado de arte, establecido en aquel momento por una tecnología intrusiva basada en detectores de loop inductivos.

3.4 Sustracción de fondo multi-pista

Analizando el problema se optó por fusionar diferentes pistas en un modelo de fondo ya que en aquel momento apuntaba a ser la mejor aproximación para obtener

resultados de sustracción de fondo precisos. Había que dirimir no obstante dos cuestiones principales: (i) que pistas utilizar; y (ii) de qué manera había que fusionarlas.

Entre las pistas más comunes que se venían utilizando podías encontrarte las siguientes: el color del píxel, la intensidad del píxel (nivel de gris) y los bordes (obtenidos de la imagen de gradientes), que habían sido utilizadas anteriormente en trabajos como [Horprasert et al.99, Huerta et al.08, Huerta et al.09]. Incluso aunque estas aproximaciones presentaban interesantes conclusiones para su aplicación en la medición de datos de tráfico, tales como conteo de vehículos y clasificación, no tenían en cuenta algunos puntos necesarios para sistemas de vigilancia de video reales y viables. Precisamente, uno de estos puntos significativos era la actualización del fondo a lo largo del tiempo, ya que en aplicaciones reales, los periodos de entrenamiento no pueden estar separados de los periodos de proceso. El fondo debe de ser entrenado de manera continua mientras el sistema observa la escena para adaptar los cambios globales, que afectan a los valores de media y desviación estándar del fondo así como las distorsiones cromáticas y de brillo. Otro factor importante era la complejidad de los algoritmos. La potencia computacional requerida por los métodos de procesamiento de imagen está relacionada con la resolución y la frecuencia de imágenes o *framerate*, incluso si estos algoritmos pueden ser paralelizados dentro de múltiples núcleos y múltiples núcleos de GPU programable con lenguajes paralelos como OpenMP, TBB, CUDA u OpenCL. Tipicamente, son necesarios más pasos de proceso, no necesariamente paralelizables, tales como procedimientos de clasificación o de seguimiento de vehículos. Por tanto, eran necesarias algunas simplificaciones y optimizaciones para la propia sustracción de fondo.

Así, en este trabajó se profundizó en una arquitectura de segmentación multi-pista para fusionar diferentes pistas en la imagen, que combinaban estrategias *bottom-up* y *top-down* para resolver los cambios de iluminación globales y locales y obtener un aprendizaje condicional del modelo de fondo. Como se muestra en los experimentos realizados en la sección 3.5, se mejoró las aproximaciones de segmentación existentes, resultando en un paso hacia adelante en el estado del arte del momento en conteo de vehículos y clasificación utilizando cámaras de vigilancia. Más concretamente se mejoraron los siguientes aspectos:

- La estrategia de abajo arriba (bottom-up) incluía un modelo de color con una sensibilidad más alta a cambios de la escena que los modelos habitualmente usados como el codebook RGB o el RGB cónico[Kim et al.04, Huerta et al.08] y las pistas de gradiente, que reforzaban eficientemente las máscaras de segmentación. Esto permitía definir fácilmente diferente regiones segmentadas de acuerdo a sus características de luminancia y cromacidad. Así, podíamos distinguir sombras proyectadas que se movían, reflejos de los faros delanteros del vehículo, cambios globales de iluminación debido a la auto-exposición de la cámara cuando un camión grande con colores claros pasaba a través de la imagen o por cambios en el ambiente.

- La estrategia de arriba a abajo (top-down) incluía información de la escena proveniente de las direcciones de las sombras detectadas directamente de las imágenes sin la necesidad de sensores adicionales, la distorsión producida por la lente de la cámara y por la perspectiva, y datos de seguimiento históricos, para obtener un modelo de fondo mejorado que pudiera también ser usado en días soleados y con situaciones de tráfico denso, ignorando otros objetos en movimiento como insectos, gotas de lluvia,etc.

- Se demostró que la estrategia de sustracción de fondo propuesta funcionaba bien en el escenario objetivo, ya que la estrategia de seguimiento 2D/3D se basaba profundamente en ella para obtener la posición y el volumen estimados del vehículo. La capacidad del método de distinguir entre sombras en movimiento proyectadas y cambios de la iluminación global reducía el número de errores de seguimiento y así permitía al sistema alcanzar ratios bajos de falsos positivos y falsos negativos.

- Adicionalmente, paralelizamos nuestros algoritmos donde era posible en nuestra implementación, alcanzando una ejecución en tiempo real de los mismos.

3.4.1 Clasificación de los píxeles de la imagen

La figura 3.3 resume la secuencia de pasos que hay que seguir para realizar la sustracción de fondo multi-pista que se propuso. En esta algoritmo, I es la imagen actual (I^{y}_{xy}, $I^{c_x}_{xy}$ y $I^{c_y}_{xy}$ son las coordenadas de luminancia y cromacidad para cada píxel

xy en el espacio de color IHLS), B es el promedio del fondo (B^{y}_{xy}, $B^{c_x}_{xy}$ y $B^{c_y}_{xy}$ las coordenadas de luminancia y cromacidad para cada píxel xy del fondo en el espacio de color IHLS), Σ_l es la varianza de la luminancia del fondo y Σ_c la varianza de la cromacidad del fondo, k_l y k_c son constantes de proporcionalidad para determinar los mapas de disparidad fondo/primer-plano (D_l y D_c), o_l, o_c, o_s y o_w son los desplazamientos aplicados para la clasificación de los píxeles, y,finalmente, t_g y t_f son los umbrales para los gradientes (Sobel$_{xy}$) y los procedimientos de rellenado de los blobs (watershed [Preteux92]).

El algoritmo clasifica los píxeles de acuerdo a múltiples pistas: (1) $disparidades$ de luminancia y cromacidad , (2) gradientes de la imagen y (3) observaciones de alto nivel como detección de blobs y dirección estimada de la sombra. Los parámetros que controla el algoritmo pueden dividirse en dos grupos: (1) aquellos relacionados con la segmentación FG/BG ($k_{(l,c)}$ y $o_{(l,c)}$) y (2) aquellos relacionados con la clasificación de píxeles segmentados ($o_{(s,h,w)}$, t_g y t_f).

El primer grupo, los umbrales de segmentación, se determinan manteniendo los ratios de falsa alarma de segmentación de píxeles por debajo de un umbral [T. Chalidabhongse03], mientras que las referencias para el segundo, los umbrales de clasificación, son las observaciones de segmentaciones de sombras oscuras en días soleados y las segmentaciones de focos delanteros cuando hay una luz ambiente menor.

En la práctica, estos umbrales pueden ser modificados como sigue: una vez el sistema ha aprendido el modelo de fondo (B, Σ_l, Σ_c), cogemos un frame que contiene vehículos pasando como una referencia para comprobar la calidad de la segmentación. Inicialmente, $o_{(l,c)}$ son asignados a cero, mientras a $k_{(l,c)}$ se les asigna un número alto. Entonces, los parámetros $k_{(l,c)}$ se van decrementando separadamente hasta que las formas de los vehículos se extraen sin ruido de segmentación alrededor. Después, incluso si el frame se segmenta correctamente, debido al ruido de los streams de video, es recomendable subir los valores de desplazamiento de segmentación $o_{(l,c)}$ de manera que las ratios de falsa alarma de segmentación de píxel se mantienen muy bajos ($< 0{,}1\,\%$ en nuestro caso). Es deseable asignarles los valores más bajos posibles de modo que el sistema presente la mayor sensibilidad posible a cambios en la escena.

Una vez los umbrales de segmentación han sido establecidos se coge un frame en el que se puedan observar vehículos proyectando sombras en un día soleado. La regla

Algorithm 1: Multi-Cue Background Subtraction Algorithm.

$D_l \leftarrow$ Per xy píxel: $|I^y_{xy} - B^y_{xy}| - k_l \cdot (\Sigma_l)_{xy}$
$D_c \leftarrow$ Per xy píxel: $\sqrt{(I^{c_x}_{xy} - B^{c_x}_{xy})^2 + (I^{c_y}_{xy} - B^{c_y}_{xy})^2} - k_c \cdot (\Sigma_c)_{xy}$
Apply CLAHE [Reza04] to D_l and D_c
t_l & $t_c \leftarrow t_{l,c} = min(D_{l,c}) + o_{l,c}$
if *($(D_l)_{xy} > t_l \cup (D_c)_{xy} > t_c$ (per xy píxel))* **then**
    if *($B^y_{xy} > I^y_{xy} \cap |B^y_{xy} - I^y_{xy}| < o_s$)* **then**
        $Mask_{xy} = shadow$;
    else if *($B^y_{xy} > I^y_{xy} \cap |B^y_{xy} - I^y_{xy}| \geq o_s$)* **then**
        $Mask_{xy} = black$
    else if *($I^y_{xy} - B^y_{xy} > o_h \cap I^y_{xy} \leq o_w$)* **then**
        $Mask_{xy} = highlight$
    else if *($I^y_{xy} > o_w$)* **then**
        $Mask_{xy} = white$
    else
        $Mask_{xy} = foreground$
    end
    if *($Mask_{xy} = (shadow \cup black) \cap (D_c)_{xy} > t_c \cap (D_l)_{xy} - t_l < (D_c)_{xy} - t_c$)* **then**
        $Mask_{xy} = foreground$
    end
else
    $Mask_{xy} = background$;
end
$E_I, E_B \leftarrow$ Apply Sobel$_{xy}$ to I and B with t_g
Add $(E_I - E_B)$ to $Mask$ as $foreground$
Crop $highlight$ regions in $Mask$ (figura 3.7)
Watershed(t_f): Fill $foreground$ & $highlight$ & $white$ blob inbetweens as $foreground$
Reinforce $Mask$ from shadow direction (figura 3.9)
Update B, Σ_l, Σ_c (figura 3.11)
return $Mask$

Figura 3.3: Multi-Cue Background Subtraction Algorithm.

para asignar los parámetros de clasificación es segmentar únicamente los vehículos, lo mejor posible, con respecto a las sombras en este caso. Esto significa que configuramos o_s y t_g de modo que los píxeles clasificados como $black$ y $foreground$, que se correspondan con que regiones oscuras y con gradiente, se asignen mayormente en las proyecciones de los vehículos y no en las sombras. El siguiente paso consiste en establecer o_h y o_w usando una imagen con vehículos con los focos encendidos. De nuevo, la regla es segmentar únicamente los vehículos, lo mejor posible, con respecto a las proyecciones de los focos en este caso. Así, primeo configuramos o_h para obtener únicamente las clasificaciones de los píxels proyectados por los focos como $highlight$, que será recortado consecuentemente utilizando el algoritmo de recorte explicado en la subsección siguiente. Como las regiones internas del vehículo pueden ser también clasificadas como $highlight$, configuramos o_w para obtener clasificaciones de píxel $white$ en lugar de $highlight$ y así evitar recortarlas. Finalmente, a t_f se le asigna el mejor valor posible siguiendo la regla de mejora de la segmentación de vehículos únicamente, observando cómo afecta este criterio rellenar los intersticios de las regiones $foreground$, $highlight$ y $white$ con $foreground$.

Podría parecer que esta parametrización no permite que la aproximación sea genérica, pero como se demostrará en la sección de pruebas, estos parámetros son bastante independientes de los cambios del entorno, por lo que necesitan ser configurados únicamente una vez para todos los casos.

Los umbrales de la segmentación resultantes $t_{l,c}$ pueden adaptarse automáticamente a los cambios ambientales porque se corresponden con los valores mínimos de los mapas de disparidad $D_{l,c}$, calculados en cada instante de tiempo, sesgados por los desplazamientos definidos por el usuario $o_{l,c}$, y por lo tanto, permiten al sistema ejecutarse satisfactoriamente en instalaciones reales. Para la implementación práctica de la aproximación se recomienda normalizar la resolución de las imágenes a un número predefinido de líneas, manteniendo la relación de aspecto. De esta manera es posible también hacer que los parámetros que dependan de la resolución se vuelvan independientes de instalación en instalación.

Las imágenes de disparidad $D_{l,c}$ se establecen comparando los valores de las diferenecias de cromacidad y luminancia de los píxeles con respecto sus correspondientes varianzas temporales. Si la diferencia es más alta que la varianza, entonces puede considerarse que no es debido al ruido de la imagen o a características

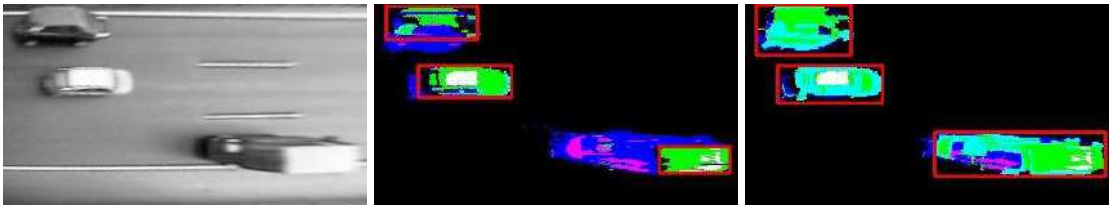
del fondo en movimiento y, por lo tanto, que sean parte de un cambio en la escena debido a un objeto en movimiento (vehículo, mosca, gotas de lluvia, etc.) o una variación en la iluminación. Los umbrales para extraer las regiones del primer-plano se establecen automáticamente con los valores de disparidad mínimos, y así, pueden adaptarse a variaciones de escenas de exterior. Como puede haber algunos vehículos con valores de disparidad mayores que otros vehículos en la misma imagen y ya que consideramos también los desplazamientos definidos por el usuario para los umbrales, con el objetivo de disminuir segmentaciones de falsos negativos, aplicamos el procedimiento de normalización CLAHE [Reza04] a las imágenes de disparidad para amplificar el contraste local de las disparidades antes de umbralizar. En esta fase, se aplican de igual modo procedimientos morfológicos tales como erosión y dilatación para eliminar áreas segmentadas pequeñas debido al ruido.

Inicialmente, se extrae una máscara midiendo las disparidades de luminancia y la cromacidad de la imagen actual con respecto a la media actualizada del fondo a lo largo del tiempo. Esta máscara contiene píxeles catalogados de acuerdo a las siguientes etiquetas: $background$, $shadow$, $black$, $foreground$, $highlight$ y $white$, donde cada una de ellas se representa como un valor de intensidad de gris en la máscara. Las etiquetas $Shadow$ y $black$ se asignan a píxeles más oscuros dependiendo de la cantidad de disparidad de luminancia con respecto a B. La razón para incluir una categoría de píxel $black$, además de $shadow$ para regiones más oscuras, es que es más plausible que esta categoría esté dentro de blobs segmentados en lugar de en sus fronteras, como se puede observar experimentalmente. Después de la primera fase de clasificación de píxeles, estas regiones más oscuras se revisan nuevamente para comprobar si su disparidad cromática tiene más importancia que la de la luminancia y se reetiquetan como $foreground$ en el caso de que así sea. De manera similar píxeles más luminosos se catalogan como $highlight$ y $white$. La categorización propuesta permite mejoras en la calidad de la segmentación con respecto a previas categorizaciones, tales como la que utiliza únicamente 4 clases ($background$, $shadow$, $highlighted\ background$ y $foreground$) presentada originalmente en [Horprasert et al.99] como se mostrará en la sección de pruebas.

En la segunda fase del proceso, la máscara es reforzada con pistas de gradiente sustraído, lo cual es especialmente útil para detectar características que se corresponden a vehículos oscuros que pueden ser confundidos con sombras de

acuerdo a su definición de modelo. Es recomendable dilatar la imagen E_B antes de sustraer ya que la sustracción de bordes puede contener características ruidosas que deberían ser eliminadas. Por otro lado, los bordes pueden aparecer en los límites de las sombras y por lo tanto debería aplicarse una erosión basada en un kernel morfológico. Es preferible realizar esta erosión únicamente cuando se haya detectado que podría haber una proyección de sombra significativa, lo cual se hace en el algoritmo mostrado en la figura 3.9, ya que los vehículos pequeños y oscuros sin sombras proyectadas podrían ser eliminados en el proceso. Esta decisión puede ser automatizada fácilmente a través de la relación del número de píxeles en la máscara de refuerzo de la dirección de la sombra con respecto a los píxeles de blobs completos.

Después, como se explicará en la siguiente subsección, la máscara se procesará para eliminar regiones iluminadas que se corresponden a cambios súbitos de iluminación, debido a la variabilidad del tiempo o a los focos de vehículos. Adicionalmente, el contenido de las máscaras de blobs ser rellenarán aplicando el procedimiento watersed a los contenidos $foreground$, $highlight$ y $white$ rellenandolos con $foreground$. Como el watershed incrementa el tamaño de los blobs compactos resultantes, se aplica una erosión proporcional para evitarlo. Así, reforzamos la segmentación de vehículos oscuros y los preparamos para la última fase en la que se estimará la dirección de la sombra para reforzar la máscara todavía más. La figura 3.4 muestra un ejemplo de este procedimiento de segmentación.



(a) (b) (c)

Figura 3.4: Resultados de la segmentación de imagen. La imagen (b) muestra la segmentación obtenida usando los mapa de disparidad de luminancia y cromacidad únicamente, mientras que la (c) también incluye las pistas de los gradientes. Los colores para la clasificación de los píxeles son: $background$=negro, $shadow$=azul, $black$=magenta, $foreground$=cyan, $highlight$=verde y $white$=blanco. Se puede observar en la imagen (c) cómo el número de píxeles clasificados como $foreground$ es mayor.

3.4.2 Procesamiento de los cambios bruscos de iluminación

Ignoraremos las regiones más oscuras para construir los blobs candidatos a vehículo para la fase de seguimiento, pero las regiones iluminadas requieren procesamiento adicional para borrar aquellas generadas por cambios de iluminación repentina que provienen de variaciones del tiempo o focos de vehículos (figuras 3.5 y 3.6). La figura 3.7 muestra el algoritmo que hace esto, teniendo en consideración la geometría del carril, donde x es la dirección a lo largo del carril e y es la dirección transversal en las imágenes que, utilizando una calibración inicial, están rectificadas de manera que estas direcciones coinciden con el eje x e y de la imagen. Dentro de un carril, se suman los píxeles correspondientes a una línea perpendicular al carril y si todos los píxeles no-fondo se corresponden con la categoría $highlight$, entonces esos píxeles son tipificados como $background$. Las proyecciones de vehículo tienen más de una categoría de píxeles en las líneas perpendiculares a los carriles, y por lo tanto, usando esta aproximación, los blobs pueden recortarse por carril eliminando áreas completamente iluminadas, pero no los vehículos. En este proceso se considera la geometría del carril ya que puede haber vehículos en carriles diferentes, paralelos a regiones iluminadas, que podrían interferir en el recorte. La categoría $white$ también se incluye en la máscara, además de $highlight$, porque puede haber vehículos pintados de blanco con poca textura característica, y aplicando este procedimiento se podrían recortar cuando no deberían. Este procedimiento puede ser aplicado en los bordes de la carretera del mismo modo.

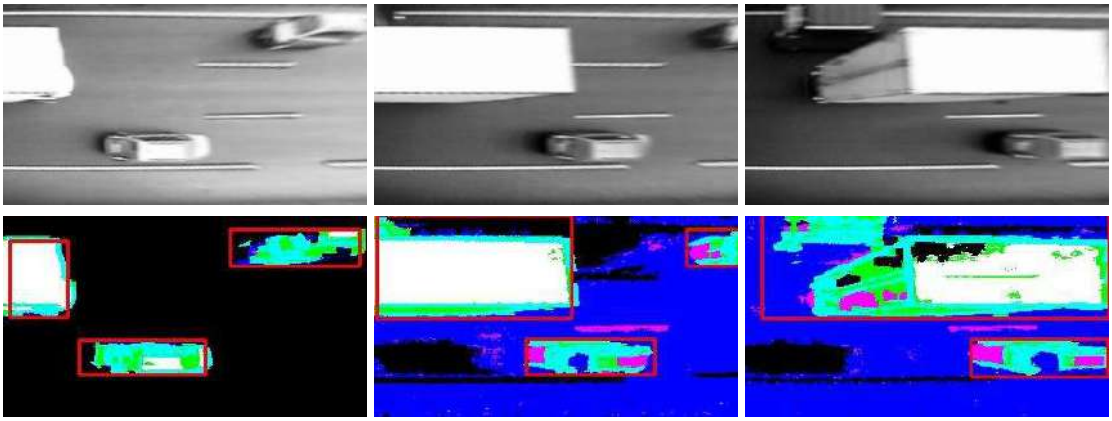
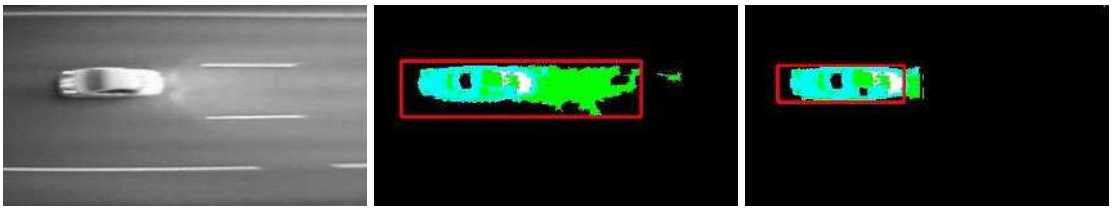


Figura 3.5: Cambios de iluminación repentinos del fondo de la imagen debido a la auto-exposición de la cámara cuando un camión de color claro pasa a través de la imagen.



(a) (b) (c)

Figura 3.6: Cambio repentino de la iluminación local debido a las luces de un vehículo. La imagen (b) muestra la segmentación obtenida inicialmente, y (c) el blob recortado resultado usando la aproximación propuesta.

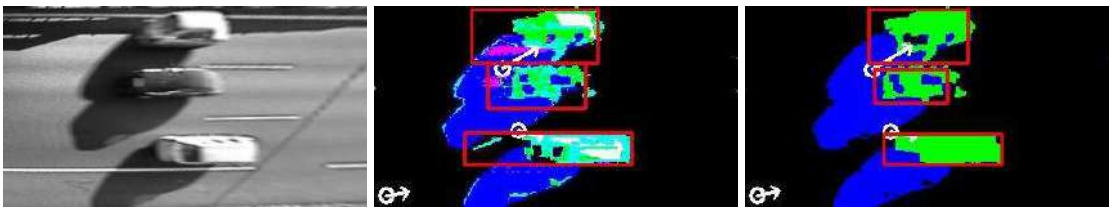
Algorithm 2: Algoritmo de recorte de iluminación fuerte.

```
1  forall lane do
2      for x = 0, lane_length do
3          count = 0
4          hCount = 0
5          for y = lane_y, lane_width do
6              if Mask_xy ≠ background then
7                  count = count + 1
8                  if Mask_xy = highlight then
9                      hCount = hCount + 1
10                 end
11             end
12         end
13         if hCount = count ∩ count ≠ 0 then
14             for y = lane_y, lane_width do
15                 Mask_xy = background
16             end
17         end
18     end
19 end
20 return Mask
```

Figura 3.7: Algoritmo de recorte de $Highlight$.

3.4.3 Estimación de la dirección de la sombra y refuerzo de mascara

La figura 3.9 muestra el procedimiento para reforzar la máscara segmentada estimando la dirección media de la sombra y etiquetando la región del blob como $highlight$, ignorando la categoría $shadow$, opuesta a ella. Usando su máscara complementaria, la región opuesta es explícitamente asignada a $shadow$ para borrar cualquier píxel restante catalogado erróneamente como píxel de primer-plano, principalmente debido a pistas de borde en fronteras de sombra. Esto es adecuado para distinguir mejor los vehículos oscuros de las sombras proyectadas, porque permite el reetiquetado automático de píxeles que tienen más probabilidad de caer dentro de vehículos y no en regiones de sombra, ya que sabemos que el vehículo está localizado en la parte opuesta de su sombra proyectada. Los procedimientos morfológicos pueden también aplicarse a las imágenes G_1 y G_2 para obtener una imagen$(G_1 - G_2)$ de refuerzo más compactada y mejor adaptada, como se muestra en la figura 3.8.



(a) (b) (c)

Figura 3.8: Estimación de la dirección de sombra y refuerzo de máscara. La imagen (b) muestra en la esquina inferior izquierda la dirección media de la sombra, y (c) el refuerzo $highlight$ y la "limpieza" de $shadow$ aplicada a los blobs a partir de ella.

El número de píxeles N para el desplazamiento de imagen se calcula experimentalmente de acuerdo a la resolución. Se recomienda asignar un valor con un tamaño similar a la anchura de los vehículos.

3.4.4 Actualización condicional del fondo

La figura 3.11 muestra el algoritmo para actualizar las imágenes de fondo B, Σ_l y Σ_c , donde I_{t-1} es el frame anterior, T es una matriz que almacena el tiempo pasado para cada píxel $estático$, $blobs_{tracked}$ son los blobs incorporados actualmente al algoritmo de seguimiento, Δt es el tiempo pasado desde el frame anterior, l_{rate} es la tasa de

Algorithm 3: Shadow Direction Reinforcement Algorithm.

$blobs \leftarrow$ get non-$background$ blobs [Suzuki and Be85]
$S \leftarrow$ get image with $shadow$ & $black$ regions
$F \leftarrow$ get image with $foreground$, $highlight$ & $white$ regions
forall $blobs$ **do**
    $G \leftarrow$ paint single blob in an image
    $S_1 \leftarrow G \cap S$
    $F_1 \leftarrow G \cap F$
    $s \leftarrow$ calculate centroid of S_1
    $f \leftarrow$ calculate centroid of F_1
    $\alpha \leftarrow$ calculate angle of vector joining s and f
end
$\bar{\alpha} \leftarrow$ calculate mean from α angles & previous $\bar{\alpha}$ when $blobs \neq 0$. Otherwise, maintain previous $\bar{\alpha}$
$G_1 \leftarrow$ get image with blobs containing $black$, $foreground$, $highlight$ & $white$ regions
$G_2 \leftarrow$ displace G_1 in $\bar{\alpha} + \pi$ direction N pixels
Add $(G_1 - G_2)$ to $Mask$ as $highlight$ and set the rest of the mask as $shadow$
return $Mask$

Figura 3.9: Algoritmo de refuerzo de la dirección de la sombra.

aprendizaje de fondo y t_{update} el umbral de tiempo para la actualización de blobs $est\acute{a}ticos$. Los píxeles $est\acute{a}ticos$ son aquellos que se encuentran dentro de blobs con seguimiento con ningún desplazamiento de frame a frame.

Este procedimiento permite incrementar la robustez del sistema en términos de actualización de fondo. Dentro de este tipo de escenarios, es crítico guardar un modelo de fondo actualizado incluso en situaciones de alta densidad de tráfico, donde los vehículos pasan más tiempo en la escena. Nuestro algoritmo es capaz de identificar estas situaciones y adaptarse a las actualizaciones del fondo, mejorando el rendimiento de los esquemas de actualización que trabajan a nivel de píxel. La figura 3.10 ilustra el efecto de aplicar la aproximación de actualización condicional de fondo propuesta comparada a las estrategias a nivel de píxel, tales como el running-average [Wren et al.97] o la actualización condicional por niveles [Kim et al.05], en un escenario de ejemplo desafiante.

Posteriormente esta máscara multipista alimentará el proceso de seguimiento que estimará la posición y el volumen de los vehículos. Este proceso de seguimiento utilizará la máscara para establecer los *bounding-boxes* de las proyecciones 2D de los vehículos en el plano de la carretera y estimar el volumen 3D de acuerdo a la calibración de cámara previamente realizada.

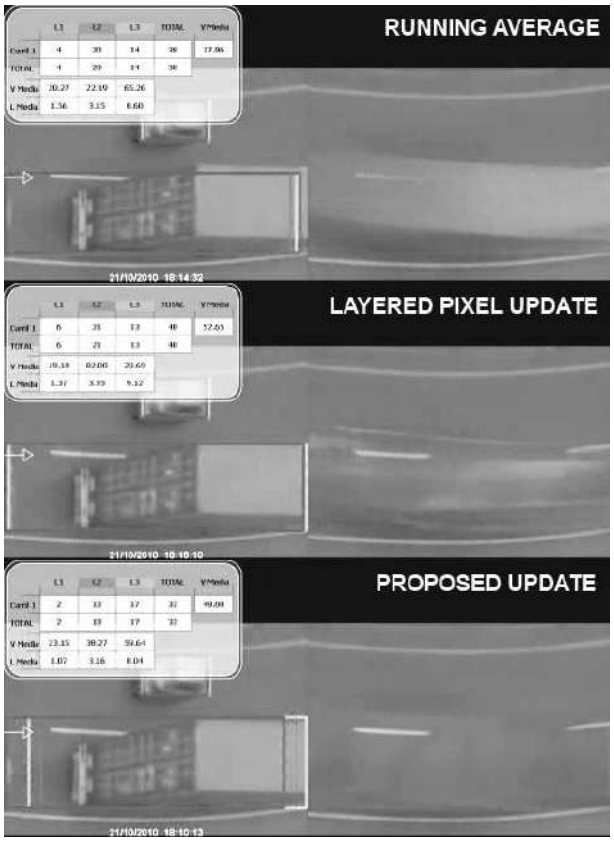


Figura 3.10: Comparación entre diferentes métodos de actualización de fondo. La imagen superior representa la aproximación básica a nivel de píxel. La imagen del medio muestra el resultado después de aplicar el algoritmo propuesto pero sin usar regiones (a nivel de píxel); y la imagen inferior muestra el excelente resultado proporcionado por el procedimiento descrito en la figura 3.11.

Algorithm 4: Conditional Background Update Algorithm.

$C \leftarrow$ paint $blobs_{tracked}$ distinguishing $static$ and $moving$ blobs with different gray-level values
forall xy *pixels in* $< I, I_{t-1}, B, \Sigma_l, \Sigma_c, T, C >$ **do**
  if $C_{xy} = static$ **then**
    $T_{xy} = T_{xy} + \Delta t$
    if $T_{xy} > t_{update}$ **then**
      $B_{xy} = I_{xy}$
      $T_{xy} = 0$
    end
  else if $C_{xy} = moving$ **then**
    $T_{xy} = 0$
  else
    $(\Delta r, \Delta g, \Delta b) = (I - B)_{xy}^{(r,g,b)}$
    $\Delta d_B = \sqrt{\Delta r^2 + \Delta g^2 + \Delta b^2}$
    $factor = l_{rate} / \Delta d_B$
    $B_{xy}^{(r,g,b)} = B_{xy}^{(r,g,b)} + factor \cdot (\Delta r, \Delta g, \Delta b)$
  end
  $\Delta y = |I_{xy}^{y} - (I_{t-1})_{xy}^{y}| - (\Sigma_l)_{xy}$
  $\Delta c = \sqrt{(I_{xy}^{c_x} - (I_{t-1})_{xy}^{c_x})^2 + (I_{xy}^{c_y} - (I_{t-1})_{xy}^{c_y})^2} - (\Sigma_c)_{xy}$
  $\Delta d_\Sigma = \sqrt{\Delta y^2 + \Delta c^2}$
  $factor = l_{rate} / \Delta d_\Sigma$
  $(\Sigma_l)_{xy} = (\Sigma_l)_{xy} + factor \cdot \Delta y$
  $(\Sigma_c)_{xy} = (\Sigma_c)_{xy} + factor \cdot \Delta c$
end
return B, Σ_l, Σ_c

Figura 3.11: Algoritmo de actualización condicional del fondo.

3.5 Resultados

Para evaluar el rendimiento de nuestra aproximación, realizamos dos tipos de tests: (1) rendimiento de la sustracción de fondo para comprobar la sensibilidad de nuestra aproximación a las variaciones de la escena con respecto a diferentes alternativas en el estado del arte de aquel momento, y (2) una evaluación del conteo y la clasificación realizada por el sistema basado en visión utilizando un conjunto de vídeos con diferentes situaciones complejas y desafiantes .

3.5.0.1 Rendimiento de la sustracción de fondo

Las alternativas con las que nuestra aproximación se comparó fueron las siguientes: (1) el modelo cilíndrico RGB descrito en [T. Chalidabhongse03] [Kim et al.04], (2) el modelo de RGB cónico descrito en [Huerta et al.09], (3) el modelo basado en el espacio de color HSV usado en [Cucchiara et al.01] y (4) una variante basada en el espacio de color HLS, que también separaba la luminancia de la cromacidad de una manera parecida al HSV pero reemplazando el brillo (o valor) por la luminosidad. Para hacer una comparación justa de sus respectivas sensibilidades a los cambios de escena, integramos estos modelos de luminancia y cromacidad en el mismo marco de trabajo de disparidad de imágenes presentado anteriormente. De esta modo, le permitíamos al sistema elegir automáticamente el umbral para separar fondo de primer-plano.

Los parámetros de cada aproximación de sustracción de fondo se determinaban experimentalmente de modo que el ratio de falsas alarmas se mantuvieran por debajo de 0,1 % de la misma manera explicada en la sección 3.4, mientras que la segmentación de vehículos tenía una calidad similar en todos los métodos. Esta comparación se realizaba usando el método de perturbación descrito en [T. Chalidabhongse03], que medía el porcentaje de píxeles de imagen considerados como primer-plano (sin distinguir subclases) con respecto al número total de píxeles de la imagen cuando se aplicaba a cada píxel una perturbación de color aleatoria. Esta perturbación aleatoria se producía mediante la adición de un vector de tamaño Δ con una dirección aleatoria a cada píxel en el espacio de color RGB. Si Δ era alto, casi todos los píxeles de la imagen debían ser catalogados como primer-plano por todos los métodos. De este modo podíamos medir la sensibilidad de los métodos de sustracción detectando objetivos de bajo contraste contra el fondo aprendido. Cuanto menor fueran los valores Δ que se

necesitaban para detectar píxeles de primer-plano mayor la sensibilidad de la aproximación de sustracción de fondo, y por lo tanto mejores los resultados. Así pues, para este test, no era necesario observar escenas con vehículos en movimiento. Observar escenas de fondo era suficiente, ya que los candidatos a píxeles de primer-plano se generaban artificialmente. Durante la observación del fondo incrementamos los valores de Δ por 1 de frame en frame, donde su unidad correspondía a valores de píxeles con 8 bits por canal de color. Durante el período de perturbación no se realizaba actualización.

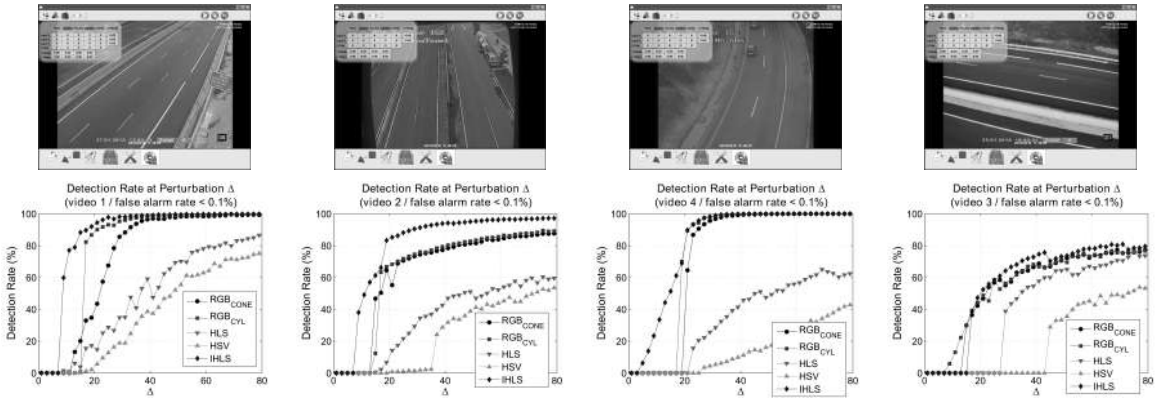


Figura 3.12: Comparación de la sensibilidad del modelo de color de las aproximaciones de disparidad FG/BG usando IHLS, conical RGB, cylindrical RGB y modelos de cromacidad y luminancia HSV y HLS, a través del método de perturbación de color aplicado en cuatro vídeos diferentes de carretera.

La figura 3.12 muestra la sensibilidad de los resultados obtenida en los cinco modelos en cuatro vídeos con tipos diferentes de fondos sin la presencia de vehículos en movimiento:: (1) una escena con un cambio de iluminación notable debido a la desaparición del sol tras unas nubes y su reaparición, (2) una escena con iluminación difusa y con gran cantidad de ruido de imagen especialmente alrededor del color blanco debido a la conversión analógico-digital, (3) una escena con iluminación difusa pero con menos contraste y (4) una escena nocturna grabada en escala de grises en lugar de en color. Son secuencias cortas de unos pocos segundos, en los que las perturbaciones no se han aplicado, todos los píxeles están etiquetados como fondo. Cuanto mayor fuera Δ , mayor la probabilidad de que los métodos de sustracción de fondo etiquetasen los píxeles como primer-plano. Así, se puede observar cómo, en

general, nuestra aproximación (marcada como IHLS) necesitaba una perturbación menor para detectar ratios más altos de píxeles de primer-plano que el resto de métodos evaluados, especialmente si se usaban imágenes de color.

Para comprobar la sensibilidad e influencia de los parámetros de segmentación en el rendimiento del proceso de sustracción de fondo, observamos como el sistema reaccionaba a este test con diferentes valores para los parámetros. Las aproximaciones con mayor sensibilidad continuaban siendo los modelos de RGB cilíndrico, RGB cónico e IHLS, con ligeras diferencias entre ellos. Sin embargo, teniendo en cuenta los resultados obtenidos en [Blauensteiner et al.06] para el IHLS con respecto al RGB y al modelo de color RGB normalizado para segmentación de imagen en regiones de baja saturación, consideramos que IHLS es la mejor opción.

Finalmente, en relación al tiempo de computación de nuestra implementación, se ejecutaba a 30 FPS usando imágenes capturadas a una resolución de 768x576 pero convertidas a 320x240 para el procesamiento en un Intel Core2 Quad CPU Q8400 @ 2.66 GHz con 3 GB RAM con una tarjeta gráfica NVIDIA 9600GT, que era suficiente para nuestros propósitos. A modo de curiosidad, el programa se implementó en C/C++, usando OpenMP y CUDA para las zonas de código que querían ejecutarse en paralelo (en diferentes núcleos de la CPU o en GPU, respectivamente). En nuestra implementación los porcentajes de tiempo medio de consumo de las diferentes etapas con respecto al total era, desde la más alta a la más baja: (1) Calculo de la máscara de color 32,3 %, (2) cálculo de la máscara de bordes 29,2 %, (3) recorte de las zonas $highlight$ 17,5 %, (4) actualización condicional 16,3 %, (5) refuerzo de sombras 2,6 %, (6) seguimiento 2D 2,0 % y (7) seguimiento 3D 0,1 %. Aplicamos herramientas de CUDA para los cálculos de bordes y de color y también para la actualización condicional, donde el problema era altamente separable. Por otro lado, no observamos cambios significativos en el tiempo de computación debido a parámetros de variación, excepto en el caso de t_f para el relleno de zonas conexas por watershed, que en nuestra implementación se podía traducir en un deterioro significativo del rendimiento si el parámetro era alto. Sin embargo, manteniendo la resolución de las imágenes bajo control, únicamente necesitamos un valor pequeño para mejorar la máscara de segmentación, por lo que en práctica no era un problema para el rendimiento general del sistema.

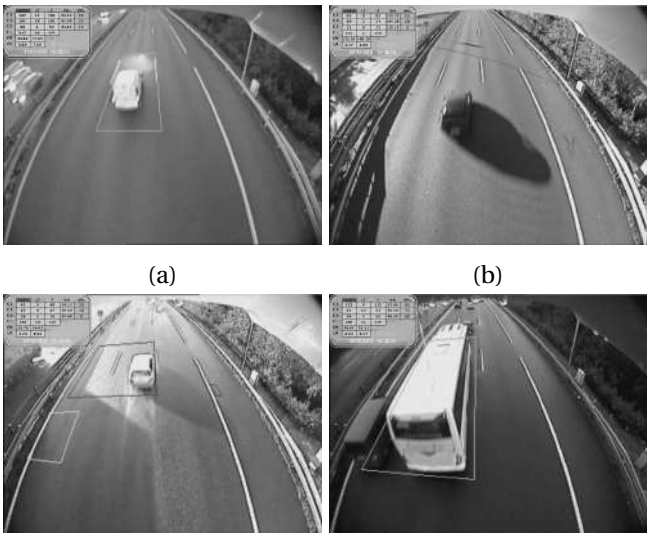


Figura 3.13: Ejemplos de fallos de detección y clasificación en cada vídeo: (a) clasificación errónea debido a una mala segmentación de la reflexión de las luces delanteras, (b) vehículo oscuro no-detectado, (c) falso positivo debido a un cambio global en la iluminación y (d) vehículo perdido debido a una oclusión parcial.

3.6 **Conclusiones**

Presentamos un sistema novedoso en el marco de la visión por computadora para realizar seguimiento y clasificar vehículos con el objetivo de reemplazar los detectores de bucle inductivo (ILDs) especialmente en autopistas. El sistema ha sido validado en condiciones climáticas diferentes especialmente duras (incluyendo días lluviosos, con el empeoramiento de la visibilidad y días soleados con la aparición de fuertes sombras proyectadas direccionales), obteniendo unos resultados similares a los de las ILDs. Adicionalmente, este sistema se distinguía de otras aproximaciones basadas en visión por computador en que podía gestionar las sombras proyectadas sin la necesidad de ningún hardware más allá de las cámaras, tales como GPS para estimar la dirección de las sombras. Por tanto, consideramos que era una alternativa viable para reemplazar a los ILDs u otras tecnologías tales como los tags instalados en los vehículos, los escáneres laser que reconstruyen la forma 3D de los vehículos u otras aproximaciones basadas en visión por computador, cuya instalación y mantenimiento era y es más compleja que usar únicamente cámaras. La evolución de las GPUs y los procesadores

multi-núcleo nos permitieron además alcanzar ejecución en tiempo-real con componentes disponibles en el mercado.

Without data you're just another person with an opinion.

W. Edwards Deming

CAPÍTULO

4

Sistema de Reconocimiento de Señales de Tráfico

4.1 Contexto

Como ya se ha comentado previamente, desplazarse, a día de hoy, es una actividad ligada a muchos aspectos de nuestra vida, ya sean desplazamientos cortos o largos, ya sean por trabajo o por ocio. Sin embargo, desplazarse siempre entraña cierto riesgo que todos asumimos y que la comunidad científica pretende minimizar. Para que el lector se haga una idea, según la Organización Mundial de la Salud en su informe de estado global, refleja que a diciembre de 2018 el número de muertes en tráfico por carretera ascendió a 1.35 millones [Organization18, Organization09], sólo en España, en la primera mitad del 2020 se han registrado del orden de 378 fallecimientos debido a accidentes de tráfico. Para paliar estas cifras la sociedad científico-tecnológica en la que vivimos va enfocando sus esfuerzos en ir reduciendo el factor humano de sus carreteras, con el objetivo de eliminar las distracciones, la somnolencia, la ansiedad por llegar antes, los comportamientos de conducción irresponsables, etc..., incrementado el intervalo de tiempo de reacción que pueda tener el conductor, advirtiéndole de desvíos de atención o de comportamientos inapropiados detectados o alertándole con antelación de posibles situaciones de riesgo y aconsejándole medidas efectivas para

evitarlas, utilizando para ello, por ejemplo, sistemas asistenciales de ayuda a la conducción (ADAS). Estos sistemas, a la postre, acabarán dotando de total autonomía a los vehículos, que completamente sensorizados, circularán por las vías de transporte de una manera segura y efectiva, proporcionando a los pasajeros una experiencia de viaje cómoda y despreocupada. De hecho para mejorar los niveles de seguridad, son indispensables también sistemas automáticos que siendo capaces de entender el entorno inmediato de los vehículos, detecten y analicen la situación y el deterioro de otros de los agentes importantes que participan en el buen funcionamiento del tráfico por vías de transporte, me estoy refiriendo aquí a la señalética. El conductor de hecho no es el único agente responsable de la seguridad de su conducción, la señalética presente en los canales de transporte también es determinante, realizando diferentes funciones de cara a los vehículos que circulan por las carreteras: definiendo un tipo de conducción en un tramo determinado, advirtiendo de peligros, indicando acciones aconsejadas en función de la situación, controlando de manera sincronizada en ocasiones los flujos de tráfico, etc. También entra en juego en el nivel de seguridad presente en una vía de transporte el estado del propio canal de transporte y por supuesto, la posibilidad de que agentes externos puedan invadirlo y provocar situaciones de peligro.

Es en este contexto donde se fragua el trabajo que se presenta a continuación, bajo el paraguas del proyecto europeo INLANE ya descrito en la sección inicial. Aunque el dossier del proyecto define ciertos objetivos, es fácil extrapolar los sistemas desarrollados a otras aplicaciones directas como alguna de las indicadas en el párrafo anterior: la geolocalización de los elementos detectados, el análisis de su estado o la detección de la invasión del canal de transporte por elementos externos.

Y aquí nuevamente recurrimos a una tecnología que te permite analizar una zona amplia de interés, siendo poco invasiva, de carácter generalista y de bajo coste: la visión por computador, que acompañada por su inseparable adlátere, el *machine learning*, han sido los elementos nucleares dentro del desarrollo que se ha realizado.

Dicho esto en el siguiente apartado entraremos a definir con mayor detalle la investigación llevada a cabo en el contexto esbozado en las líneas previas.

4.2 Descripción del problema

Dentro del contexto de este proyecto europeo, se definió la tarea de desarrollar un sistema de reconocimiento de señales de tráfico basado en visión artificial. Este sistema, aunque en principio, estaría especializado en señales de limitación de velocidad, habría de ser extensible a un mayor número de tipos de señales y de hecho, a diferentes elementos que pudiesen encontrarse en un canal de transporte: conos, semáforos, marcas en el suelo, animales, vehículos, etc... Además debía de funcionar en tiempo real y en todos los países de la zona europea. Esta solución había de proporcionar la geolocalización de las diferentes señales detectadas utilizando para ello una única cámara, sin duda pensando en hacer un sistema de bajo coste para su futura productización.

El diagrama de etapas o *pipeline* ideado para alcanzar los objetivos definidos, no es muy diferente, hablando en términos muy generales, del que se usa habitualmente en muchos de los proyectos basados en visión artificial y son muchas las disciplinas que interactúan en él de manera sistemática y cooperativa.

Dentro de estas disciplinas, encontramos una de las técnicas clave y por otro lado emergente a día de hoy, perteneciente al campo del *machine learning*: la detección de objetos. Como puede verse en la figura 2.1 la fase de detección es una sección de gran relevancia dentro de la etapa de análisis, ya que proporcionará información acerca de la ubicación de elementos que consideremos significativos en la escena. La detección visual de objetos en un escenario adquiere así un papel fundamental en los sistemas que se basan en técnicas de visión artificial que pretenden analizar la información que presentan las diferentes imágenes capturadas por los sensores.

Como ya se ha comentado previamente, es la combinación de la visión artificial junto con el *machine learning* el marco de trabajo en el que encontramos la líneas de investigación hacia la solución de este problema. Dicha combinación, posibilita la localización de los elementos en una imagen, asignándoles una categoría general a dichos elementos detectados que se resolverá con una tipificación particular en el proceso de reconocimiento, dando lugar , finalmente, a la posibilidad de realizar un análisis semántico en la imagen. A modo de ejemplo, imagine el lector un coche que va circulando a una velocidad de 100 km/h por una vía con límite de velocidad de 80 km/h. Llegará un punto en la vía en el que, a modo disuasorio y recordatorio, se habrán

instalado unas señales de velocidad límite de 80 km/h. Cuando el vehículo llegue a la zona donde se halle instalada la señal, el TSR instalado abordo realizará una detección de objetos en la imagen. De esta detección, se obtendrán una lista de señales de tráfico, quizás una lista de vehículos y alguna que otra lista de otros elementos cuya detección hayamos considerado relevante. El sistema pasará posteriormente a la etapa de reconocimiento donde se clasificará cada señal detectada, en este caso como señal de limitación de velocidad máxima a 80km/h. Finalmente, tras la ubicación espacial de la señal para comprobar que es una señal que se corresponde con la vía por la que se circula, se pasaría a la fase de comprensión de la escena donde los elementos y sus relaciones adquieren un nivel semántico; así, si la señal se encuentra en el carril que le corresponde al vehículo en el que está montado el TSR, el sistema debería comprobar la velocidad del vehículo e informar al usuario de que se está circulando a una velocidad no permitida o, en un contexto de mayor autonomía, interactuar con los sistemas del coche para ir adecuándolo a la velocidad de la vía.

Precisamente, es esta amalgama de diferentes técnicas y disciplinas lo que hace que la implementación de un sistema de esta índole sea tan desafiante y complejo. Y los retos a enfrentar son varios y localizados en las diferentes etapas de este *pipeline* 2.1 aunque en el marco que define esta tesis nos hemos centrado principalmente en los que mantienen cierta relación con las etapas más vinculadas a los procesos de tratamiento de imagen e interpretación de datos.

Entre los desafíos nos encontramos la necesidad de recopilar grandes cantidades de datos para poder entrenar los modelos, siendo uno de los principales retos del entrenamiento supervisado utilizando técnicas de *deep learning* (DL). A menudo, la falta de un *dataset* representativo suficientemente grande disuade del uso del DL en ciertas aplicaciones. Típicamente, la adquisición de la cantidad de datos requeridos tiene costes de tiempo, de material y de esfuerzo considerables. Para mitigar este problema, una aproximación popular para entrenar de manera efectiva diferentes modelos, es la incorporación de imágenes sintéticas a una base de datos de imágenes reales.

Esto nos llevó a analizar hasta donde podíamos llegar con los datos sintéticos, sin la incorporación de datos reales en la base de datos objetivos. ¿Esta aproximación sería suficiente? ¿Tendría suficiente capacidad de generalización para utilizarlo en entornos reales? ¿Cómo podemos realizar un análisis comparativo con otros modelos que no esté

sesgado a las peculiaridades y especificidades de un contexto concreto? Estas son algunas de las preguntas que intentamos responder en este estudio, examinando el potencial del entrenamiento basado en datos sintéticos en el campo de los sistemas de transporte inteligentes. Nos enfocamos principalmente en la aplicación de reconocimiento de señales de tráfico basados en cámara para sistemas avanzados de asistencia a la conducción y para conducción autónoma. La secuencia de etapas propuesta incluye procesos nuevos de aumentación de datos tales como sombras estructuradas y brillos especulares gaussianos. Para este análisis, se decidió utilizar un modelo DL ampliamente conocido, AlexNet, con diferentes *datasets* para comparar el rendimiento de este modelo entrenado con imágenes sintéticas y reales. Se ha propuesto y elaborado, adicionalmente, un método nuevo y detallado para comparar objetivamente estos modelos. Las imágenes sintéticas se generan utilizando métodos semi-supervisados guiados por el error que también se describe. Nuestros experimentos muestran que una aproximación basada en imagen sintética supera el rendimiento en la mayoría de los casos, de los modelos entrenados con imágenes reales cuando se aplican a conjuntos de test en un entorno de dominios cruzados, es decir, dominios para los que no ha sido entrenado y, en consecuencia, la generalización del modelo se mejora decrementando el coste de la adquisición de las imágenes.

Otro de los retos que presentaba este sistema era su ejecución en tiempo real. Este punto es importante si tu sistema tiene que dar aviso de la presencia de una señal de advertencia en un momento determinado, como ocurre durante la conducción. Para sistemas de mobile-mapping, no sería tan crítico ya que lo único necesario realmente es la posición GPS para cada frame que vayas a tratar. A partir de ahí el proceso de identificación y localización de la señal podría ser offline. En el caso que nos ocupa, que fuese en tiempo real era un factor crítico.

4.3 Objetivos y Retos

Uno de los principales desafíos era diseñar una base de datos de aprendizaje adecuada. Realizar la base de datos por nuestra cuenta, hubiese supuesto un despliegue de recursos que no era viable. La base de datos debía de ser significativa, estar balanceada, debía de presentar una variabilidad alta (véase la figura 4.1) y no presentar sesgos que impactasen en la precisión del os algoritmos.

Figura 4.1: Diferentes apariencias que adoptan las señales de tráfico. Esta variedad es uno de los principales retos en los problemas de clasificación.

La adquisición de este tipo de información supone un coste alto de recursos y tiempo y no siempre es viable. Por ello se decidió recurrir a la generación de datos sintéticos, generados a partir de diferentes fuentes y con técnicas distintas. (1) Por un lado se generaron imágenes partiendo del renderizado de un simulador de conducción, (2) por otro, se generaron a partir de unas imágenes canónicas seleccionadas de fuentes públicas en internet y se las sometió a un intenso proceso de aumentación para modelar las diferentes situaciones que podríamos encontrar en un entorno real. Las técnicas aplicadas en este proceso se han introducido parcialmente en el capítulo 2 y se volverá sobre ellas en un apartado posterior.

En lo referente al análisis, hubo en concreto dos puntos que motivaron su origen: Por un lado establecer si con datos totalmente sintéticos era factible entrenar modelos que luego se aplicarían sobre datos reales y por otro, comprobar si se producían sesgos en la comprobación de la precisión de los diferentes modelos que se presentaban al entrenarlos y testearlos con imágenes de un mismo dominio, que venía siendo lo habitual en muchos artículos relacionados con esta materia. Las conclusiones derivadas de los resultados de la experimentación de las pruebas mostraban este tipo de sesgo y concluían que el modelo entrenado con imagenes sintéticas podía competir en igualdad de condiciones en experimentaciones con dominios cruzados con modelos entrenados con datos reales, superándolos en muchos casos. Si bien es cierto que para

el dominio específico con cuyos datos reales se había entrenado el modelo, este era precisamente el que obtenía un mejor resultado al compararlo contra el sintético.

La ejecución en tiempo real fue un criterio a valorar que nos llevó por diferentes caminos. Nos enfrentamos primero a la elaboración de un algoritmo de detección multi-escala que no fuese computacionalmente inabordable como lo es los algoritmos de detección basados en muestreo por ventana deslizante. Para ello en la primera aproximación se utilizó un sistema de jerarquías de detectores SVM que analizaban la imagen en busca de muestras, dividiendo estas muestras en regiones pertenecientes a diferentes tipos de señal, basándonos en primitivas geométricas (triangulares, cuadradas, redondas).

La aproximación más tradicional a la hora de trabajar con imágenes en visión artificial pasa por obtener una representación de esa imagen en el espacio de características. Esta representación suele adoptar la forma de un vector de números, llamado también vector de características, y una vez obtenida es lo que se analizará para determinar la información asociada a ese parche de imagen. Con esta representación numérica se consiguen dos objetivos principalmente: (1) reducir la dimensionalidad en la información que representa la imagen y (2) eliminar la información espuria que no tiene que ver con el contenido de lo representado en la imagen, quedándonos únicamente con la información que subyace al elemento presente. Estos vectores de características habitualmente, se suelen obtener a partir de filtros que aplica el investigador, recibiendo el nombre de características *hand-crafted* o *hand-engineered*.

Posteriormente y debido a los avances en el hardware de procesamiento acabamos optando por un modelo de detección DL para esta primera localización de objetos. Con la aparición del DL, es el propio sistema el que tras someterlo a un proceso de entrenamiento supervisado o no-supervisado, establece y genera los filtros adecuados para representar la información dentro de los datos que permitirá la diferenciación de los diferentes tipos de elementos presentes en la imagen. Y serán estos filtros los que determinarán el vector de características que representará a cada uno de los elementos que se le presente a la red neuronal. Podríamos decir que es el propio modelo el que, a través del entrenamiento, selecciona las características visuales que va a utilizar para inferir información de la entrada, eliminando al investigador de ese proceso de selección.

Una vez aprendidos estos filtros se podrán aplicar en otros dominios utilizando técnicas de transfer-learning, reentrenando esos modelos aprendidos con información obtenida del dominio objetivo.

Por otro lado, el tratamiento de los falsos positivos (zonas que no son de objetos de interés que el detector identifica como objetos de interés) y negativos (zonas que no se detectan como objetos de interés siendo objetos de interés) fue otro de los puntos problemáticos que había que resolver. Aplicando técnicas de seguimiento a las detecciones y combinándolas con predicciones y detecciones que les proporcionasen validez se resolvió este problema.

Finalmente, la localización de los diferentes elementos detectados con una única cámara fue otro de los puntos que representaba un desafío. Se utilizaron técnicas de calibración monocular para el plano del suelo y a partir de los tamaños esperados de señales se desarrolló un algoritmo para estimar la ubicación espacial de los elementos con respecto a la cámara montada en el vehículo.

Las contribuciones realizadas por esta tesis a este campo son las siguientes: Primero, presentamos un análisis en profundidad del uso únicamente de datos sintéticos en el proceso de aprendizaje de una red neuronal convolucional(CNN) de clasificación simple, aplicada a un sistema TSR. Segundo, proponemos un nuevo método para comparar clasificadores de manera objetiva cuando se dispone de diferentes *datasets* heterogéneos, esto es, cuando los *datasets* tienen diferente cardinalidad y tipos de señales de tráfico. Y tercero, se presenta una propuesta de encarar el entrenamiento de modelos de clasificación, siguiendo las directrices obtenidas de un análisis empírico.

4.4 Estado del Arte

Estado del arte de modelos de clasificación y detección Antes de comenzar a profundizar en el estado del arte relacionado con los procesos de clasificación y detección conviene establecer en primer lugar cual es la diferencia entre estas dos técnicas diferentes, figura 4.2.

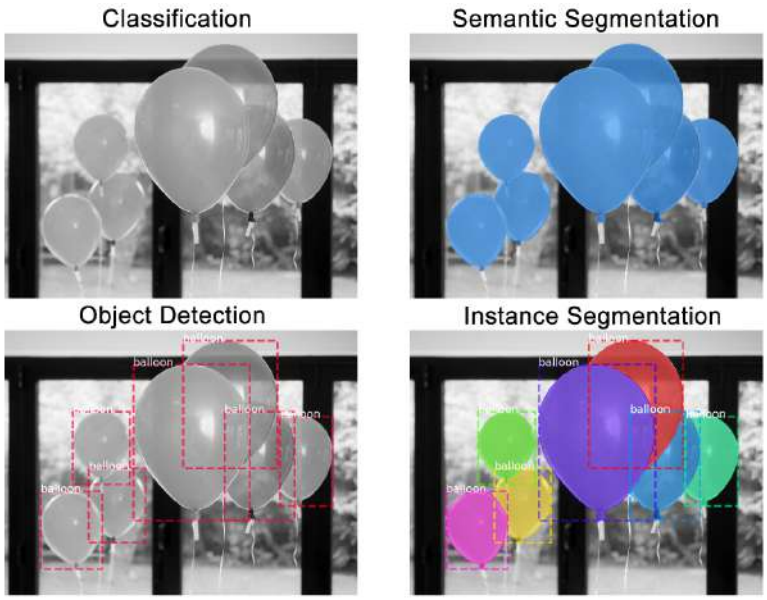


Figura 4.2: Diferencia entre clasificación, detección y segmentación.

Llamamos clasificación cuando a partir de una imagen lo que se espera del modelo es una clasificación, una etiqueta. Observando la imagen superior izquierda de la figura 4.2, un modelo de clasificación respondería a la pregunta: ¿Que objeto o tipo de objetos hay en la imagen? respondiendo diligentemente con la etiqueta globos. Por otro lado, el proceso de detección persigue el objetivo de no sólo identificar sino también localizar espacialmente los diferentes elementos presentes en la imagen. Por este motivo la clasificación se podría entender como algo más particular que el proceso de detección e incluso como parte del mismo. Asociada a la tipificación de un elemento, además, suele proporcionarse también el porcentaje de pertenencia a la clase predicha o en ocasiones, una lista de valores cada uno de ellos representando el porcentaje de pertenencia a una de las clases para las que se aprendió el modelo.

La literatura al respecto es extensa, sobre todo en el ámbito del DL, y está fuera del ámbito de esta disertación hacer un recorrido exhaustivo por cada uno de los métodos de detección existentes. Sin embargo, y para que el lector pueda tener una visión global del estado de la detección en el momento actual, pasaremos brevemente por los métodos y modelos de detección más significativos en el panorama científico.

Figura 4.3: Esquema de detección tradicional y extremo a extremo.

Existen diferentes métodos para acometer el proceso de detección de objetos (véase la figura 4.3). Los métodos de visión artificial tradicionales usados para la detección aparecieron a finales de los 90. Una técnica muy común hasta la aparición del *deep learning* consistía en la realización de un muestreo de la imagen, que vendría seguido por una transformación de las diferentes muestras en el dominio de la imagen al dominio de los datos generando un vector descriptor de características para dicha muestra, y finalmente, la combinación de estas características generadas, con algún algoritmo de *machine learning* como KNN, SVM o Adaboost para su clasificación. Existen diversos estudios a modo de visión general sobre diferentes aproximaciones clásicas al problema de la detección visual de objetos genéricos [Liu et al.19] o más específicos [Zafeiriou et al.15][Sun et al.06] [Enzweiler and Gavrila09]. Desde la perspectiva tradicional los métodos basados en características invariantes a la iluminación como HOG combinadas con clasificadores lineales binarios como el SVM eran habituales. También se utilizaban otras aproximaciones que buscaban puntos característicos utilizando mecanismos de asociación con los puntos característicos de imágenes de referencia de las clases de interés para realizar la clasificación. Esto fue muy habitual hasta la llegada de AlexNet en 2012. A partir de ese momento comenzó a

surgir una tendencia clara a la utilización de modelos de *deep learning* para la detección. Y así, motivados por la explosión aplicativa del DL debido a diferentes razones explicadas en el capítulo 2, muchos procesos de visión artificial han sufrido cambios profundos. Han sido modelados por redes neuronales que realizan el proceso como una función cuya expresión analítica es aproximada tras una etapa de entrenamiento a partir de los datos, sin que el investigador tenga que indicar que procesos generarán el vector de características y que modelo utilizará para separar el espacio multidimensional de datos que utilizará para la clasificación. Una aproximación extremo a extremo (*end-to-end*) en la que todo el proceso se optimiza igual sin diferenciar entre sus partes visualizándose como un sistema compuesto de entrada, modelo y salida. Y frente a los métodos más tradicionales nos encontramos con los basados en DL. Existen infinidad de revisiones sobre todos los campos y aplicaciones que ha impulsado con su imparable propagación [Liu et al.19] [Verschae and Ruiz-del Solar15] [Zhiqiang and Jun17] [Lu et al.19] [Xiao et al.20] [Zhao et al.19], por citar algunos.

Los modelos de detección basados en *deep learning* se suelen dividir típicamente en dos grupos diferenciados: los modelos de detección de objetos de dos etapas y los modelos de detección de objetos de una etapa.

Los modelos de dos etapas tuvieron su arranque con la red RCNN [Girshick et al.14]. Esta red sustituyó los algoritmos de extracción de características tradicionales como HOG o SIFT por una red que utilizó para dicha extracción (*backbone*) y lo combinó con un algoritmo de proposición de regiones. Los pasos que se siguen en este tipo de redes pasan por el algoritmo de proposición de regiones generando una serie de candidatos, de regiones en la imagen que se inyectan en el *backbone* para extraer las características asociadas a cada candidato. Finalmente, las características se clasifican usando un algoritmo SVM. Este nuevo modelo de detección fue evolucionando incluyendo las mejoras que resolvían algunos de los problemas que presentaba. Así, apareció la SPPNET [He et al.14] que añadió una capa de *Spatial Pyramid Pooling* para evitar el recorte y las deformaciones que sufrían las imagenes en el RCNN antes de entrar al backbone. Haciendo además que la inferencia fuese bastante más rápida. Luego apareció la red Fast R-CNN [Girshick15] que proponía un nivel de *pooling* de la región de interés. este modelo presentaba una precisión mayor que las redes anteriores y además el entrenamiento era de extremo a extremo ya que ahora la clasificación

también se realizaba dentro del propio modelo, y el entrenamiento actualizaba todos las capas de la red. En este punto el cuello de botella se había vuelto el algoritmo de proposición de regiones, hasta que llegó el modelo FasterRCNN [Ren et al.15] que mejoró este punto proponiendo una red de proposición de regiones (RPN) en lugar del algoritmo selective search utilizado hasta el momento.

Por otro lado los modelos de una etapa comenzaron con el modelo OverFeat [Sermanet et al.14] que integraban la clasificación y la localización del objeto en una única arquitectura de red. Este modelo era más rápido que su homólogo en los de dos etapas pero era menos preciso. Y así llegamos hasta el modelo Yolo [Redmon et al.16] [Redmon et al.16] [Redmon and Farhadi18] [Bochkovskiy et al.20] [Long et al.20], explicado en el capítulo 2 y que evolucionó del modelo Yolov1 al modelo v5 a día de hoy. Este modelo entraría dentro de este grupo, donde también encontraríamos el modelo SSD (single shot multibox detector) [Liu et al.16] que utiliza como *backbone* VGG16 para la extracción de características y que consiguió una velocidad de proceso mayor que el modelo Yolo.

Técnicas de generación sintética de imágenes El entrenamiento en las redes neuronales presenta el inconveniente de requerir grandes cantidades de datos de entrenamiento. Esto se puede resolver invirtiendo ingentes cantidades de esfuerzo, tiempo y dinero en grabar, anotar y procesar el metraje de vídeo desde vehículos equipados. Aparte del coste, el contenido multimedia obtenido podría no representar de manera adecuada todas las condiciones de exteriores requeridas para obtener un modelo entrenado tan genérico como fuera posible.

Las técnicas de aumentación de datos [Shorten and Khoshgoftaar19] [Bloice et al.19] son una aproximación popular a la hora de resolver este inconveniente ya que añaden variabilidad al espacio de datos de las señales de tráfico que los modelos aprenden. La mayor parte de las técnicas que se aplican actualmente se diseñan de manera manual y no siempre producen los resultados deseados. En [Afifi and Brown19], los autores exponen que los métodos de constancia de color computacional usados a día de hoy (también conocido como balance de color) no generan resultados de aspecto real; por lo tanto, proponen un método nuevo para generar imágenes más realistas capaces de alimentar procesos de entrenamiento. Las tecnologías de aumentación de datos han probado una alta efectividad en procesos de clasificación [Perez and Wang17]. En [Cubuk et al.19], los autores proponen una

búsqueda automática para políticas de aumentación de datos mejoradas en lugar de aplicar de manera aleatoria procesos de aumentación de datos diferentes [Baird et al.92, Simard et al.03, Krizhevsky et al.12]. Una vez se realiza la aumentación de datos, el *dataset* de salida se usa para entrenar modelos que eviten las limitaciones impuestas por una adquisición de datos real.

Hay también otros métodos relacionados con la generación de datos sintéticos, tales como [Richter et al.16] donde se propone el uso de la salida de juegos comerciales para generar datos de ground-truth de alta calidad, con precisión a nivel de píxel y de gran escala para entrenar sistemas de segmentación semántica sin acceder al código fuente del juego. El trabajo realizado en [Kar et al.19] muestra la generación de escenarios de carretera 3D con un modelo generativo entrenado previamente. En [Wang et al.19], se usaron datos etiquetados generados sintéticamente en el contexto de conteo de multitudes para proponer una red totalmente convolucional entrenada con estos datos sintéticos y afinándola (finetune) con datos reales. Otro estudio analiza la manera de transferir los efectos generados por la cámara en el proceso de adquisición de datos (aberraciones cromáticas, difuminados, exposición excesiva o insuficiente, ruido) a los datos sintéticos y proponen una secuencia de pasos novedosa [Carlson et al.18].

En [Mogelmose et al.12], los autores analizan la aplicación de datos sintéticos al entrenamiento de modelos de detección y concluyen que los datos sintéticos de entrenamiento para detección no producen buenos resultados. Más adelante, en el campo de la detección, los autores de [Tremblay et al.18] usan una técnica llamada *domain randomization* para generar muestras sintéticas para entrenar detectores DNN. Esta técnica implica la aleatorización de los parámetros del simulador para generar escenarios con condiciones diferentes. Más relacionado con el mundo de la clasificación, [Chigorin and Konushin13] entrenaron un clasificador con imágenes generadas de manera sintética, cambiando el ámbito del dominio de la imagen un *dataset* ruso de señales de tráfico.

En [Moiseev et al.13] los autores intentaron demostrar que es posible entrenar una red neuronal usando únicamente datos sintéticos mejorando los resultados obtenidos con la misma red entrenada con datos reales. Compararon un K-NN, un LDA, un SVM y una CNN con dos niveles de convolución y dos niveles totalmente conectados. Sin embargo, usaron únicamente dos *datasets* diferentes (GTSRB y STSD) y compararon los

resultados de evaluar sus muestras de test de una manera clásica. Los autores entrenaron los clasificadores con imágenes obtenidas de la wikipedia, aplicando diferentes procesos de aumentación para generar el *dataset* y usando HOG como características para entrenar los clasificadores más tradicionales. Concluyeron que utilizando los datos sintéticos se obtenían mejores datos.

En otros dominios, [Ranjan et al.18] renderizaron 1M de imágenes de ojo sintéticas usando *ray tracing* para mejorar el entrenamiento de los modelos, incorporando así información de imágenes nuevas en el modelo utilizando transfer learning. Entrenando con datos sintéticos primero y luego haciendo ajuste fino con datos reales. De esta manera el modelo no se entrena desde el principio sino a partir de un modelo entrenado ya con una serie de datos y se aprovecha la configuración actual del modelo entrenado para utilizarlo como punto de inicio del modelo a entrenar, traduciéndose esto en la necesidad de una cantidad inferior de datos para el nuevo entrenamiento.

Los sistemas de ayuda avanzada a la conducción (ADAS) son los sistemas nucleares que guiarán la conducción hacia escenarios con vehículos autónomos en el futuro cercano. Existe literatura científica extensa sobre estos sistemas, como por ejemplo las siguientes revisiones [B.V. and Karthikeyan18] [Velez et al.15] [Satzoda and Trivedi17]. En esta tesis, nos hemos enfocado en los sistemas TSR que son un elemento crucial requerido no únicamente por los ADAS, sino también por los sistemas aplicados a vehículos autónomos, el mapeo móvil y los sistemas de inspección visual de la seguridad de la carretera. A lo largo de las últimas décadas y antes de la adopción de aproximación DL, se han utilizado diferentes técnicas de aprendizaje máquina en sistemas TSR [Aghdam et al.17, Wali15]. Un logro significativo se alcanzó en 2011, cuando fue presentado el benchmark de reconocimiento de señales de tráfico alemanas (GTSRB) [Ciresan et al.11]. Desde entonces, el *dataset* propuesto ha sido adoptado como referencia por muchos autores. En este desafío, la clasificación de las señales de tráfico sobrepasó el rendimiento humano usando aproximaciones *deep learning*. En [Ciresan et al.11], los autores, por ejemplo, utilizaron un comité de redes neuronales para mejorar la precisión humana, alcanzando una tasa de acierto del 99,15 %. Después de los primeros resultados del desafío, algunas aproximaciones alcanzaron esas cotas en el ranking. Posteriormente en el 2012, el mismo equipo [Cireşan et al.12] mejoró la precisión del reconocimiento llegando aun 99,46 % combinando varias redes neuronales (DNN) en una DNN muti-columna, haciendo así

al sistema insensible a las variaciones de contraste e iluminación. En [Jin et al.14], se usaron veinte CNNs independientes con una función de pérdida "bisagra"para obtener una precisión de 99,65 % en el *dataset* GTSRB. Hasta ahora , el mejor resultado en este *dataset* es uno descrito en [Zhu et al.16a], donde los autores usaron una etapa de extracción de propuestas guiada por una red neuronal totalmente convolucional basada en R-CNN y el algoritmo EdgeBox [Zitnick and Dollár14] para alcanzar una precisión del 99,7 %. En [Aghdam et al.16], se introdujo una nueva arquitectura convolucional para reducir el coste operacional manteniendo el 99,55 % de precisión. Los autores usaron una red convolucional optimizada y ligera para detectar las señales de tráfico, usando LReLu como función de activación. El modelo fue entrenando aplicando también procesos de aumentación a los datos para reducir el sobre-ajuste (*over-fitting*). En [Arcos-García et al.18], los autores probaron diferentes algoritmos de optimización de redes y combinaron un modelo con una red de transformación espacial, alcanzando una precisión del 99,7 % en el *dataset* GTSRB. En [Wong et al.18], sin embargo, los autores se enfocaron en mantener una buena precisión 98,9 % mientras incrementaba la eficiencia y reducían el número de parámetros y operaciones al realizar la inferencia.

Habibi et al [Aghdam et al.17, Aghdam et al.16] se enfocaron en obtener una red convolucional ligera para detectar y clasificar las señales de tráfico usando los niveles de la red convolucional para simular una estrategia de ventana deslizante. Esta estrategia fue un intento de minimizar el coste computacional.

La mayoría de estas aproximaciones están centradas en las etapas de reconocimiento y detección. Otras, sin embargo, se enfocan en el funcionamiento del sistema en general.

Así, en [Timofte et al.14], los autores proponen un sistema completo que puede detectar, clasificar y localizar las señales que aparecen en la imagen en tres dimensiones. Combinaron un sistema de detección de señales de tráfico basada en una única vista bidimensional con geometría de escena multi-vista en un proceso de optimización de múltiples vistas monoculares off-line. En [Zhu et al.19] desarrollan un algoritmo de detección y reconocimiento de señales de tráfico basado en RetinaNet. En [Doval et al.19], los autores proponen un sistema completo para detectar, clasificar y localizar la posición tridimensional de las señales detectadas usando Yolov3 para detección y tipificación, y un sistema estéreo para la estimación tridimensional de la

señal. Generan además un nuevo *dataset* llamado LSITSD. Finalmente, en [Wali et al.19], los autores analizan el estado actual de los TSRs y las futuras tendencias y desafíos. En el trabajo realizado por [William et al.19] evalúan diferentes detectores (F-RCNN, SSD, Tiny-YOLOv2) con diferentes generadores de características (MobileNet v1, Inception v2) aplicados a la detección de señales de tráfico sobre el German Traffic Signs Detection Benchmark (GTSDB) analizando precisión y tiempo de ejecución obteniendo las mejores precisiones con F-RCNN e Inception v2 y el mejor tiempo de ejecución con Tiny-YOLOv2. En [Tabernik and Skočaj19] utilizan el modelo de segmentación Mask R-CNN para resolver el pipeline completo de detección y reconocimiento contra un *dataset*, que aprovechan para presentar en el artículo, de 200 categorías.

4.5 Vista general del sistema

El sistema de reconocimiento de señales de tráfico que se desarrolló en el proyecto *Inlane* se basó en una serie de módulos ejecutándose de manera concurrente, que resolvían diferentes tareas: módulo de detección, módulo de seguimiento, de clasificación, de localización y de captura.

El sistema estaba basado en cuatros módulos principales, en rectángulos coloreados en la figura 4.4. Previendo de antemano que la tarea de detección de señales era bastante pesada y que la elección de la estrategia de muestreo junto con la clasificación de cada muestra iba a ser computacionalmente muy costoso, se optó por realizar la detección utilizando descriptores HOG y clasificadores SVM, ya que tras la revisión bibliográfica pertinente, la combinación de HOG-SVM parecía prometedora y comenzamos a andar el camino en esa dirección.

Establecimos para empezar el algoritmo de detección una estrategia de muestreo de ventana deslizante multiescala que nutriría a los diferentes detectores SVM binarios entrenados con antelación en cada uno de los diferentes tipos de señales que nos interesaban, atendiendo a su estructura geométrica (circulares, romboidales, cuadradas, triangulares). Estos clasificadores manejaban descriptores de 4356 dimensiones y con ellos se perseguía el objetivo de disminuir la tasa de *true negatives*, esto es, aumentar el recall o sensitividad, de tal manera que no nos dejásemos ninguna señal sin detectar en la imagen. Esto aumentó el número de falsos positivos candidatos,

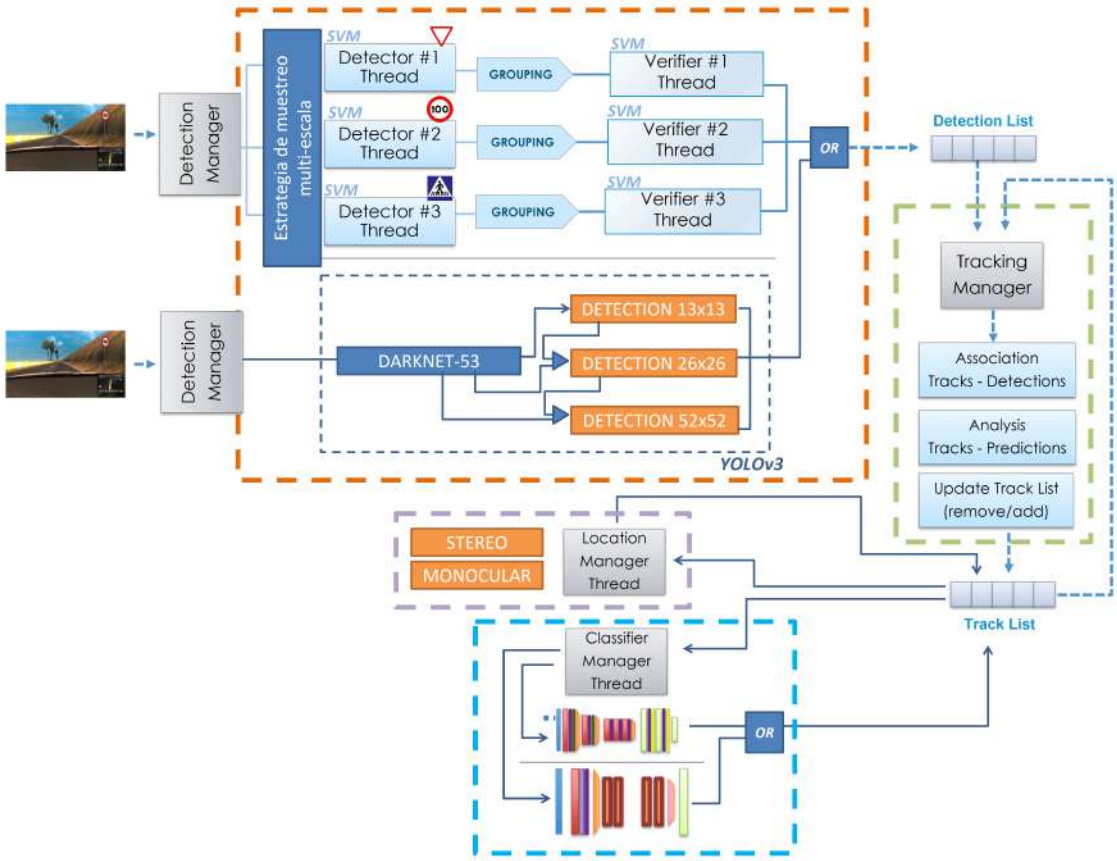


Figura 4.4: Arquitectura modular del sistema de reconocimiento de señales. En naranja discontinuo el módulo de detección, en verde discontinuo el módulo de seguimiento, en púrpura discontinuo el módulo de localización y en azul discontinuo el módulo de clasificación.

por lo que decidimos dividir en dos etapas la fase de detección: tendríamos por un lado, una etapa de detectores SVM binarios que en paralelo hacían las veces de detector multiclase y tras una fase de agrupación de candidatos, una segunda etapa de verificación. Esta etapa manejaba vectores descriptores de 144 valores por lo que era más ligera; además únicamente se aplicaba a las muestras que el primer nivel de detectores dejaba pasar. ¿Su finalidad? Descartar falsos positivos para cada tipo de señal. Una vez generadas las detecciones estas pasaban al módulo de seguimiento, donde se combinaban con las señales para las que se estaba realizando ya un seguimiento previo, de aquí en adelante *tracks*. Estos dos conjuntos de datos se asociaban mediante métricas de distancia y similitud. También se realizaba un proceso predictivo de posición basado en el filtro de kalman, para cada *track* para el que no se había encontrado una asociación previa entre las detecciones. Si esta predicción nos ofrecía una región de la imagen cuya similitud con la señal fuese parecida, entonces aun no habiendo detección nueva, la señal analizada se consideraba asociada. Finalmente en este módulo de seguimiento se llegaba a una fase de actualización de los diferentes *tracks*. Se podían dar diferentes situaciones en este punto. Podíamos encontrarnos con señales que habían sido detectadas para las que no se había encontrado un *track*, esto denotaba que había que añadir un *track* nuevo a partir de esa señal, que pasaría en un estado de candidato a *track* durante unos cuantos frames, antes de convertirse en un *track* real. Con esto evitábamos los posibles errores generados por la detección, ya que los errores no solían tener consistencia temporal. Podía ocurrir también, que un *track* se hubiese quedado sin asociación, momento en el cuál se le activaba un contador de pérdida, que de no recuperarse inevitablemente en un lapso arbitrario pero corto de tiempo lo convertiría en una *track* a eliminar, considerándose entonces que había desaparecido de la imagen. Así el sistema de seguimiento continuaba con la gestión de esta lista de *tracks* que a la postre eran la información visual que se le devolvía al usuario. Entretanto y mientras el tracking era agnóstico a la clase específica de los diferentes *tracks*, un tercer módulo se encargaba de ir confiriéndoles etiquetas a los *tracks* en curso. Cuando un *track* tenía en su histórico un número suficiente de tipificaciones coherentes, era entonces cuando se le advertía al usuario del tipo de señal detectada. Al ser, esta clasificación un procedimiento costoso, se realizaba de manera concurrente e independiente al resto de procesos. Con el paso del tiempo y las mejoras aportadas por la tecnología en lo

referente a la potencia de cálculo se decidió pasar a un sistema *end-to-end* basado en DL. Así se optó por un modelo que en el momento de su aparición ofrecía tasas de imágenes muy tentadoras, el Yolov3 del que ya se ha hablado previamente.

En lo referente al módulo de clasificación se comenzó utilizando el modelo AlexNet obtenido del análisis que se describe en el siguiente apartado. Este modelo realizaba la inferencia en un hilo independiente, accediendo a los *tracks* disponibles y siguiendo una serie de reglas de actualización, asignaba al nivel de histórico del *track* que correspondía la clase que infería. En el proceso de actualización de los *tracks*, una de las tareas que se realizaba con respecto a la clasificación es determinar a partir del histórico de clasificaciones que estaban asignadas, el tipo de señal al que pertenecía el *track*, dándole mayor peso a las clases de la zona intermedia del *track*. Posteriormente se ha reemplazado el modelo AlexNet con un modelo con una capacidad representativa mayor, el ResNet, para tratar de aumentar la precisión de la clasificación.

Algorithm 5: Algoritmo de seguimiento

```
1  I =Imagen;
2  D =Lista de detecciones;
3  T =Lista de tracks;
4  th_iou =Umbral para el IoU entre bounding boxes;
5  th_correlacion =Umbral mínimo de correlacion entre parches;
6  DoTrack (I, D, T)
       input  :la imagen, la lista de detecciones y la lista de tracks actuales.
       output :la lista de tracks actuales actualizada
7      A = CreateAssociationMatrix(T, D);
8      foreach t_i ∈ T do
9          Associate(I, D, t_i, A, th_iou, th_correlacion);
10     updateTracks(I, T, D);
11 Associate (I, D, t, A, th_iou, th_correlation)
       input  :la imagen, la lista de detecciones, el track a asociar y los umbrales de
                asociación.
12     iou_i = GetBestIoU(i, A);
13     if iou_i > th_iou then
14         marcar t_i como asociado;
15         marcar d_j como utilizado;
16     else
17         pred_box_i = predecir la posición actual de t_i ;
18         correlation_i = correlate(I(pred_box_i), I(t_i_box));
19         if correlation_i > th_correlation then
20             marcar t_i como asociado con pred_box_i;
21 UpdateTracks (D, T)
       input  :la lista de detecciones y la lista de tracks.
22     foreach t_i ∈ T do
23         if t_i notassociated then
24             t_i_not_found ++;
25         else
26             t_i_not_found = 0;
27             t_i_ocurrences ++;
28             if t_i == proto_track then
29                 if t_i_ocurrences > th_ocurrences then
30                     t_i = track;
31     foreach d_j ∈ D do
32         añadir d_j a T como proto_track;
```

Algorithm 6: Algoritmo de creación de la matriz de asociación.

```
1 I =Imagen;
2 D =Lista de detecciones;
3 T =Lista de tracks;
4 CreateAssociationMatrix (D, T)
      input  :la lista de detecciones y la lista de tracks actuales.
      output :Matriz de Asociacion
5     foreach t_i ∈ T do
6         foreach d_i ∈ D do
7             A_{i,j} = IoU(t_i, d_j);
8     returnA;
```

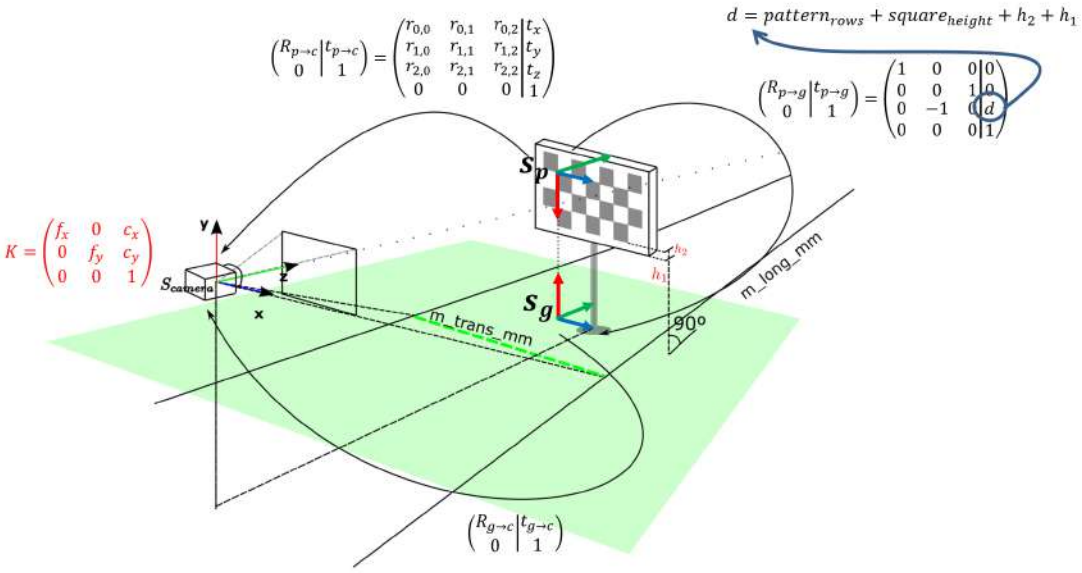


Figura 4.5: Método de calibración del suelo esquematizado.

Finalmente para la localización de la señal en principio, se atacaron dos posibilidades, un sistema monocámara y un sistema estéreo. En el sistema monocámara basamos la detección de la posición de la señal tanto en el tamaño típico de la señal que habíamos detectado como en su proyección con respecto al plano del suelo calibrado previamente. Utilizar el plano del suelo tenía el problema de que la

señal podía no estar en un plano con la misma inclinación que el plano calibrado para la cámara y eso suponía una fuente de error que se intentó mitigar con el tamaño típico de señal.

Para ambos casos, se calibraban lo intrínsecos de la cámara, en rojo en la figura 4.5. Estos parámetros determinan el modelo matemático de la proyección de un punto de la escena en el plano de imagen y están formados por los valores f_x y f_y que representarían el valor de focal en píxeles y cuya relación con la focal de la cámara sería $f_x = s_x f$ y $f_y = s_y f$ donde s_x y s_y serían los píxeles por milímetro en el eje x e y respectivamente, y el punto en el que intersecta el eje óptico el plano de la cámara (c_x, c_y) que suele coincidir con el punto medio de la imagen. Además de estos parámetros, el proceso de calibración también estima los valores que conforman el modelo matemático de la distorsión que produce la propia lente. En resumen el proceso de transformación de un punto tridimensional del mundo (x, y, z) en coordenadas de la cámara, en un punto bidimensional del plano (x_i, y_i), se rige en una cámara pinhole por las siguientes ecuaciones:

$$
\begin{aligned}
x' &= \frac{x_c}{z_c} \\
y' &= \frac{y_c}{z_c} \\
r &= x'^2 + y'^2 \\
x_d &= x' \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 k_6 r^6} + 2 p_1 x' y' + p_2 (r^2 + 2 x'^2) \\
y_d &= y' \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 k_6 r^6} + p_1 (r^2 + 2 y'^2) + 2 p_2 x' y' \\
\begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} &\triangleq \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_d \\ y_d \\ 1 \end{pmatrix}
\end{aligned}
\tag{4.1}
$$

Esta calibración de intrínsecos se realiza siguiendo un método definido en [Tsai86]. Este método consiste en obtener los parámetros de la cámara a partir de la asociación entre puntos tridimensionales del mundo y puntos en la imagen. Estas asociaciones se realizan detectando un patrón de dimensiones conocidas en las imágenes. Esta calibración también proporciona la calibración de extrínsecos que son los parámetros

de rotación y traslación para llevar un punto desde un sistema de coordenadas diferente del de la cámara (uno establecido en el patrón por convenio) al sistema de coordenadas de la cámara. Esto es debido a que normalmente no se puede medir de manera relativamente sencilla la posición del patrón con respecto de la cámara así que hay que añadir un sistema de coordenadas adicional que por conveniencia, se sitúa en uno de los puntos del patrón. Así pues el proceso de calibración también calcularía esta transformación entre sistemas de coordenadas (véase la ecuación 4.2).

$$\begin{pmatrix} x_c \\ y_c \\ z_c \\ 1 \end{pmatrix} = P \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} = \left(\begin{array}{ccc|c} r_{0,0} & r_{0,1} & r_{0,2} & t_x \\ r_{1,0} & r_{1,1} & r_{1,2} & t_y \\ r_{2,0} & r_{2,1} & r_{2,2} & t_z \\ 0 & 0 & 0 & 1 \end{array}\right) \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} \tag{4.2}$$

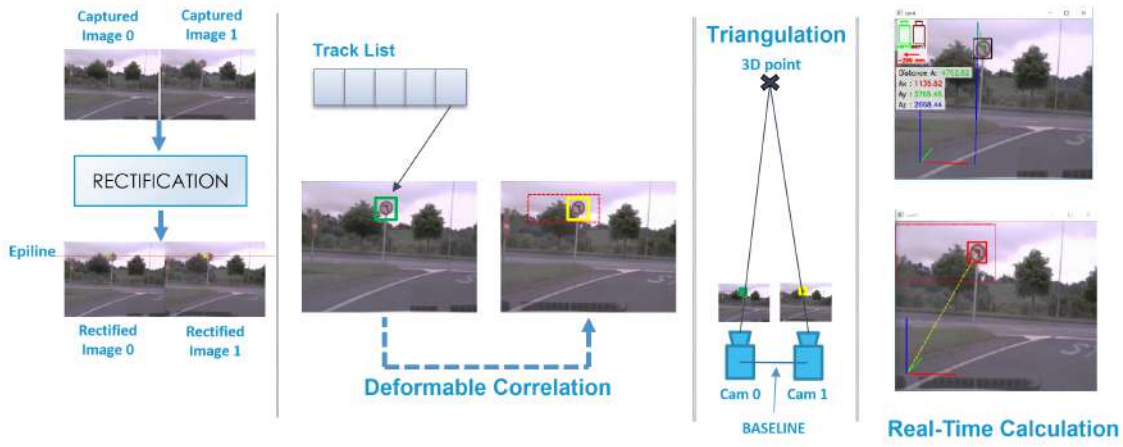
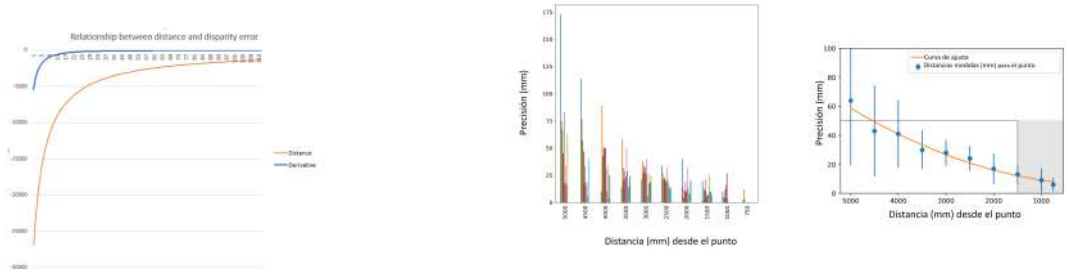


Figura 4.6: Cálculo de la localización de una señal utilizando triangulación a partir de un par estéreo calibrado.

En lo referente al par estéreo, el sistema en general funcionaría de manera parecida realizando una detección y clasificación con una única cámara y es en el módulo de localización donde encontraríamos una diferencia sustancial. En este caso las dos cámaras estarían calibradas entre sí y el modo de proceder tras la detección sería, localizar la señal detectada en una cámara en la imagen de la otra cámara y mediante triangulación ubicarla tridimensionalmente. De este modo los errores que manejamos



(a) (b)

Figura 4.7: (a) Análisis entre la disparidad y la distancia. (b) Análisis de mediciones realizadas sobre un punto.

si bien es cierto que con la distancia aumentarían a partir de los dos metros y media rondarían los cinco centímetros de error con respecto al par estéreo.

Se realizaron dos análisis en relación al error de distancias cometido con el par estéreo: uno a nivel más teórico para entender como evolucionaban las distancias con respecto a la disparidad y otro más empírico. Este último consistía en un análisis de la evolución del error para un punto concreto, midiendo su distancia con relación a un punto de referencia y acercando dicho punto en sucesivas iteraciones hacia el par estéreo. La distancia se medía posteriormente con un medidos laser y se comparaba con la medida estimada. Esta medición se repetía varias veces para el punto a cada distancia y también se realizó para diferentes posiciones (x, y) del punto de medida con parecidos resultados a la figura 4.7. Los resultados obtenidos con las diferentes pruebas establecieron un error menor de 5 cm a partir aproximadamente de los 4 metros de distancia al punto de referencia.

En lo referente a la validación de este sistema nos encontramos en el proceso de la generación de un ground-truth representativo para detección que junto con los resultados obtenidos con la aplicación del nuevo modelo de clasificación formarán parte del contenido de la futura publicación de un artículo nuevo. En este artículo pretendemos analizar y describir los resultados tanto de tiempos de ejecución como de precisión del sistema en general.

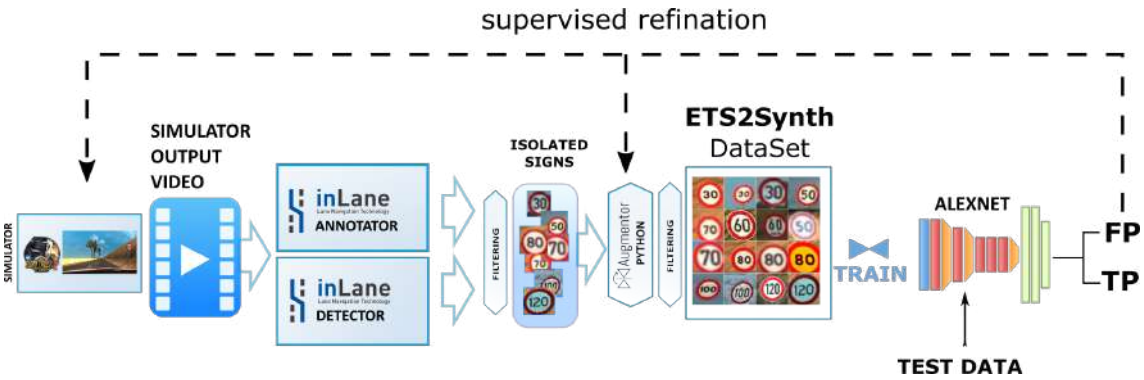


Figura 4.8: Secuencia de etapas para generar el *dataset* ETS2Synth. Se genera una secuencia de vídeo desde el software ETS2. Posteriormente, se realiza una etapa de etiquetado, aumentación y validación, antes de continuar con la etapa de entrenamiento para obtener un modelo entrenado. Testear este clasificador con datos reales guiará el proceso, de manera iterativa para incorporar nuevos elementos en la escena o nuevos procesos en las aumentaciones.

4.6 Metodología empleada

4.6.1 Métodos de generación de señales sintéticas

Para minimizar el ingente coste de producir y anotar datos reales, proponemos la utilización de *datasets* sintéticos que pueden usarse tanto para entrenar modelos como para generar datos para la etapa de inferencia. A continuación, explicamos el método propuesto para elaborar la base de datos para entrenar un clasificador de señales de tráfico utilizando únicamente imágenes sintéticas.

Se han utilizado dos métodos diferentes para producir dos *datasets* sintéticos (véanse las figuras 4.8 y 4.9). Por un lado, se siguió una aproximación basada en simuladores para generar un *dataset* que llamamos ETS2Synth. Por otro lado, se siguió otra línea, basada en aumentación directa, para la creación de otro *dataset* al que llamamos Synth.

En este experimento, se seleccionó un subconjunto de clases de señales de tráfico para el análisis. Este subconjunto, llamado Subconjunto de Clases Seleccionadas (SCS)(véase la figura 4.10), se creó en aras de una mayor agilidad en relación a los entrenamientos, bajo la premisa de que el comportamiento del clasificador no

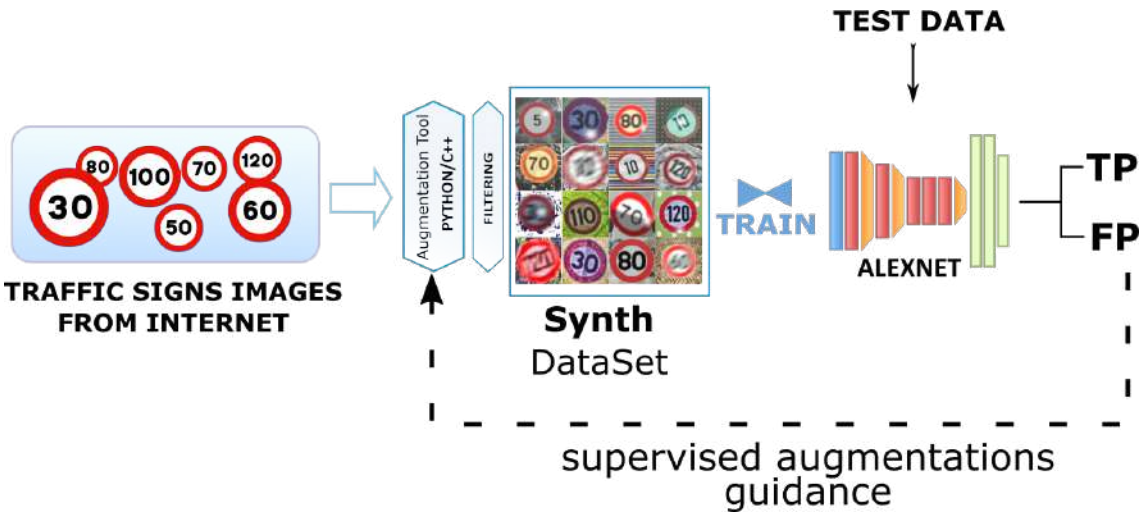


Figura 4.9: Secuencia de etapas para la generación del *dataset* Synth. Se recogen imágenes canónicas originales de internet , se pasa posteriormente a una etapa de aumentación de datos, y se entrena el clasificador. Después de testear el clasificador con *datasets* reales, se utilizan los falsos positivos para guiar iterativamente la adición de nuevos procesos de aumentación a la generación del *dataset*.

Figura 4.10: Subconjunto de Clases Seleccionadas (SCS).

cambiaba cuando el dominio se volvía mayor, ya que este subconjunto es, estadísticamente hablando, suficientemente representativo.

El SCS está compuesto por un grupo de señales de tráfico con diferencias visuales claras, haciéndolo más fácil para clasificar, y por otro grupo de señales de limitación de velocidad con una mayor similaridad inter-clases (relación entre las diferentes clases), haciéndolas más difíciles de clasificar.

4.6.1.1 Generación de *dataset* basado en simuladores (*Dataset* ETS2Synth)

Como se puede ver en la figura 4.8, para obtener el *dataset*, el proceso comienza con la generación de un vídeo a partir de la salida de un motor software de simulación donde aparecen las señales de tráfico requeridas, siguiendo, posteriormente, la secuencia de pasos. La etapa de adquisición fue realizada capturando imágenes desde un vídeo generado desde la salida de renderizado del simulador Euro Trucks Simulator 2. Hay algunos beneficios en usar simuladores en lugar de simples imágenes:

- Es posible obtener en poco tiempo, secuencias de vídeo largas y densas (con gran cantidad de elementos de interés) de un escenario concreto, permitiendo así la generación de bases de datos considerablemente grandes con relativamente poco esfuerzo.
- Es posible, de igual modo, controlar el tiempo, la iluminación e incluso las especificaciones del sensor, produciendo, por lo tanto, imágenes que de otro modo serían muy difíciles de obtener.
- En algunos casos, la información tridimensional (es decir, la localización de las señales) está disponible desde el simulador para poder utilizarla posteriormente en el proceso de anotación (si bien es cierto que esto no ocurre en el software que seleccionamos para este estudio).

Hoy en día, es fácil encontrar fuentes alternativas para generar datos sintéticos muy realistas, especialmente usando motores de renderizado de video juegos. Las alternativas en el mercado son numerosas pero usualmente no ofrecen la posibilidad de controlar todas las condiciones que pueden darse en escenarios de carretera (condiciones meteorológicas, texturas de señales, localización de señales relativas a la

cámara, etc). Por eso la selección final pasa por establecer una compensación equilibrada, un término medio si se quiere, entre lo que te aporta y lo que no.

Existen diversos simuladores para la generación de datos sintéticos, tales como EuroTruck Simulator 2 [Simulator12], NVIDIA AutoSIM, Prescan[Prescan], Carla[Dosovitskiy et al.17], AirSim[Shah et al.17], Bus Simulator[Studios16], Grand Theft Auto[Design13]. En el mundo de la automoción. Prescan es uno de los más ampliamente utilizados porque puede definir diferentes escenarios personalizados y eventos en estos escenarios usando un lenguaje de script. Sin embargo, para obtener imágenes más realistas nos hemos enfocado en el Eurotruck Simulator 2 [Simulator12], que posee un motor de renderizado superior con numerosos parámetros relacionados con la escena.

Este software también le permite al usuario crear sus circuitos personalizados, controlar la luz ambiental, las condiciones meteorológicas y añadir señales de tráfico de diferentes países. En nuestro caso, para aumentar la frecuencia de aparición de señales, se modeló un circuito simple con señales de varios países diferentes, dispuestas en intervalos de 10 m aprox.

Se generaron dos horas de metraje (secuencias de aproximadamente 10 minutos), incluyendo escenarios con lluvia (intensa y ligera), niebla, soleados, nubosos y condiciones diurnas y nocturnas (véase la figura 4.11). Tras la aumentación, el *dataset* contenía 17,297 imágenes de señales de velocidad de limitación de velocidad.

Una vez grabados los vídeos con las imágenes renderizadas por el simulador, estos se etiquetaban siguiendo uno de los dos caminos propuestos a continuación: automáticamente (usando un detector de señales de tráfico existente) o manualmente, vía anotación manual usando una aplicación de anotación. Para el proceso de anotación manual, se analizaron diferentes *softwares* públicos presentes a día de hoy en internet. Finalmente, al no encontrar ninguno que se adaptase exactamente a lo que necesitábamos, y teniendo claro cuáles eran las funcionalidades que precisábamos para realizar un proceso de anotación menos engorroso y más liviano y fluido, desarrollamos una herramienta de anotación de vídeo [Cortés et al.15] con funcionalidades concretas y muy dirigidas a agilizar y acelerar el proceso de anotación. Esta herramienta permitía la posibilidad de capturar desde vídeos con diferentes codificaciones o desde carpetas con imágenes. Nos permitía de igual modo, salvar los

(a) Image at sunset (b) Clear day

(c) Foggy day (b) Image at night

Figura 4.11: Images taken from ETS2 simulator custom circuit.

datos en formato VCD [1], usado para almacenar las anotaciones en disco, con la posibilidad de convertir a formatos específicos de otros *frameworks* para *deep learning*. Entre las funciones que se implementaron introdujimos la capacidad de cortar y copiar nuevas detecciones, navegar con la rueda del ratón tanto temporalmente como a nivel de zoom, la posibilidad de aplicar una interpolación lineal entre las posiciones del mismo objeto en dos fotogramas claves y la realización de seguimiento activo mediante flujo óptico de una zona anotada.

[1] https://vcd.vicomtech.org/

Tabla 4.1: Lista de herramientas de anotación

Annotator Name	Features	License	Dependencies
LabelImg	PASCAL VOC format	MIT License	python
cvat	custom format	MIT License	web-based
Viulab	custom format (VAT)	propietary license	c++
APS2.0	custom format	propietary license	c++
VIA url	custom format	BSD 2-clause "Simplified"License	web-based
ALT	KITTI *dataset* format	GPL	Fiji plugin
LabelMe	custom format	MIT License	web-based
LabelBox	custom format	Apache License	web-based
PolygonRR	custom format	Custom License	web-based
RectLabel	custom format	Custom License	python
Hitachi Segmentation Editor	custom format	MIT License	web
Dataturks	custom format	propietary	web
Anno-Mage	custom format	Apache License 2.0	python
BBox Label Tool	Boxes	MIT License	python
makesense	Boxes	GPLv3	web
coco annotator	Boxes	MIT License	web
VoTT	Boxes	MIT License	web
YoloMark	Boxes	The Unlicense License	web
SuperAnnotate	Custom format	commertial (free demo)	desktop

Para la alternativa automática, utilizamos un detector basado en clasificadores multi clase SVM sobre características HOG desarrollado en el contexto del proyecto Europeo Inlane del H2020 [Commission18].

Sin embargo, la aproximación automática se enfrenta a ciertos inconvenientes. Típicamente, los detectores proponen no sólo elementos de interés si no también falsos positivos que el usuario debería filtrar en un paso de validación manual. En el caso del detector de INLANE, se añadió una etapa de verificación manual porque este detector usa una estrategia de muestreo de ventana deslizante y se enfoca en identificar las formas geométricas de las señales de tráfico, y por tanto, genera algunas muestras que se corresponden con falsos positivos que deben de ser descartadas.

Una vez esta selección se completa, el núcleo de la base de datos se somete a una etapa de aumentación. No obstante, algunas de las distorsiones deseadas las produce

por el propio simulador. Por ejemplo, distorsiones relacionadas con la perspectiva, condiciones climatológicas o oclusiones, parámetros propios de la cámara, etc. Por lo tanto, no es necesario realizar un proceso de aumentación intensivo para estas imágenes. Para esta tarea de aumentación en concreto, se utilizó un programa codificado en python llamado Augmentor [Bloice et al.19]. Define una secuencia de pasos con los procesos a aplicar y una probabilidad de aplicación para cada uno de dichos procesos. La salida de este proceso de aumentación conforma el *dataset* ETS2Synth. Finalmente, el modelo de clasificación es entrenado a partir del *dataset* de salida y testeado contra un *dataset* real. Para la generación del *dataset* definitivo, será necesario iterar sobre la secuencia de pasos del diagrama definido en la figura 4.8 hasta que la precisión del modelo contra los datos reales sea adecuada. Estas iteraciones estarán guiadas por los falsos positivos del test y definirán qué nuevas señales habría que incorporar al escenario virtual, los cambios de parámetros en dicho escenario y los nuevos procesos de aumentación sobre las señales detectadas.

4.6.1.2 **Generación de *dataset* basado en aumentación de imágenes canónicas (*dataset* Synth)**

La segunda aproximación (véase la figura 4.9) implica la aplicación de un proceso de aumentación intensivo a un conjunto de imágenes canónicas recogidas de internet [wikimediaa] [wikimediab] [wikipediaa]. Comparada con la aproximación basada en el simulador, hay principalmente dos ventajas que merece la pena mencionar: el tiempo requerido para generar muestras es considerablemente menor y la varianza intra-clase (dentro de la clase) de los datos generados por aumentación es mayor, aunque no necesariamente más realista.

Primero, se escogen unas pocas imágenes por país y tipo para crear el conjunto de datos canónico. Posteriormente, se inicia un proceso empírico e iterativo de aumentación de datos en el que se aplican diferentes técnicas para transformar las imágenes. Después de la etapa de aumentación, el modelo seleccionado es clasificado primero, y testeado contra imágenes reales posteriormente. Esto producirá un subconjunto de true positives y otro subconjunto de false positives que definirán cierta precisión para este modelo. Es precisamente el análisis empírico de este subconjunto de falsos positivos lo que guiará el tipo de aumentaciones aplicadas en la siguiente

iteración hasta obtener los resultados deseados (es decir, una precisión aceptable), al igual que ocurría en la aproximación anterior.

Aumentación de datos

El proceso de aumentación de datos ha probado ser tremendamente útil en diferentes trabajos [A. Mikolajczyk18] [Arcos-García et al.18] [Wong et al.18]. Sin embargo, no es evidente definir una secuencia de aumentaciones adecuada que genere muestras útiles que mejoren la base de datos existente.

Desarrollamos una herramienta de aumentación personalizada basada en los procesos aplicados típicamente con modificaciones en los parámetros que manejaban e incorporándole además nuevos procesos que consideramos de presencia ineludible (sombras estructurales y brillos especulares), ejecutándose en el mismo modo de secuencia de procesos de aumentación estocástico que el Augmentor.

Así, son diferentes los procesos de aumentación que se han aplicado a las imágenes canónicas: variaciones de luminosidad, adición de fondo, transformaciones afines y de perspectiva, difuminados, adición de ruido, normalización de histograma, distorsiones elásticas, padding, sombras, brillos especulares y operaciones morfológicas, etc. Estos procesos emulan la distorsión producida por los dispositivos de captura y por las condiciones del escenario y los elementos de interés, que pueden distorsionar las imágenes de diferentes maneras. Algunos ejemplos de aumentación se ilustran en la figura 4.12

Figura 4.12: Ejemplo de procesos de transformación aplicados a una imagen. De izquierda a derecha: original, escalado, cambio de iluminación, sombra, deformación elástica, rotación, operador morfológico, ruido, reflejo especular, difuminado por movimiento, padding, transformación perspectiva, shearing o inclinación.

Para el *dataset* ETS2Synth , se realizó un proceso ligero de aumentación para añadir la variabilidad que el simulador no podía proporcionar. Para el *dataset* Synth, el proceso de aumentación fue considerablemente más intenso donde se modificaban los

parámetros presentados en la tabla 4.2. En ambos casos, se realizó un paso adicional de verificación para evitar introducir muestras corruptas o inútiles en los *datasets*.

4.6.2 Entrenamiento y métricas

El objetivo de este estudio fue determinar si los datos sintéticos pueden reemplazar a los datos reales para reducir el esfuerzo y los recursos dedicados a la adquisición de datos en soluciones basadas en DL. Para empezar, se seleccionó y entrenó una arquitectura CNN con diferentes conjuntos de datos reales y sintéticos y, posteriormente fueron comparados los resultados de dos en dos.

Los estadísticos utilizados para la evaluación del modelo fueron principalmente los verdaderos positivos y falsos negativos para calcular la sensitividad del modelo o cómo de bien se comporta para una clase determinada. Estos estadísticos se explican en la figura 4.13.

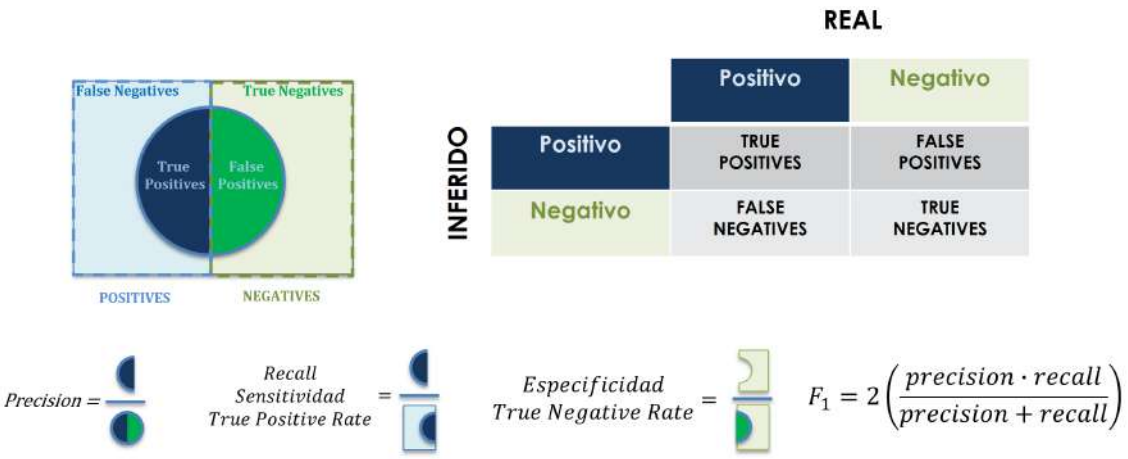


Figura 4.13: Estadísticos empleados en la validación.

El modelo seleccionado fue un modelo de clasificación AlexNet [Krizhevsky et al.12] (véase la figura 2.40), que ganó el desafío de reconocimiento visual de gran escala ImageNet en 2012 y ha sido extensivamente estudiado en la literatura. Seleccionamos un modelo bien conocido y maduro con el objetivo de evitar potenciales fallos en el comportamiento del modelo que nos pudieran desviar de los objetivos del estudio.

Cuando tratamos con *datasets* públicos, es necesario advertir que están, habitualmente, compuestos de un número y tipologías de elementos de interés de lo

NOMBRE	DESCRIPCIÓN	MÉTODO	PARÁMETROS	
			Nombre	Rango
Lighting variations	Luminancia general de la imagen	**HSI Color Space**	α_V	(0.5, 0.9)
			α_S	(0.5, 0.9)
Backgrounds	Fondo tras las señales	Los fondos los proporcionan las texturas del escenario renderizado por el simulador.		
		La incorporación de fondo se ha realizado mediante un proceso de fusión explicado en el apartado 2 con imágenes aleatorias de [Cimpoi et al.14] en el *dataset*.		
Affine Transformation	Transformaciones afines sobre la imagen	traslation	t_x	(-5, 5)
			t_y	(-5,5)
		Rotación	θ_x	(-10, 10)
			θ_y	(-10, 10)
			θ_z	(-35, 35)
		Escalado	s_x	(0.4,0.8)
			s_y	(0.4, 0.8)
		Shear	sh_x	(0, 0.1)
			sh_y	(0, 0.1)
Perspective Transformation	Transformación de perspectiva sobre la imagen	Homography (Calculada a partir de cuatro puntos)	d_x^{tl}	(1, 8)
			d_x^{tr}	(1, 4)
			d_x^{bl}	(1, 4)
			d_x^{br}	(1, 4)
			d_y^{tl}	(1, 4)
			d_y^{tr}	(1, 4)
			d_y^{bl}	(1, 4)
			d_y^{br}	(1, 4)
Blurring	Este proceso simula el desenfoque gausiano y de movimiento	filtro de caja	k_x	(0, 5)
			k_y	(0, 5)
			θ	(0, 2π)
		Gausiano	k_x	(3, 23)
			k_y	(3, 23)
			σ_x	(4, 15)
			σ_y	(2, 5)
			θ	(0, 35)
Ruido	Ruido en la imagen	Gaussian	k	(3, 7)
			μ	(-80, 80)
Cropping	Recorte de diferentes partes de la imagen.	El recortes se hace cuando se anotan las muestras.		
Corrección Gamma	Aplica una corrección Gamma a la imagen	Gamma	γ	(0.0, 10.0)

NOMBRE	DESCRIPCIÓN	MÉTODO	PARÁMETROS	
			Nombre	Rango
Normalización Histograma	Modificaciones en el histograma de la imagen	MIN MAX	min	0
			max	255
		LOG	α	(0.2, 0.8)
		Prune	α	(0.2, 0.3)
		Clahe	xdivs	(8, 12)
			ydivs	(8, 16)
			limit	(0.1, 10.5)
Distortiones	Deformaciones no lineales sobre la imagen	Elastic	σ	(3, 8)
			escala	(30, 40)
			k	(6, 12)
		MLS Wrapping	N	(5, 10)
			M	(5, 10)
			α	(0, 3)
			grid	(1, 20)
			intensity	(4, 6)
		Grid	N	(5, 30)
			M	(5, 30)
Fonts or Aspect	Relación de aspecto de la imágen	Relación de aspecto y fuentes proporcionadas por el simulador.		
Oclussions	Objectos en frente de las señales	Las oclusiones son proporcionadas por el simulador.		
Padding	Añade padding con ceros alrededor de la imágen		P_{left}	(2, 15)
			P_{top}	(2, 15)
			P_{right}	(2, 15)
			P_{bottom}	(2, 15)
shadows	Este proceso añade una sombra sobre la imagen	estructural	α	(0.4, 0.7)
			dataset	Imágenes para generar sombras obtenidas de [Cimpoi et al.14]
		linear	α	(0.1, 0.3)
Specular highlights	Generación de reflejo especular	Gaussiano	w	(0.3,0.5)
			h	(0.3, 0.5)
			d_x	(0, 1)
			d_y	(0, 1)
			σ_x	(0.1, 0.2)
			σ_y	(0.1, 0.2)
			α	(-45, 45)
Morphological Operator	Aplica un operador morfológico de dilatación o erosión sobre la imágen.	dilatación o erosión	k	(3, 5)

Tabla 4.2: Parámetros en los procesos de aumentación.

más heterogéneos. En este caso particular, donde el objeto de interés son señales de tráfico, ocurre de esa manera como puede verse en la figura 4.15. Esto hace que sea difícil comparar clasificadores que han sido entrenados con diferentes *datasets* y esbozar unas conclusiones objetivas y justas como resultado de esta comparación.

Tabla 4.3: Lista de *datasets* públicos online de señales de tráfico

Name	Classes	Country
Stereopolis (2010)	10	FR
STSD [Larsson and Felsberg11] (2011)	7	SW
UKOD [Maddern et al.17] (2012)	> 100	UK
LISA [Davis and Goadrich06] (2012)	49	US
GTSRB [Houben et al.13] (2013)	43	DE
RTSD [Shakhuro and Konushin16] (2013)	156	RU
BTS KULD [Timofte et al.14] (2014)	> 100	BE
rMASTIF [Šegvic et al.10] (2015)	31	HR
TT 100K [Zhu et al.16b] (2016)	> 45	CN
DITS (2016)	58	IT
MTSD [Ahmed Madani16] (2016)	66	MY
EMTSD (2016)	66	MY
Mapillary [Ertler et al.20](2018)	1500	Global
CCTSDB [Zhang et al.17](2017)	48	CN
CURE-TSR [Temel et al.17](2018)	14	BE
TSRD	58	CN
VDB	35	SP
DFG[Tabernik and Skočaj19](2019)	200	SI

ISO 3166-1 códigos alpha-2 [wikipediab] se usan como códigos de países. Global se usa cuando el origen es de más de un país.

En este análisis, seleccionamos seis *datasets* distintos de señales de tráfico públicos, D = { GTSRB, DITS, rMASTIF, BTSC, TSRD, VDB }, disponibles en internet (Table 4.3) y que pertenecían además a diferentes países.

Sin embargo, sólo se ha utilizado un subconjunto de clases de los diferentes *datasets* (SCS, véase la figura 4.10). El objetivo de esta selección fue ejecutar los tests con mayor eficiencia basándonos en la asunción de que las clases del SCS eran suficientes para extrapolar los resultados al uso de un mayor número de clases. Las clases comunes con el SCS para cada *dataset* público están descritas en detalle en la figura 4.15 y se puede

Figura 4.14: Imagenes de los diferentes *datasets*.

comprobar la heterogeneidad de las imágenes en la figura 4.14), habiendo sido seleccionadas imágenes de buena calidad visual dentro de cada dataset para la gráfica.

Una vez elegidos los *datasets*, el siguiente paso fue el entrenamiento. En esta fase, se generó un modelo M_i (usando la arquitectura AlexNet) para cada subconjunto de entrenamiento de cada dominio D_i^{TR}. Sin embargo, para otras experimentaciones esta etapa de entrenamiento podría ser innecesaria ya que el modelo podría venir entrenado y aún así ser usado en la comparación.

Se realizó la inferencia con cada modelo entrenado para las señales proporcionadas por los *datasets* seleccionados, tanto para la parte de entrenamiento como de test, obteniendo así una matríz de resultados como la presentada en la figura 4.16. Aquí nos encontramos uno de los principales problemas que podían enturbiar el análisis: no todos los datasest compartían el mismo tipo de señales. Por ello las comparaciones habrían de realizarse sobre la intersección entre los datasets que participaban en la comparación. Además como redujimos el número de clases a un subconjunto representativo, finalmente estas clases obtenidas habíamos de intersectarlas a su vez con el SCS. Así nos asegurábamos de que los *datasets* de testeo usados para cada modelo estaban formados por las muestras que pertenecían las clases dentro de la intersección entre las clases de la parte de training de ese *dataset*, las clases del *dataset* a testear y el SCS.

De esa tabla, extraíamos los ratios de éxito (Positivos Verdaderos, TP) y de fracaso (Falsos Positivos, FP) para cada modelo contra todos los *datasets* considerados. Estos

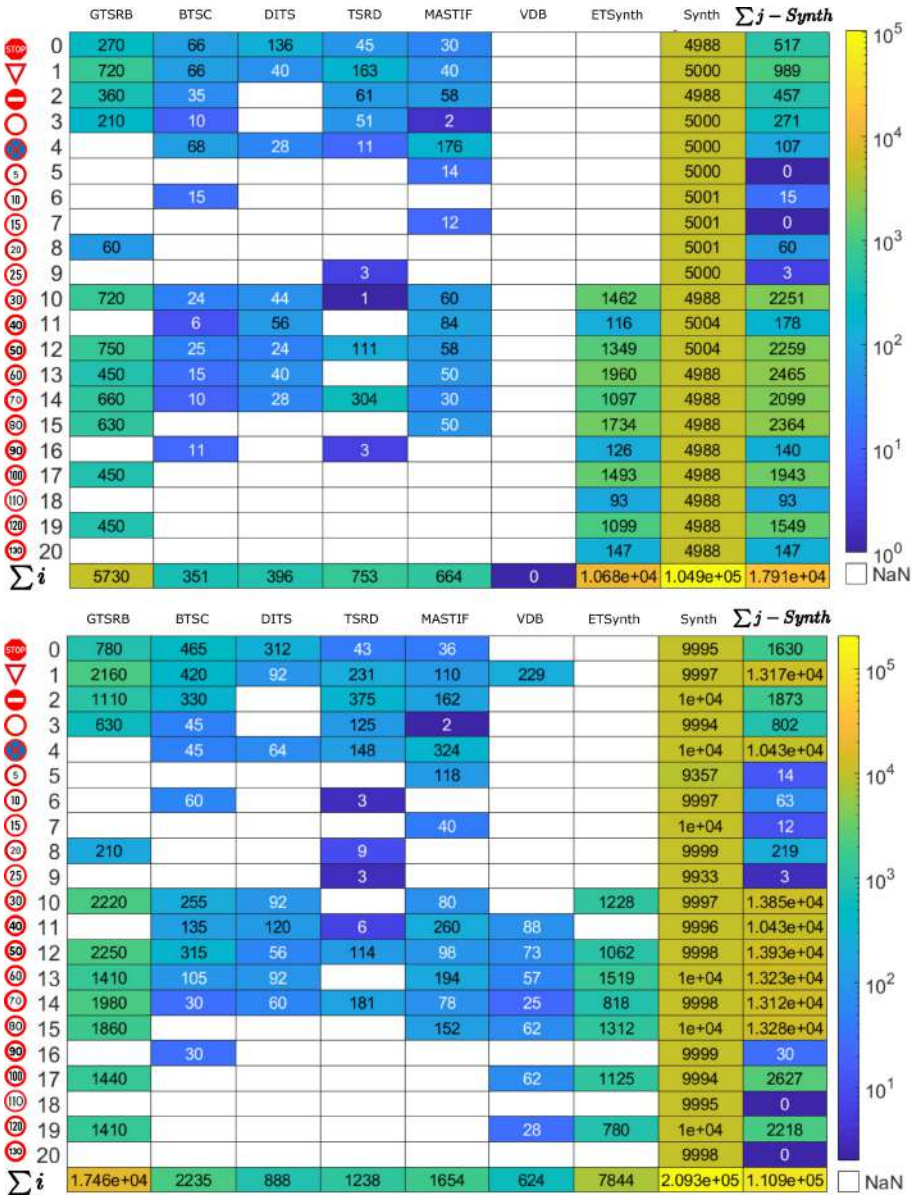


Figura 4.15: Número de imágenes por *dataset*, únicamente para clases en el SCS: (top) muestras de test, (bottom) muestras de training.

GTSRB	GTSRB			DITS			MASTIF			BTSC			TSRD			Synth			ETS2Synth			All Datasets (SUM)				
	TP	FP		TP	FP		TP	FP		TP	FP		TP	FP		TP	FP		TP	FP		TP	FP	global	geometric	aritmetic
0	269	1	0,996	61	5	0,924	133	3	0,978	45	0	1,000	30	0	1,000	2806	2192	0,561	0	0	NaN	3314	2201	0,601	0,872	0,910
1	718	2	0,997	65	1	0,985	40	0	1,000	163	0	1,000	40	0	1,000	3576	1424	0,715	0	0	NaN	4562	1427	0,762	0,932	0,950
2	345	15	0,958	13	22	0,371	0	0	NaN	61	0	1,000	44	14	0,759	3159	1839	0,632	0	0	NaN	3578	1876	0,656	0,689	0,744
3	209	1	0,995	10	0	1,000	0	0	NaN	50	1	0,980	2	0	1,000	3911	1089	0,782	0	0	NaN	4180	1091	0,793	0,935	0,952
4	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	0,000	0,000	0,000
5	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	0,000	0,000	0,000
6	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	0,000	0,000	0,000
7	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	0,000	0,000	0,000
8	60	0	1,000	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	1182	3819	0,236	0	0	NaN	1242	3819	0,245	0,486	0,618
9	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	0,000	0,000	0,000
10	719	1	0,999	14	10	0,583	31	13	0,705	1	0	1,000	52	8	0,867	2423	2575	0,485	1018	444	0,696	4206	3043	0,580	0,719	0,762
11	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	0,000	0,000	0,000
12	746	4	0,995	21	4	0,840	22	2	0,917	110	1	0,991	58	0	1,000	2620	2384	0,524	893	456	0,662	4412	2851	0,607	0,800	0,847
13	428	22	0,951	9	6	0,600	1	39	0,025	0	0	NaN	50	0	1,000	2618	2380	0,524	1711	249	0,873	4767	2696	0,639	0,366	0,662
14	647	13	0,980	10	0	1,000	25	3	0,893	302	2	0,993	26	4	0,867	1722	3276	0,345	998	99	0,910	3704	3393	0,522	0,805	0,855
15	589	41	0,935	0	0	NaN	0	0	NaN	0	0	NaN	46	4	0,920	3497	1501	0,700	1623	111	0,936	5709	1653	0,775	0,849	0,873
16	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	0,000	0,000	0,000
17	443	7	0,984	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	1640	3358	0,328	418	1075	0,280	2501	4440	0,360	0,449	0,531
18	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	0,000	0,000	0,000
19	438	12	0,973	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	1607	3391	0,322	803	296	0,731	2848	3699	0,435	0,612	0,675
20	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	NaN	0	0	0,000	0,000	0,000
arithmetic 21	5611	119	0,979	203	48	0,809	252	60	0,808	732	4	0,995	348	30	0,921	30761	29228	0,513	7464	2730	0,732	45023	32189	0,583	0,788	0,822

(TEST Datasets)

Figura 4.16: Resultados obtenidos para el subconjunto de test de cada *dataset* para el modelo aprendido con GTSRB.

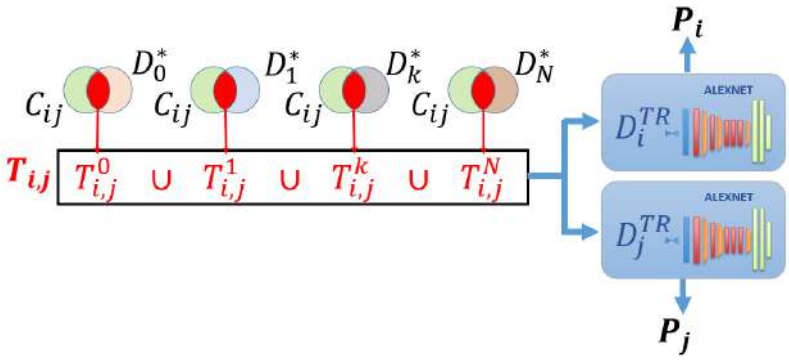
falsos positivos se podrían considerar falsos negativos atendiendo a la clase que se trata, aunque tratándose de un asunto meramente interpretativo mantuvimos el apelativo de falsos positivos a lo largo del análisis.

Después de reunir los resultados, se generaban estadísticas diferentes para proporcionar interpretaciones adicionales: una matriz de confusión de modelo contra modelo (G) y el comportamiento de cada modelo en relación a los tipos de señal, esto es, las matrices de señales contra modelos (S). Con estas estadísticas, podíamos esbozar conclusiones objetivas y justas relacionadas con el rendimiento del clasificador bajo análisis.

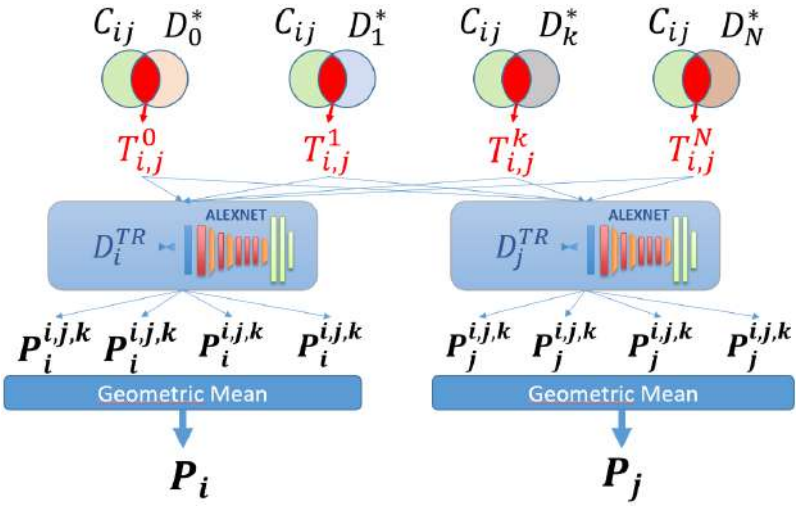
Para comparar resultados entre los modelos M_i y M_j y para generar G, era necesario el uso de una métrica. Los modelos eran comparados en pares usando el valor de precisión obtenido (véase la figura 4.17). Las clases que participaban en la comparación de dos modelos M_i y M_j se establecen como $\{C_{i,j}\} = \{C_i\} \cap \{C_j\} \cap \{SCS\}$, donde $\{C_i\}$ y $\{C_j\}$ son los conjuntos de clases usados para entrenar el modelo M_i y M_j respectivamente.

Esta métrica evitaba la influencia en los resultados de las clases no comunes a ambos *datasets*, lo que podía distorsionar la comparación. Por lo tanto, los modelos se testeaban contra el mismo conjunto de test $T_{i,j}$ buscando imparcialidad.

El *dataset* que era usado para comparar el modelo i y el modelo j, $T_{i,j}$, se obtenía de la unión de cada uno de los $T_{i,j}^k$ cara cada uno de los k *datasets* en el análisis.



(a) Usando la unión de las intersecciones de los *datasets*



(b) Usando la intersección de los *datasets* de manera disjunta

Figura 4.17: Cálculo de los valores de precisión.

$$T_{i,j}^{k} = \{C_{i,j}\} \cap \{D_{k}^{*}\} \tag{4.3}$$

donde $\{D_k^*\}$ es el conjunto de muestras de entrenamiento del *dataset* $\{D_k^{TR}\}$ o el conjunto de muestras de test del *dataset* $\{D_k^{TE}\}$, dependiendo del tipo de muestra que se va a analizar, test o train.

$$T_{i,j} = \bigcup_{k=0}^{N} \{T_{i,j}^{k}\} \tag{4.4}$$

Aquí, N es el número de *datasets* usados en el análisis. Una vez que el *dataset* de test se definía y para calcular cada valor de la matriz de confusión $G_{i,j}$, era necesario calcular cada una de las precisiones P_i y P_j.

Estos valores se obtenían testeando $T_{i,j}$ contra M_i and M_j.

$$\begin{aligned} f : \{T, M\} &\rightarrow TP, FP \\ (TP_p, FP_p) = f(T_{i,j}, M_p) \quad &\setminus \quad \forall p \in \{i, j\} \end{aligned} \tag{4.5}$$

Donde f representa la ïnferencez (TP_p, FN_p) son los verdadores positivos, falsos negativos del clasificador, respectivamente; $p \in \{i, j\}$. Con estos valores, se calcula la precisión P, donde $P_p = \frac{TP_p}{TP_p + FN_p}$.

En algunos casos, los modelos se entrenaban con *datasets* muy favorables que ofrecían poca complejidad. Esto favorecía a estos modelos durante el test. Para evitar esta arbritariedad, ambos modelos se enfrentan a las mismas señales de modo que los resultados se podían comparar de manera objetiva. Sin embargo, calcular las precisiones de este modo puede llevar a interpretaciones incorrectas. En este caso que nos ocupa, y como puede verse en la figura 4.15, la cardinalidad para la misma clase en algunos de los *dataset* es mucho mayor que en los demás; por lo tanto, unir todas las clases comunes en un *dataset* mayor y único producirá un sesgo favorable en los resultados hacia aquellos *datasets* en perjuicio de los otros con cardinalidad más pequeña.

Para solucionar este problema, realizamos una aproximación diferente para calcular la precisión. Primero, cada *dataset* era considerado de manera independiente y testeado con ambos modelos i y j como sigue:

$$\{(TP_p^{i,j,k}, FP_p^{i,j,k}) = f(T_{i,j}^k, M_p)\} \tag{4.6}$$

donde N es el número de *datasets*, $k \in \{0..N\}$ y $p \in \{i, j\}$, son los dos modelos bajo comparación. Por tanto, se calculaba un conjunto de tuplas (TP, FP), uno por cada modelo en la comparación.

$$P_p^{i,j,k} = \frac{TP_p^{i,j,k}}{TP_p^{i,j,k} + FP_p^{i,j,k}} \tag{4.7}$$

$$P_p = \sqrt[n]{\prod_{k=0}^{n} P_p^{i,j,k}} = \sqrt[n]{\prod_{k=0}^{n} \frac{TP_p^{i,j,k}}{TP_p^{i,j,k} + FP_p^{i,j,k}}} \tag{4.8}$$

con esta aproximación se calculaban un conjunto de N valores de precisión ($Pr_i^{i,j,k}$ y $Pr_j^{i,j,k}$) para M_i y M_j. La precisión real para el modelo i y el j (Pr_i and Pr_j) se obtenía como una media geométrica de esos N valores.

En ambos casos, los *datasets* usados para comparar ambos modelos eran los mismos; por lo tanto, eran totalmente comparables.

Cada celda de la matriz modelo vs modelo $G_{i,j}$ comparaba dos modelos que fueron entrenados con un *dataset* de entrenamiento diferente. El valor que aparece en la celda era la precisión del modelo correspondiente a esa fila si ese valor de precisión resultaba mayor que su complementario, o la diferencia (en negativo) entre los valores de precisión de ambos modelos si el valor era menor, como se explica en Eq. (4.9).

$$G_{i,j} = \begin{cases} P_i & \text{if } P_i >= P_j \\ P_i - P_j & \text{if } P_i < P_j \end{cases} \tag{4.9}$$

Se ilustran otras estadísticas en este estudio en Figs. 4.20 y 4.21. Por un lado, en la figura 4.20, se testeó M_i contra la parte de test del *dataset* para ese dominio D_i^{TE}, y por otro lado, en la figura 4.21, el *dataset* de test utilizado era $T_{i,j}$ que se genera por la unión de todas las clases comunes entre el SCS y las muestras que pertenecían a la parte de test de cada uno de los *dataset* públicos utilizados. La figura 4.21 relaciona los tipos de las señales de tráfico con los modelos de una manera independiente. Muestra los resultados obtenidos por cada modelo cuando se testeó con cada clase de señal de tráfico, haciendo posible analizar el impacto de dicha clase en el valor de rendimiento

Algorithm 7: Data Generation algorithm

$T_{i,j}^{k,*}$ =Get From D_k^* all samples which belong to common classes between D_i^*, D_j^* and SCS. $* \in \{TEST, TRAIN\}$;

GenerateDataTables (D^{TR}, D^{TE})

input : N training and test public *datasets*.

output : TP and FP rates traffic sign-wise tables for n models trained with each of D^{TR} and tested with D^{TR} and D^{TE}

foreach $M_i \in M$ **do**

  foreach $D_j \in D$ **do**

    foreach $* \in \{TR, TE\}$ **do**

      $DataTable[i][j]^* = Test(M_i, T_{i,j}^{j,*})$;

ComputePrecision (i, j)

input : first model index, second model index

output : Precision values for P_i and P_j

foreach $D_k \in D$ **do**

  $TP_i^{i,j,k}, FP_i^{i,j,k}$=Test($M_i$, $T_{i,j}^{k,*}$);

  $TP_j^{i,j,k}, FP_j^{i,j,k}$=Test($M_j$, $T_{i,j}^{k,*}$);

  $\boldsymbol{P_i}^{i,j,k} = TP_i^{i,j,k} / (TP_i^{i,j,k} + FP_i^{i,j,k})$;

  $\boldsymbol{P_j}^{i,j,k} = TP_j^{i,j,k} / (TP_j^{i,j,k} + FP_j^{i,j,k})$;

P_i, P_j=GeometricMean($\boldsymbol{P_i}, \boldsymbol{P_j}$);

GenerateConfusionMatrix

input : Data tables

output : Model vs Model matrix (G)

foreach $M_i \in M$ **do**

  foreach $M_j \in M$ **do**

    $P = ComputePrecisions(i, j)$;

    if $P_i >= P_j$ **then**

      $G_{i,j} = P_i$;

    else

      $G_{i,j} = P_i - P_j$;

del modelo. Se pueden extraer algunas conclusiones interesantes de la segunda tabla, como se discutirá más adelante. Además estos resultados pueden guiar el proceso de mejora del clasificador, señalando la clase donde el clasificador podría no generalizar de manera efectiva.

4.7 Resultados

Estos fueron los resultados obtenidos de los test realizados: tablas de precisión, matriz de confusión modelo vs modelo, tabla señal vs modelo.

Interpretar los datos es difícil cuando se usan *datasets* heterogéneos. Primero, como se aprecia en la figura 4.15, hay varias diferencias entre estos *datasets*, no sólo en los tipos de señales que presentan y en las diferentes representaciones dentro de la misma señal, sino también en el número de señales por clase, volviendo más compleja la tarea de análisis de los resultados. Consideramos no obstante que, para posibilitar que los analistas determinen cuál es el mejor modelo, es mejor el uso *datasets* de testeo heterogéneos como tenemos en este caso y por ello proponemos este método para guiar al analista hacia la consecución del mejor modelo para su caso concreto.

También debería ser advertido que algunos *dataset* están sesgados a ciertas señales; algunos están más balanceados y otros presentan muestras insuficientes. El *dataset* Synth es el conjunto de muestras de señales de tráfico más completo y balanceado debido a su inherente simplicidad y proceso de formación. El primer paso en este análisis era entrenar el modelo AlexNet con las muestras de entrenamiento de cada uno de los *datasets*. Era importante mantener todos los parámetros de la red para todos los clasificadores para asegurar la misma configuración para todos. Subsecuentemente, cada modelo fue testeado contra todos los *datasets* (con ambas partes, la de test y la de entrenamiento). Los resultados se muestran en [Cortés19] tanto para las partes de test como para las partes de entrenamiento. En estas tablas de resultados se calculan los positivos verdaderos, los falsos negativos y la precisión para cada tipo de señal de tráfico en cada *dataset*. Esto nos ha posibilitado posteriormente unir los datos de la manera más apropiada para el análisis. Cada celda representa el comportamiento del proceso de clasificación para cada tipo de señal de tráfico en un dominio. Los valores marginales también se calculan ya que proporcionan buena información sobre el rendimiento del modelo con un *dataset* particular (columnas) o la complejidad

asociada a cada tipo de señal, es decir, como de difícil es clasificar ese tipo de señal (filas).

Algunos de los valores introducen sesgos en la precisión para algunas de las clases debido a la cardinalidad de las señales. Así, el número de señales del *dataset* Synth, siendo tan grande, hace que el resto de los resultados sean irrelevantes, y la precisión media del GTSRB se vuelve 0.583 %. Por lo tanto, se decidió calcular el rendimiento del clasificador a partir de los porcentajes de precisión de la media geométrica. Esto produce un resultado más balanceado porque todos los *datasets* se consideran al mismo nivel y el *dataset* con más muestras no se beneficia.

Una vez está completa la tabla de resultados para cada modelo, se genera la matriz de confusión entre los diferentes modelos para identificar el modelo que mejor generaliza. Para esto, se comparan los modelos en pares con el mismo conjunto de test. Este conjunto de test se elabora como ya se explicó anteriormente en la sección metodología.

4.7.1 **Resultado de evaluación entre modelos**

Podemos ver en los resultados (véase la figura 4.18) que los mejores modelos son, en orden, el modelo entrenado con el *dataset* Synth , ETS2Synth y GTSRB. Esto confirma nuestra hipótesis inicial de que los datos sintéticos pueden ser usados para entrenar modelos que pueden aplicarse a datos del mundo real. También confirma que incluso aunque el modelo GTSRB tiene tanto en la parte de test como en la parte de entrenamiento un mejor rendimiento contra su propio modelo (véase la figura 4.20) que los modelos Synth y ETS2Synth, su capacidad de generalización es peor como se muestra en la figura 4.18).

El peor comportamiento del *dataset* Synth ocurre cuando se compara al *dataset* ETS2Synth. Sin embargo, el *dataset* Synth produce mejores resultados que el *dataset* ETS2Synth y lo hace mejor comparado a los otros modelos.

Se ha realizado un experimento adicional para incorporar el poder de generalización del *dataset* Synth en un dominio específico (véase la figura 4.19). Se han comparado diferentes combinaciones con el objeto de mejorar los modelos. Este experimento se realizó unicamente con los *dataset* GTSRB y Synth. Para entrenar estos nuevos modelos, los *datasets* fueron combinados, y se han refinado de modelos

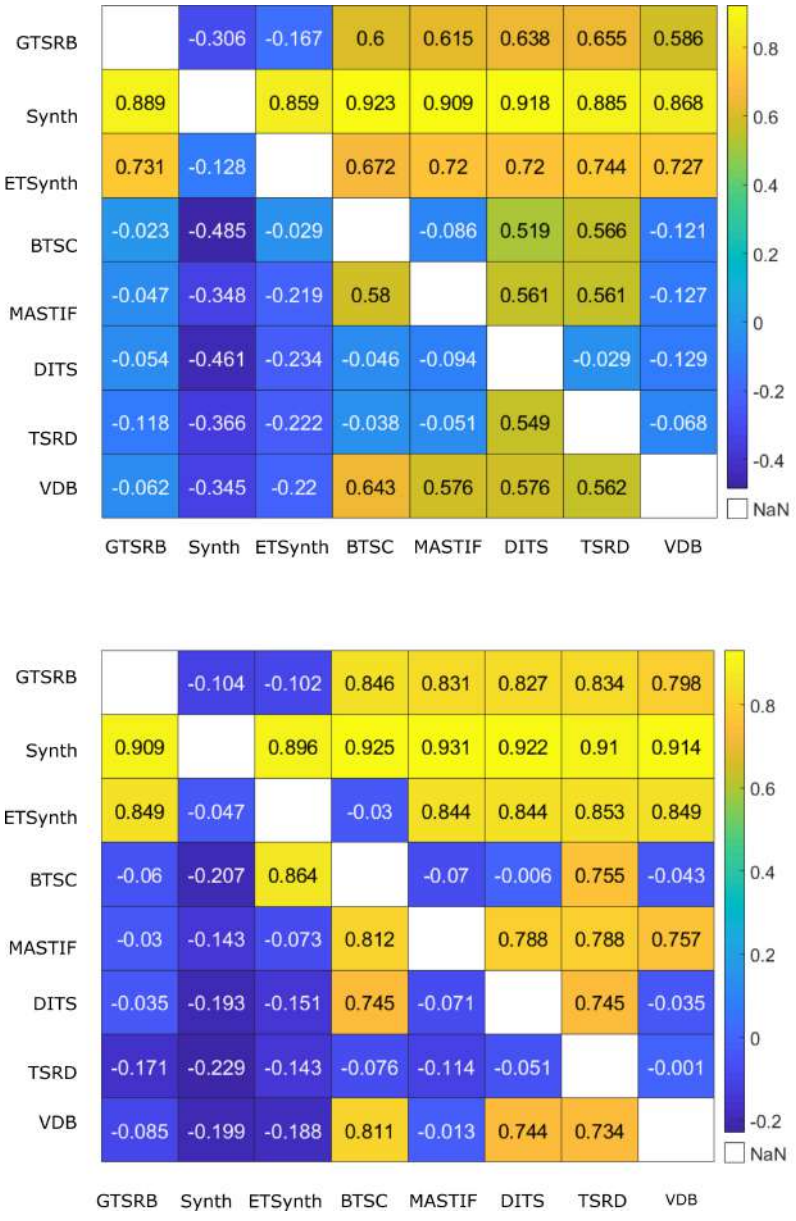


Figura 4.18: Comparación entre pares de clasificadores usando las muestras de la parte de test de los *datasets*. Un valor positivo indica que el modelo de la fila devuelve un mejor resultado que el modelo de la columna.

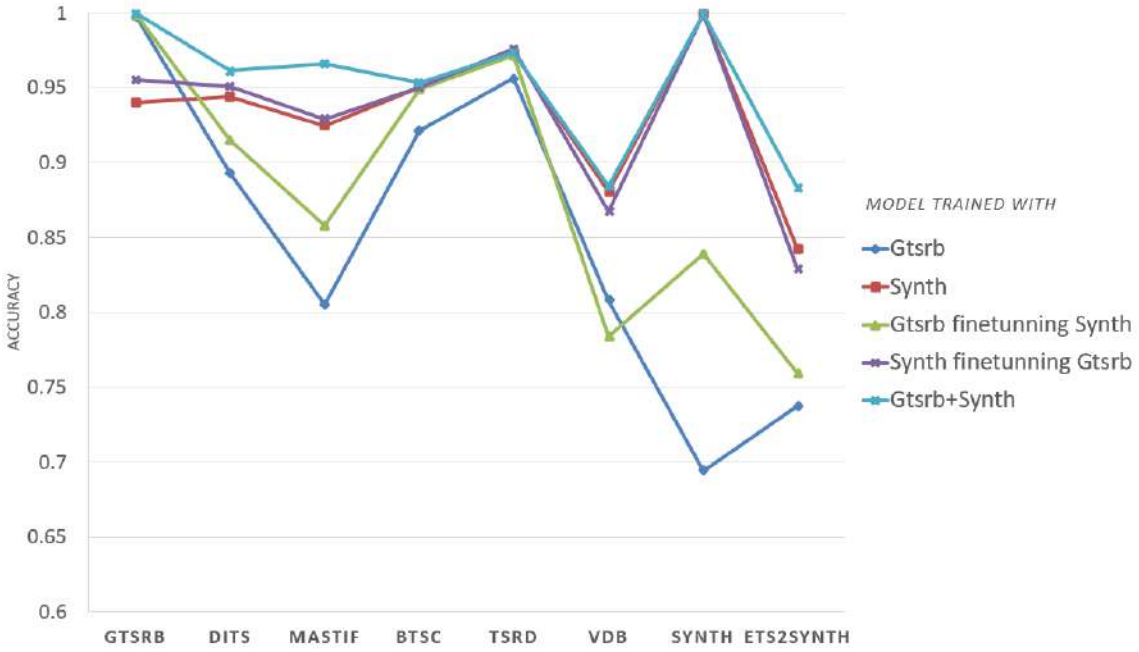


Figura 4.19: Comparación entre modelos, evaluado a través de los dominios.

aprendidos previamente e incluso se ha entrenado uniendo varios *datasets*. Todos estos nuevos modelos fueron testeado contra todos los *datasets* obteniendo resultados a través de los diferentes dominios que se muestran en la figura 4.19.

Finalmente, para testear el comportamiento de modelo entrenados con el *dataset* Synth contra un *dataset* de imagen real extremadamente desafiante, realizamos un último experimento para evaluar el *dataset* CureTSD (la parte real), cuya reputación lo posiciona como uno de los más difíciles *datasets* de señales de tráfico disponibles. Esta evaluación se realizó con los modelos de clasificación entrenados con GTSRB, Synth y ETS2Synth. Los resultados muestran que incluso con imagenes muy complicadas, los clasificadores de imagenes basados únicamente en imagenes sintéticas pueden competir contra modelos reales con buenos resultados. En este caso, cuando se compara GTSRB contra el *dataset* Synth, observamos que los resultados incrementalmente llegan a ser muy similares en ambos con los *datasets* de test (0.51 para GTSRB, 0.57 para Synth, y 0.76 para ETS2Synth) y con los *datasets* de train (0.65 para GTSRB , 0.63 para Synth, y 0.55 para ETS2Synth). Es necesario considerar que las imágenes de Cure-TSD se adquirieron en Bélgica y que este dominio es muy parecido al

dominio del GTSRB; esto podría ser beneficioso para modelos entrenados con el *dataset* GTSRB y aún así, el modelo entrenado con Synth proporciona buenos resultados. Cabría mencionar los bajos valores de precisión que ofrecen en general los clasificadores evaluados contra el *dataset* CureTSD y esto es debido a la presencia de imágenes distorsionadas y de baja calidad en el *dataset*.

4.7.2 Resultados de evaluación por señal de tráfico

Figura 4.20: Resultados obtenidos de evaluar los modelos contra la parte de test del *dataset* en el mismo dominio. La primera columna la evaluación del modelo entrenado con la parte de entrenamiento del *dataset* GTSRB utilizando las muestras de evaluación del propio *dataset* GTSRB. Se aplica un *heatmap* por columna para visualizar las variaciones inter-clase.

Otro análisis interesante se realizó considerando la complejidad de clasificación para cada tipo de señal de tráfico. No todas las señales son igualmente difíciles de clasificar y esto depende no sólo del tipo de señal sino también del *dataset* de entrenamiento utilizado. Esto puede verse en la figura 4.21, donde se presentan los resultados relacionados con las diferentes señales para cada *dataset*.

Debe considerarse el número de imágenes para cada señal de tráfico dentro de cada *dataset* porque puede introducir sesgos en los resultados. Se puede ver en la figura 4.20 que los datos sintéticos no proporcionan la mejor precisión cuando se testean contra la

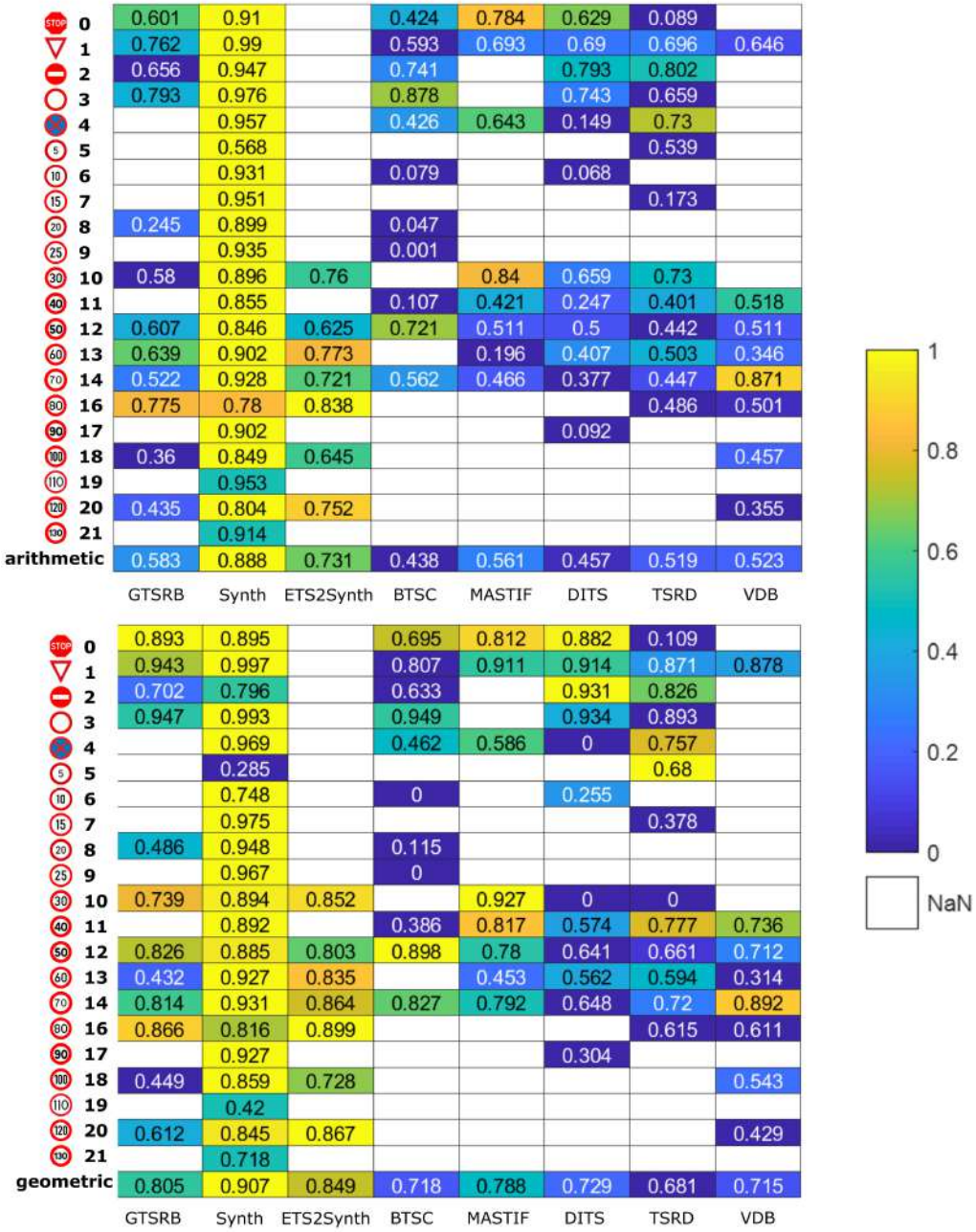


Figura 4.21: Resultados obtenidos de evaluar los modelos contra el *dataset* de testeo común, generado de la intersección de las clases comunes entre todos los *datasets* y el SCS. El heatmap está aplicado por filas. La tabla superior se calculó usando la media aritmética y la inferior usando la media geométrica.

parte de test de su *dataset*. GTSRB, por ejemplo, obtiene un 0.98 de precisión atendiendo a la media geométrica mientras que el modelo entrenado con sintéticos obtiene un 0.88. Así, aunque el resultado contra el *dataset* de test del dominio es bueno, no significa que la capacidad de generalización del modelo sea mejor que otro que obtuvo peor resultado.

También es reseñable que no hay una clase que destaque por su dificultad a nivel global para todos los clasificadores; es más, cada modelo presenta su propio grupo de señales de compleja clasificación. Por ejemplo, el GTSRB presenta dificultades cuando clasifica señales de limitación de velocidad a 100, mientras que Synth encuentra más problemáticas las señales de limitación de velocidad de 5 o las señales de no pasar. En la etapa de generación del *dataset* sintético, estas clases pueden guiarnos hacia modificaciones en los procesos aplicados que mejorarán el *dataset* usado para entrenar el modelo de clasificación.

Típicamente, un clasificador se evalúa contra las muestras de test correspondientes proporcionadas por el *dataset* del dominio. Los resultados pueden estar sesgados porque los modelos entrenados aprenden las peculiaridades del dominio, tales como las transformaciones de perspectiva, o las distorsiones de las imágenes producidas por el tipo de cámara usada. Consecuentemente, usualmente el rendimiento suele ser muy bueno cuando se evalúan contra su propio *dataset* de test.

4.7.3 Conclusiones

En este estudio se discutió como un modelo entrenado con un *dataset* sintético puede tener un rendimiento similar a un modelo entrenado con datos reales si se aplican los procesos de generación de imagen adecuados a las imágenes sintéticas para prevenir sesgos contra peculiaridades de los *dataset* reales.

Ambas hipótesis fueron validadas en nuestro análisis. Por un lado, ambos modelos entrenados con sintéticas (Synth y ETS2Synth) funcionan mejor o con un rendimiento similar (alcanzando valores de precisión de aproximadamente 0.9) a la hora de generalizar que los modelos entrenados usando *datasets* reales como se ve en la figura 4.18.

Por lo tanto, si el dominio objetivo es general, la mejor opción es crear un modelo entrenado con *dataset* sintéticos, ya que tiene un mejor comportamiento comparado

con otros modelos (véase la figura 4.18). Sin embargo, si el dominio objetivo está muy enfocado, una buena alternativa podría ser usar un modelo entrenado con imágenes de ese dominio, mientras que una mejor alternativa (véase la figura 4.19) sería mezclar imágenes reales con sintéticas para entrenar el modelo. Esta es la mejor opción para mantener una buena precisión frente a dominios no vistos todavía.

Por otro lado, el proceso de entrenamiento puede ser guiado considerando el análisis de señal vs modelo. Usamos esta aproximación y los resultados mejoraron en cada iteración. Los modelos basados en los *datasets* Synth y ETS2Synth tienen un mejor comportamiento a la hora de generalizar. Aquí, podemos observar el sesgo producido por entrenar con un único *dataset* ya que el modelo basado en sintéticas produce peores resultados que los modelos a partir del GTSRB o DITS en sus propios *datasets* de test (véase la figura 4.20). Esto obedece a que la variabilidad en el *dataset* sintético es muy alta; así, los conjuntos de test y entrenamiento difieren significativamente, comparados a al situación de los *dataset* reales. El modo de proceder es continuar detectando las señales más complicadas para los modelos Synth o ETS2Synth y generar nuevas muestra mediante una elección empírica de los procesos de aumentación que se aplicarán a las muestras canónicas. Para iterar, el entrenamiento empieza desde el último modelo aprendido, como un proceso de *transfer learning* entre iteraciones del modelo, o congelando zonas del modelo que sean lo suficientemente buenas para forzar al modelo a mejorar la clasificación de las señales.

Los modelos basados en sintéticas pueden alcanzar resultados con buena precisión cuando se aplican a datos reales. Además, el proceso para obtener estos datos es significativamente menos costoso que el de obtener imágenes reales. Adicionalmente, las especificidades o peculiaridades no relevantes de los datos reales típicamente disminuyen las capacidades de generalización de los modelos.

Es también muy interesante apuntar la posibilidad de aplicar este tipo de análisis al campo de la detección, generando imágenes que usen muestras e imágenes de fondo, uniéndolas de manera aleatoria usando las aumentaciones y procesos usados en este estudio.

Adicionalmente, para validar las conclusiones alcanzadas, extendimos el análisis evaluando los modelos contra las partes de entrenamiento de los *datasets* de los diferentes dominios, ya que son significativamente mayores que las partes de test. Los resultados fueron similares [Cortés19].

Lo que sabemos es una gota de agua; lo que ignoramos es el océano

Isaac Newton

CAPÍTULO

5

Conclusiones y trabajo futuro

En esta disertación se ha analizado la evolución de los enfoques en el ámbito de la visión artificial y el *machine learning* que han sufrido las soluciones aplicadas a las ITS, particularizándolo en dos casos concretos asociados a proyectos de cierta envergadura y relevancia tanto a nivel nacional como internacional. A nivel más específico se ha desarrollado un modelo de sustracción de fondo multipista dentro del contexto de un proyecto a nivel nacional y en el marco de una solución de conteo y clasificación de vehículos para sistemas de peaje free-flow. Por otro lado, se ha definido, estructurado y desarrollado una solución de reconocimiento de señales de tráfico englobada en un proyecto de proyección europea. Para este último además se realizó un estudio pormenorizado de los efectos a nivel de precisión de la generación de los diferentes modelos a partir de imágenes sintéticas.

Todo esto dentro del mundo del transporte inteligente y aumentando un poco la perspectiva, en el ámbito de las ayudas avanzadas a la gestión de infraestructura y de las ayudas avanzadas a la conducción.

5.1 Conclusiones

Teniendo en cuenta el intervalo temporal en los trabajos descritos en esta disertación se percibe claramente el cambio de paradigma predominante a nivel

científico en los desarrollos científicos académicos. A nivel general, se percibe la clara tendencia de que en los tiempos venideros continuará esta proliferación de sistemas basados en *deep learning*, por los resultados tan interesantes que pueden llegar a ofrecer. De hecho será fascinante asomarse, participar y sorprendernos del mundo al que nos lleva esta tecnología que impregnará cada uno de los ámbitos y recovecos de nuestro día a día. Sin embargo, uno de los principales inconvenientes que habrá que sortear es, precisamente, la necesidad abrumadora de datos necesarios para hacer que estos modelos aprendan. Esto o bien pasará por hacer evolucionar a las redes de manera que dependan de menos datos de entrada (como el caso del *one-shot learning*) o bien por un trabajo combinado entre los gráficos por computador y la visión artificial, generando renderizados y aumentaciones de gran realismo y variabilidad, de la que surgirán ingentes *datasets* que podrán nutrir las necesidades de aprendizaje de estas redes.

A modo de conclusiones particulares, para el primer caso aplicativo, se partía del objetivo de desarrollar un sistema de sustracción de fondo que nos permitiera lidiar con los cambios lumínicos que pudiera sufrir la escena así como cambios en la meteorología y con las sombras proyectadas por vehículos sin la resolución de estos puntos fuese origen de nuevos problemas. Finalmente se llegó al desarrollo de un sistema que conseguía lidiar con la mayor parte de las dificultades aquí expresadas sin repercutir negativamente en otras fuentes de problemas. Concluimos en aquel entonces que añadir a los modelos de sustracción de fondo información adicional con la que manejar los problemas propios de esta técnica favorecía los resultados al poder discriminar entre las zonas de sombra y las zonas de los vehículos que la proyectaban, tanto para la generación de la región como para la fase de actualización del propio fondo.

En lo referente al segundo caso, los objetivos aquí fueron diversos y dieron lugar al profundo análisis que aquí se ha detallado sobre el alcance de la aplicación de datos sintéticos en la resolución de problemáticas con imágenes reales.

La principal conclusión alcanzada es que los datos generados sintéticamente pueden dar buenos resultados en dominios particulares y presentan una capacidad de generalización superior que los modelos entrenados en dichos dominios. También se propuso una metodología de comparación para diferentes modelos de clasificación a partir de *datasets* heterogéneos en categorías y en cardinalidad y que a día de hoy son la

gran mayoría en casi todos los ámbitos. Consideramos además que esto proporciona una medida más imparcial y por ende, más justa, entre modelos entrenados con diferentes *datasets*. Otra conclusión extraída del estudio es que al entrenar con un dominio concreto aún particionando el *dataset* en dos mitades, los modelos entrenados en un dominio tienden a modelar las especificidades de dicho dominio dando mejores resultados de precisión contra sus propios tests. También concluimos que es factible mediante la simple observación de los falsos negativos realizar un proceso de guiado en la incorporación de nuevos procesos que generen nuevas imágenes que añadir al entrenamiento.

En lo referente a la ejecución en tiempo real, es curioso advertir que es el hardware el que nos lleva unos cuantos pasos de ventaja y aunque en ocasiones, técnicas de optimización computacionales pueden proporcionar mejores tiempos de ejecución, la evolución del hardware acaba proporcionándole esta posibilidad a multitud de algoritmos que en otro momento se aplicaban off-line precisamente por su coste computacional.

5.2 Trabajo Futuro

Sustracción de fondo Como proyección a futuro una línea prometedora sería trasladar este sistema al ámbito del DL y analizar los resultados de ambas aproximaciones al problema. Otra opción pasaría por divergir de esta aproximación y aplicar técnicas de segmentación de imagen o de reconstrucción tridimensional. A día de hoy se han dado pasos muy significativos en el mundo de la reconstrucción tridimensional y en el de la segmentación semántica de imágenes utilizando técnicas de DL. Una línea que se abre es la aplicación de estas técnicas para segmentar las imágenes atendiendo a zonas en sombra y zonas de elementos de interés. Esto podría llegar a sustituir al modelo de fondo por un nuevo modelo extremo a extremo que etiquetase cada píxel de la imagen como perteneciente a una de las diferentes categorías establecidas, eliminando las sombras y las aberraciones lumínicas que se pudieran producir en el escenario. O bien, inferir la reconstrucción tridimensional de la escena utilizando esta información para eliminar zonas de sombra y obtener el volumen de los vehículos para su posterior clasificación y segmentación, combinado o no con un modelo de sustracción de fondo.

Detección de objetos A la hora de entrenar los modelos de detección aquí utilizados, una línea de investigación prometedora que se abre, es la aplicación de esta metodología de generación e incorporación de imágenes sintéticas al conjunto de datos utilizados para entrenar. Esto reduciría los tiempos de la anotación de imágenes para detección y permitiría incrementar la variabilidad de casos presentes en los *datasets* generados. Se podían estudiar políticas de *transfer learning* para transferir el conocimiento adquirido en el aprendizaje de los modelos de clasificación a los modelos de detección y tras evaluar los resultados, pasar a prescindir del clasificador si procediere. Se podrían resumir posibles futuras líneas de investigación en:

- Entrenar el detector con imágenes sintéticas, adecuando el pipeline descrito en esta tesis y comprobando la diferencia de precisión al prescindir del clasificador.
- Extender la detección de objetos no solo a señales si no a la mayoría de elementos presentes en un canal de transporte (semáforos, conos, vallas, peatones, otros vehículos, marcas en el suelo, etc.), siguiendo la misma metodología.

Generación de imagen sintética Hay todavía un mundo que recorrer en la línea de la generación de imágenes cada vez más realistas, incorporando por ejemplo las salidas de redes generativas adversarias entrenadas previamente o mejorando y añadiendo nuevos procesos de aumentación o incluso, y acompañados por el mercado del videojuego, mejorando los renderizados de los motores de generación 3d hasta que prácticamente sean indistinguibles de la realidad. También sería interesante modificar estos procesos de generación de imagen para adecuarlos al entorno de la detección. Una actuación adicional como resultado de esta generación de datos sintéticos, es poner a disposición de la comunidad científica un *dataset* sintético de un número de señales de tráfico sensiblemente mayor del disponible actualmente que consiste únicamente en las clases asociadas al estudio y proveer de las herramientas adecuadas para reproducir dicho *dataset*, ampliándolo también a otros ámbitos de aplicación. A modo de resumen:

- Adecuar aumentación para poderse aplicar a la detección
- Analizar el uso de redes generativas adversarias para la generación de nuevas muestras a la base de datos.

- Modificar o añadir nuevos procesos de aumentación
- Automatizar el proceso de guiado de aplicación de procesos de aumentación.

Seguimiento Visual de objetos Al igual que ocurre con los modelos de fondo, sería interesante embarcarse en una línea de seguimiento visual de objetos trasladándola a una perspectiva más cercana al ámbito del DL.

Sistema de reconocimiento y localización de señales de tráfico Esta línea de trabajo implica la necesidad de establecer protocolos y métricas de evaluación de la calidad de un sistema global de reconocimiento y localización de señales de tráfico. En el mundo de los sistemas de transporte no es fácil realizar una evaluación holística de los diferentes sistemas. Teniendo en cuenta que se tratan de sistemas de aplicación directa al mundo real y con un efecto tan acusado en nuestra vida cotidiana el cambio a una sociedad futura donde diferentes grados de automatización estén presentes en los vehículos que atestan las carreteras, pasa por severos protocolos de calidad que los sistemas habrán de cumplir. A día de hoy no obstante no resulta sencillo realizar dichas evaluaciones, traduciéndose esto en la ausencia de estándares que seguir a la hora de evaluar los sistemas. En el caso que ocupa a esta tesis, la evaluación es un proceso abierto todavía sobre el sistema desarrollado y que dará lugar a nuevos resultados científicos en los años venideros. La evaluación holística del sistema implica también la generación de un *dataset* de evaluación más complejo que el utilizado para evaluar las diferentes partes por separado. Este *dataset* debería de estar compuesto de diferentes vídeos etiquetados, donde por cada señal además de su clase y su posición en la imagen habría que guardar su posición tridimensional en relación a la cámara o par estéreo instalado en el coche.

Elige un trabajo que te guste y no tendrás que trabajar ni un día de tu vida.

Confucio

APENDICE

A

Publicaciones relacionadas

Estas publicaciones son resultado directo de los trabajos aquí presentados:

A.1 Analysis of classifier training on synthetic data for cross-domain datasets

Título: Analysis of classifier training on synthetic data for cross-domain datasets
Autores: Andoni Cortés, Clemente Rodríguez, Gorka Vélez, Javier Barandiarán, and Marcos Nieto
Campo: Advanced driving assistance systems
Revista: IEEE Transactions on Intelligent Transportation Systems
tipo: Q1
Año: 2020
DOI: `http://dx.doi.org/10.1109/TITS.2020.3009186`
Relación:: Este artículo fue el resultado del análisis realizado sobre datos sintéticos del capítulo 4
Abstract: *A major challenges of deep learning (DL) is the necessity to collect huge amounts of training data. Often, the lack of a sufficiently large dataset discourages the use of DL in certain applications. Typically, acquiring the required amounts of data costs*

considerable time, material and effort. To mitigate this problem, the use of synthetic images combined with real data is a popular approach, widely adopted in the scientific community to effectively train various detectors. In this study, we examined the potential of synthetic data-based training in the field of intelligent transportation systems. Our focus is on camera-based traffic sign recognition applications for advanced driver assistance systems and autonomous driving. The proposed augmentation pipeline of synthetic datasets includes novel augmentation processes such as structured shadows and gaussian specular highlights. A well-known DL model was trained with different datasets to compare the performance of synthetic and real image-based trained models. Additionally, a new, detailed method to objectively compare these models is proposed.

A.2 **Adaptive Multi-Cue Background Subtraction for Robust Vehicle Counting and Classification**

Título: Adaptive Multi-Cue Background Subtraction for Robust Vehicle Counting and Classification

Autores: L. Unzueta, M. Nieto, A. Cortés, J. Barandiaran, O. Otaegui and P. Sánchez

Campo: ATM

Revista: IEEE Transactions on Intelligent Transportation Systems

Editor: IEEE Press

Año: 2012

Páginas: 527-540

Tipo: Q1

Relación: Este artículo fue el resultado del desarrollo del modelo de fondo multi-pista del capítulo 3

DOI: `http://dx.doi.org/10.1109/TITS.2011.2174358`

Abstract: *In this paper, we present a robust vision-based system for vehicle tracking and classification devised for traffic flow surveillance. The system performs in real time, achieving good results, even in challenging situations, such as with moving casted shadows on sunny days, headlight reflections on the road, rainy days, and traffic jams, using only a single standard camera. We propose a robust adaptive multicue*

segmentation strategy that detects foreground pixels corresponding to moving and stopped vehicles, even with noisy images due to compression. First, the approach adaptively thresholds a combination of luminance and chromaticity disparity maps between the learned background and the current frame. It then adds extra features derived from gradient differences to improve the segmentation of dark vehicles with casted shadows and removes headlight reflections on the road. The segmentation is further used by a two-step tracking approach, which combines the simplicity of a linear 2-D Kalman filter and the complexity of a 3-D volume estimation using Markov chain Monte Carlo (MCMC) methods. Experimental results show that our method can count and classify vehicles in real time with a high level of performance under different environmental situations comparable with those of inductive loop detectors.

A.3 **Semi-automatic tracking-based labeling tool for automotive applications**

Título: Semi-automatic tracking-based labeling tool for automotive applications
Autores: A. Cortes, O. Senderos, N. Aranjuelo, M. Nieto, and O. Otaegui
Congress: 22st ITS World Congress
Año: 2015-10-05
Relación: Este artículo fue el resultado del desarrollo de la herramienta de anotación utilizada para la generación de los *datasets* del capítulo 4
DOI: `http://dx.doi.org/10.1109/BMSB.2017.7986179`
Abstract: *The trend toward smart cars continues to build momentum in the automotive industry. As vision based sensors proliferate in the vehicles the quantity and variety of data will become overwhelming and difficult to be processed effectively. Computer vision-based approaches can be trained and evaluated with such data sources once adequately labeled. However, accurately annotating entities in video is labor intensive and an expensive task. As the quantity of available video grows, traditional solutions to this task are unable to scale to meet the needs of sectors like automotive or security. We present a semi-automatic multi-purpose annotation tool which reduces the manual annotation effort by enabling the user to verify automatically generated annotations,*

rather than annotating from scratch. This tool has been successfully used in a variety of example applications in the domain of Advanced Driver Assistance Systems.

Estas publicaciones que se presentan a continuación surgieron dentro del contexto de trabajo en el que se enmarcaba la tesis:

A.4 **Real-time lane tracking using Rao-Blackwellized particle filter**

Título: Real-time lane tracking using Rao-Blackwellizedparticle filter
Autores: M. Nieto, A. Cortés, O. Otaegui, J. Arrospide, L. Salgado
Revista: JOURNAL OF REAL-TIME IMAGE PROCESSING
Páginas: 179-191
Año: 2016
Tipo: Q2
DOI: `https://doi.org/10.1007/S11554-012-0315-0`
Abstract: *A novel approach to real-time lane modeling using a single camera is proposed. The proposed method is based on an efficient design and implementation of a particle filter which applies the concepts of the Rao-Blackwellized particle filter (RBPF) by separating the state into linear and non-linear parts. As a result the dimensionality of the problem is reduced, which allows the system to perform in real-time in embedded systems. The method is used to determine the position of the vehicle inside its own lane and the curvature of the road ahead to enhance the performance of advanced driver assistance systems. The effectiveness of the method has been demonstrated implementing a prototype and testing its performance empirically on road sequences with different illumination conditions (day and nightime), pavement types, traffic density, etc. Results show that our proposal is capable of accurately determining if the vehicle is approaching the lane markings (Lane Departure Warning), and the curvature of the road ahead, achieving processing times below 2 ms per frame for laptop CPUs, and 12 ms for embedded CPUs.*

A.5 Perspective Multiscale Detection and Tracking of Persons

Título: Perspective Multiscale Detection and Tracking of Persons
Autores: Nieto, Marcos and Ortega, Juan and Cortes, Andoni and Gaines, Seán
Proceedings: MultiMedia Modeling
Páginas: 92–103
Editor: Springer International Publishing
Año: 2014
Abstract: *The efficient detection and tracking of persons in videos has widrespread applications, specially in CCTV systems for surveillance or forensics applications. In this paper we present a new method for people detection and tracking based on the knowledge of the perspective information of the scene. It allows alleviating two main drawbacks of existing methods: (i) high or even excessive computational cost associated to multiscale detection-by-classification methods; and (ii) the inherent difficulty of the CCTV, in which predominate partial and full occlusions as well as very high intra-class variability. During the detection stage, we propose to use the homograhy of the dominant plane to compute the expected sizes of persons at different positions of the image and thus dramatically reduce the number of evaluation of the multiscale sliding window detection scheme. To achieve robustness against false positives and negatives, we have used a combination of full and upper-body detectors, as well as a Data Association Filter (DAF) inspired in the well-known Rao-Blackwellization-based particle filters (RBPF). Our experiments demonstrate the benefit of using the proposed perspective multiscale approach, compared to conventional sliding window approaches, and also that this perspective information can lead to useful mixes of full-body and upper-body detectors.*

A.6 Computer vision: the emerging cost-effective technology for vehicles

Título: Computer vision: the emerging cost-effective technology for vehicles
Autores: M. Nieto, J. D. Ortega, O. Otaegui, A. Cortés, J. Barandiarán, L. Unzueta

Proceedings: 9th ITS European Congress, At Dublin, Ireland

Año: 2013

Abstract: *In this paper we analyze the recent exponential growth of applications based on video processing in the framework of Advanced Driver Assistance Systems (ADAS). Specifically, we focus on the cost-effective solutions provided by computer vision methods for services like lane departure warning systems, collision avoidance systems, etc. Along the paper our own contributions are described as examples of the state of the art from the perspective of real-time by design, searching a trade-off between the accuracy and reliability of the designed algorithms, and the restrictive computational, economical and design requisites of embedded platforms.*

A.7 **Perspective Multiscale Detection of Vehicles for Real-Time Forward Collision Avoidance Systems**

Título: Perspective Multiscale Detection of Vehicles for Real-Time Forward Collision Avoidance Systems

Autores: Ortega, Juan Diego and Nieto, Marcos and Cortes, Andoni and Florez, Julian"

Proceedings: Advanced Concepts for Intelligent Vision Systems

Editor: Springer International Publishing

Páginas: 645–656

Año: 2013

Abstract: *This paper presents a single camera vehicle detection technique for forward collision warning systems suitable to be integrated in embedded platforms. It combines the robustness of detectors based on classification methods with an innovative perspective multi-scale procedure to scan the images that dramatically reduces the computational cost associated with robust detectors. In our experiments we compare different implementation classifiers in search for a trade-off between the real-time constraint of embedded platforms and the high detection rates required by safety applications.*

A.8 On creating vision-based advanced driver assistance systems

Título: On creating vision-based advanced driver assistance systems
Autores: M. Nieto, O. Otaegui, G. Velez, J. D. Ortega, A. Cortés
Revista: IET INTELLIGENT TRANSPORT SYSTEMS
Páginas: 59-66
Año: 2015
Tipo: Q3
DOI: https://doi.org/10.1049/iet-its.2013.0167
Abstract: *In this study, the authors analyse the exponential growth of advanced driver assistance systems based on video processing in the past decade. Specifically, they focus on how research and innovative ideas can finally reach the market as cost-effective solutions. They explore well-known computer vision methods for services like lane departure warning systems, collision avoidance systems and point out potential future trends according to a review of the state-of-the-art. Along this study, the authorsówn contributions are described as examples of such systems from the perspective of real-time by design, pursuing a trade-off between the accuracy and reliability of the designed algorithms and the restrictive computational, economical and design requisites of embedded platforms.*

A.9 Vehicle tracking and classification in challenging scenarios via slice sampling

Título: Vehicle tracking and classification in challenging scenarios via slice sampling
Autores: M. Nieto, L. Unzueta, J. Barandiarán, A. Cortés, O. Otaegui, P. Sanchez
Revista: EURASIP JOURNAL ON ADVANCES IN SIGNAL PROCESSING
Año: 2011
Tipo: Q2
DOI: https://doi.org/10.1186/1687-6180-2011-95
Abstract: *This article introduces a 3D vehicle tracking system in a traffic surveillance*

environment devised for shadow tolling applications. It has been specially designed to operate in real time with high correct detection and classification rates. The system is capable of providing accurate and robust results in challenging road scenarios, with rain, traffic jams, casted shadows in sunny days at sunrise and sunset times, etc. A Bayesian inference method has been designed to generate estimates of multiple variable objects entering and exiting the scene. This framework allows easily mixing different nature information, gathering in a single step observation models, calibration, motion priors and interaction models. The inference of results is carried out with a novel optimization procedure that generates estimates of the maxima of the posterior distribution combining concepts from Gibbs and slice sampling. Experimental tests have shown excellent results for traffic-flow video surveillance applications that can be used to classify vehicles according to their length, width, and height. Therefore, this vision-based system can be seen as a good substitute to existing inductive loop detectors.

Estas últimas son otras publicaciones dentro del mundo de la visión artificial:

A.10 **Single camera railways track profile inspection usingan slice sampling-based particle filter**

Título: Single camera railways track profile inspection usingan slice sampling-based particle filter

Autores: Nieto, Marcos, Cortés, Andoni, Barandiaran, Javier, Otaegui, Oihana and Etxabe, Iñigo

Proceedings: Computer Vision, Imaging and Computer Graphics. Theory and Application

Páginas: pages

Editor: Springer Berlin Heidelberg

Año: 2013

Tipo: Q4

DOI: `http://dx.doi.org/10.1007/978-3-642-38241-3_22`

Abstract: *An automatic method for rail inspection is introduced in this paper. The method detects rail flaws using computer vision algorithms. Unlike other methods*

designed for the same goal, we propose a method that automatically fits a 3D rail model to the observations. The proposed strategy is based on the novel combination of a simple but effective laser-camera calibration procedure with the application of an MCMC (Markov Chain Monte Carlo) framework. The proposed particle filter uses the efficient overrelation slice sampling method, which allows us to exploit the temporal coherence of observations and to obtain more accurate estimates than with other sampling techniques. The results show that the system is able to robustly obtain measurements of the wear of the rail. The two other contributions of the paper are the successfull introuction of the slice sampling technique into MCMC particle filters and the proposed online and flexible method for camera-laser calibration.

A.11 MCMC particle filter with overrelaxated slice sampling for accurate rail inspection

Título: MCMC particle filter with overrelaxated slice sampling for accurate rail inspection

Autores: Nieto, Marcos and Cortes, Andoni and Otaegui, Oihana and Etxabe, I."

Proceedings: Proceedings of the International Conference on Computer Vision Theory and Applications

Páginas: 164-172

Año: 2012

Abstract: *This paper introduces a rail inspection system which detects rail flaws using computer vision algorithms. Unlike other methods designed for the same purpose, we propose a method that automatically fits a 3D rail model to the observations during regular services and normal traffic conditions. The proposed strategy is based on a novel application of the slice sampling technique with overrelaxation in the framework of MCMC (Markov Chain Monte Carlo) particle filters. This combination allows us to efficiently exploit the temporal coherence of observations and to obtain more accurate estimates than with other techniques such as importance sampling or Metropolis-Hastings. The results show that the system is able to efficient and robustly obtain measurements of the wear of the rails, while we show as well that it is possible to*

introduce the slice sampling technique into MCMC particle filters.

A.12 Fast Multistage Algorithm for K-NN

Título: Multidimensional Multistage K-NN Classifier for Handwritten Character Recognition

Autores: I. Soraluze Arriola, C. Rodríguez Lafuente, F. Boto Sánchez, A. Cortés

Proceedings: CIARP

Editor: Springer Berlin Heidelberg

Páginas: 448-455

Año: 2003

DOI: https://doi.org/10.1007/978-3-540-24586-5_55

Abstract: *In this paper we present a way to reduce the computational cost of k-NN classifiers without losing classification power. Hierarchical or multistage classifiers have been built with this purpose. These classifiers are designed putting incrementally trained classifiers into a hierarchy and using rejection techniques in all the levels of the hierarchy apart from the last. Results are presented for different benchmark data sets: some standard data sets taken from the UCI Repository and the Statlog Project, and NIST Special Databases (digits and upper-case and lower-case letters). In all the cases a computational cost reduction is obtained maintaining the recognition rate of the best individual classifier obtained.*

A.13 Noisy Digit Classification with Multiple Specialist

Título: Noisy Digit Classification with Multiple Specialist

Autores: A. Cortés, F. Boto, C. Rodríguez

Revista: Lecture Notes in Computer Science

Páginas: 601-608

Año: 2005

DOI: https://doi.org/10.1007/11551188_66

Abstract: *A multi-classifier formed by specialised classifiers for noise produced by an*

image is shown in this work. A study has been carried out in the case of structure noisy images. Classifiers based on neighbourhood criteria are used in this work, the zoning global feature and the Euclidean distance too. The experiments have been carried out with images of typewritten digits, taken from forms of the Bank of Spain. Trying to obtain a strong database to support the experiments, we have added noise to the images of the digits. The recognition rate improves from 64.58% to 96.18%.

A.14 Cut Digits Classification with k-NN Multi-specialist

Título: Cut Digits Classification with k-NN Multi-specialist
Autores: F. Boto, A. Cortés, C. Rodríguez
Título del libro: Document Analysis Systems VII
Páginas: 496–505
Publisher: Springer Berlin Heidelberg
Año: 2006
Isbn: 978-3-540-32157-6
Abstract: *A multi-classifier formed by specialised classifiers for noise produced by an image is shown in this work. A study has been carried out in the case of cut images, where tree cases of specialization are considered. Classifiers based on neighbourhood criteria are used, the zoning global feature and the Euclidean distance too. Furthermore, the paper explains a modification of the Euclidean distance for classifying cut digits. The experiments have been carried out with images of typewritten digits, taken from real forms. Trying to obtain a strong database to support the experiments, we have cut images deliberately. The recognition rate improves from 84.6% to 97.70%, but whether the system provides information about the disturbance of the image, it can achieve a 98.45%.*

Bibliografía

[A. Mikolajczyk18] M. Grochowski A. Mikolajczyk. Data augmentation for improving deep learning in image classification problem. In *2018 International Interdisciplinary PhD Workshop (IIPhDW)*, pages 117–122, May 2018. 130

[Afifi and Brown19] Mahmoud Afifi and Michael S. Brown. What else can fool deep learning? addressing color constancy errors on deep neural network performance. In *International Conference on Computer Vision (ICCV)*, 2019. 110

[Aghdam et al.16] Hamed Habibi Aghdam, Elnaz Jahani Heravi, and Domenec Puig. A practical approach for detection and classification of traffic signs using convolutional neural networks. *Robotics and Autonomous Systems*, 84(Supplement C):97 – 112, 2016. 113

[Aghdam et al.17] Hamed Habibi Aghdam, Elnaz Jahani Heravi, and Domenec Puig. A practical and highly optimized convolutional neural network for classifying traffic signs in real-time. *Int. J. Comput. Vision*, 122(2):246–269, apr 2017. 112, 113

[Ahmed Madani16] Rubiyah Yusof Ahmed Madani. Malaysian traffic sign dataset for traffic sign detection and recognition systems. In *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 2016. 134

[Akilan et al.18] T. Akilan, Q. M. Jonathan Wu, W. Jiang, A. Safaei, and J. Huo. New trend in video foreground detection using deep learning. In *2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 889–892, 2018. 74

[Arcos-García et al.18] A. Arcos-García, J. A. Álvarez García, and L. M. Soria-Morillo. Deep neural network for traffic sign recognition systems: An analysis of spatial transformers and stochastic optimisation methods. *Neural networks : the official journal of the International Neural Network Society*, 99:158—165, March 2018. 113, 130

[Aubert et al.04] D. Aubert, F. Guichard, and S. Bouchafa. Time-scale change detection applied to real-time abnormal stationarity monitoring. *Real-Time Imaging*, 10(1):9–22, 2004. 75

[Baird et al.92] Henry S. Baird, K. Yamamoto, and H. Bunke. *Structured Document Image Analysis*. Springer-Verlag, Berlin, Heidelberg, 1992. 111

[Benezeth et al.10] Y. Benezeth, Pierre-Marc Jodoin, Bruno Emile, Helene Laurent, and Christophe Rosenberger. Comparative study of background subtraction algorithms. *Journal of Electronic Imaging*, 19(3):1 – 12, 2010. 74

[Blauensteiner et al.06] P. Blauensteiner, H. Wildenauer, A. Hanbury, and M. Kampel. Motion and shadow detection with an improved colour model. In *IEEE Proc. International Conference on Signal and Image Processing*, pages 627–632, 2006. 77, 95

[Bloice et al.19] Marcus D Bloice, Peter M Roth, and Andreas Holzinger. Biomedical image augmentation using augmentor. *Bioinformatics*, 04 2019. 110, 129

[Bochkovskiy et al.20] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020. 110

[Bouwmans14] Thierry Bouwmans. Traditional and recent approaches in background modeling for foreground detection: An overview. *Computer Science Review*, 11-12:31 – 66, 2014. 74

[Buch et al.10] Norbert Buch, J. Orwell, and Sergio Velastin. Urban road user detection and classification using 3d wire frame models. *Computer Vision, IET*, 4:105 – 116, 07 2010. 71

[B.V. and Karthikeyan18] S. S. B.V. and A. Karthikeyan. Computer vision based advanced driver assistance system algorithms with optimization techniques-a review. In *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 821–829, March 2018. 112

[Carlson et al.18] Alexandra Carlson, Katherine A. Skinner, and Matthew Johnson-Roberson. Modeling camera effects to improve deep vision for real and synthetic data. *CoRR*, abs/1803.07721, 2018. 111

[Chigorin and Konushin13] A Chigorin and Anton Konushin. A system for large-scale automatic traffic sign recognition and mapping. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-3/W3:13–17, 10 2013. 111

[Cimpoi et al.14] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014. 31, 132, 133

[Cireşan et al.12] D. Cireşan, Ueli Meier, Jonathan Masci, and Jurgen Schmidhuber. Multi-column deep neural network for traffic sign classification. *Neural Networks*, 32:333 – 338, 2012. 112

[Ciresan et al.11] D. Ciresan, U. Meier, J. Masci, and J. Schmidhuber. A committee of neural networks for traffic sign classification. In *The 2011 International Joint Conference on Neural Networks*, pages 1918–1921, July 2011. 112

[Commission18] The European Commission. Inlane project, 2018. 128

[Cortés et al.15] Andoni Cortés, Orti Senderos, Marcos Nieto, Nerea Aranjuelo, and Oihana Otaegui. Semi-automatic tracking-based labeling tool for automotive applications. In *22nd ITS World Congress*, Bordeaux, France, 2015. 126

[Cortés19] Andoni Cortés. Paper data, 2019. 142, 149

[Cubuk et al.19] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation strategies from data. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 110

[Cucchiara et al.00] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati. Statistic and knowledge-based moving object detection in traffic scenes. In *IEEE Proc. Intelligent Transportation Systems*, pages 27–32, 2000. 75

[Cucchiara et al.01] R. Cucchiara, C. Grana, G. Neri, M. Piccardi, and A. Prati. The Sakbot system for moving object detection and tracking. In P. Remagnino, G. A. Jones, N. Paragios, and C. S. Regazzoni, editors, *Video-Based Surveillance Systems-Computer Vision and Distributed Processing*, chapter 12, pages 145–157. Springer, 2001. 76, 77, 93

[Davis and Goadrich06] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 233–240, New York, NY, USA, 2006. ACM. 134

[Design13] DMA Design. Grand theft auto. [PC], [PlayStation] y [Game Boy Color], 2013. 126

[Dosovitskiy et al.17] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017. 126

[Doval et al.19] G. N. Doval, A. Al-Kaff, J. Beltrán, F. G. Fernández, and G. Fernández López. Traffic sign detection and 3d localization via deep convolutional neural networks and stereo vision. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1411–1416, Oct 2019. 113

[Elgammal et al.00] A. Elgammal, D. Harwood, and L. Davis. Non-parametric model for background subtraction. In *Proc. of the European Conference on Computer Vision-Part II, LNCS 1843*, pages 751–767, 2000. 75

[Enzweiler and Gavrila09] M. Enzweiler and D. M. Gavrila. Monocular pedestrian detection: Survey and experiments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2179–2195, 2009. 108

[Ertler et al.20] Christian Ertler, Jerneja Mislej, Tobias Ollmann, Lorenzo Porzi, and Yubin Kuang. Traffic sign detection and classification around the world. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 134

[Garcia-Garcia et al.20] Belmar Garcia-Garcia, Thierry Bouwmans, and Alberto Jorge Rosales Silva. Background subtraction in real applications: Challenges, current models and future directions. *Computer Science Review*, 35, February 2020. 74

[Gershenfeld et al.04] Neil Gershenfeld, Raffi Krikorian, and Danny Cohen. The internet of things. *Scientific American*, 291(4):76–81, oct 2004. 3

[Ghosh and Lee00] Sumit Ghosh and Tony Szu-Hsien Lee. Intelligent transportation systems : New principles and architectures. In *INTELLIGENT TRANSPORTATION SYSTEMS : NEW PRINCIPLES AND ARCHITECTURES*, 2000. 13

[Girshick et al.14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, page 580–587, USA, 2014. IEEE Computer Society. 109

[Girshick15] Ross Girshick. Fast r-cnn. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, page 1440–1448, USA, 2015. IEEE Computer Society. 109

[Goodfellow et al.16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`. 50

[Gupte et al.02] S. Gupte, O. Masoud, R. Martin, and N. Papanikolopoulos. Detection and classification of vehicles. *IEEE Trans. Intell. Transp. Syst.*, 3:37–47, 2002. 75

[Haag and Nagel00] M. Haag and H. H. Nagel. Incremental recognition of traffic situations from video image sequences. *Image and Vision Computing*, 18:137–153, 2000. 71

[Hanbury and Serra02] Allan Hanbury and Jean Serra. A 3d-polar coordinate colour representation suitable for image analysis. *Computer Vision and Image Understanding - CVIU*, 01 2002. 18

[He et al.14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 346–361, Cham, 2014. Springer International Publishing. 109

[He et al.16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 58

[Horprasert et al.99] T. Horprasert, D. Harwood, and L. Davis. A statistical approach for real-time robust background subtraction and shadow detection. In *IEEE Proc. of the International Conference on Computer Vision (ICCV'99)*, pages 256–261, 1999. 75, 76, 79, 84

[Houben et al.13] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. Detection of traffic signs in real-world images: The german traffic sign detection benchmark. In *International Joint Conference on Neural Networks*, 2013. 1288. 134

[Hu et al.06] J. Hu, T. Su, and S. Jeng. Robust background subtraction with shadow and highlight removal for indoor surveillance. In *IEEE/RSJ Proc. International Conference on Intelligent Robots and Systems*, pages 4545–4550, 2006. 75

[Huerta et al.08] I. Huerta, D. Rowe, and M. Mozerov. Background subtraction fusing colour, intensity and edge cues. In *Proc. Conference on Articulated Motion and Deformable Objects (AMDO'08), LNCS 5098*, pages 279–288, 2008. 79, 80

[Huerta et al.09] I. Huerta, M. Holte, T. Moeslund, and J. González. Detection and removal of chromatic moving shadows in surveillance scenarios. In *IEEE Proc. International Conference of Computer Vision*, pages 1499–1506, 2009. 77, 79, 93

[Javadi18] Mohammad Saleh Javadi. Computer vision algorithms for intelligent transportation systems applications, 2018. 13

[Jin et al.14] J. Jin, K. Fu, and C. Zhang. Traffic sign recognition with hinge loss trained convolutional neural networks. *IEEE Transactions on Intelligent Transportation Systems*, 15(5):1991–2000, Oct 2014. 113

[Johansson et al.09] B. Johansson, J. Wiklund, P. Forssén, and G. Granlund. Combining shadow detection and simulation for estimation of vehicle size and position. *Pattern Recognition Letters*, 30:751–759, 2009. 71, 77

[Kar et al.19] Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba, and Sanja Fidler. Meta-sim: Learning to generate synthetic datasets. *CoRR*, abs/1904.11621, 2019. 111

[Kim et al.04] K. Kim, T. Chalidabhongse, D. Harwood, and L. Davis. Background modeling and subtraction by codebook construction. In *Proc. International Conference on Image Processing (ICIP'04)*, pages 2–5, 2004. 80, 93

[Kim et al.05] K. Kim, D. Harwood, and L.S. Davis. Background updating for visual surveillance. In *Proc. International Symposium on Visual Computing (ISVC'05), LNCS 3804*, pages 337–346, 2005. 75, 90

[Krizhevsky et al.12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, 2012. 111, 131

[Larsson and Felsberg11] Fredrik Larsson and Michael Felsberg. Using fourier descriptors and spatial models for traffic sign recognition. In Anders Heyden and Fredrik Kahl, editors, *Image Analysis*, pages 238–249, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. 134

[Liu et al.16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 21–37, Cham, 2016. Springer International Publishing. 110

[Liu et al.19] Li Liu, Wanli Ouyang, X. Wang, P. Fieguth, J. Chen, Xinwang Liu, and M. Pietikäinen. Deep learning for generic object detection: A survey. *International Journal of Computer Vision*, 128:261–318, 2019. 108, 109

[Loce et al.17] Robert P. Loce, Raja Bala, and Mohan Trivedi. *Computer Vision and Imaging in Intelligent Transportation Systems*. Wiley-IEEE Press, 1st edition, 2017. 13

[Long et al.20] Xiang Long, Kaipeng Deng, Guanzhong Wang, Yang Zhang, Qingqing Dang, Yuan Gao, Hui Shen, Jianguo Ren, Shumin Han, Errui Ding, and Shilei Wen. Pp-yolo: An effective and efficient implementation of object detector, 2020. 110

[Lu et al.19] J. Lu, S. Tang, J. Wang, H. Zhu, and Y. Wang. A review on object detection based on deep convolutional neural networks for autonomous driving. In *2019 Chinese Control And Decision Conference (CCDC)*, pages 5301–5308, 2019. 109

[Lydia and Francis19] Agnes Lydia and Sagayaraj Francis. A survey of optimization techniques for deep learning networks. pages 2454–9150, 05 2019. 51

[Maddern et al.17] Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. 1 year, 1000 km: The oxford robotcar dataset. *The International Journal of Robotics Research*, 36(1):3–15, 2017. 134

[Mayo and Tapamo09] Z. Mayo and J. R. Tapamo. Background subtraction survey for highway surveillance. In *Proc. of the Annual Symposium of the Pattern Recognition Association of South Africa (PRASA'09)*, pages 77–82, Stellenbosch, South Africa, 2009. 71

[Mech and Ostermann99] R. Mech and J. Ostermann. Detection of moving cast shadows for object segmentation. *IEEE Trans. Multimedia*, 1:65–76, 1999. 75

[Mikic et al.00] I. Mikic, P. Cosman, G. Kogut, and M. Trivedi. Moving shadow and object detection in traffic scenes. In *IEEE Proc. International Conference on Pattern Recognition*, pages 321–324, 2000. 76

[Mine et al.19] Tsunenori Mine, Akira Fukuda, and Shigemi Ishida. *Intelligent Transport Systems for Everyone's Mobility*. Springer, 01 2019. 13

[Mogelmose et al.12] A. Mogelmose, M. M. Trivedi, and T. B. Moeslund. Learning to detect traffic signs: Comparative evaluation of synthetic and real-world datasets. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 3452–3455, Nov 2012. 111

[Moiseev et al.13] Boris Moiseev, Artem Konev, Alexander Chigorin, and Anton Konushin. Evaluation of traffic sign recognition methods trained on synthetically generated data. In Jacques Blanc-Talon, Andrzej Kasinski, Wilfried Philips, Dan Popescu, and Paul Scheunders, editors, *Advanced Concepts for Intelligent Vision Systems*, pages 576–583, Cham, 2013. Springer International Publishing. 111

[NREL00] NREL. SOLPOS 2.0, distributed by the national renewable energy laboratory. Center for renewable energy resources. Renewable resource data center, 2000. 68

[Organization09] World Health Organization. Global status report on road safety. Technical report, World Health Organization, 2009. 99

[Organization18] World Health Organization. Global status report on road safety 2018. Technical report, World Health Organization, 12 2018. 99

[Pang et al.07] Clement Pang, William Lam, and Nelson Yung. A method for vehicle count in the presence of multiple-vehicle occlusions in traffic images. *Intelligent Transportation Systems, IEEE Transactions on*, 8:441 – 459, 10 2007. 71

[paperswithcode] paperswithcode. Image classification on imagenet. 59

[Parks and Fels08] D. H. Parks and S. S. Fels. Evaluation of background subtraction algorithms with post-processing. In *2008 IEEE Fifth International Conference on Advanced Video and Signal Based Surveillance*, pages 192–199, 2008. 74

[Perez and Wang17] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *CoRR*, abs/1712.04621, 2017. 110

[Prati et al.03] A. Prati, I. Mikic, M. M. Tridevi, and R. Cucchiara. Detecting moving shadows: Algorithms and evaluation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25:918–923, 2003. 76, 77

[Prescan] TASS International Prescan. Prescan: Simulation of adas and active safety. 126

[Preteux92] E. Preteux. Watershed and skeleton by influence zones: A distance-based approach. *Journal of Mathematical Imaging and Vision*, 1:239–255, 1992. 81

[Ranjan et al.18] Rajeev Ranjan, Shalini De Mello, and Jan Kautz. Light-weight head pose invariant gaze tracking. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2237–22378, 2018. 112

[Redmon and Farhadi18] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018. 110

[Redmon et al.16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. 110

[Ren et al.15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28, pages 91–99. Curran Associates, Inc., 2015. 110

[Reza04] A. M. Reza. Realization of the contrast limited adaptive histogram equalization (CLAHE) for real-time image enhancement. *Journal of Mathematical Imaging and Vision*, 38(1):35–44, 2004. 82, 84

[Richter et al.16] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 102–118, 2016. 111

[Roller et al.93] D. Roller, K. Daniilidis, and H. H. Nagel. Model-based object tracking in monocular image sequences of road traffic scenes. *International Journal of Computer Vision*, 10:257–281, 1993. 76

[Satzoda and Trivedi17] Ravi Satzoda and Mohan Trivedi. *Vision-Based Integrated Techniques for Collision Avoidance Systems*, chapter 12, pages 305–320. John Wiley and Sons, Ltd, 2017. 112

[Schaefer et al.06] Scott Schaefer, Travis McPhail, and Joe Warren. Image deformation using moving least squares. *ACM Trans. Graph.*, 25:533–540, 07 2006. 24

[Sermanet et al.14] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks, 2014. 110

[Shah et al.17] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2017. 126

[Shakhuro and Konushin16] Vladislav Shakhuro and Anton Konushin. Russian traffic sign images dataset. *Computer Optics*, 40(2):294–300, 2016. 134

[Shorten and Khoshgoftaar19] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, Jul 2019. 110

[Simard et al.03] Patrice Y. Simard, Dave Steinkraus, and John Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition - Volume 2*, ICDAR '03, page 958. IEEE Computer Society, August 2003. 25, 111

[Simulator12] SCS EuroTruck Simulator. Euro truck simulator 2. [PC CD-ROM], [ONLINE-Steam], 2012. v1.35. 126

[Stauffer and Grimson99] C. Stauffer and W. E. Grimson. Adaptive background mixture models for real-time tracking. In *IEEE Proc. Conference on Computer Vision and Pattern Recognition (CVPR'99)*, volume 2, pages 246–252, 1999. 75

[Studios16] TML Studios. Fernbus simulator. [PC CD-ROM], [ONLINE-Steam], 2016. 126

[Sun et al.06] Zehang Sun, George Bebis, and Ronald Miller. On-road vehicle detection: A review. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(5):694–711, May 2006. 108

[Suzuki and Be85] S. Suzuki and K. Be. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985. 89

[T. Chalidabhongse03] D. Harwood L. Davis T. Chalidabhongse, K. Kim. A perturbation method for evaluating background subtraction algorithms. In *IEEE Proc. Joint International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VSPETS'03)*, 2003. 75, 81, 93

[Tabernik and Skočaj19] Domen Tabernik and Danijel Skočaj. Deep Learning for Large-Scale Traffic-Sign Detection and Recognition. *IEEE Transactions on Intelligent Transportation Systems*, 2019. 114, 134

[Temel et al.17] D. Temel, G. Kwon, M. Prabhushankar, and G. AlRegib. Cure-tsr: Challenging unreal and real environments for traffic sign recognition. In *Neural Information Processing Systems (NeurIPS) Workshop on Machine Learning for Intelligent Transportation Systems*, 2017. 134

[Tezcan et al.20] M. O. Tezcan, P. Ishwar, and J. Konrad. Bsuv-net: A fully-convolutional neural network for background subtraction of unseen videos. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 2763–2772, 2020. 75

[Timofte et al.14] Radu Timofte, Karel Zimmermann, and Luc Van Gool. Multi-view traffic sign detection, recognition, and 3d localisation. *Machine Vision and Applications*, 25(3):633–647, Apr 2014. 113, 134

[Tremblay et al.18] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. *CoRR*, abs/1804.06516, 2018. 111

[Tsai86] R. Tsai. An efficient and accurate camera calibration technique for 3d machine vision. In *CVPR 1986*, 1986. 120

[Šegvic et al.10] S. Šegvic, K. Brkić, Z. Kalafatić, V. Stanisavljević, M. Ševrović, D. Budimir, and I. Dadić. A computer vision assisted geoinformation inventory for traffic infrastructure. In *13th International IEEE Conference on Intelligent Transportation Systems*, pages 66–73, Sep. 2010. 134

[Velez et al.15] Gorka Velez, Oihana Otaegui, Juan Ortega, Marcos Nieto, and Andoni Cortes. On creating vision-based advanced driver assistance systems. *IET Intelligent Transport Systems*, 9:59–66, 02 2015. 112

[Verschae and Ruiz-del Solar15] Rodrigo Verschae and Javier Ruiz-del Solar. Object detection: Current and future directions. *Frontiers in Robotics and AI*, 2:29, 2015. 109

[Wali et al.19] Safat Wali, Majid Abdullah, M. A. Hannan, Aini Hussain, Salina Samad, Pin Jern Ker, and Muhamad Mansor. Vision-based traffic sign detection and recognition systems: Current trends and challenges. *Sensors*, 19:2093, 05 2019. 114

[Wali15] Safat Wali. Comparative survey on traffic sign detection and recognition: a review. *Przeglad Elektrotechniczny*, 1:40–44, 12 2015. 112

[Wang et al.19] Qi Wang, Junyu Gao, Wei Lin, and Yuan Yuan. Learning from synthetic data for crowd counting in the wild. *CoRR*, abs/1903.03303, 2019. 111

[wikimediaa] wikimedia. Road signs. 129

[wikimediab] wikimedia. Speed limit. 129

[wikipediaa] wikipedia. Comparison of european road signs. 129

[wikipediab] wikipedia. Country codes. 134

[William et al.19] M. M. William, P. S. Zaki, B. K. Soliman, K. G. Alexsan, M. Mansour, M. El-Moursy, and K. Khalil. Traffic signs detection and recognition system using deep learning. In *2019 Ninth International Conference on Intelligent Computing and Information Systems (ICICIS)*, pages 160–166, 2019. 114

[Wong et al.18] Alexander Wong, Mohammad Javad Shafiee, and Michael St. Jules. muNet: A Highly Compact Deep Convolutional Neural Network Architecture for Real-time Embedded Traffic Sign Classification. *CoRR*, abs/1804.00497, 2018. 113, 130

[Wren et al.97] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19:780–785, 1997. 74, 90

[Xiao et al.20] Youzi Xiao, Zhiqiang Tian, Jiachen Yu, Yinshu Zhang, Shuai Liu, Shaoyi Du, and Xuguang Lan. A review of object detection based on deep learning. *Multimedia Tools and Applications*, 79, 09 2020. 109

[Yu and Koltun16] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations (ICLR)*, May 2016. 45

[Zafeiriou et al.15] Stefanos Zafeiriou, Cha Zhang, and Zhengyou Zhang. A survey on face detection in the wild: Past, present and future. *Computer Vision and Image Understanding*, 138:1 – 24, 2015. 108

[Zhang et al.17] Jianming Zhang, Manting Huang, Xiaokang Jin, and Xudong Li. A Real-Time Chinese Traffic Sign Detection Algorithm Based on Modified YOLOv2. *Algorithms*, 10(4), 2017. 134

[Zhao et al.19] Zhong-Qiu Zhao, Peng Zheng, Shou-Tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, PP:1–21, 01 2019. 109

[Zhiqiang and Jun17] W. Zhiqiang and L. Jun. A review of object detection based on convolutional neural network. In *2017 36th Chinese Control Conference (CCC)*, pages 11104–11109, 2017. 109

[Zhu et al.16a] Yingying Zhu, Chengquan Zhang, Duoyou Zhou, Xinggang Wang, Xiang Bai, and Wenyu Liu. Traffic sign detection and recognition using fully convolutional network guided proposals. *Neurocomputing*, 214(Sup. C):758 – 766, 2016. 113

[Zhu et al.16b] Zhe Zhu, Dun Liang, Songhai Zhang, Xiaolei Huang, Baoli Li, and Shimin Hu. Traffic-sign detection and classification in the wild. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 134

[Zhu et al.19] Meixin Zhu, Jingyun Hu, Ziyuan Pu, Zhiyong Cui, Liangwu Yan, and Yinhai Wang. Traffic sign detection and recognition for autonomous driving in virtual simulation environment. *CoRR*, abs/1911.05626, 2019. 113

[Zitnick and Dollár14] C. Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 391–405, Cham, 2014. Springer International Publishing. 113