Universidad Zaragoza
1542

Trabajo Fin de Grado

Aplicación de un sensor acústico distribuido sobre fibra óptica al control de carreteras

Optical fiber distributed acoustic sensor for smart road control

Autor/es

Guillermo López Lanuza

Director/es

Íñigo Salinas Áriz

Escuela de Ingeniería y Arquitectura

Curso 2019

Aplicación de un sensor acústico distribuido sobre fibra óptica al control de carreteras

Resumen

Este Trabajo de Fin de Grado es una contribución al desarrollo de una aplicación de control de tráfico en carreteras, a partir de los datos de un sensor acústico distribuido basado en fibra óptica funcionando mediante reflectometría coherente.

En primer lugar se estudian y evalúan las trazas recibidas de dicho sensor, tanto en el dominio temporal como espectral, con el objetivo de mejorar los datos brutos a partir de una serie de técnicas de procesado en frecuencia.

Posteriormente se realiza un análisis de forma manual de las trazas obtenidas a lo largo de un conjunto de días de funcionamiento del sensor, con el principal fin de encontrar eventos inhabituales que son posibles de detectar con el sistema del que se dispone.

Por último, se desarrolla una aplicación que permite controlar la velocidad media y el número medio de coches en diferentes tramos de reducida longitud a lo largo de la vía donde está instalado el sensor, lo cual permitirá un control en tiempo real de la situación de la carretera y servirá para alarmar en caso de que se produzca una situación anormal.

Este trabajo supone un avance en la aplicación práctica del mencionado sensor distribuido, así como un punto de partida para futuros estudios y aplicaciones.

Agradecimientos

Me gustaría agradecer en primer lugar a toda mi familia por el esfuerzo y la confianza depositados en mí durante estos cuatro años de duración del grado.

Por otro lado, dar gracias a los compañeros y las compañeras del laboratorio, integrantes del Grupo de Tecnologías Fotónicas de la Universidad de Zaragoza, que me han ayudado en todo lo posible durante la elaboración del trabajo. En especial, agradecer a Iñigo Salinas por darme la oportunidad de llevar a cabo este proyecto y guiarme durante la realización del mismo.

Además, agradecer a la empresa Aragón Photonics Labs por confiar en mí para la continuación de este proyecto y haberme ayudado y proporcionado todo el material necesario.

Por último, gracias a todos mis amigos y compañeros que me han acompañado estos cuatro años haciendo que todo sea más sencillo.

Índice general

Índice de figuras

1. Introducción

1.1. Contexto y motivación

Este Trabajo de Fin de Grado se ha realizado dentro del Grupo de Tecnologías Fotónicas (GTF) de la Universidad de Zaragoza, enmarcado dentro del proyecto con nombre "Sensado Acústico Distribuido de Largo Alcance por Retrodifusión Rayleigh Coherente (SACOH)", en colaboración con la empresa Aragón Photonics Labs (APL), y con especial interés tanto para el Grupo de Red Eléctrica de España (REE) como para CINTRA, operador a nivel internacional de infraestructuras de transporte.

Tradicionalmente, la vigilancia y regulación de infraestructuras públicas de gran tamaño (líneas ferroviarias, vías de automóviles…) se ha realizado mediante una serie de tecnologías que suponen un coste elevado -tanto en el aspecto económico como en el de gestión-, ya sea mediante el uso de cámaras de vídeo vigilancia como de sensores y/o detectores de movimiento.

Otro de los inconvenientes que presentan dichas tecnologías es su localización puntual, que permite a algunos de los usuarios de las mencionadas infraestructuras alterar su comportamiento y/o entrar dentro de la legalidad únicamente en los puntos concretos donde se encuentra instalado el dispositivo de control, puesto que fuera de los mismos no hay ningún sistema que permita la monitorización.

Es aquí donde la fibra óptica puede desempeñar un papel importante. De manera general, es conocida por su capacidad como medio de transmisión de datos a alta velocidad en sistemas de telecomunicación, y es una de las tecnologías más implantadas en el mundo actual con este fin. Sin embargo, existe también un cada vez más amplio número de aplicaciones diferentes, siendo una de las principales utilizar la fibra como elemento sensor distribuido. Con ella, es posible registrar distintas magnitudes físicas dentro de un amplio abanico: temperatura, tensión, sonido, vibraciones…

Las ventajas que ofrece su utilización en sustitución de otros elementos de control y regulación son elevadas: compatibilidad y aprovechamiento de sistemas de comunicaciones ópticas ya instalados –para mismos o diferentes usos-, reducido tamaño, peso y coste, alta sensibilidad, inmunidad a interferencias electromagnéticas, flexibilidad geométrica, resistencia ambiental a elevadas temperaturas, etc.

Su versatilidad hace que el número de aplicaciones de la fibra como elemento de sensado y monitorización en diferentes ámbitos e industrias esté actualmente en

continuo aumento: medicina, arqueología, aplicaciones militares, ingeniería civil, ingeniería aeronáutica, ingeniería nuclear… Dentro de las tecnologías de sensado distribuidas basadas en fibra óptica, el COTDR (*Coherent Optical Time Domain Reflectometer*) tiene un interés especial para la aplicación descrita en este proyecto, tal y como se explicará en capítulos posteriores.

1.2. Objetivo y alcance

El principal objetivo del presente Trabajo de Fin de Grado es el análisis, evaluación y procesado de los datos recogidos por el sistema de sensado acústico distribuido para su aplicación al control y monitorización de carreteras. Para ello, se estudiarán las características tanto temporales como espectrales de su señal con el fin de mejorar la información extraída directamente del sensor, que posteriormente será tratada y evaluada.

1.3. Metodología

La realización del proyecto se ha organizado y dividido en cuatro etapas diferentes. En primer lugar, se comienza con un estudio teórico del sistema de sensado acústico distribuido basado en fibra óptica, el cual se basa en la técnica COTDR. Tras finalizar lo anterior, y ya comprendiendo el funcionamiento del sistema, se pasa a conocer los antecedentes en el trabajo, puesto que es un proyecto ya empezado previamente y con ciertos avances que es preciso considerar como punto de partida para la continuación del mismo. Esto incluye una visita a un emplazamiento donde está instalado un sistema de sensado similar.

Una vez llegado este punto, se realiza y evalúa la mejora en el sistema tras la aplicación de un método de filtrado en el dominio frecuencial mediante un filtro paso banda.

A continuación, y con el objetivo de demostrar la potencia y capacidad del sistema, se procede a buscar de manera manual diferentes eventos que pueden darse en carreteras. Resultan de mayor interés aquellos que no se dan con normalidad y frecuencia, entre los cuales se encuentran frenazos bruscos, accidentes, paradas inesperadas… Para ello, se dispone de datos offline que han sido tomados a lo largo del año.

Por último, se finaliza con el desarrollo de una aplicación capaz de calcular velocidades medias y número de coches en los diferentes tramos de la carretera. Esto permite tener medidas estadísticas y comprobar que éstas avanzan de acuerdo a lo habitual en una autovía, de tal forma que sea posible actuar del modo que proceda

sobre situaciones de alarma, manteniendo así el orden y la normalidad en las vías que implanten el sistema de sensado.

1.4. Cronograma

El desarrollo del trabajo se ha producido entre los meses de febrero y junio, siendo el diagrama de Gantt correspondiente al mismo el que se muestra en la Figura 1.1.

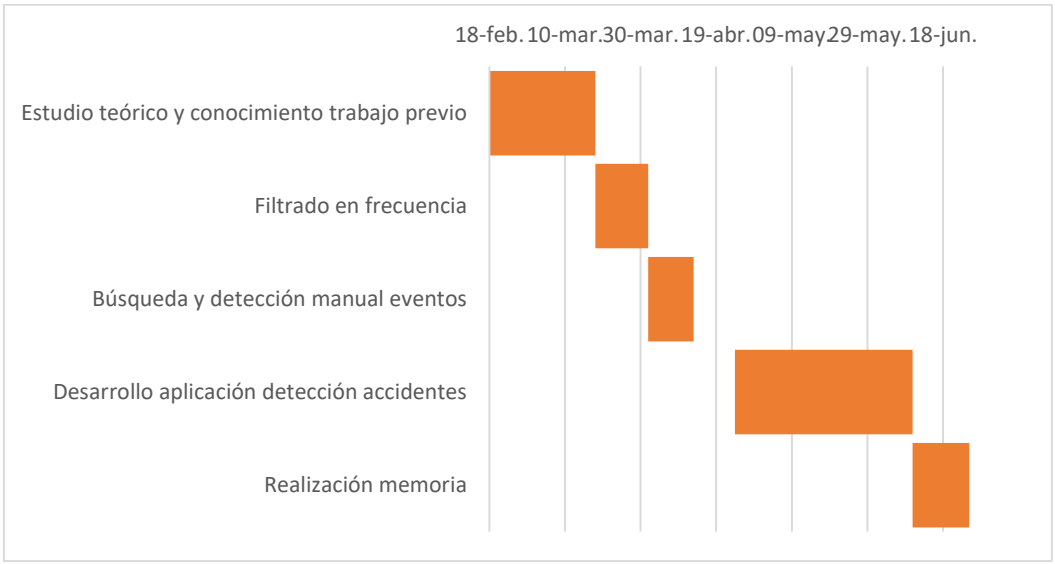


Figura 1.1. Diagrama de Gantt

1.5. Contenido de la memoria

Además de este capítulo introductorio, la memoria se organiza en otros cinco capítulos y tres anexos, los cuales se resumen de manera breve a continuación.

Capítulo 2 – Fundamentos teóricos. Se incluye una explicación teórica del funcionamiento del sistema de sensado acústico distribuido así como de sus principales componentes.

Capítulo 3 – Filtrado en frecuencia. Se examina el espectro en frecuencia de los datos para aprovechar sus propiedades y tratar de mejorar el aspecto de las representaciones para un futuro procesado.

Capítulo 4 – Búsqueda y detección manual de eventos. A partir de datos recogidos por el sensor durante largos intervalos de tiempo, se buscan y estudian apariciones de eventos –tanto habituales como imprevistos- en las carreteras.

Capítulo 5 – Algoritmos para automatización de detección de eventos. Siguiendo los eventos encontrados en el capítulo anterior, se intentan desarrollar una serie de algoritmos que permitan automatizar el proceso realizado antes de manera manual para la detección de algunos de los eventos que podrían calificarse como anormales en el tráfico.

Capítulo 6 – Conclusiones y líneas futuras. Se recogen los resultados de todos los capítulos anteriores, dejando abiertas una serie de líneas que será posible tratar en un futuro para mejorar el sistema, aprovechando lo realizado en el presente Trabajo de Fin de Grado.

Anexo A. Transformada de Fourier Localizada (STFT).

Anexo B – Transformada de Hough en Matlab.

Anexo C – Código Matlab.

2. Fundamentos teóricos

2.1. Sistemas basados en fibra

En los sistemas que utilizan la fibra óptica en cualquiera de sus aplicaciones ya mencionadas en el capítulo introductorio, hay varios fenómenos característicos que pueden afectar al funcionamiento general de los mismos, y que es pertinente presentar a continuación:

- Atenuación: es la reducción o pérdida de potencia óptica con la propagación de la luz a través de la fibra óptica. Cuanto más larga es la fibra y mayor es la propagación de la luz a través de ella, más se atenúa la señal óptica. Hay varias causas de atenuación: absorción, difusión (*scattering*) y radiación.

- Dispersión:

    - Cromática: es el ensanchamiento del pulso debido al hecho de que diferentes longitudes de onda de la luz se propagan a velocidades ligeramente diferentes a través de la fibra debido a que el índice de refracción de las fibras de vidrio tiene dependencia con la longitud de onda [1].

    - De polarización: las ondas electromagnéticas están caracterizadas por dos polarizaciones (a lo largo del eje X y a lo largo del eje Y); y debido al cambio del índice de refracción por variaciones en la temperatura o presión de la fibra, las velocidades de propagación en los diferentes ejes del núcleo de la fibra son distintas [2].

- *Scattering* no lineal:

    - Brillouin: en un proceso de emisión espontánea, un fotón incidente se transforma en un *scattered photon* (fotón dispersado) y en un fonón, dando lugar a otra onda con una frecuencia menor a la que la originó. Esta onda en fibras ópticas puede propagarse a lo largo de decenas de kilómetros sin atenuarse, lo que hace que sea un fenómeno indeseado [3].

    - Raman: cuando un rayo de luz monocromática se propaga a lo largo de la fibra óptica, aparece el *scattering* Raman, transfiriendo varios fotones a nuevas frecuencias –los cuales pueden ganar o perder energía-. Su origen es la respuesta no lineal de tercer orden de la polarización del material [4].

En concreto, la propiedad de la fibra óptica en la que está basado el sensor es uno de los factores de atenuación: la difusión o *scattering* Rayleigh. Este fenómeno consiste en

que la energía luminosa guiada se convierte en energía luminosa radiada debido a su propagación por un medio diferente del vacío [5]. Este tipo de *scattering* es lineal (no modifica la longitud de onda de la luz difundida), y supone pérdidas de la luz que va avanzando a lo largo de la fibra. Estas pérdidas se deben a variaciones de la densidad del material, ocasionadas principalmente por defectos en la fabricación de la fibra, heterogeneidades, acumulaciones de materia…

Otro aspecto muy importante en estos sistemas es el tipo de emisor utilizado para la transmisión. Principalmente se pueden clasificar en dos: diodo emisor de luz (LED) y láser. Este último presenta una serie de características que lo hacen más apto para el desarrollo del proyecto:

- Direccionalidad: la emisión de la luz en un láser es guiada preferencialmente en una única dirección.

- Monocromaticidad: la luz monocromática es aquella que solamente posee componentes de un color, es decir, una única longitud de onda. En la realidad no existen emisores monocromáticos puros, pero sí que hay unos que tienen una anchura espectral menor que otros, lo cual es interesante desde el punto de vista de la capacidad de transmisión de información. Es el caso de los láseres, ya que los fotones emitidos poseen longitudes de onda muy cercanas entre sí.

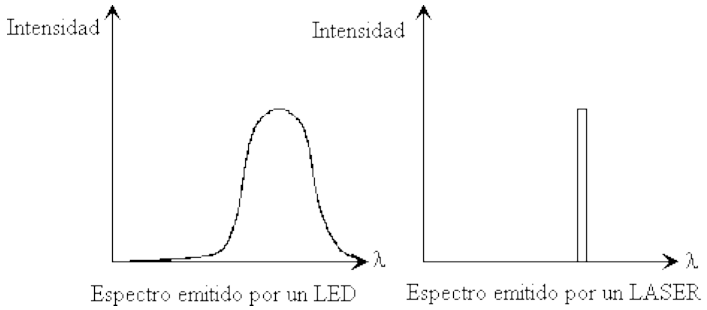


Figura 2.1.1. Aproximación del espectro de los distintos emisores.

- Coherencia. Surge del proceso de emisión estimulada que proporciona la amplificación láser. Como el proceso de emisión surge de un estímulo común, los fotones emitidos tienen una relación definida de fase los unos con los otros. De este modo se habla de dos tipos de coherencia: temporal, relacionada con la monocromaticidad; y espacial, relacionada con la direccionalidad. Así, la luz se considera coherente cuando posee ambos tipos de coherencia a la vez.

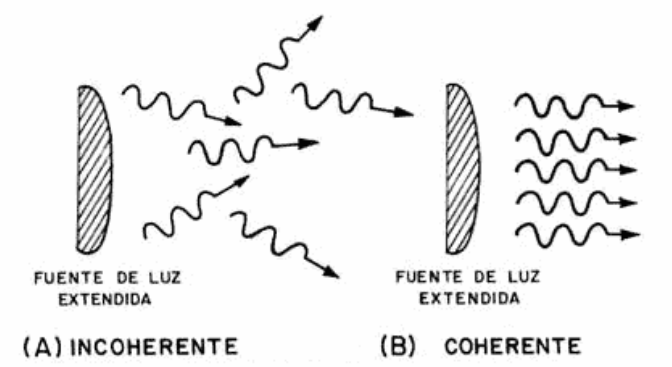


Figura 2.1.2. Diferentes tipos de luz según su coherencia.

2.2. Reflectometría óptica coherente en el dominio del tiempo (COTDR)

El sensor acústico distribuido en el que se basa el proyecto se fundamenta en la utilización de la técnica COTDR, la cual se desarrolla a partir del OTDR (*Optical Time Domain Reflectometer*), pero con la diferencia de que la luz acoplada en la fibra es altamente coherente, y el tiempo de coherencia mucho mayor que el ancho del pulso, de tal manera que el pulso pueda interferir consigo mismo. De esta manera, se consigue un funcionamiento distinto que permite desarrollar la aplicación en la que se fundamenta el proyecto.

En un OTDR se introduce la señal por uno de los extremos de la fibra y en un receptor situado en ese mismo extremo se recibe la luz reflejada por *back-scattering* Rayleigh. Este *scattering* se produce en todas las direcciones, y es proporcional a la potencia existente en el punto en el que se produce la difusión; sin embargo, la que realmente se puede capturar mediante un detector en el extremo inicial de la fibra es la que vuelve hacia atrás retrodifundida.

La potencia recibida P_{rx} debido a la luz reflejada por *back-scattering* desde un punto z de la fibra atiende a la siguiente expresión:

$$P_{rx} = \frac{1}{2} \cdot P_{tx} \cdot v_g \cdot \tau \cdot B \cdot K \cdot e^{-2\alpha z} \qquad (2.2.1)$$

donde P_{tx} es la potencia transmitida, v_g es la velocidad de grupo, τ es la anchura del pulso, B es la fracción de potencia óptica capturada, K es el coeficiente de *scattering* Rayleigh, y α es la atenuación por unidad de longitud de la fibra. Así se obtienen gráficas

en función de z como la que se muestra a continuación, que permiten analizar el comportamiento de la atenuación a lo largo de toda la fibra:

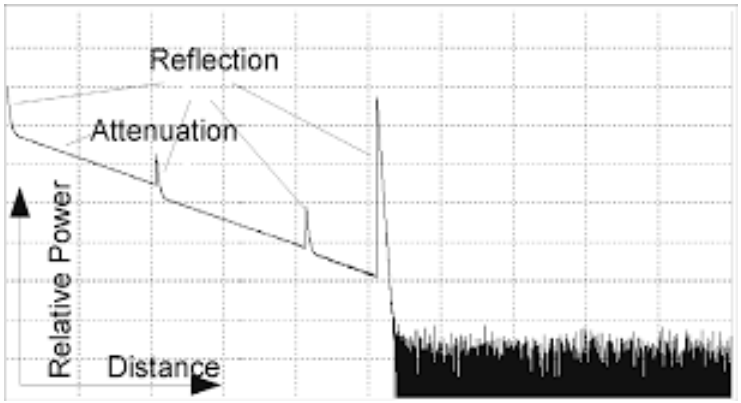


Figura 2.2.1. Traza OTDR.

Por otro lado, midiendo el retardo entre el momento de transmisión y el momento de recepción de la señal es posible determinar en qué punto z de la fibra se localiza cada uno de los posibles eventos:

$$z = \frac{v_g \cdot \Delta t}{2} \tag{2.2.2}$$

Otro parámetro importante del OTDR es la resolución espacial *Δz*, es decir, cuál es la mínima separación en distancias que es posible resolver. Viene dado por la Ecuación 2.2.3, existiendo de esta manera un compromiso a la hora de determinar la anchura de pulso entre distancia explorable (nivel de potencia) y resolución espacial.

$$\Delta z = \frac{1}{2} v_g \tau \tag{2.2.3}$$

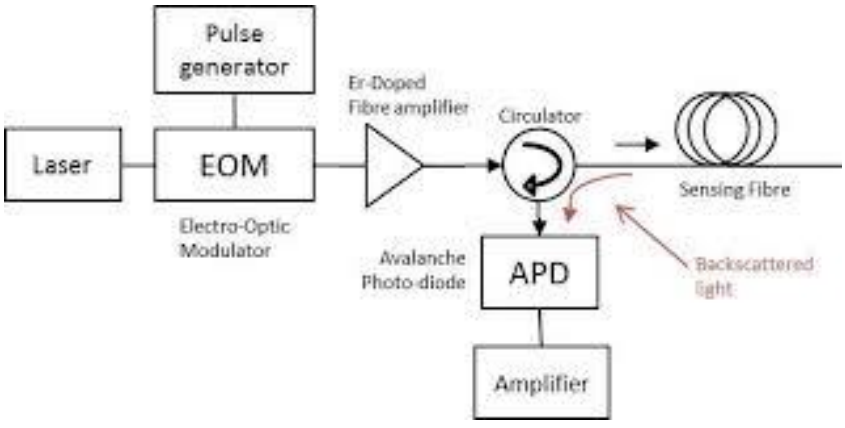


Figura 2.2.2. Esquema de un sistema COTDR.

Un sistema COTDR tiene algunas diferencias en su implementación física con respecto a un sistema convencional, como puede verse en la Figura 2.2.2. En primer lugar, es necesario utilizar un láser coherente con alta monocromaticidad, dado que su longitud de coherencia es inversamente proporcional a su anchura espectral. También se necesita que el láser emita un elevado nivel de potencia (unos 20 dBm -0.1 W-) para alcanzar distancias de decenas de kilómetros.

Posteriormente, la luz del láser se inyecta a un modulador que presente una alta relación de extinción, es decir, que el nivel de señal a la salida sea el mínimo posible cuando no se están transmitiendo pulsos, para evitar ruido de fondo.

A continuación, la señal pasa a uno o varios EDFAs (*Erbium Doped Fiber Amplifier*) en serie para conseguir un alto nivel de potencia. Un EDFA basa su funcionamiento en la emisión estimulada de fotones, que ocurre cuando un fotón decae de un alto nivel de energía a otro de menor debido al impacto de un fotón incidente de la señal de entrada. De esta manera, el fotón emitido tiene misma longitud de onda, fase y polarización que el incidente [6]. Con todo ello, se aumenta el rango de exploración del sistema, pues la potencia se va perdiendo a lo largo de la fibra debido a la atenuación presente, y a distancias alejadas del transmisor la señal es inapreciable como se podrá apreciar en capítulos próximos.

Sin embargo, un EDFA, además de amplificar la señal, introduce ruido ASE (*Amplified Spontaneous Emission*), el cual tiene una densidad espectral de potencia directamente proporcional a la energía del fotón considerado (E = hf), la ganancia del amplificador y la figura de ruido de éste. Además, es nocivo puesto que ocupa por completo el espectro frecuencial donde la amplificación de señal es posible [7]. Por lo tanto, es muy perjudicial y afecta de manera notable y negativa a las prestaciones del dispositivo. Así, aunque en el esquema anterior no aparece por tratarse de un diseño simplificado, es recomendable utilizar un filtro paso banda que permita eliminar las componentes de dicho ruido fuera de banda, ajustando los parámetros del filtro para no alterar el comportamiento de la señal.

A la salida del filtro se coloca un circulador óptico, un elemento que permite aislar a los extremos receptores y transmisores. Así, se consigue por un lado enviar la señal a la fibra; y por el otro, detectar lo recibido por *back-scattering* para después enviarlo al detector.

La principal diferencia del COTDR con respecto al OTDR radica en que la traza recibida en este último depende de la amplitud de la señal óptica en cada punto de la fibra, por lo que un evento que ocurre al principio del tramo influye en los que puedan ocurrir a continuación. Sin embargo, en COTDR, para la detección, la señal de la que se dispone

es una combinación de interferencias coherentes que son reflejadas por *back-scattering* Rayleigh, a lo largo de la fibra, durante la duración completa del pulso. Dicho de otra forma, el valor de la intensidad medida en un punto determinado de la fibra depende de las fases relativas de los campos reflejados por *scattering*. De esta manera, los diferentes eventos que suceden a lo largo de la fibra son independientes, pudiendo llegar a detectar todos ellos siempre y cuando la atenuación por propagación no sea demasiado elevada.

Estas interferencias son aleatorias e impredecibles, pero idealmente, en un punto *z* concreto, deberían ser constantes a lo largo del tiempo. Sin embargo, cuando se produce un incidente o perturbación, se producen variaciones abruptas debidas a cambios de fase; aunque hay que tener en cuenta que no todas pertenecen a eventos, ya que, debido a la elevada sensibilidad de la fibra, también se registran variaciones notables debidas a cambios de presión, temperatura…

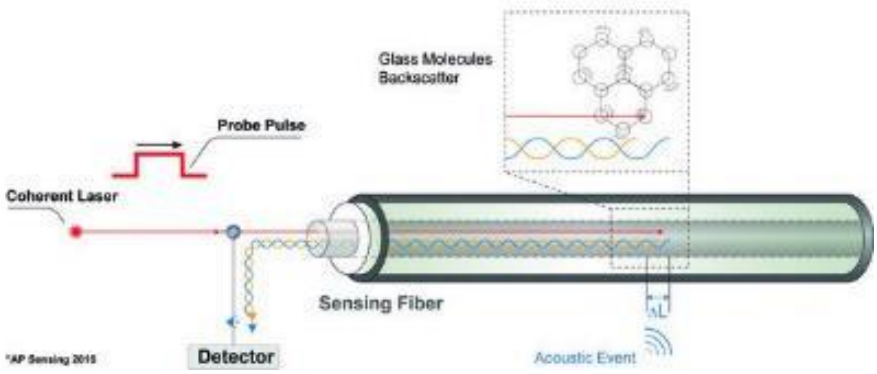


Figura 2.2.3. Cambio de fase por evento acústico.

Asumiendo [8] que los elementos de *scattering* Rayleigh (visibles en la Figura 2.2.3.) están distribuidos aleatoriamente en diferenciales de longitud ΔL y que la polarización de la señal reflejada es la misma para todos los pulsos, se puede expresar la señal en detección de acuerdo a la Ecuación 2.2.4, donde r_k y Φ_k son respectivamente la amplitud y la fase de la señal compleja suma de los M pulsos recibidos por *scattering*; mientras que α_i y Ω_i son la amplitud y la fase del dispersor i-ésimo. Estas cuatro últimas variables se suponen aleatorias, como ya se ha enunciado anteriormente.

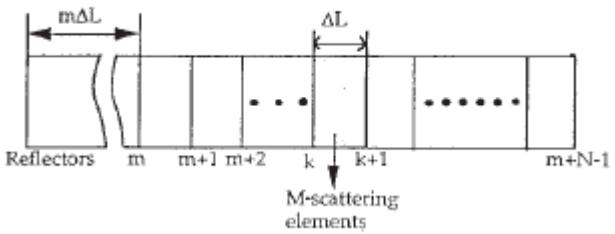


Figura 2.2.4. Discretización del modelo de Rayleigh.

$$r_k \exp(j\Phi_k) = \sum_{i=0}^{M} \alpha_i \exp(j\Phi_i) \qquad (2.2.4)$$

En cuanto al campo interferente en recepción, puede expresarse así:

$$E_b(m\Delta L) = E_0 \sum_{k=m}^{m+N-1} P_k r_k \exp j(\theta_k + \Phi_k) e^{-\alpha k \Delta L} \qquad (2.2.5)$$

donde r_k, Φ_k y P_k son la reflectividad, fase y polarización aleatorias del k-ésimo elemento reflector; Θ_k representa el cambio de fase debido a la perturbación; α supone el coeficiente de atenuación de potencia de la fibra; ΔL es la longitud del diferencial de fibra examinado; y por último, E_0 representa el valor de amplitud del campo incidente.

A estas señales disponibles en detección se les denominará de ahora en adelante *raws.*

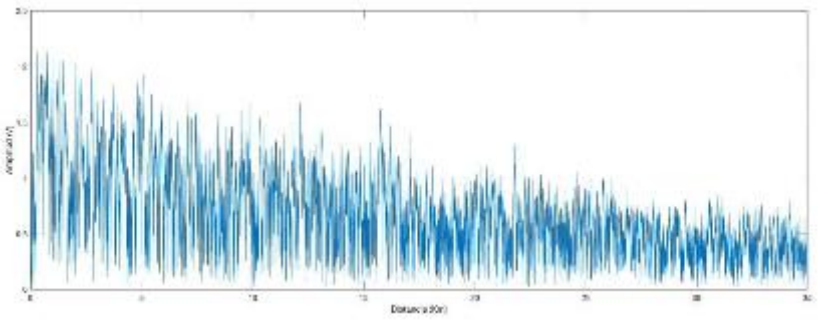


Figura 2.2.5. Traza óptica de un COTDR o raw [9].

A partir de la representación de un *raw* (Figura 2.2.5.) se corrobora la dificultad de extraer información en una única traza debido a la aleatoriedad de las interferencias que se producen en los distintos puntos de la fibra, así como la pérdida de potencia con la propagación de la luz a lo largo de la misma.

2.3. Generación de *waterfalls*

Una vez se dispone en el receptor de los datos brutos, es posible realizar una representación tiempo-distancia de los diferentes *raws*, construyendo así lo que se denomina un *waterfall* (cascada). Este nombre se deriva de la forma en la que se van construyendo las imágenes cuando el proceso se realiza en tiempo real.

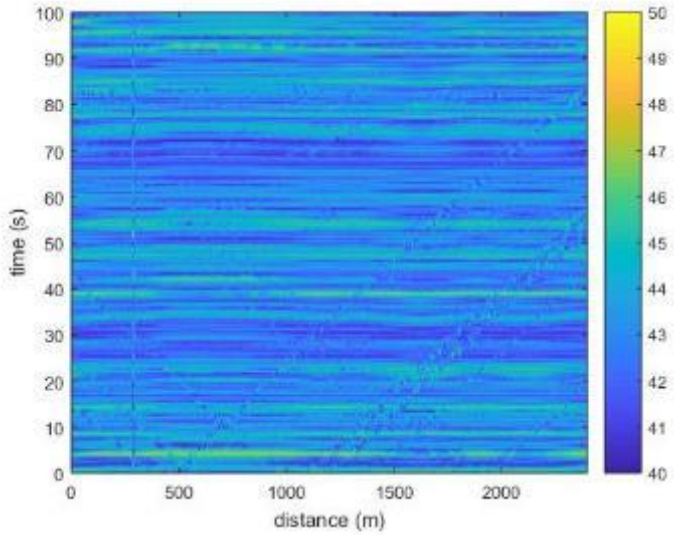


Figura 2.3.1. Porción de waterfall original.

La intensidad y el color de la superficie representan la intensidad del estímulo detectado en cada punto e instante temporal, lo cual está directamente relacionado con la presencia de vehículos. En la Figura 2.3.1. pueden apreciarse de manera diferenciada tres vehículos avanzando en la misma dirección de propagación de la luz en la fibra y uno en sentido contrario, es decir, desde el final del tramo de fibra hacia el principio. Aquí se ve la importancia de implementar un sistema basado en COTDR en vez de OTDR, el cual permite la detección de múltiples eventos simultáneos a lo largo del tramo de fibra instalado.

Las líneas diagonales que pueden observarse en el *waterfall* anterior representan vehículos circulando por la carretera donde está instalado el sistema. La mayoría de los vehículos que es posible capturar son camiones debido a que las interferencias producidas por los mismos tienen mayor amplitud que las de los automóviles o motocicletas. En el eje de las abscisas se representa la distancia, mientras que en el de ordenadas se coloca la referencia temporal. De esta manera, cuanto más vertical sea una línea menor velocidad lleva el vehículo detectado; y viceversa.

También se ve una línea vertical en torno a los 300 m del tramo representado. Esto, tal y como se explicará en capítulos posteriores, es un punto muerto, una zona sin sensibilidad debida a algún fenómeno físico en ese punto concreto que provoca altos niveles de amplitud en todo momento.

3. Filtrado en frecuencia

Como se ha visto en el Capítulo 2.3., es posible la representación de *waterfalls* a partir de los *raws*. Sin embargo, es conveniente aplicar técnicas de procesado de señal para limpiar la imagen obtenida y facilitar la extracción de información y características relevantes que es posible obtener con el sistema de sensado acústico distribuido.

En este aspecto, se partía de un procesado realizado por la empresa *Aragón Photonics Labs* que aplica una serie de técnicas para finalmente acabar comparando con un umbral. Este procesado, aunque puede tener aplicación al control de tráfico, está inicialmente pensado para otro objetivo diferente relacionado con la detección de intrusismos.

De esta manera es posible representar el *waterfall* umbralizado, el cual ofrece la misma información, pero gracias a las técnicas aplicadas el resultado es más ventajoso y permite que posteriormente el tratamiento de los datos sea más sencillo.

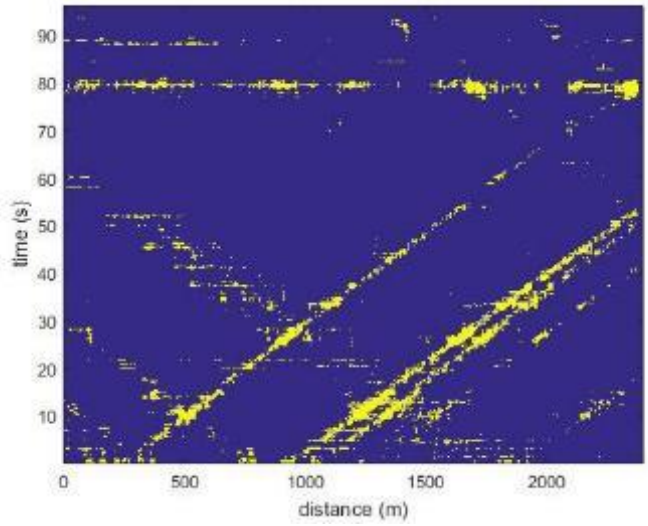


Figura 3.1. Porción de waterfall umbralizado tras el procesado realizado por APL.

A partir de la representación de un *waterfall* es posible la observación y el seguimiento de un modo visual de los diferentes vehículos que atraviesan la vía donde está instalada el sistema de sensado acústico. Es pertinente aclarar que las interferencias provocadas por camiones tienen una amplitud mucho mayor en comparación con las originadas por vehículos más simples como coches o motocicletas, por lo que es más fácil su localización y tratamiento.

Todo lo mostrado hasta ahora se limita al dominio temporal. Sin embargo, transformando los datos al dominio frecuencial es posible extraer información relevante que permite mejorar los *waterfalls* para el futuro tratamiento de los mismos. Además, las técnicas de procesado para la representación del *waterfall* presentadas en el Capítulo 2.3. suponen un elevado coste computacional y hacen que, a pesar de que los resultados mejoren considerablemente, el algoritmo no sea tan óptimo. Esto es debido principalmente a que dicho procesado está pensado para una aplicación diferente consistente en detección de intrusos por motivos de seguridad, pero que también implementa este sistema de carreteras. Sin embargo, efectuando el cambio al dominio frecuencial para el tratamiento de los datos se consigue que la realización de los cálculos sea más rápida y eficaz. Podría pensarse una combinación de ambas técnicas de procesado de señal que aproveche las ventajas que ofrece cada una de ellas.

Así, en este capítulo se analiza en Matlab el espectro que presentan las señales en detección para observar el rango de frecuencias en el que se encuentra la información verdaderamente importante, con el objetivo de filtrar dicho intervalo frecuencial y descartar el resto de componentes, consiguiendo así eliminar el ruido de fuera de banda y otras interferencias que no son interesantes en la aplicación considerada, para consecuentemente mejorar así las prestaciones.

El método que se va a aplicar para la obtención del espectro es el de la transformada de Fourier localizada (STFT: *Short-Time Fourier Transform*) (**Anexo A**). Para ello, se aplica la transformada de Fourier a la señal enventanada. En este caso concreto, en una medida de compromiso entre resolución frecuencial y relación lóbulo primario – lóbulo secundario (NLPS) [10], se escoge la ventana de Hamming con solape –desplazamiento- del 50%.

En la parte de transmisión, la frecuencia de repetición de pulsos es de 1 KHz. Por lo tanto, lo que se tiene es información de cada punto *z* de la fibra óptica muestreado a 1 KHz. Para una resolución frecuencial de aproximadamente 1 Hz, se escoge un número de muestras igual a 1024 para la realización de la transformada de Fourier localizada.

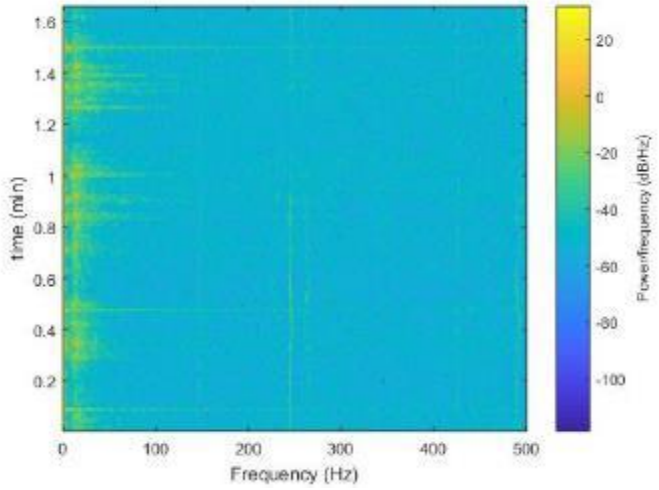


Figura 3.2. Espectrograma para d = 1500 m.

En la Figura 3.2. se ve un espectrograma a una distancia fija del láser, elegida a partir de la Figura 2.3.1. de tal forma que se diera la presencia en la carretera de uno o varios vehículos. Un espectrograma es una representación tiempo - frecuencia en la que se calcula la energía del contenido frecuencial de la señal a medida que ésta va avanzando a lo largo del tiempo. Puede observarse el aspecto paso bajo del espectro de la señal en detección, puesto que las frecuencias donde se concentra la mayor parte de la potencia de la señal están próximas a los 0 Hz.

Vistas las componentes que presentan las señales recibidas, a continuación se realiza un filtrado paso banda en frecuencia, con frecuencia de corte inferior igual a 20 Hz y frecuencia de corte superior 70 Hz. La elección de estas frecuencias se justifica por los siguientes motivos:

- Hasta 20 Hz, las componentes son ruidosas y dificultan la extracción de información del *waterfall*. Además, es en frecuencias en torno a la continua donde se encuentra la mayor parte del ruido.

- A partir de 70 Hz, todas las frecuencias quedan prácticamente eliminadas debido a la atenuación del suelo y la información no cambia por mucho que se aumente la frecuencia de corte superior.

- Si se limita todavía más la frecuencia de corte superior (por ejemplo, disminuyéndola a 50 Hz) se obtiene un *waterfall* en el que los vehículos que se mueven hacia distancias positivas están menos diferenciados. Esto se debe a que las altas frecuencias sufren mayor atenuación, y los coches que circulan por el lado opuesto de la vía no presentan energía a frecuencias más elevadas. De este modo, estableciendo la frecuencia de corte superior en 70 Hz, se consigue reforzar las líneas de los automóviles en mismo sentido a la luz.

Consiguientemente, se calcula la energía en cada una de las ventanas temporales en la banda de frecuencias argumentada previamente, y se compara con un umbral de energía consistente en el promedio de energías en un determinado punto *z* de la carretera.

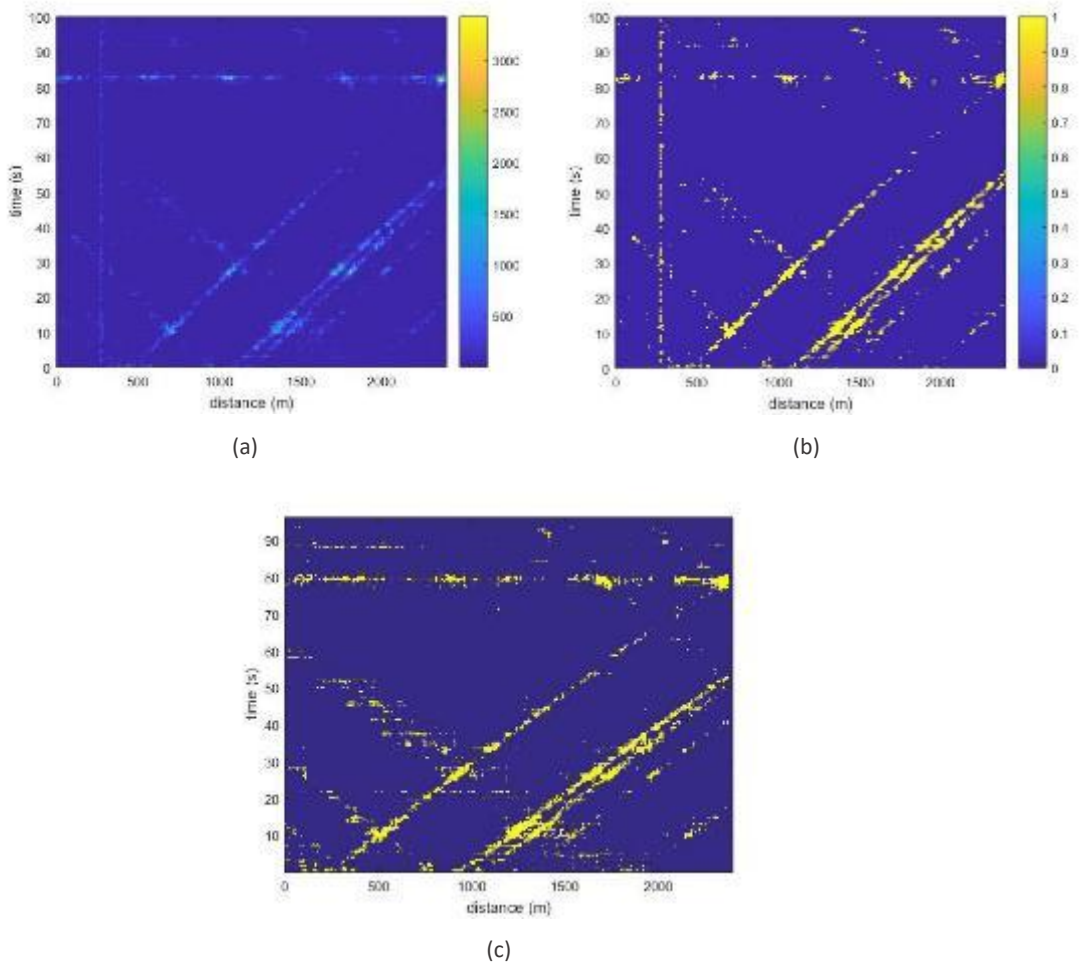


Figura 3.3. (a) Porción de waterfall tras filtrado paso banda de 20 Hz a 70 Hz. (b) Porción de waterfall umbralizado tras filtrado paso banda de 20 a 70 Hz. (c) Porción de waterfall umbralizado tras el procesado realizado por APL.

Así, queda demostrada la mejora presentada tras la aplicación de este filtro paso banda de 20 a 70 Hz, pues los vehículos que antes era posible distinguir con el cálculo de energía localizada quedan realzados; y además se visualizan otros que anteriormente apenas eran distinguibles. Aparece también una línea horizontal para t = 80 s aproximadamente, debida a un fallo puntual del sistema de sensado que supone un instante muerto en la detección.

4. Búsqueda y detección manual de eventos

Para el posterior desarrollo de una aplicación dotada con inteligencia artificial, resulta importante e interesante la realización de una búsqueda meticulosa de eventos en los datos recogidos por el sensor con el fin de conocer el alcance del sistema de sensado, así como de saber a qué tipo de situaciones va a haber que hacer frente. Además, es una tarea que requiere cierto conocimiento del sistema y del método de representación de los datos para caracterizar e interpretar los *waterfalls*.

Se trabaja con datos grabados durante el funcionamiento del sistema desde diciembre del año 2018 hasta marzo del 2019. Para ello, se hace uso de una aplicación desarrollada en Visual Studio que permite la visualización de *waterfalls*. Paralelamente, se muestra una imagen del mapa donde está ubicado físicamente el sistema de sensado acústico distribuido: en la autovía A28 de Portugal, cubriendo un total de 51200 m. El cable de fibra óptica está dispuesto en el arcén derecho de la misma, lo cual supone que los vehículos que circulan por el lado derecho presenten mayor facilidad para ser detectados al ser mayores las amplitudes de las interferencias registradas. Cabe resaltar que la búsqueda de eventos es un proceso relativamente lento, pues la apertura de *waterfalls* de largos intervalos temporales requiere mucha memoria RAM, por lo que hay que analizarlos poco a poco -con los ordenadores de los que se disponía para la realización del trabajo, en intervalos de cuatro horas-. Además, hay que ajustar el zoom para tener una buena resolución tanto temporal como espacial para la detección óptima de eventos.

Es pertinente remarcar que el aspecto de los *waterfalls* de ahora en adelante va a ser distinto de lo mostrado anteriormente. Esto se debe a que, durante la realización del trabajo, se realizaron cambios en el hardware del sistema de sensado que mejoraron la sensibilidad del sistema.

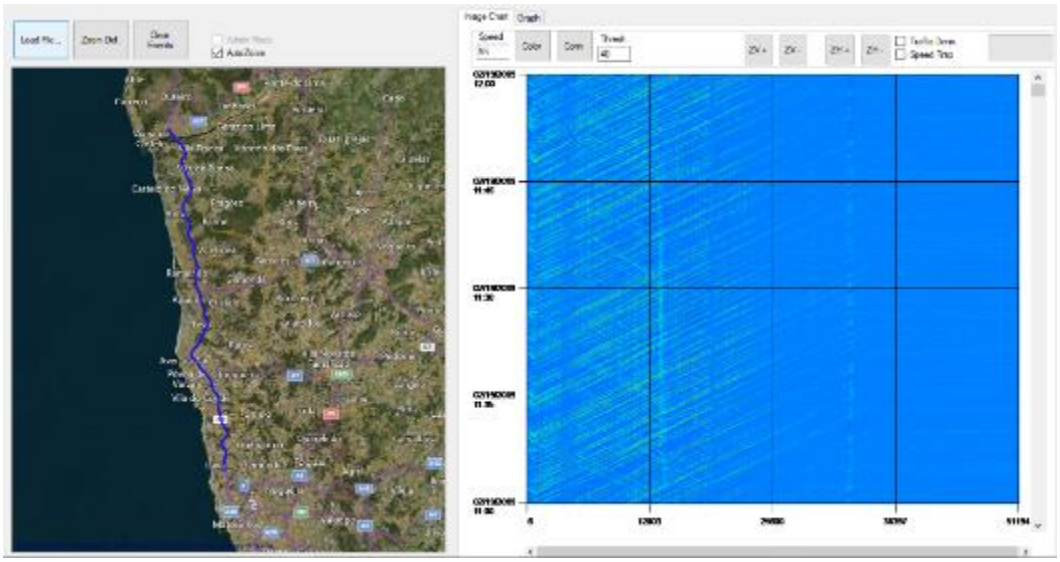


Figura 4.1. Aplicación para búsqueda y detección de eventos.

Figura 4.2. Emplazamiento donde está ubicado el DAS [11].

En una visión general de la representación se observa que, a partir de una determinada distancia, la potencia recibida por *back-scattering* en el detector es tan pequeña que el seguimiento de vehículos se convierte en una tarea muy complicada.

Hay zonas en las que la sensibilidad es nula o demasiado elevada debido a la presencia de puentes, ríos… y que aparecen como líneas verticales con alta intensidad en todo momento temporal, imposibilitando así la detección y el seguimiento de vehículos en esas zonas.

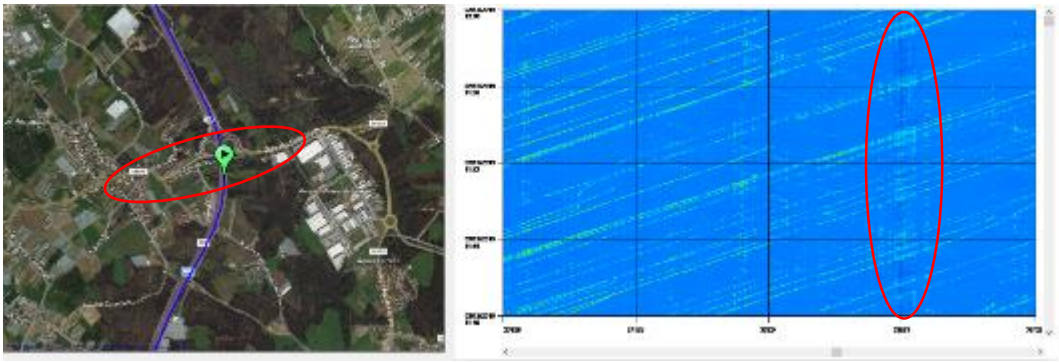


Figura 4.3. Pérdida de sensibilidad debida a la presencia de un vía pasando por debajo.

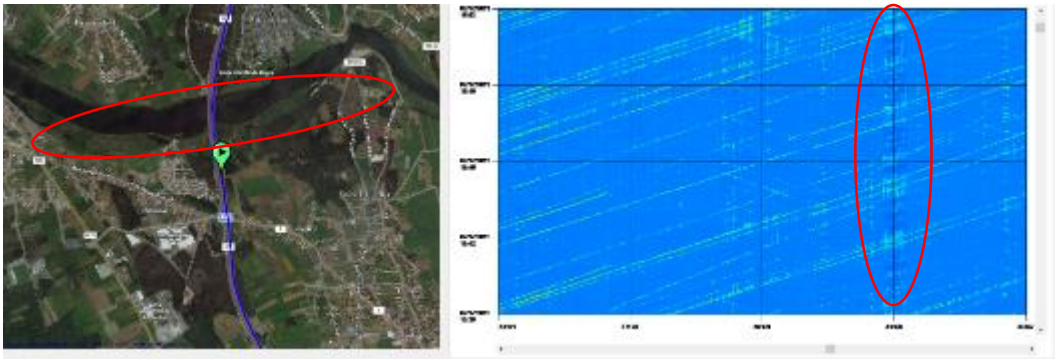


Figura 4.4. Pérdida de sensibilidad debida a la presencia del río Cávado.

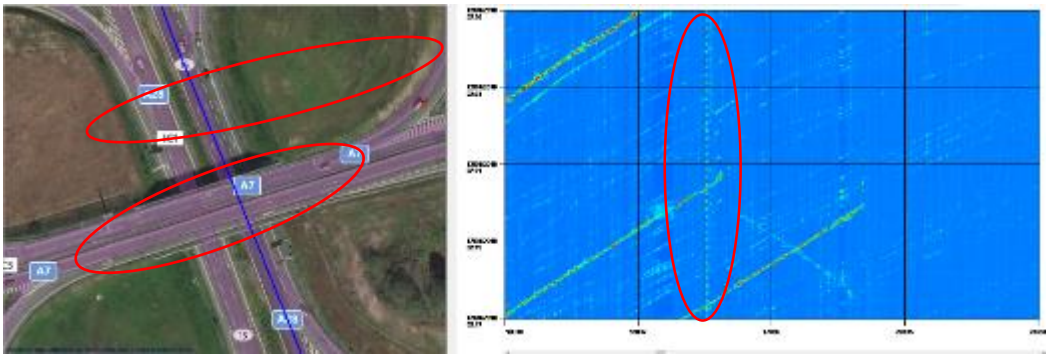


Figura 4.5. Pérdida de sensibilidad debida a la presencia de un puente.

4.1. Eventos habituales en carreteras

A modo introductorio para conocer la aplicación y el funcionamiento del sistema, es conveniente mostrar situaciones comunes en carreteras. Además, con una conversión de píxeles horizontales a kilómetros y píxeles verticales a horas es posible marcar dos puntos en el *waterfall* y mostrar la velocidad en un cuadro de texto.

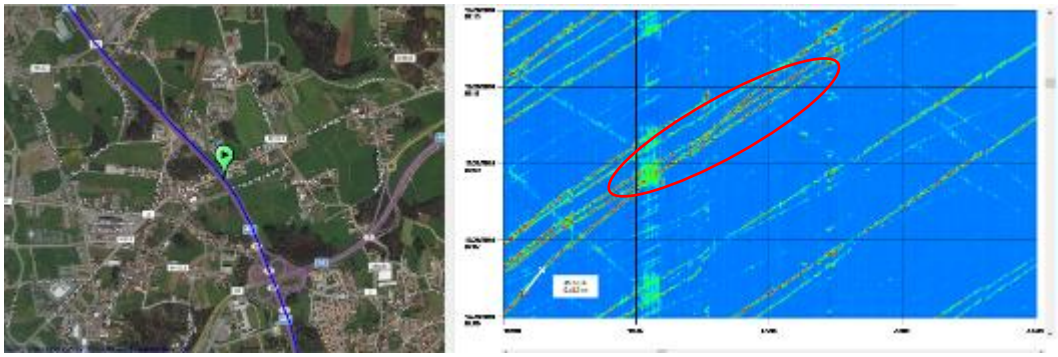


Figura 4.1.1 Adelantamiento múltiple.

En el caso de la Figura 4.1.1. puede observarse un adelantamiento que se produce a lo largo de la carretera. La caracterización de los adelantamientos es, pues, el cruce de líneas –que representan vehículos- en un punto concreto de distancia (eje x). De este modo, los vehículos con menor pendiente están adelantando al de mayor pendiente al estar representado el eje de distancias en las abscisas y el de tiempos en el de ordenadas.

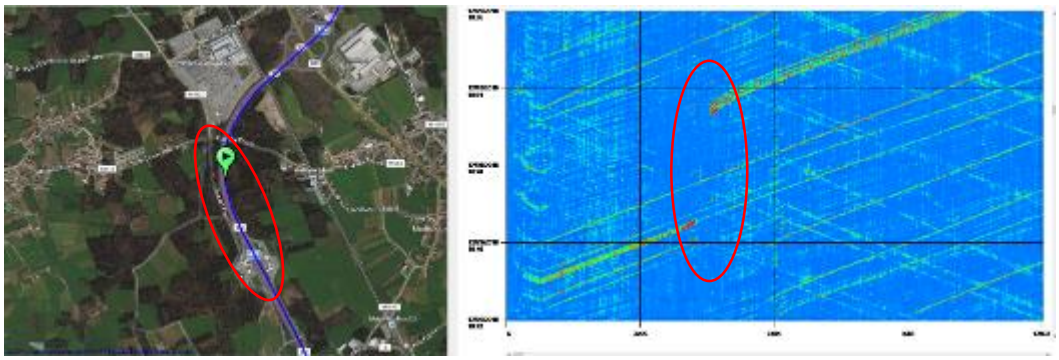


Figura 4.1.2. Salida y entrada múltiple en la autovía.

En la Figura 4.1.2. se aprecia en la parte inferior la salida de varios vehículos de la autovía, marcada por la desaparición de líneas; y posteriormente, la entrada de un elevado número de vehículos a la misma, es decir, líneas que brotan. Hay puntos a lo largo de la vía que se caracterizan por gran cantidad de entradas y salidas. En concreto, en el que se muestra en la imagen se producen abandonos e incorporaciones con gran frecuencia, pues es un puesto obligatorio de paso para convoys que vayan a atravesar la frontera con España. Además, en la parte izquierda de la imagen puede verse un cambio de pendiente de cada uno de los vehículos detectados, que se debe a que en los primeros metros la fibra va en sentido contrario antes de dar la vuelta.

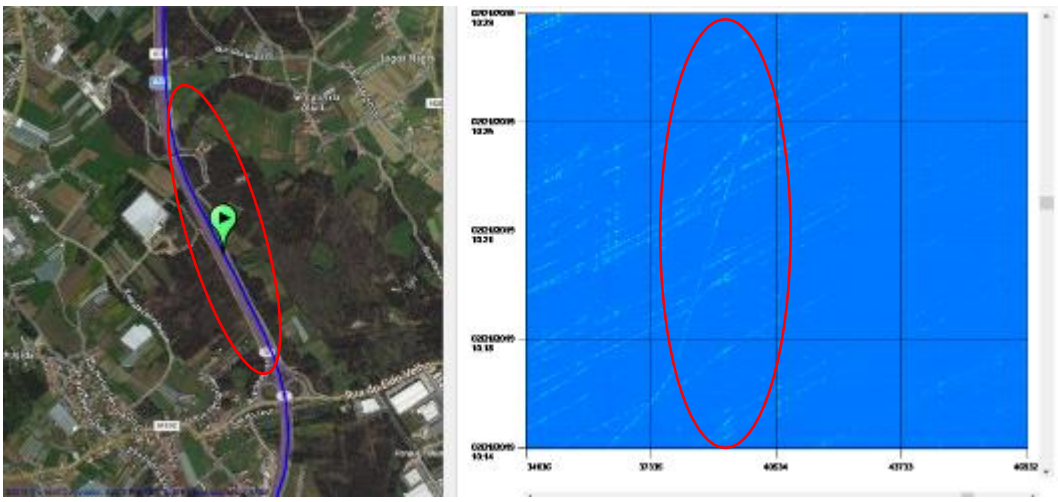


Figura 4.1.3 Cuesta localizada entre los kilómetros 37 y 40.

Se ve en la Figura 4.1.3. como algunos de los vehículos ven reducida su velocidad (líneas prácticamente verticales) desde el kilómetro 37 hasta el 40. Esto se debe a la presencia de una cuesta en dicha localización, que hace que sobre todo los camiones, pero también algunos coches, lleven velocidades menores.

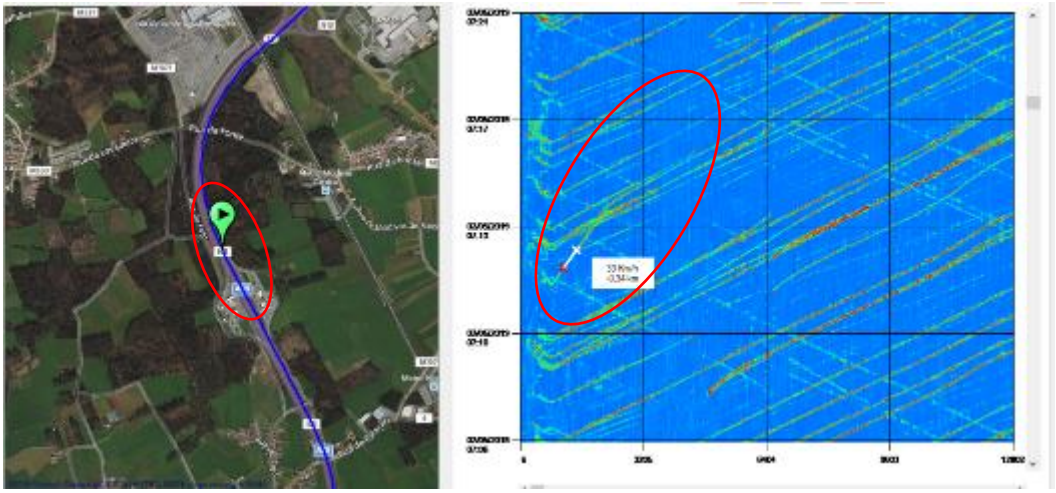


Figura 4.1.4. Entrada lenta a la autovía por presencia de otros vehículos en la misma.

En la situación que se da en la Figura 4.1.4., se localiza un vehículo que lleva una velocidad reducida en su incorporación a la A28, pero que en cuanto se encuentra dentro de la misma comienza a acelerar y adquiere un ritmo normal. La causa de esta entrada lenta se debe a la presencia de otros vehículos en la carretera que dificulta la incorporación, debiéndose hacer así ésta a poca velocidad.

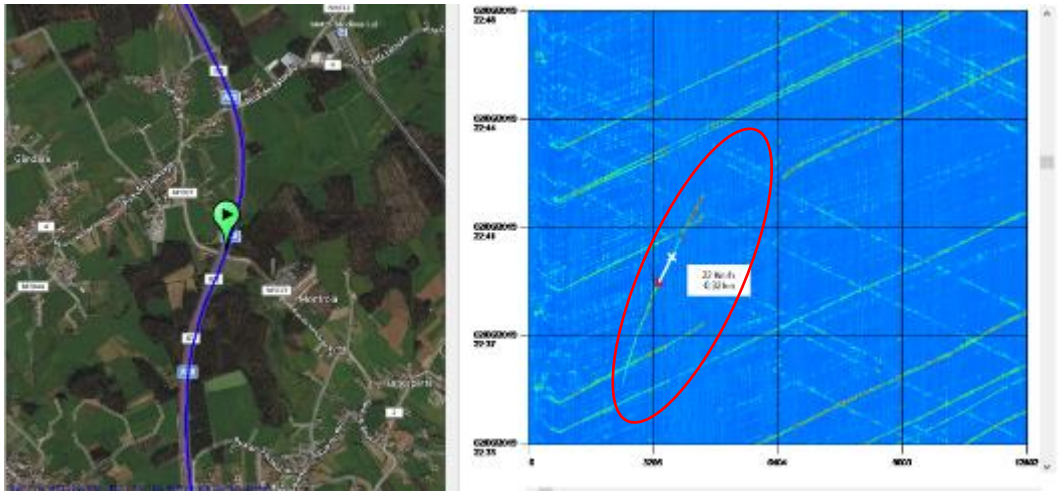


Figura 4.1.5. Vehículo lento.

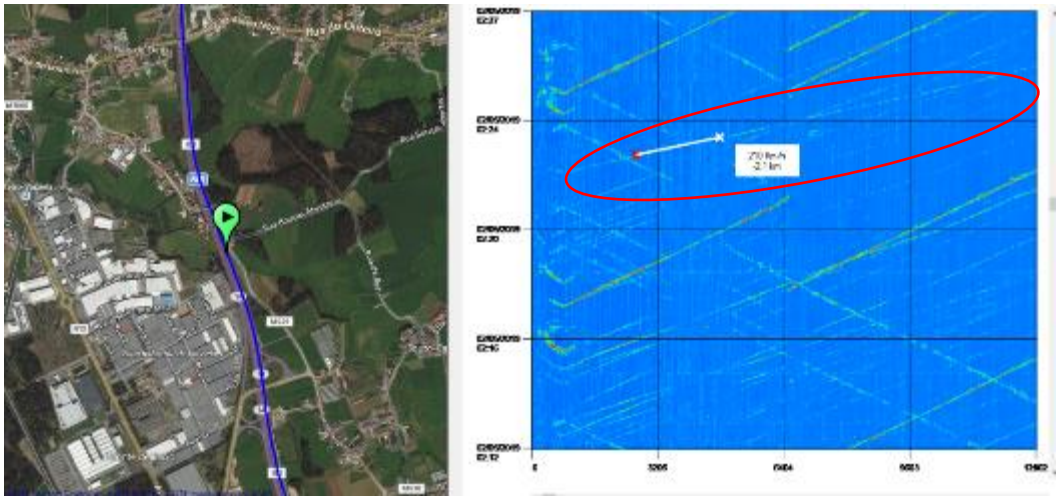


Figura 4.1.6. Vehículo rápido.

También es posible encontrar vehículos cuyas velocidades no se adaptan a las de la vía; ya sea por ser mucho mayores o mucho menores de lo esperado. En la Figura 4.1.5. se ve un vehículo muy lento, cuya representación es casi vertical; mientras que en la Figura 4.1.6. el coche lleva una velocidad muy elevada, con una línea prácticamente horizontal. Además, los cuadros de texto incorporados permiten añadir información que corrobora dichas situaciones.

4.2. Eventos puntuales en carreteras

No obstante, resulta todavía de mayor interés detectar una serie de eventos que no son corrientes y que pueden suponer situaciones en las que sea necesario una actuación externa urgente; o simplemente que sea importante monitorizar desde un puesto de control.

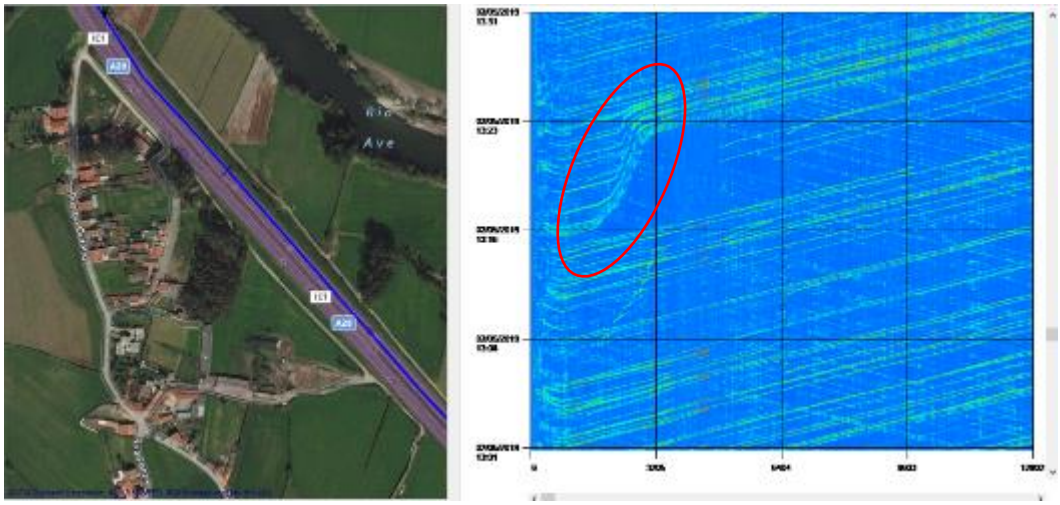


Figura 4.2.1. Frenazo general de todos los vehículos: breve incidente en la vía.

En la Figura 4.2.1. se muestra el ejemplo de varios vehículos circulando a velocidad normal, pero que tienen que frenar de repente debido a un incidente en la carretera. Posteriormente, tras unos minutos prácticamente parados, vuelven a reestablecer la marcha anterior. Es característico de estos eventos la ausencia de vehículos en los tramos posteriores al incidente –excepto cuando hay entradas próximas-.

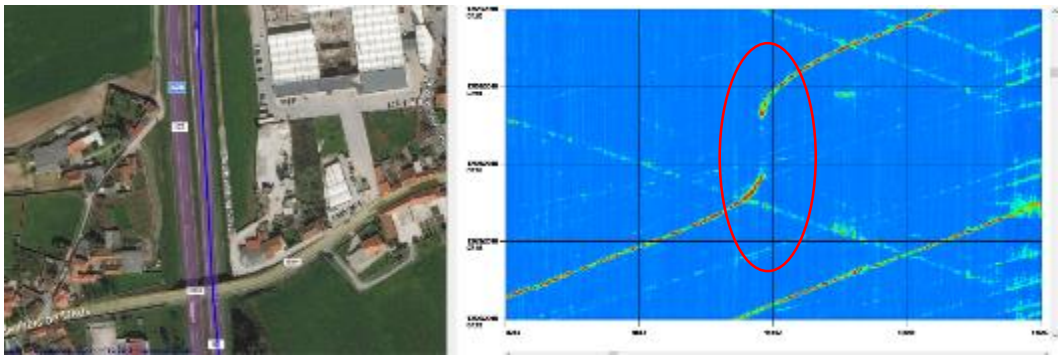


Figura 4.2.2. Frenazo brusco de un único vehículo en medio de la carretera hasta quedar parado durante unos minutos.

El caso de un vehículo que realiza una parada inesperada en medio de la autovía se ilustra en la Figura 4.2.2. Como puede apreciarse, el vehículo circula de manera habitual hasta que realiza un frenazo brusco: la línea que representa al vehículo pasa a tener una pendiente mucho mayor. La línea llega incluso a desaparecer -puesto que si el vehículo no se mueve, deja de generar interferencias que puedan llegar a ser detectadas- para posteriormente aparecer e ir perdiendo pendiente (ganando velocidad). Esta situación podría confundirse con una salida y una entrada de dos vehículos diferentes; pero en este caso no es así porque en el punto kilométrico que se ha producido no hay posibles incorporaciones y abandonos de la autovía.

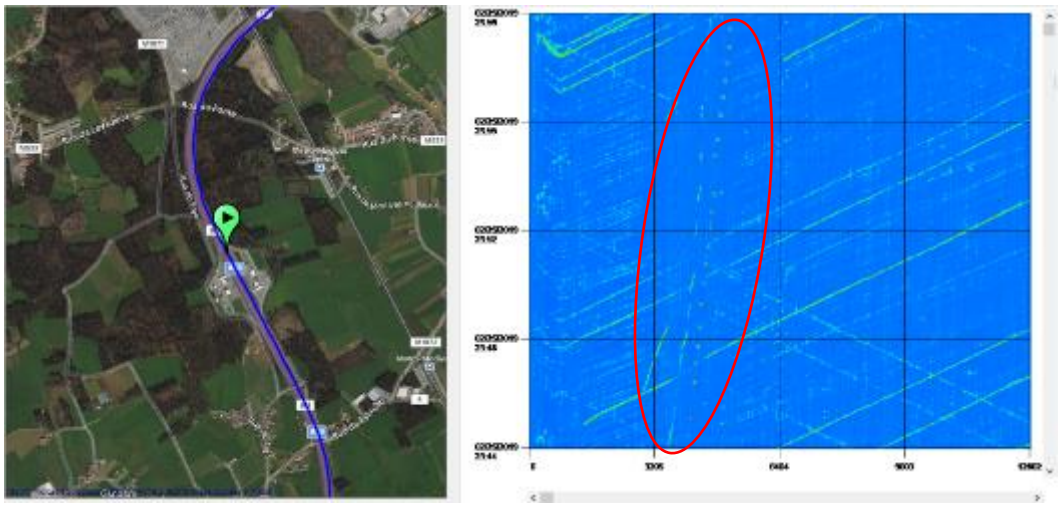
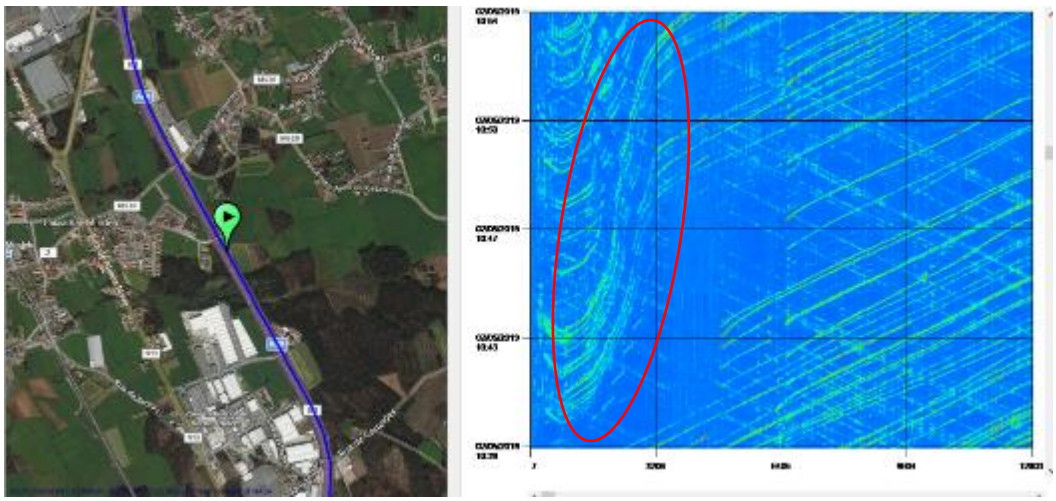
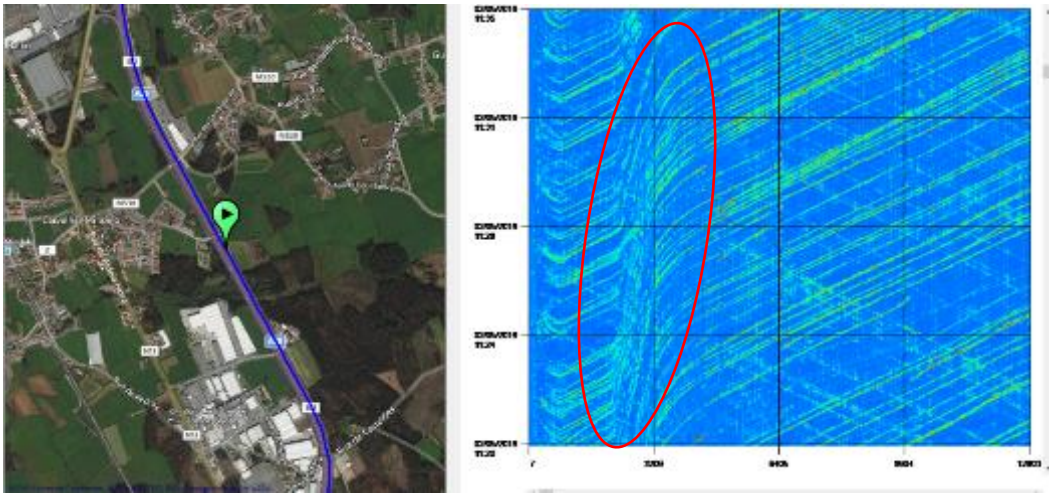


Figura 4.2.3. Tres vehículos circulando lento e interrumpidamente: vehículos de mantenimiento.

En la Figura 4.2.3. se observan tres vehículos avanzando a velocidades muy lentas y realizando una especie de *percusión* en el *waterfall*. Eventos como éste representan vehículos especiales realizando tareas de mantenimiento en la carretera, que circulan a velocidades reducidas y realizan parones a lo largo de la misma para llevar a cabo las acciones que sean necesarias.



(a)



(b)

Figura 4.2.4 (a), (b). Retención de más de una hora de duración: accidente.

El caso de la Figura 4.2.4. es similar al de la Figura 4.2.1., pues se ha producido un accidente en la vía que hace reducir drásticamente la velocidad a los vehículos que circulan por el tramo correspondiente; aunque las dos principales diferencias son la duración del mismo –en el anterior eran varios minutos, aquí es más de una hora- y la continuación del tráfico a pesar de incidente.

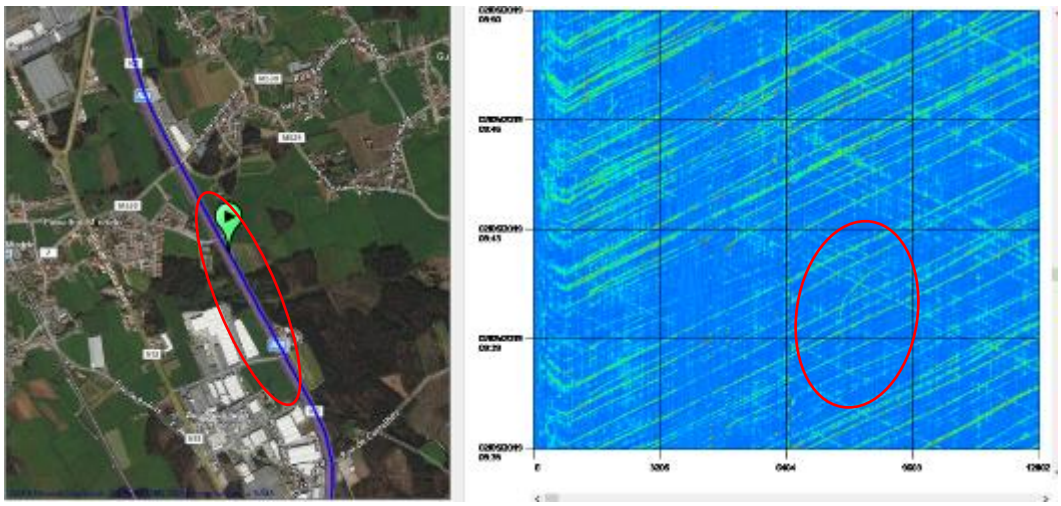


Figura 4.2.5. Trayectoria extraña de un vehículo.

En la Figura 4.2.5. se muestra un evento de difícil caracterización: un vehículo avanzando a un ritmo lento e irregular y que en determinados instantes parece retroceder.

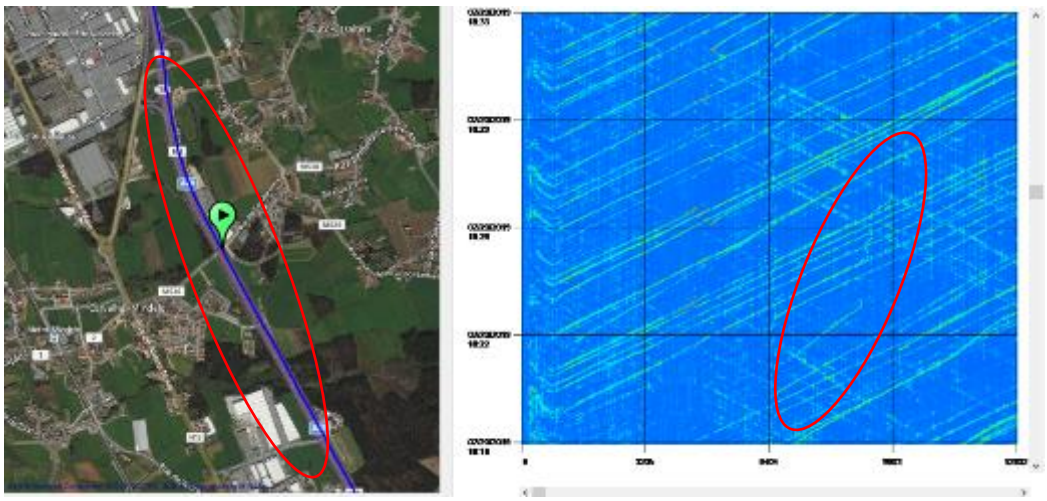


Figura 4.2.6. Parada inesperada en medio de la autovía, continuación a ritmo lento y salida por el primer desvío: posible avería en el coche.

En el caso de la Figura 4.2.6., un vehículo que avanza a un ritmo normal realiza una parada inesperada en medio de la carretera. Seguidamente, continúa a ritmo inusualmente lento e irregular, y abandona la autovía por el primer desvío posible. Esta situación puede ser identificada como una avería en el vehículo.

5. Algoritmos para automatización de detección de eventos

En el Capítulo 4 queda plasmada la posibilidad de hallar –manualmente, tras una minuciosa búsqueda de datos recogidos a lo largo de un extenso periodo anterior- una serie de situaciones de especial interés en la vía en la que está instalado el sistema de sensado acústico.

No obstante, en el presente Trabajo de Fin de Grado va a desarrollarse un algoritmo para el procesado de los *waterfalls* que permita automatizar la detección de algunos de los eventos ilustrados anteriormente, y que puede resultar de gran ayuda para centros de control y monitorización de las carreteras.

5.1. Procesado de *waterfalls*

El procesado se realiza completamente en Matlab. Para comenzar, es pertinente conocer que los datos recogidos por el sensor son *waterfalls* con una duración temporal corta de aproximadamente 45 segundos –del tramo de fibra completa-, que de ahora en adelante serán denominados *lonchas*, de tal manera que la composición de los *waterfalls* completos se realiza mediante la superposición de todas las lonchas. De esta forma, los diferentes procesos que se ejecutan y llevan a cabo para el tratamiento de eventos se aplican individualmente a cada una de las lonchas. Esto supone que, a pesar de que este trabajo se realice con datos grabados, se pueda incorporar todo el desarrollo para su aplicación a tiempo real, puesto que el procesado se realiza en un tiempo medio de aproximadamente 5 segundos, muy por debajo del tiempo de duración de una loncha.

El nombre con el que el sistema etiqueta a cada una de las lonchas proporciona información que es extraída para su posterior análisis. El formato que presentan los nombres de las lonchas es el siguiente:

20190221120152706_00006_51194_2448538127416_Das007_814476.bmp

donde:

- Los primeros ocho caracteres (20190221) son la fecha: 21/02/2019.
- Las siguientes cuatro cifras (1201) representan la hora: 12:01

- El número después del primer '_' (00006) es la distancia en metros donde inicia el tramo de fibra.

- El próximo número, posterior al siguiente '_' (51194), es la distancia en metros donde acaba el tramo de fibra. Así, la duración total de la fibra son aproximadamente 51.2 km.

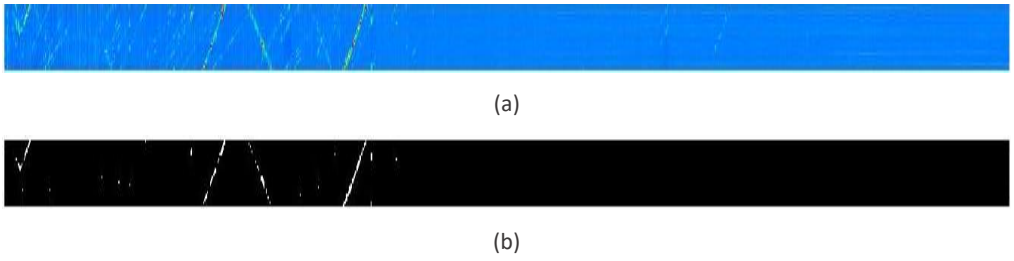
- Después del consiguiente '_', se muestra el tiempo absoluto en milisegundos (2448538127416) cuando fue adquirida la loncha. Este valor como tal carece de información, pero será importante para compararlo con el de lonchas anteriores o posteriores, para saber el tiempo que ha transcurrido entre una y otra, y por lo tanto, la duración de la loncha en cuestión.

- A continuación, se muestra el nombre y versión del sensor (Das007).

- El siguiente dato es una referencia (814476).

- Por último, se tiene la extensión de las imágenes o lonchas (.bmp).

Al leer la imagen, se obtienen tres matrices de las correspondientes componentes R, G y B. Para etapas posteriores, es necesaria la binarización de la imagen, es decir, su conversión a blanco y negro.



(a)

(b)

Figura 5.1.1. a) Loncha original. b) Loncha binarizada.

A simple vista, en la Figura 5.1.1. parece evidente la presencia de tres vehículos en sentido "positivo" y uno en sentido contrario al de propagación de la luz.

5.2. Cálculo de velocidades medias por tramos

El primero objetivo en el que se va a centrar el trabajo referido a la automatización es el cálculo de la velocidad de cada uno de los vehículos presentes en la loncha a procesar. Para ello, es esencial la utilización de la transformada de Hough y de las tres funciones utilizadas para su cálculo: *hough*, *houghpeaks* y *houghlines* (**Anexo B**). Básicamente, esta transformada se utiliza para detectar de manera

computacionalmente eficiente la presencia de grupos de puntos pertenecientes a una misma línea o curva. Para lo cual, se transforma cada uno de los puntos de la figura en una linea en un "espacio paramétrico", es decir, los puntos quedan representados en base a sus coordenadas ρ (distancia al origen de coordenadas) y Θ (ángulo respecto al eje de las abscisas positivas) [12]. A partir de una línea referida a cada vehículo, es relativamente sencillo el cálculo de las velocidades, conocida la relación de aspecto en píxeles de la imagen, y el tiempo y la distancia reales representados en la misma.

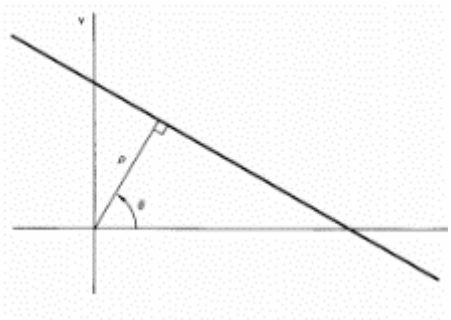


Figura 5.2.1. Representación de una línea según sus coordenadas ρ, Θ [14].

De este modo, ajustando una serie de parámetros de entrada de las funciones de la transformada Hough y que permiten regular la resolución, todos los puntos pertenecientes a un mismo *bin,* entendido éste como un intervalo de valores ρ y Θ determinados, estarán asociados a una misma línea. De este modo, se produce una discretización de la imagen donde se aplica la transformada, al reducirse la búsqueda de segmentos a toda la imagen a la detección de los mismos en cada uno de los *bins* definidos. Es aquí donde reside una de las mayores dificultades del procesado: en el compromiso existente en la elección de parámetros de las distintas funciones empleadas para la aplicación de la transformada de Hough y la detección de picos y líneas a través de la misma (**Anexo B**). Por un lado, hay que escoger la resolución tanto en ρ como en Θ que permita que segmentos pertenecientes a un mismo vehículo pertenezcan en la medida de lo posible al mismo *bin*; por otro, es necesario ajustar las variables *FillGap*, *MinLength* referidas a las líneas trazadas por la función *houghlines* para que no sean considerados como segmentos que representan a un vehículo la unión entre dos picos cualquiera detectados por la función *houghpeaks*.

Tras la aplicación de dicha transformada, se obtiene a la salida un conjunto de puntos donde inician las líneas; y a su vez, los puntos donde las mismas finalizan. Mediante la representación de los mismos y unidos por líneas, se obtiene un resultado como el mostrado en la Figura 5.2.2.:

(a)

(b)

Figura 5.2.2. a) Resultado de la transformada de Hough. b) Zoom a una de las líneas (vehículos) detectadas.

En la Figura 5.2.2. b) se aprecia que, al tener las líneas detectadas una anchura de varios píxeles, son encontradas a su vez varias "sublíneas" dentro del mismo *bin*, y que representan algo sin información alguna, a pesar de que en la Figura 5.2.2. a) parezca que el seguimiento es verdaderamente óptimo.

Para solucionar este problema y reducir el número de líneas representando un único coche, se realiza un diezmado en dos dimensiones. Esto es, se divide la imagen en cuadrados de dimensiones 7 x 7 y todos esos píxeles se representan por un '1' si se supera un determinado umbral. La elección del factor de diezmado presenta un compromiso: si es pequeño, no se soluciona el problema presentado en la Figura 5.2.2.; sin embargo, si es muy grande, pueden perderse algunas líneas. A base del método de prueba y error con varias lonchas representativas, se decide elegir un factor 7. Además, se escoge el mismo factor en la dimensión horizontal que en la vertical para mantener la relación de aspecto. De esta manera es posible disminuir el número de píxeles en la anchura de cada una de las líneas, minimizando así el número de segmentos que la transformada de Hough traza.

(a)

(b)

Figura 5.2.3. a) Loncha diezmada. b) Zoom a la loncha.

De esta manera, aplicando ahora la transformada de Hough a la imagen diezmada, el número de líneas sigue siendo elevado, pero todas siguen una dirección similar pues son prácticamente paralelas. Este fenómeno es más fácil de solucionar mediante cierto

postprocesado que el mostrado en la Figura 5.2.2., donde las líneas seguían direcciones muy dispersas y poco representativas.

(a)

(b)

Figura 5.2.4. a) Resultado de la transformada de Hough sobre la imagen diezmada. b) Zoom a unas de las líneas (vehículos) detectadas.

El siguiente paso se basa en una aproximación consistente en que los vehículos que sean detectados en la loncha van a permanecer durante la misma desde el principio hasta el final; es decir que, si el coche entra a la autovía en medio de la loncha, se considerará que lo ha hecho al principio; y si sale a mitad, se dará por supuesto que ha abandonado la carretera en el momento en el que termina la loncha. Esta es una aproximación que simplifica mucho el procesado y el seguimiento de vehículos; mientras que el error que puede suponer es relativamente pequeño, pues se está realizando el tratamiento en intervalos de unos 45 segundos.

Consiguientemente, se ordenan las líneas de principio a final y se establece una distancia mínima que deben estar separados los puntos inicial y final para representar a vehículos distintos. Así, cuando se encuentran líneas devueltas por la transformada de Hough que no superan el umbral mínimo en cuanto a separación de distancias, se ponderan sus puntos iniciales y finales con el objetivo de conseguir una única línea que verdaderamente represente al vehículo en cuestión.

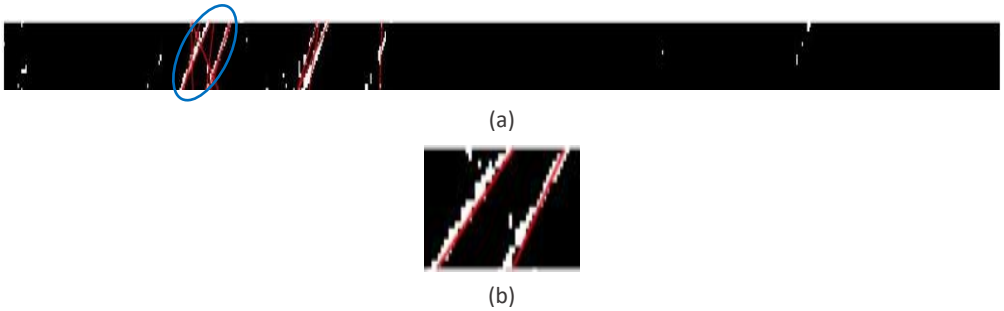
(a)

(b)

Figura 5.2.5. a) Resultado tras juntar líneas que representan a mismo vehículo. b) Zoom a las líneas.

A pesar de que el resultado en la Figura 5.2.5. parece ya óptimo, en situaciones con coches circulando muy próximos entre sí (habitual en horas punta de tráfico), no se ha considerado hasta ahora que se puedan unir líneas cuyo punto inicial esté en un vehículo, y el punto final en otro distinto pero cercano; o viceversa. Para ello, se aplica

otra técnica que descarte las líneas que representen casos como el descrito, u otros similares.



(a)

(b)

Figura 5.2.6. a) Líneas que empiezan en un vehículo y acaban en otro. b) Zoom a las líneas rodeadas, postprocesadas para evitar el problema de a).

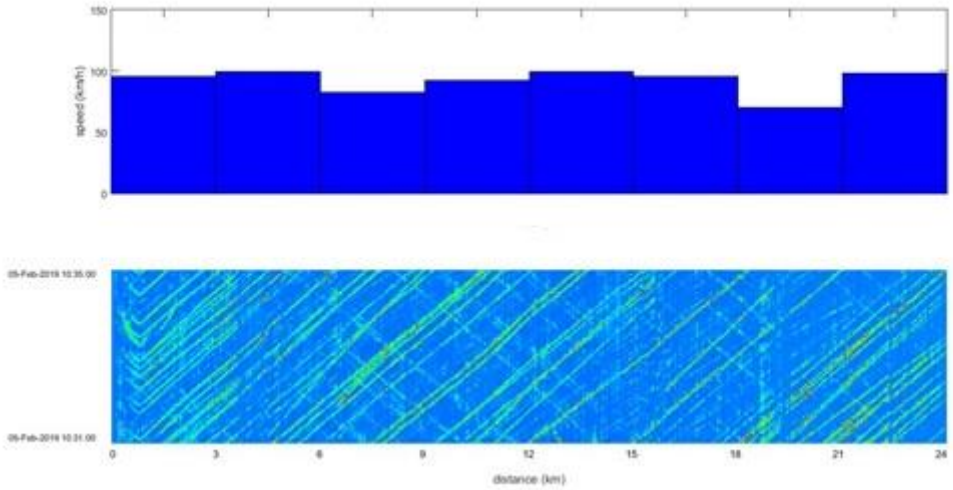
Conseguido ya el seguimiento de vehículos, con las líneas obtenidas es posible calcular las velocidades de cada uno de ellos. Puede resultar interesante dividir el tramo entero de fibra en "subtramos" de reducida longitud –en función de la precisión que se quiera pues, cuanto más corto es el tramo, menor número de vehículos y menor fiabilidad ofrece el resultado-, de tal forma que el caudal de vehículos en cada uno de ellos sea suficiente para el cómputo de las velocidades medias en dichos subtramos. De esta forma pueden registrarse estadísticas monitorizando el funcionamiento del sensor para conseguir detectar anomalías en el flujo de vehículos correspondientes a atascos o incidentes en el caso de una bajada pronunciada de la velocidad media en un tramo concreto.

Para el cálculo de estas velocidades medias, como el número de vehículos que se desplazan desde el final de la fibra en el sentido hacia donde está ubicado el sensor es reducido, se elabora una función distinta que siga únicamente a los vehículos que se desplazan en el mismo sentido que se propaga la luz, considerando así únicamente estas velocidades. Así mismo, para que la imagen presente mejor visibilidad, se representan solamente los primeros 24 kilómetros, pues queda comprobado que a mayores distancias el cálculo de velocidades medias carece de sentido al no ser capaz de detectar vehículos debido a la elevada atenuación de la señal. A pesar de ello, ha intentado adaptarse la etapa de binarización mediante la aplicación de un umbral adaptativo para poder detectar vehículos presentes a distancias más lejanas. Sin embargo, estas señales procedentes de puntos ubicados al final del sistema tienen amplitudes tan bajas que quedan totalmente enmascaradas en el ruido.

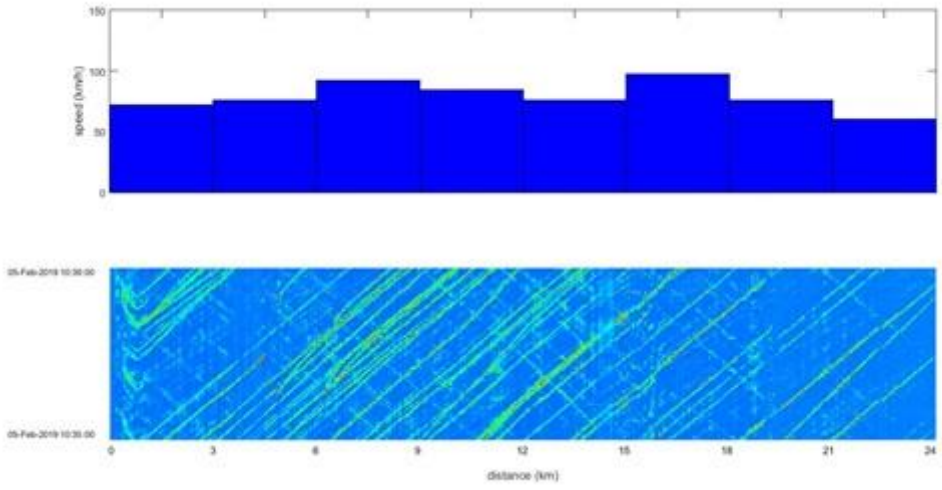
Para obtener mayor precisión en el cálculo, pese a que cada loncha se procesa individualmente como ya ha sido explicado, no es hasta que se procesan cinco lonchas cuando se proyectan los resultados. Esto es debido a que se realiza un promediado de

las velocidades medias por tramo calculadas en cada una de las lonchas, lo cual ofrece mayor robustez y fiabilidad.

Para la representación de velocidades medias por tramo, se opta por dividir la pantalla en dos, colocando en la mitad superior un diagrama de barras representando dichas velocidades; y en la parte inferior, el *waterfall* –concatenación de cinco lonchas- que se ha procesado y del que se están representando los datos. Así, se hacen coincidir los ejes espaciales de las dos imágenes (eje x: distancia en kilómetros) para que los resultados sean más visuales. Hay que considerar que, como ya se ha explicado anteriormente, la mayoría de los vehículos detectados corresponden a camiones por la diferencia de intensidad en las interferencias registradas. Esto hace que las velocidades medias obtenidas en cada uno de los tramos tomen valores de velocidades habituales en este tipo de vehículos.



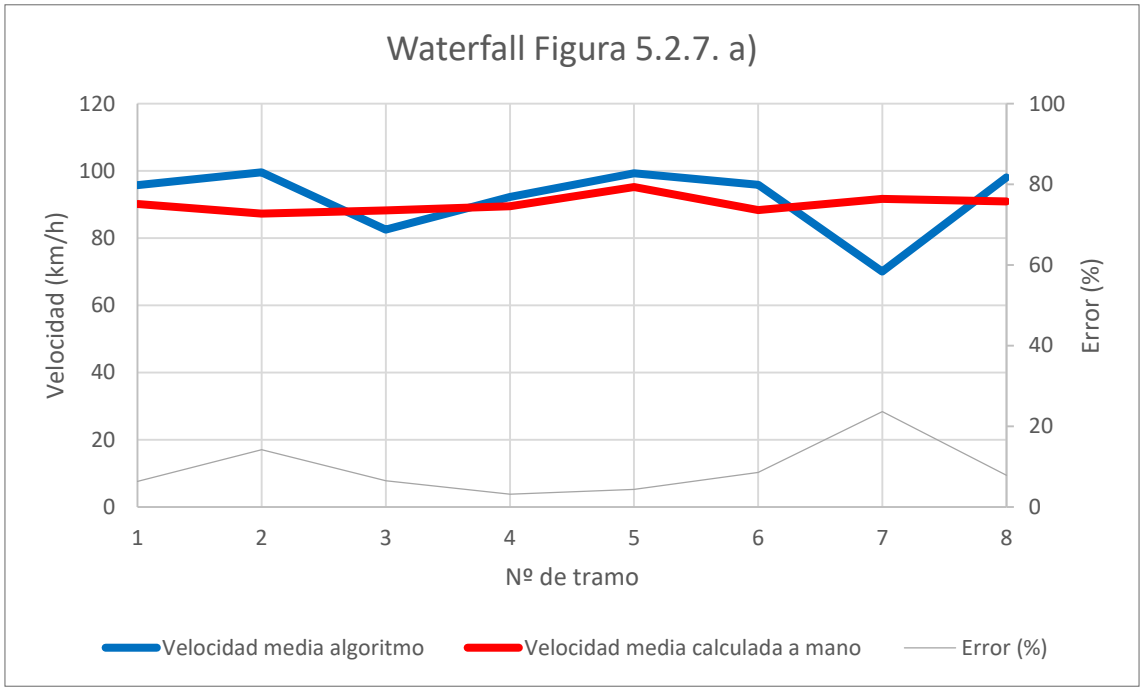
(a)



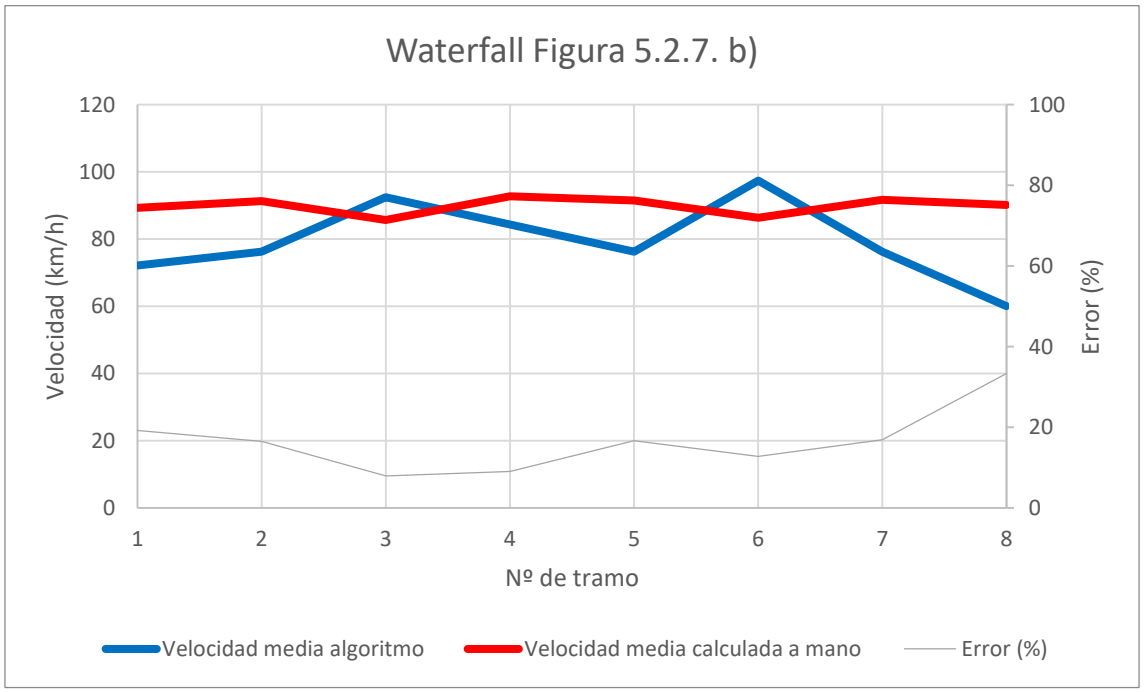
(b)

Figura 5.2.7 a), b). Velocidades medias en tramos de 3 kilómetros en una situación de tráfico normal.

Para contrastar la precisión de los resultados ofrecidos por el algoritmo, se procede a realizar un cálculo manual de las velocidades medias en cada uno de los tramos en los que se divide la autovía, utilizando para ello la aplicación presentada en el Capítulo 4.



(a)



(b)

Figura 5.2.8 a), b). Comparación del cálculo de velocidades medias por tramos mediante el algoritmo presentado y de forma manual.

Waterfall	Error medio 1-8 (%)	Error medio 1-6 (%)
Figura 5.2.7. a)	9.3	7.1
Figura 5.2.7. b)	16.4	13.7

Tabla 5.2.1. Cálculo del error medio en el cómputo de velocidades

En la Tabla 5.2.1. puede verse como el error medio resultante calculado a partir del error cometido en cada uno de los ocho tramos está en porcentajes relativamente bajos, en torno a la décima parte del total. No obstante, analizando las gráficas presentadas en la Figura 5.2.8., se observa que en la mayoría de los tramos el error cometido es menor, pero hay dos situaciones que elevan el porcentaje final:

- Zonas de nula o muy elevada sensibilidad.

- Distancias alejadas del inicio del tendido de la fibra óptica, donde ya se ha comentado en capítulos anteriores que debido a la atenuación de la señal las trazas recibidas no permiten un control preciso de los vehículos.

Por ello, en la tabla se presenta una segunda medida que únicamente muestra el error medio obtenido en los seis primeros tramos, comprobando que se obtienen valores de error más bajos debido a las dos razones mencionadas previamente, lo cual confiere mayor fiabilidad al algoritmo desarrollado.

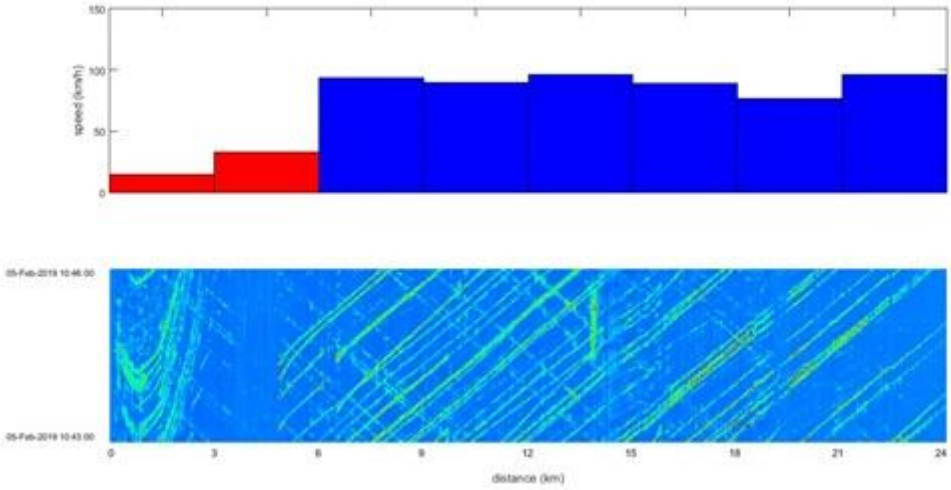
5.2.1. Aplicación a detección de atascos y accidentes

Con lo desarrollado hasta ahora, en la Figura 5.2.7. queda reflejada la capacidad del algoritmo para calcular las velocidades medias en diferentes segmentos de la autovía donde está instalado el sistema.

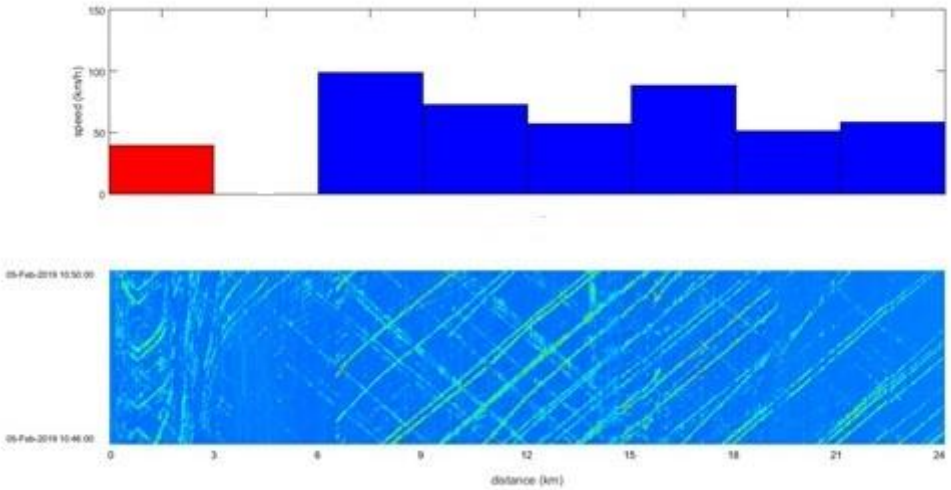
De todos los eventos presentados en el Capítulo 4, la principal aplicación que puede tener este cálculo es la detección automática de atascos y/o accidentes que puedan darse. En casos de este tipo, la velocidad media del tramo en el que se produce se reduce bruscamente; en situaciones extremas, en los siguientes kilómetros no habrá vehículos por lo que la velocidad será muy baja o nula. No obstante, hay que tener en cuenta que puede producirse en una zona próxima a una entrada a la autovía. En dicho supuesto, la velocidad del siguiente segmento no será cero porque, a pesar de que por la autovía

principal en el tramo en cuestión no haya vehículos, seguirán incorporándose por el margen.

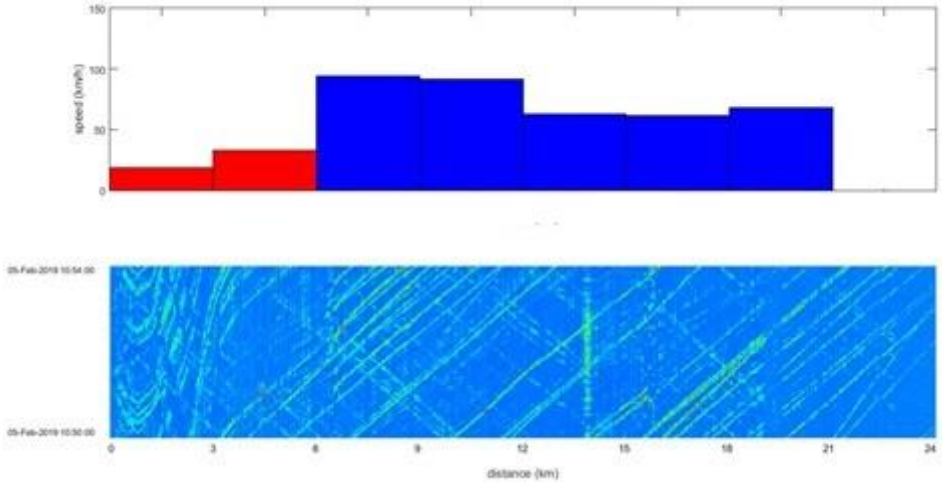
A continuación se examina el accidente que se produjo en la autovía A28 y que ha sido ilustrado en la Figura 4.2.4.. Para ello, se elige que las barras del diagrama representado sean de color azul cuando la velocidad está por encima de una mínima (40 km/h, velocidad media bastante reducida en una autovía); y rojas cuando se están dando situaciones de alerta.



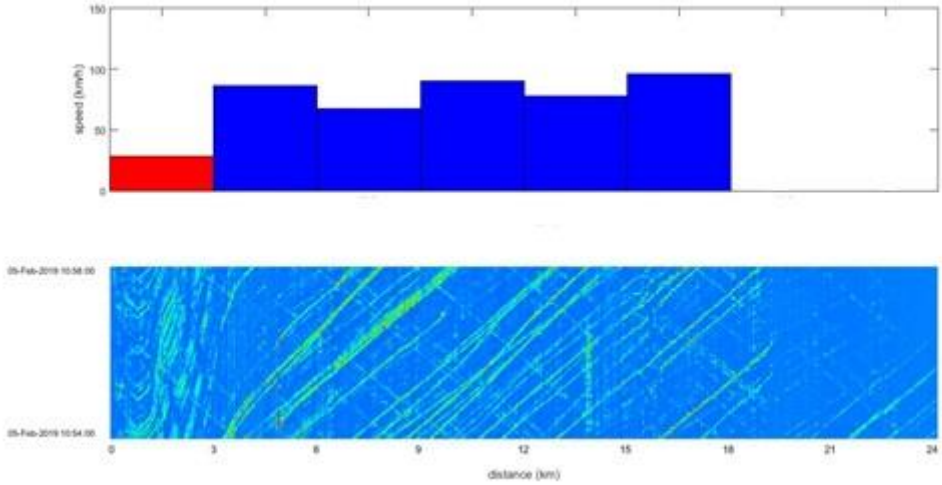
(a)



(b)



(c)



(d)

Figura 5.2.1.1 a), b), c), d). Evolución temporal de las velocidades medias en tramos de 3 kilómetros del accidente expuesto en la Figura 4.2.4.

En la Figura 5.2.1.1. puede corroborarse el buen funcionamiento del algoritmo, capaz de detectar velocidades por debajo de un umbral que supone una situación atípica en una autovía, quedando manifestado en el diagrama de barras para su control y posterior actuación si así es considerado necesario.

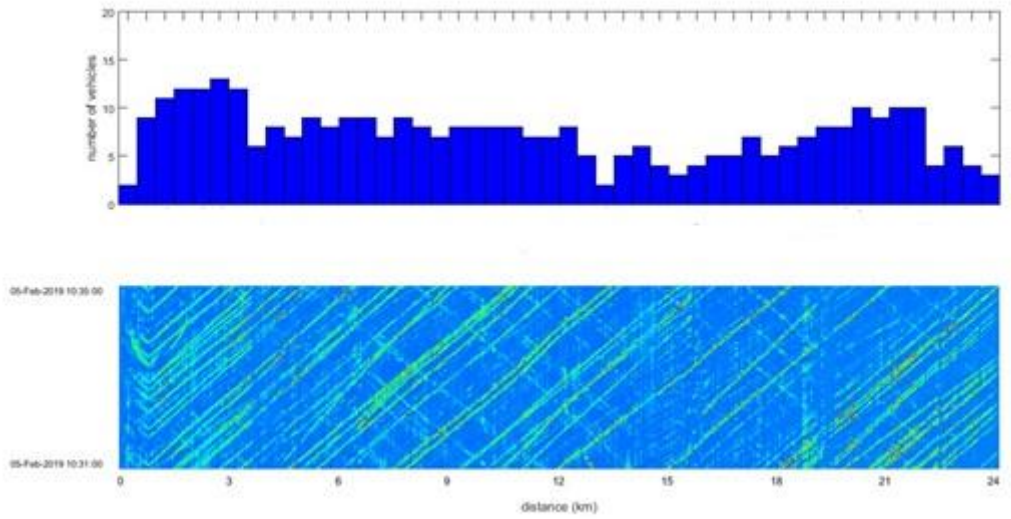
5.3. Cálculo del número de vehículos por tramos

Posteriormente, se desarrolla una función que permita contabilizar el número de vehículos. La función utilizada para el cálculo de velocidades también serviría para contar los coches y camiones que hay en la carretera en una loncha determinada. Sin embargo, cuando hay un tráfico denso, aparecen más líneas de las que deben –aunque

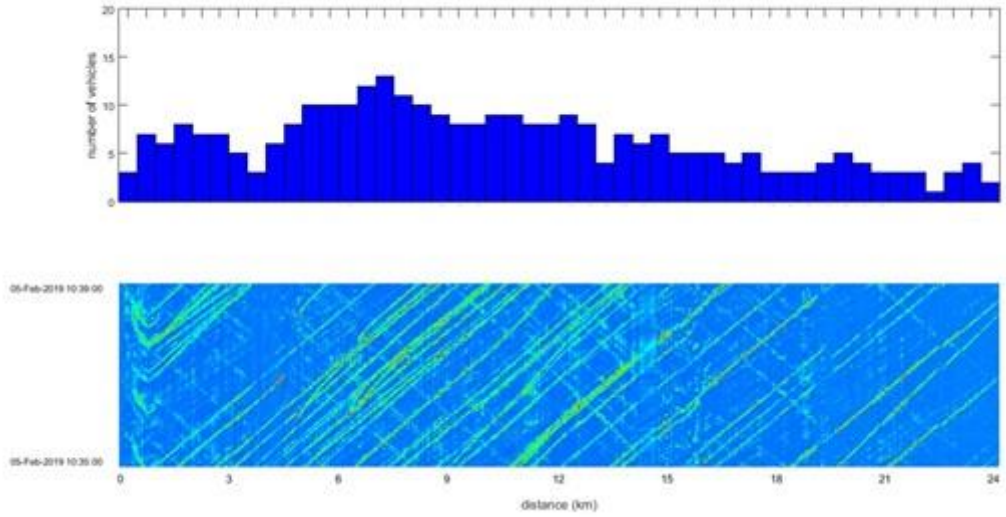
la velocidad media sigue resultando buena- y por lo tanto no es apta para controlar el caudal de tráfico.

Así, a la imagen diezmada y binarizada se le aplica un algoritmo consistente en contar el número de alternancias entre '1' y '0' en cada punto *z* de la carretera, entendiendo cada uno de los cambios como un vehículo presente. Se realiza para todas las columnas de cada una de las lonchas, promediando luego el resultado de los vehículos presentes en segmentos de una longitud determinada para obtener la cifra final de vehículos presentes en dichos puntos kilométricos.

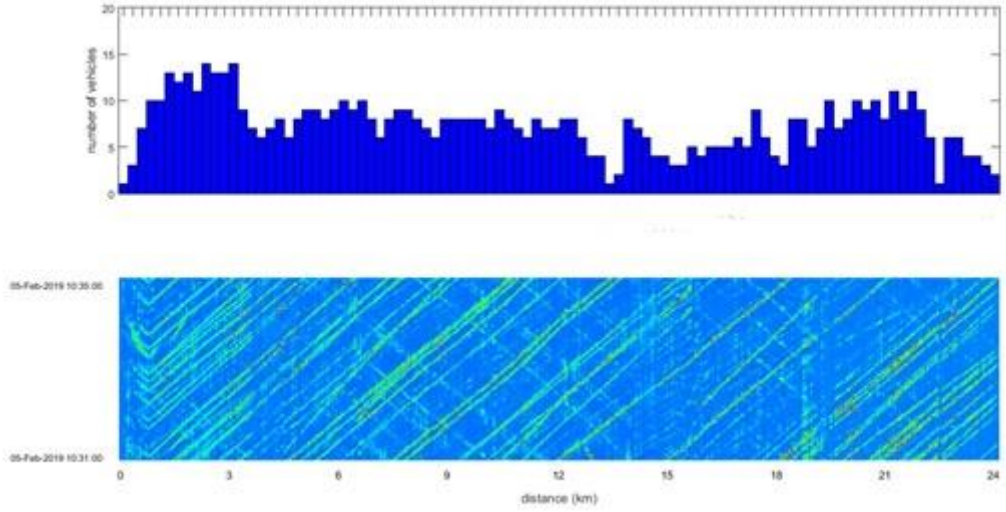
Del mismo modo que ocurría en el Capítulo 5.2., existe un compromiso a la hora de determinar la longitud del tramo donde evaluar el número de vehículos: cuanto menor sea dicho segmento, más interesante puede resultar para aplicaciones como las del Capítulo 5.2.1., pero menor es el número de vehículos y menor es el promediado que se realiza, por lo que menos precisos son los resultados. Seguidamente, se presentan en la Figura 5.3.1. los resultados obtenidos para tramos de 0.25 kilómetros y de 0.5 kilómetros en situaciones de tráfico corriente, donde queda reflejado dicho compromiso y las ventajas y contrapartidas que presentan cada uno de ellos.



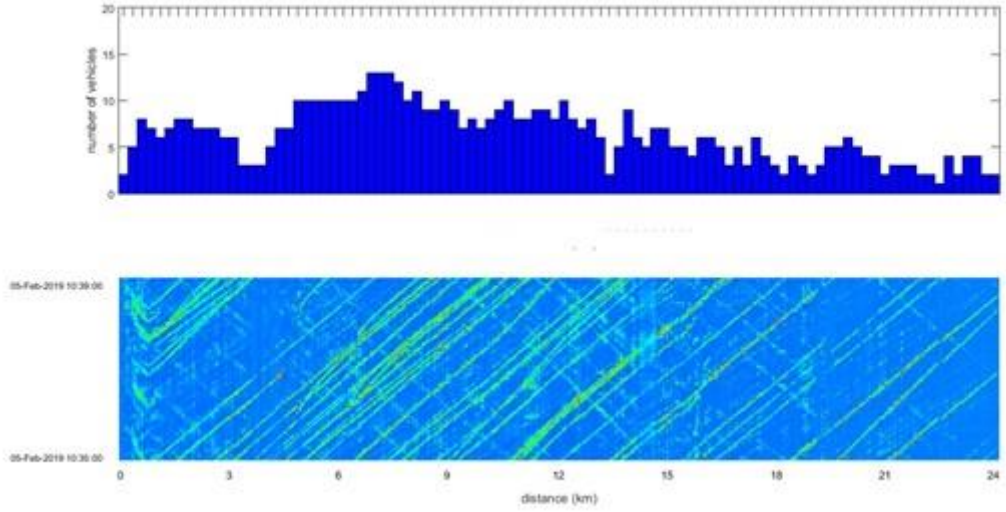
(a)



(b)



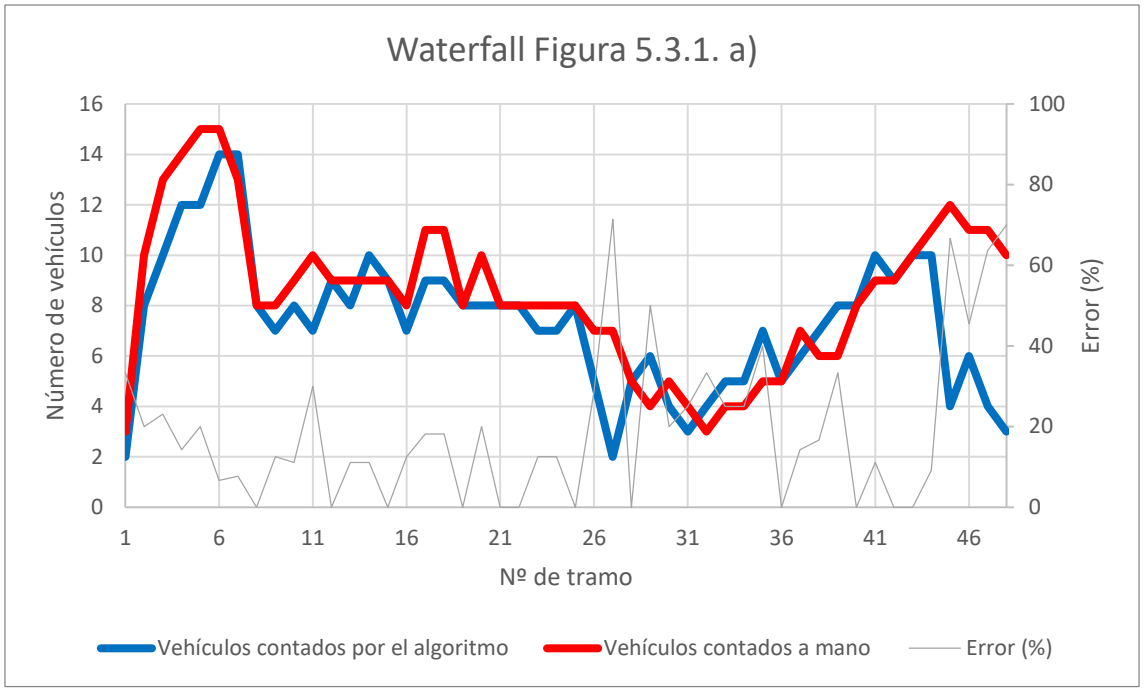
(c)



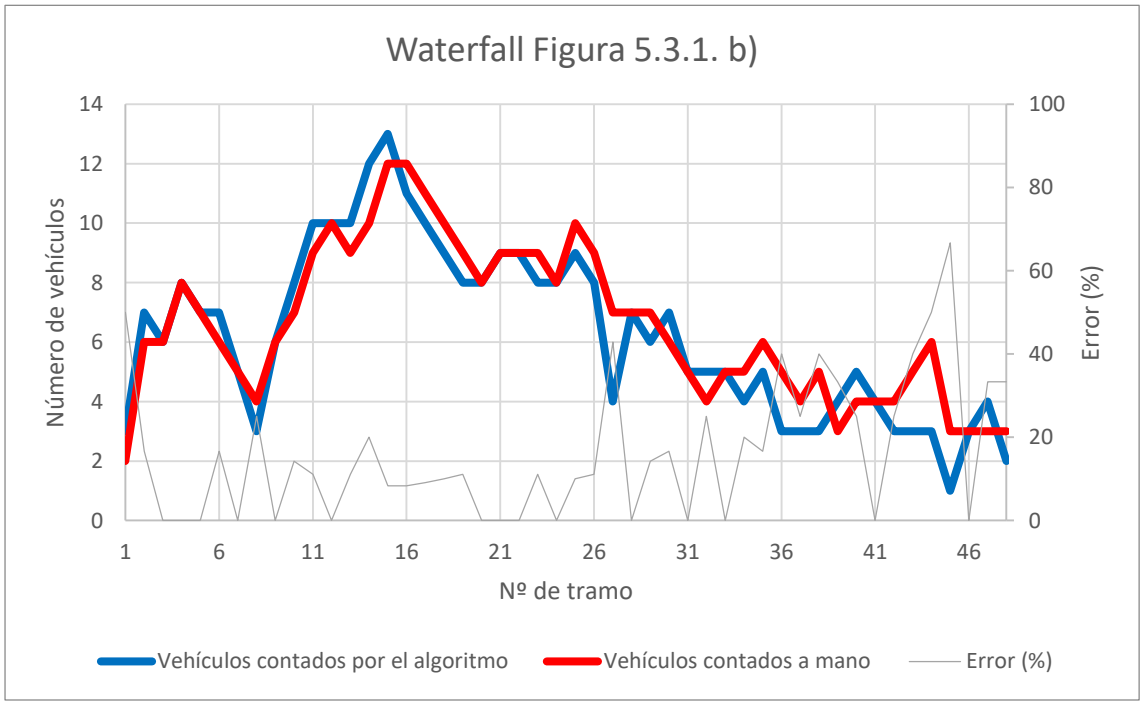
(d)

Figura 5.3.1 a), b). Cálculo del número medio de vehículos en tramos de 0.5 kilómetros en una situación de tráfico normal. c), d). Cálculo del número medio de vehículos en tramos de 0.25 km en una situación de tráfico normal.

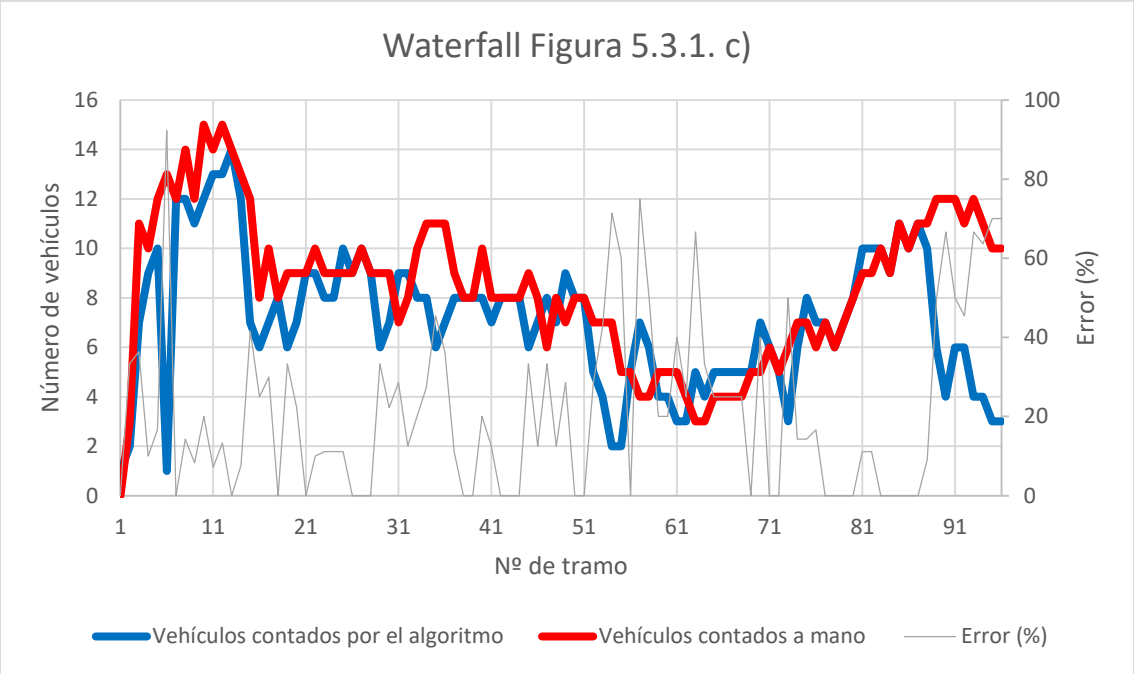
De la misma manera que se hizo en el Capítulo 5.2. para el cálculo de velocidades, se realiza un cómputo manual del número de vehículos en cada uno de los tramos para comprobar la exactitud del algoritmo desarrollado,



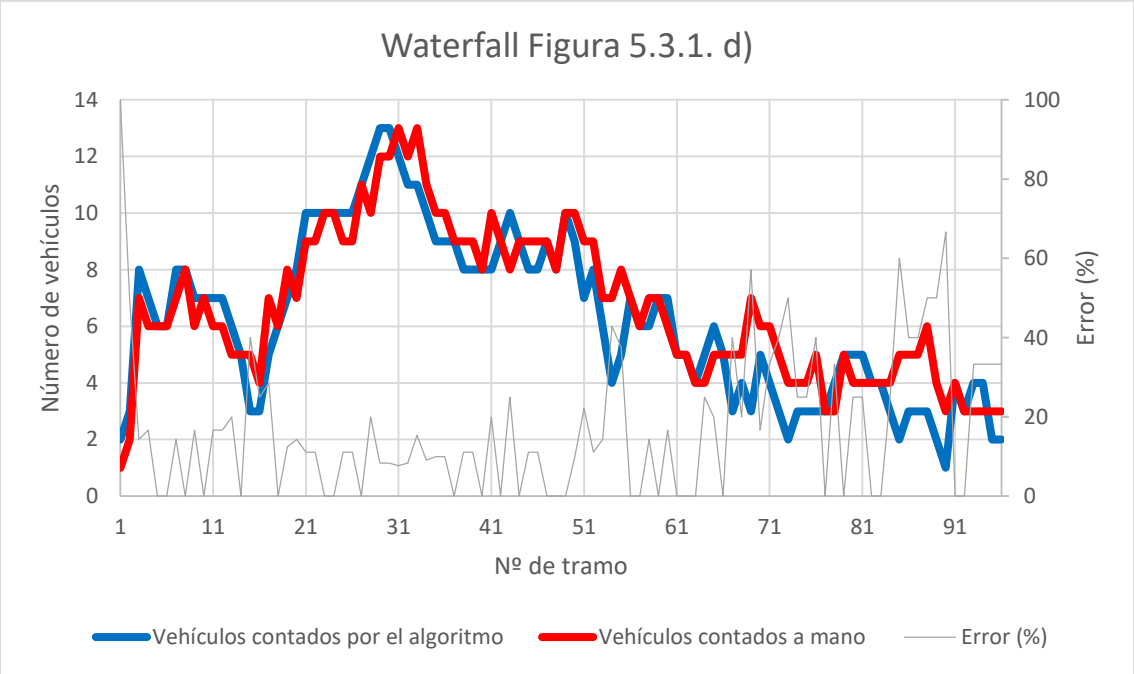
(a)



(b)



(c)



(d)

Figura 5.3.2. a), b), c), d). Comparación del cálculo del número medio de vehículos por tramos mediante el algoritmo presentado y de forma manual.

Waterfall	Error medio 1 (%)	Error medio 2 (%)
Figura 5.3.1. a)	19.7	10.8
Figura 5.3.1. b)	16.5	10.3
Figura 5.3.1. c)	22.2	14.2
Figura 5.3.1. d)	18.0	11.9

Tabla 5.3.1. Cálculo del error medio en el cómputo de número de vehículos.

En la Tabla 5.3.1. puede analizarse el error medio cometido por el algoritmo en el cálculo del número medio de vehículos por tramo. Se presentan dos tipos de errores: el primero de ellos es la media de los errores cometidos en cada uno de los segmentos. Para el cálculo del segundo error se tienen en cuenta únicamente los tramos "no conflictivos" –aquellos que no presentan problemas de sensibilidad-, pues a partir de la observación de la Figura 5.3.2. se concluye que en los kilómetros 13-16, y 23-24 (tramos 26-33 y 45-48 para división en segmentos de 0.5 kilómetros; tramos 52-66 y 89-96 para los de 0.25 kilómetros) se comete un error elevado por la presencia de infraestructuras en lugares próximos a la autovía que disminuyen la precisión en la recogida de datos.

Así, se corrobora el compromiso presentado previamente: cuanto más amplio es el tramo donde se examinan el número de vehículos, más preciso es el cálculo. No obstante, el porcentaje de acierto del algoritmo en cualquiera de los casos es elevado. Además, si se consideran solo los tramos donde la información recogida es realmente representativa, las prestaciones mejoran todavía más, logrando errores en torno al 10%.

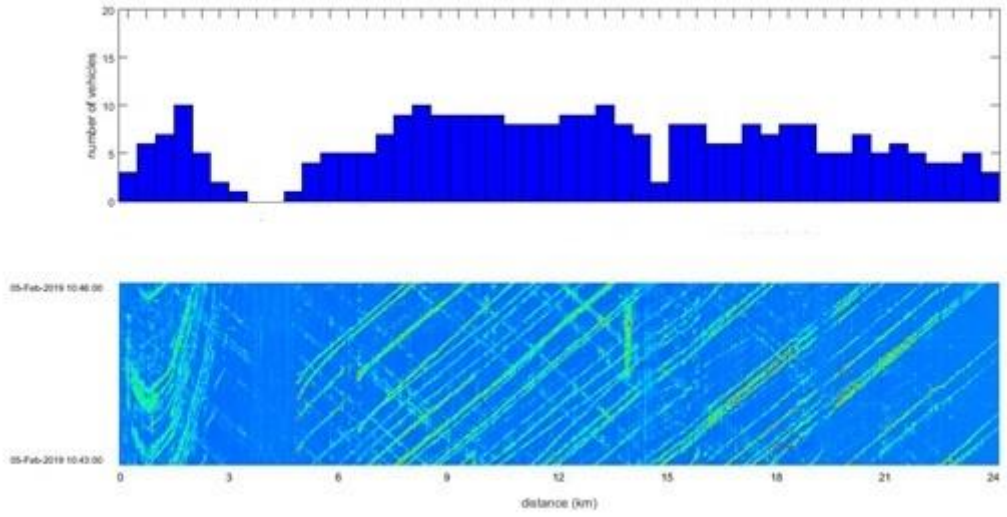
5.3.1. Aplicación a detección de atascos y accidentes

De la misma forma que el cálculo de velocidades medias por tramo se puede utilizar para encontrar atascos y accidentes, el número medio de vehículos también puede aplicarse para este tipo de situaciones.

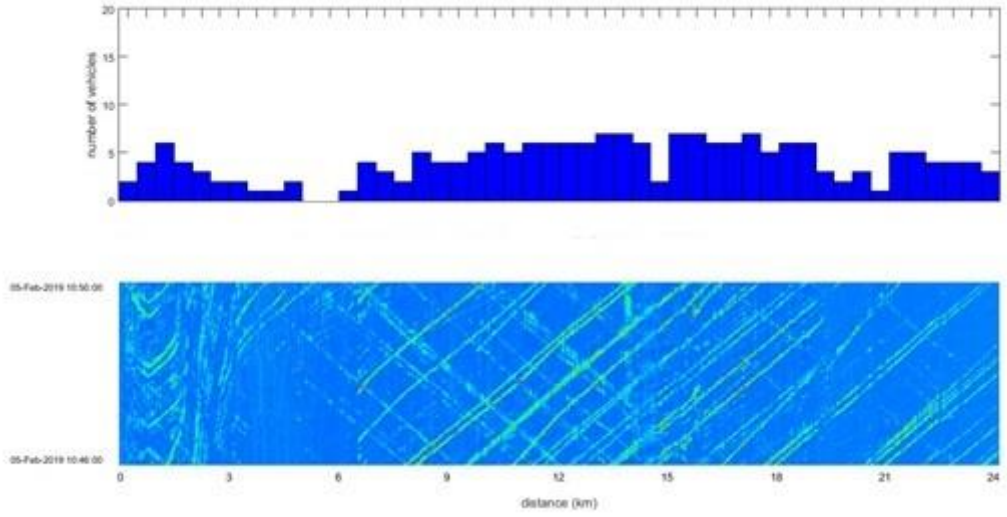
De esta manera, en condiciones habituales de tráfico el número de vehículos que atraviesan un punto kilométrico es aproximadamente constante (dependiente, eso sí, de la franja horaria). Así, si se detecta un caudal de vehículos por debajo del esperado durante el procesado de varias lonchas, puede considerarse como una situación en la que el tráfico no está fluyendo como debería hacerlo.

Este cálculo también ofrece resultados potentes, aunque es necesario considerar, aparte de la hora del día que se está monitorizando, que hay determinados puntos con entradas/salidas en la autovía que registrarán mayores fluctuaciones.

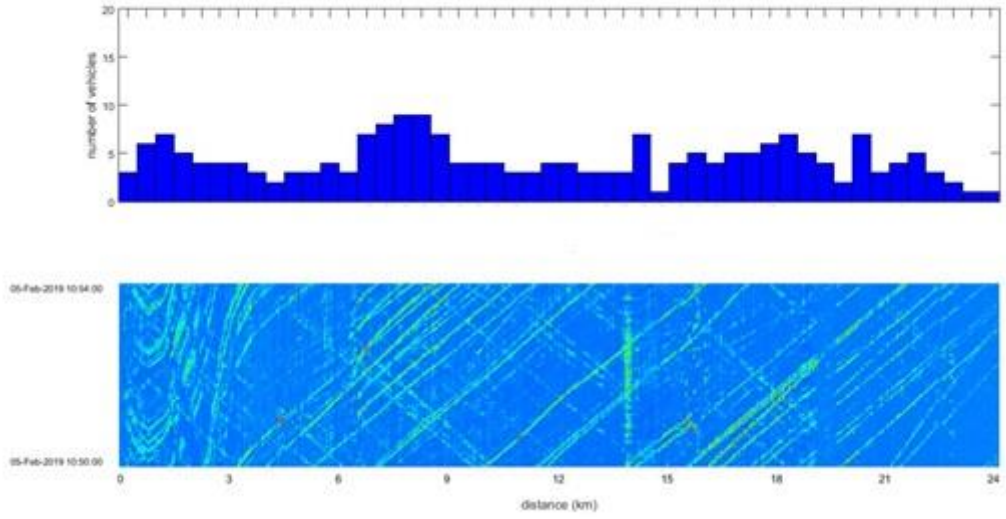
Se muestran así los datos obtenidos para el mismo momento y suceso ilustrado en la Figura 5.2.1.1., de tal manera que es posible contrastar la información dada por uno y otro algoritmo y concluir que ambos ofrecen resultados interesantes y con directa aplicación a la detección de parones largos del tráfico en carreteras. No obstante, como puede observarse, hay determinadas zonas donde la sensibilidad es tan mala que el algoritmo no funciona y los resultados que ofrece no son fiables.



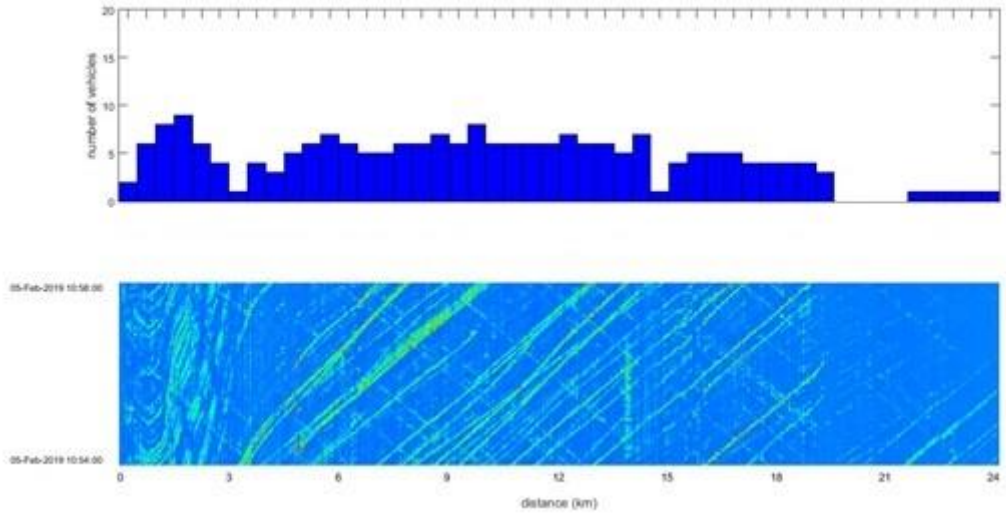
(a)



(b)

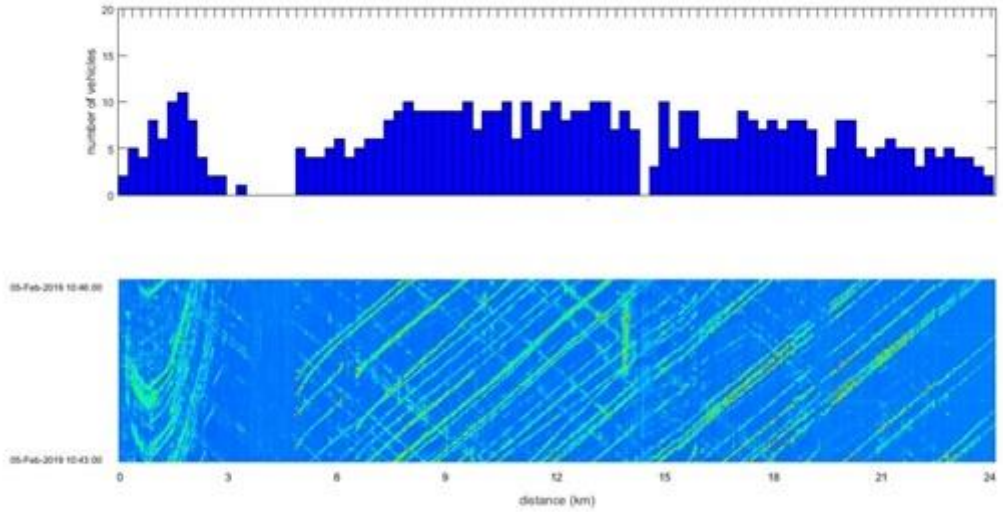


(c)

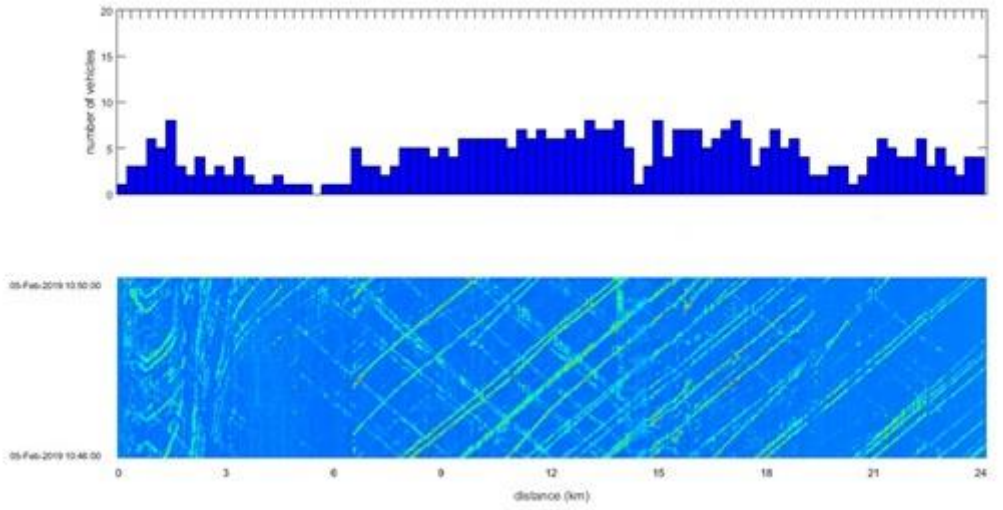


(d)

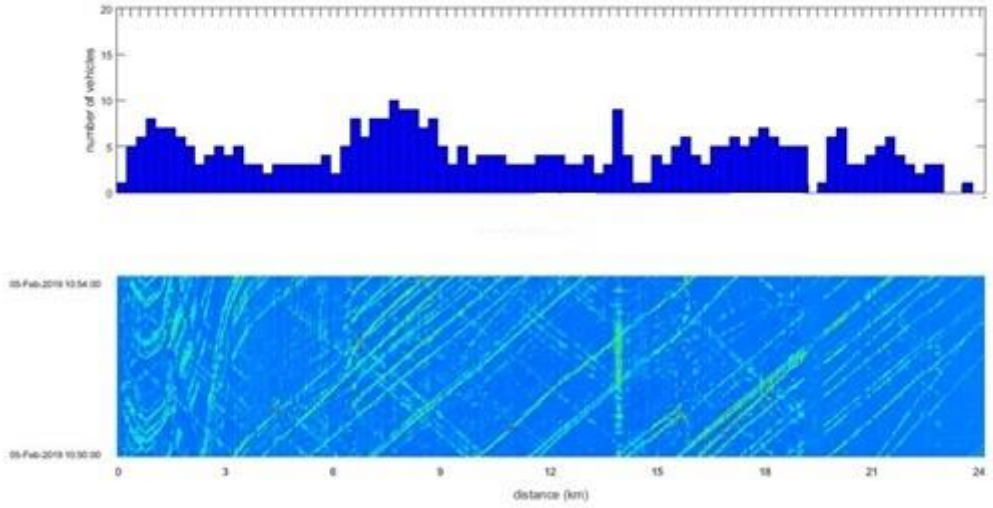
Figura 5.3.1.1 a), b), c), d). Evolución temporal del número medio de vehículos en tramos de 0.5 kilómetros del accidente expuesto en la Figura 4.2.4.


number of vehicles
05-Feb-2019 10:46:00
05-Feb-2019 10:43:00
0
3
6
9
12
15
18
21
24
distance (km)

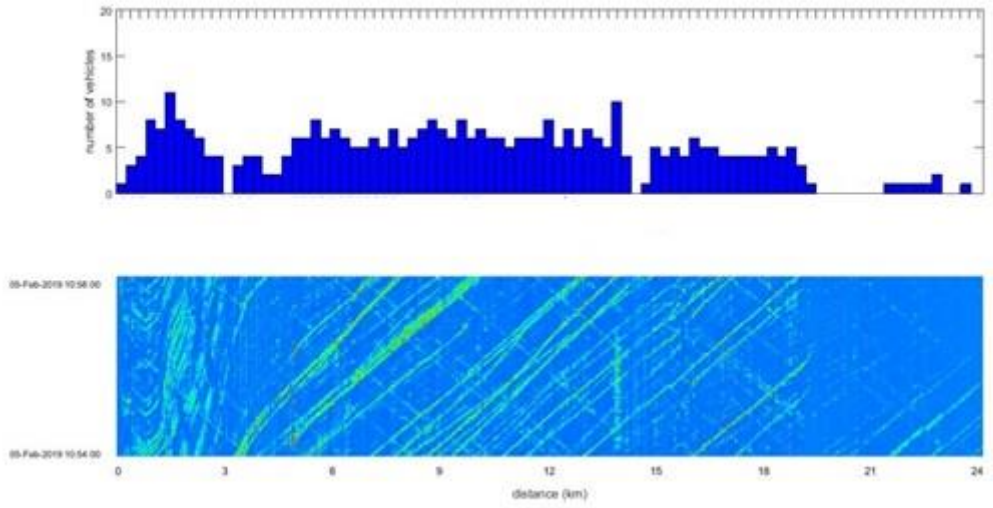
(a)


number of vehicles
05-Feb-2019 10:50:00
05-Feb-2019 10:46:00
0
3
6
9
12
15
18
21
24
distance (km)

(b)



(c)



(d)

Figura 5.3.1.2 a), b), c), d). Evolución temporal del número medio de vehículos en tramos de 0.25 kilómetros del accidente expuesto en la Figura 4.2.4.

6. Conclusiones y líneas futuras

6.1. Aplicación del filtrado frecuencial a la generación de nuevos *waterfalls*

En el Capítulo 3 se demuestra la **mejora en la representación de los *waterfall*s tras la aplicación del método de filtrado en el dominio frecuencial en la banda de 20 – 70 Hz**.

Por tanto, podría ser de gran utilidad para la consecución de unos *waterfall*s más limpios y representativos el hecho de incorporar el filtrado frecuencial en el propio software de generación de *waterfall*s en tiempo real, incluso combinándolo con parte del procesado proporcionado por *Aragón Photonics Labs* para aprovechar también los beneficios que de la aplicación de estas técnicas se obtienen. De esta manera, podría conseguirse mejorar las tareas relativas a detección y caracterización de eventos, así como el seguimiento de vehículos, optimizando la automatización de la aplicación en sí.

6.2. Optimización del algoritmo de automatización para mejorar los resultados

En el Capítulo 5 se ha explicado el desarrollo de una serie de técnicas de procesado implementadas para el seguimiento de vehículos y la monitorización de la autovía a partir del mismo. A partir de comparaciones realizadas entre datos reales y resultados obtenidos mediante la aplicación del algoritmo, ha quedado plasmado un **porcentaje de acierto elevado tanto para el cálculo de velocidades (≈ 90%) como para el número de vehículos (≈ 90%).**

Sin embargo, en condiciones de tráfico abundante, el ajuste de los parámetros de la Transformada de Hough (**Anexo B**) resulta muy complicado debido al fuerte compromiso en la resolución de las variables ρ y Θ. Por lo tanto, en situaciones con gran número de vehículos transitando la carretera, el seguimiento de los mismos no es del todo óptimo ya que las líneas procesadas no son capaces de seguir con exactitud y precisión a cada uno de ellos. De esta forma, pueden pensarse nuevas técnicas que aplicar que consigan solucionar este problema y que permitan realizar un control adecuado de cada uno de los vehículos que circulan.

Por otro lado, durante el desarrollo del algoritmo se ha intentado realizar una binarización distinta para distancias alejadas del comienzo de la fibra, con el objetivo de poder llegar a detectar eventos ocurridos en dichos puntos; pero las amplitudes tienen

valores tan pequeños que no es posible conseguir una representación óptima binaria de la imagen: a partir de una determinada distancia, las señales recibidas quedan enmascaradas por el ruido.

El siguiente paso en el desarrollo del algoritmo sería el ajuste automático de sus parámetros independientemente de las características del *waterfall* tratado, para que por ejemplo pudiera adaptarse a muestras tomadas con equipos de sensibilidades diferentes.

6.3. Aplicación del algoritmo a la detección de nuevos eventos

En el Capítulo 5 se ha presentado el procesado realizado para el seguimiento de vehículos, así como el cálculo de velocidades y número de coches medios por tramo y su aplicación directa a la detección de accidentes. En concreto, una vez corroborada la fiabilidad del algoritmo implementado para las dos aplicaciones mencionadas, se ha demostrado su utilidad en la **aplicación a detección de frenazos generales y/o accidentes en carreteras, obteniendo unos resultados en ambos casos que se ajustan a los fenómenos existentes**, con una disminución de la velocidad media y una acumulación de vehículos en el punto donde se ha producido el incidente, así como una reducción del número de vehículos en tramos posteriores al mismo.

No obstante, es posible adaptar las técnicas aplicadas para centrarse en los demás eventos expuestos en el Capítulo 4.2. y tratar de detectarlos y caracterizarlos. Con una detección mejorada de cada uno de los vehículos presentes en la autovía se facilitaría la extracción de estadísticas y detección de eventos, pues solo habrá que pensar en la situación en la que aplicar el algoritmo y ajustar el procesado para lograrlo. De esta forma, será posible encontrar otro tipo de situaciones atípicas que ya han sido presentadas en el Capítulo 4.2., como por ejemplo reducciones individuales de velocidad y/o frenazos, trayectorias extrañas, paradas en el arcén…

Referencias

[1] Pawan Kumar Dubey, Vibha Shukla. "*Dispersion in Optical Fiber Communication"*. International Journal of Science and Research (IJSR), Octubre 2014.

[2] R. Gowri Manohari, Mr. T. Sabapathi. "*Analysis and Reduction of Polarization Mode Dispersion in an Optical Fiber"*. Internation Conference on Recent Advancements in Electrical, Electronics and Control Engineering, 2011.

[3] Andrey Kobyakov, Michael Sauer, Dipak Chowdhury. "*Stimulated Brillouin scattering in optical fibers"*. Advances in Optics and Photonics, Vol. 2, Issue 1, pp. 1-59, (2010).

[4] Saimunur Rahman. "*An introduction to Stimulated Raman Scattering and its Applications in Optical Fiber Communications*". International Journal of Scientific Footprints, Julio 2014.

[5] Javier Mateo, María Ángeles Losada, Ignacio Garcés. "*Dispositivos y Sistemas de Transmisión Óptica"*. Escuela de Ingeniería y Arquitectura de la Universidad de Zaragoza, 2017.

[6] Dipika Pradhan, Vivekanand Mishra. "*Analysis and Rewiev of EDFA"*. International Journal of Computer Science and Network, Volume 4, Issue 6, Diciembre 2015.

[7] Peter J. Winzer, Andras Kalmar, Walter R. Leeb. "*Role of amplified spontaneous emission in optical free-space communication links with optical amplification – impact on isolation and data transmission; utilization for pointing, acquisition, and tracking*". Instituto de Ingeniería de Comunicaciones e Ingeniería de Radiofrecuencia, Universidad de Tecnología de Viena, Austria.

[8] Jaehee Park, Henry F. Taylor. "*Fiber Optic Intrusion Sensor using Coherent Optical Time Domain Reflectometer"*. Keimyung University, Department of Electronic Engineering, Junio 2003.

[9] Lucía Hidalgo Monge. "*Trabajo Fin de Grado: Análisis de un sistema de sensado acústico distribuido basado en fibra óptica"*. Escuela de Ingeniería y Arquitectura de la Universidad de Zaragoza, 2018.

[10] Bharti Thakur, Rajesh Mehra. "*Discrete Fourier Transform with Different Window Techniques Algorithm".* International Journal of Computer Applications, Volume 135 – No. 12, Febrero 2016.

[11] Juan José Martínez, Asier Villafranca, Pascual Sevillano, Javier Vidal, Abel García. "*Second generation Distributed Acoustic Sensing (DAS) measurements samples for road traffic supervision at A28 Motorway"*. NORTE LITORAL Project, Aragon Photonics Labs S.L.U., R+D deparment, Junio 2018.

[12] Richard O. Duda, Peter E. Hart. "*Use of the Hough Transformation To Detect Lines and Curves in Pictures*", Stanford Research Institute, Volume 15, No. 1, Menlo Park, California, Enero 1972.

[13] Tatsuro Baba. "*Time-Frequency Analysis Using Short Time Fourier Transform"*, Baba Professional Engineer Office, 1920-2, Usuba, Otawara, Tochigi, 324-0035, Japan.

[14] MathWorks, Documentación de Matlab. https://es.mathworks.com/, Versión 2018b.

Anexo A. Transformada de Fourier localizada (STFT)[13]

Una señal puede analizarse tanto en el dominio temporal como en el frecuencial. En este aspecto, existe un principio de compromiso entre la resolución temporal Δt y la resolución en frecuencia Δf. Si Δt es pequeño y la respuesta temporal mejora, Δf es mayor y la respuesta frecuencial presenta peor comportamiento; y viceversa.

La STFT encuentra este problema de resolución temporal vs. frecuencial, pues divide la señal temporal en pequeños segmentos para realizar la FFT (*Fast Fourier Transform*) y pasar al dominio frecuencial. De ahí el nombre de transformada de Fourier localizada.

Esta transformada utiliza una ventana de tamaño fijo y predefinido. Sin embargo, hay otro tipo de transformadas más avanzadas para aplicaciones que requieren un tratamiento más especializado, que utilizan ventanas temporales pequeñas para el análisis de bajas frecuencias (baja resolución frecuencial); y ventanas temporales más pequeñas para el estudio de altas frecuencias, aumentando así la resolución temporal.

De esta manera, en la STFT hay tres parámetros principales que controlan la resolución: el tamaño de la ventana utilizada para dividir la señal en pequeños fragmentos; el desplazamiento de la ventana; y el número de muestras utilizadas para la FFT. Si se denomina por *x(n)* a la señal de entrada, W_n a la ventana, S_n el desplazamiento temporal, y *N* el número de muestras de la FFT, se puede expresar la función de muestreo temporal de la STFT por la Ecuación A.1.:

$$x(n) \otimes W_n (n-k \cdot S_n) \qquad \text{(A.1.)}$$

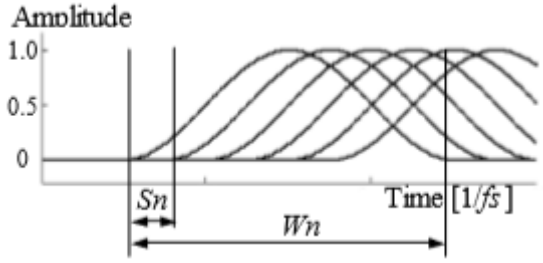


Figura A.1. Parámetros de la STFT

Que, tras calcular la transformada de Fourier, toma la expresión que se muestra en la Ecuación A.2.:

$$X(k,i\omega)=\left(\sum_{n=-\infty}^{+\infty} x(n)\cdot\exp(-jn\omega)\right)\cdot\left(\sum_{n=-\infty}^{+\infty} Wn(n-k\cdot Sn)\cdot\exp(-jn\omega)\right) \qquad \text{(A.2.)}$$

donde el primer término representa el espectro de la señal; y el segundo es el equivalente a una función de transferencia que decide la resolución tiempo-frecuencia.

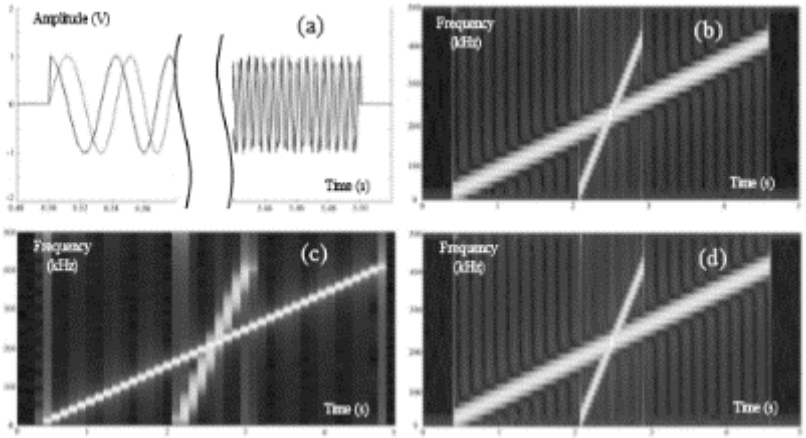


Figura A.2. Trascendencia de los parámetros de la STFT. a) Señal chirp en dominio temporal. b) STFT (N = 32, S_n/N = 1/2). c) STFT (N=128, S_n/N = 1/2). d) STFT (N = 32, S_n/N = 1/16).

Anexo B. Transformada de Hough en Matlab [14]

hough

[H,theta,rho] = hough(BW) computa el estándar Hough Transform (SHT) de la imagen binaria BW. La función hough está diseñada para detectar líneas. La función utiliza la representación paramétrica de una línea: rho = x*cos(theta) + y*sin(theta). La función devuelve rho, la distancia desde el origen a la línea a lo largo de un vector perpendicular a la línea, y theta, el ángulo en grados entre el eje xy este vector. La función también devuelve la transformación Hough estándar, H, que es una matriz espacial de parámetros cuyas filas y columnas corresponden a valores rho y theta respectivamente.

[H,theta,rho] = hough(BW,Name,Value,...) computa el estándar Hough Transform (SHT) de la imagen binaria BW, donde los parámetros nombrados afectan el cómputo.

Argumentos de entrada

'BW': imagen binaria, especificada como una matriz lógica o numérica real, 2-D, no Sparse.

Argumentos de par nombre-valor

'RhoResolution': espaciamiento de los *bin*s de transformación Hough a lo largo del eje rho , especificado como el par separado por comas consistente en 'RhoResolution' y un escalar real, numérico entre 0 y norm(size(BW)), exclusivo.

'Theta': valor Theta para la columna correspondiente de la matriz de salida H, especificada como el par separado por comas consistente en 'Theta' y un vector real, numérico dentro del rango [-90, 90).

Argumentos de salida

'H': matriz de transformación Hough, devuelta como una matriz numérica, nrho x ntheta de tamaño. Las filas y columnas corresponden a valores rho y theta.

'theta': ángulo en grados entre el eje x y el vector rho, devuelto como una matriz numérica de la clase double.

'rho': distancia desde el origen a la línea a lo largo de un vector perpendicular a la línea, devuelto como una matriz numérica de la clase double.

houghpeaks

peaks = houghpeaks(H,numpeaks) localiza los picos en la matriz de transformación Hough, H, generada por la función hough. numpeaks especifica el número máximo de picos a identificar. La función devuelve peaks una matriz que contiene las coordenadas de fila y columna de los picos.

peaks = houghpeaks(___,Name,Value,...) localiza los picos en la matriz de transformación Hough, donde los parámetros denominados controlan los aspectos de la operación.

Argumentos de entrada

'H': matriz de transformación Hough, especificada como una matriz numérica de la clase double. Las filas y columnas corresponden a valores rho y theta.

'numpeaks': número máximo de picos a identificar, especificado como escalar numérico.

Argumentos de par nombre-valor

'Threshold': valor mínimo que se considera un pico, especificado como escalar numérico no negativo. El valor puede ser cualquier valor entre 0 y Inf.

'NHoodSize': tamaño del vecindario de supresión, especificado como un vector de dos elementos positivos impares. El *barrio de supresión* es el vecindario alrededor de cada pico que se establece a cero después de que se identifique el pico.

Argumentos de salida

'peaks': se encontraron coordenadas de fila y columna de picos, devueltos como una matriz Q x 2, donde el valor Q puede variar de 0 a numpeaks.

houghlines

lines = houghlines(BW,theta,rho,peaks). Extrae segmentos de línea en la imagen BW asociada a *bin*s particulares en una transformación Hough. theta y rho son vectores devueltos por función hough. peaks es una matriz devuelta por la función houghpeaks que contiene las coordenadas de fila y columna de los contenedores de transformación Hough que se usarán en la búsqueda de segmentos de línea. El valor devuelto lines es un array de estructura cuya longitud equivale al número de segmentos de línea fusionados encontrados.

lines = houghlines(___,Name,Value,...) Extrae segmentos de línea en la imagen BW, donde los parámetros con nombre afectan a la operación.

Argumentos de entrada

'BW': imagen binaria, especificada como una matriz lógica o numérica real, 2-D, no Sparse.

'theta': ángulo de rotación de línea en radianes, especificado como una matriz lógica o numérica real, 2-D, no Sparse.

'rho': distancia del origen de la coordenada, especificada como una matriz lógica o numérica real, de 2-D, no Sparse. El origen de la coordenada es la esquina superior izquierda de la imagen (0, 0).

'peaks': coordenadas de fila y columna de los contenedores de transformación Hough, especificados como una matriz numérica real, no Sparse.

Argumentos de par nombre-valor

'FillGap': distancia entre dos segmentos de línea asociados con el mismo *bin* de transformación Hough, especificado como un escalar real positivo. Cuando la distancia entre los segmentos de línea es menor que el valor especificado, la función houghlines combina los segmentos de línea en un único segmento de línea.

'MinLength': longitud de línea mínima, especificada como escalar real positivo. houghlines descarta las líneas que son más cortas que el valor especificado.

Argumentos de salida

'lines': líneas encontradas, devueltas como una matriz de estructura cuya longitud equivale al número de segmentos de línea fusionados encontrados. Cada elemento de la matriz de estructura tiene estos campos:

- 'point1': vector de dos elementos [X Y] especificando las coordenadas del punto inicial del segmento de línea.

- 'point2': vector de dos elementos [X Y] especificando las coordenadas del punto final del segmento de línea.

- 'theta': ángulo en grados del *bin* de transformación Hough.

- 'rho': posición del eje rho del *bin* de transformación Hough.

Anexo C. Código Matlab

C.1. Filtrado frecuencial

```
% La variable 'fichero' debe estar previamente asignada con el nombre
% completo, incluyendo la ruta, del fichero que se quiera tratar.

%% -- Definición de constantes y lectura de parámetros del fichero

vluz = 2e8;
Fs = 1.25e8;       % Frecuencia de muestreo

fichero = 'F:\Pruebas\2019 01 CINTRA norte litoral\RAWs\etd08_1800';
infoID = fopen([fichero, '_info.txt'], 'rt');
line = fgetl(infoID);
while ischar(line),
    sep = strfind(line, ':');
    switch line(1:sep-1)
        case 'Muestras por trama'
            % Numero de muestras espaciales
            Ns = sscanf(line(sep+1:end), '%d');
        case 'Frecuencia de repetición de pulso'
            % Frecuencia de generacion de pulsos
            Fp = sscanf(line(sep+1:end), '%d');
        case 'Segundos'
            % Tiempo total de medida (s)
            T = sscanf(line(sep+1:end), '%d');
        case 'F_Diezmado'
            % Factor de diezmado Hw en dimensión espacial
            Ds = sscanf(line(sep+1:end), '%d');
        case 'Anchura de pulso'
            % Anchura de pulso
            Pw = sscanf(line(sep+1:end), '%f');
    end
    line = fgetl(infoID);
end;
fclose(infoID);

Nt = T * Fp;    % Numero de muestras temporales
Ds2 = 1;        % Factor de diezmado Sw en dimensión espacial
Dt = 1;         % Factor de diezmado en dimensión temporal

%% -- Lectura de datos del fichero

 bitsPerSample=14;
 codeZero=bitshift(1,(bitsPerSample - 1))-0.5;
 inputRange_volts =100;
 codeRange=bitshift(1,(bitsPerSample - 1))-0.5;
 bitShift=2;

Ns = Ns / Ds2;              % Numero de muestras espaciales tras
diezmado Sw
Fp = Fp / Dt;               % Frecuencia de pulsos tras diezmado
Nt = Nt / Dt;               % Numero de muestras temporales tras
diezmado
clear A A1 A2 fileID infoID line sep
```

```
%% -- Procesado
distance = (0:Ns-1) * Ds * Ds2 / Fs * vluz / 2;
time = (0:Nt-1) / Fp;
numMuestras=Ns;
numCanales=1;
muestrasIntervalo = numMuestras*numCanales;
muestrasIntervalo_range = numMuestras*numCanales;

T_offset_Bytes = 0;

 pfichero = memmapfile(fichero,...
               'Offset',T_offset_Bytes,...
               'Format','uint16',...
               'Writable',false);
aux0=0;


% Bucle temporal principal
tic
x = zeros(length(distance),100000);
trozo = 1; %Seleccionar que intervalo de muestras coger
tiempo_analiz = 100000; %Seleccionar cuantas muestras coger (Nt si
todas)

for n = tiempo_analiz*(trozo-1)+1:trozo*tiempo_analiz,
    n
    clearvars aux0
    aux0 = pfichero.Data((1+(n-
1)*muestrasIntervalo:muestrasIntervalo_range+(n-
1)*muestrasIntervalo));
    sampleCode = double( bitshift(aux0,-bitShift,'uint16'));

    x(:,n) = inputRange_volts * ((sampleCode - codeZero) / codeRange);

    nn = mod(n-1,Dv)+1;

end


%%
cmin = 40;
cmax = 50;

d = 1500; %Distancia a examinar (m)
punto_d = round(d/0.8); %Índice del vector distance, pues cada índice
son 0.8 m
coord_y = [1 length(x(1,:))];
coord_x = [d d];

t = 20; %Tiempo a examinar (s)
punto_t = t*1000; %Índice del vector time, pues cada índice son 0.001
s
coord_y2 = [t t];
coord_x2 = [1 length(x(:,1))];


figure(1);
imagesc(distance,time(tiempo_analiz*(trozo-1)+1:trozo*tiempo_analiz),
imrotate(x,180));
```

```
set(gca,'Ydir','normal')
hold on
plot(coord_x2, coord_y2,'r')
xlabel('distance (m)');
ylabel('time (s)');
%title([fichero ' - Original'], 'Interpreter', 'none');
colorbar
caxis([cmin cmax])

figure
plot(distance,x(:,punto_t))
xlabel('distance (km)'), ylabel ('amplitude')



%% FILTRADO FRECUENCIAL
NFFT = 1024; %Para conseguir resolución de 1 Hz
frec_pulso = 1000;
f = linspace(0,frec_pulso,NFFT);
f_fftshift = linspace(-frec_pulso/2,frec_pulso/2,NFFT);


N = NFFT; %Ventana temporal de N muestras
M = round(N/2); %Solape del 50% para la STFT
X_f_total = zeros(N,round(tiempo_analiz/M),length(x(:,1)));


%Cálculo de la STFT
for punto_d = 1:length(X_f_total(1,1,:))
    %punto_d
    X_f_total(:,:,punto_d) = buffer(x(punto_d,:),N,N-M,'nodelay');
%Matriz NxL con N tamaño de ventana, L numero de ventanas
    X_f_total(:,:,punto_d) =
X_f_total(:,:,punto_d).*repmat(hamming(length(X_f_total(:,1,1))),1,len
gth(X_f_total(1,:,1)));
    X_f_total(:,:,punto_d) = abs(fft(X_f_total(:,:,punto_d),NFFT,1));
%Con matrices, se hace la fft a cada columna
end


%Espectrograma
figure
spectrogram(x(punto_d,:),hamming(N),M,N,1000), ylabel('time (min)')


%Filtrado eliminando las frecuencias por debajo de f_min, las de por
encima
%de f_max
f_min = 20; %(Hz)
f_max = 70; %(Hz)

X_f_total(1:f_min-1,:,:) = 0;
X_f_total(length(X_f_total(:,1,1)):-1:length(X_f_total(:,1,1))-
f_min+1,:,:) = 0;
X_f_total(f_max+1:length(X_f_total(:,1,1))-f_max-1,:,:) = 0;


%Cálculo de la energía por bandas
```

```
    %Filtros de ancho N_f Hz
N_f = f_max - f_min;

E_ff_total = zeros(numMuestras,length(X_f_total(1,:,1)),ceil((f_max-
f_min)/N_f));
E_ff_total_umb =
zeros(numMuestras,length(X_f_total(1,:,1)),ceil((f_max-f_min)/N_f));
umbral = zeros(1,numMuestras);

%Cálculo de la energía por bandas
for j = 1:length(E_ff_total(1,1,:)) %Banda de N_f Hz examinada
    j
    for w = 1:length(E_ff_total(1,:,1)) %Numero ventana
        for  k = 1:numMuestras %Punto distancia
            E_ff_total(k,w,j) = sum(X_f_total(f_min+(j-
1)*N_f:f_min+j*N_f-1,w,k).^2)/N_f;
            umbral(k) =
sum(E_ff_total(k,:,:))/length(E_ff_total(1,:,1)); %umbral: promedio
energía en un punto d determinado
            if (E_ff_total(k,w,j)<1.5*umbral(k))
                E_ff_total_umb(k,w,j) = 0;
            else
                E_ff_total_umb(k,w,j) = 1;
            end
        end
    end
end


%Waterfall de energías
time_aux = linspace(0,100,length(X_f_total(1,:,1)));
for q = 1:ceil((f_max-f_min)/N_f)
    figure(10+q);
    imagesc(distance, time_aux, imrotate(E_ff_total(:,1:end-
1,q),180));
    set(gca,'Ydir','normal')
    ylabel('time (s)');
    xlabel('distance (m)');
    str = sprintf('Waterfall con filtro paso banda de %d a %d',
f_min+(q-1)*N_f, f_min+q*N_f);
    %title(str);
    colorbar

    figure(20+q);
    imagesc(distance, time_aux, imrotate(E_ff_total_umb(:,1:end-
1,q),180));
    set(gca,'Ydir','normal')
    ylabel('time (s)');
    xlabel('distance (m)');
    str = sprintf('Waterfall umbralizado con filtro paso banda de %d a
%d', f_min+(q-1)*N_f, f_min+q*N_f);
    %title(str);
    colorbar
    caxis([0 1])
end
```

C.2. Algoritmos para automatización de detección de eventos

```
%El waterfall que se quiera analizar debe estar en el directorio
%que se indica en la variable 'path'

%%
clear all; close all; clc;
path = 'D:\TFG\Datos Das Directa Cintra\20190205\08-12';
addpath(path);
listing = dir(path);
for i = 3:length(listing) %El 1 y el 2 son . ..
    names{i-2} = listing(i).name;
    names_sp{i-2}= strsplit(names{i-2},{'_','.'});
end
%% Algoritmo
%names_sp{i}{1} -> Fecha
%names_sp{i}{2} -> Distancia inicio fibra (m)
%names_sp{i}{3} -> Distancia fin fibra (m)
%names_sp{i}{4} -> Tiempo absoluto (ms)
%names_sp{i}{5} -> Das007
%names_sp{i}{7} -> Extension (.BMP)

dist_fibra = str2num(names_sp{1}{3})-str2num(names_sp{1}{2}); %
Distancia total de la fibra (m)

emp = 1; % En qué loncha de todas las que hay en 'path' empieza

j = emp;
num_lonchas = 5; % Número de lonchas que se van a superponer


while (j<size(names,2))

    % Escoger loncha(s)
        filename1 = names{j};
        filename2 = names{j+1};
        filename3 = names{j+2};
        filename4 = names{j+3};
        filename5 = names{j+4};
        tic = str2num(names_sp{j}{4});
        tic2 = str2num(names_sp{j+1}{4});
        tiempo = (tic2-tic)*10^-7; % Tiempo duración loncha (s)


        %Tiempo de inicio de la primera loncha
        fecha1 = names_sp{j}{1};
        anyo1 = str2num(fecha1(1:4));
        mes1 = str2num(fecha1(5:6));
        dia1 = str2num(fecha1(7:8));
        hora1 = str2num(fecha1(9:10));
        minutos1 = str2num(fecha1(11:12));

        date1 = datetime(anyo1,mes1,dia1);
        date1 = date1 + hours(hora1);
        date1 = date1 + minutes(minutos1);


        %Tiempo de inicio de la última loncha
```

```
        fecha2 = names_sp{j+num_lonchas}{1};
        anyo2 = str2num(fecha2(1:4));
        mes2 = str2num(fecha2(5:6));
        dia2 = str2num(fecha2(7:8));
        hora2 = str2num(fecha2(9:10));
        minutos2 = str2num(fecha2(11:12));

        date2 = datetime(anyo2,mes2,dia2);
        date2 = date2 + hours(hora2);
        date2 = date2 + minutes(minutos2);

        %Tiempo de finalización de la primera loncha
        fecha3 = names_sp{j+1}{1};
        anyo3 = str2num(fecha3(1:4));
        mes3 = str2num(fecha3(5:6));
        dia3 = str2num(fecha3(7:8));
        hora3 = str2num(fecha3(9:10));
        minutos3 = str2num(fecha3(11:12));

        date3 = datetime(anyo3,mes3,dia3);
        date3 = date3 + hours(hora3);
        date3 = date3 + minutes(minutos3+1);


    % Leer imágenes y binarizarlas
        imagen1 = imread(filename1);
        imagen2 = imread(filename2);
        imagen3 = imread(filename3);
        imagen4 = imread(filename4);
        imagen5 = imread(filename5);
        imagen_total = [imagen5; imagen4; imagen3; imagen2; imagen1];
        imagenbyn1 = abyn(imagen1);
        imagenbyn2 = abyn(imagen2);
        imagenbyn3 = abyn(imagen3);
        imagenbyn4 = abyn(imagen4);
        imagenbyn5 = abyn(imagen5);


    % Diezmado
        M = 7; % Factor dimension vertical
        N = M; % Factor dimension horizontal, para no perder relación
de aspecto
        umb = M; % Umbral
        imagen_diezm1 = diezmado(imagenbyn1, N, M, umb);
        imagen_diezm2 = diezmado(imagenbyn2, N, M, umb);
        imagen_diezm3 = diezmado(imagenbyn3, N, M, umb);
        imagen_diezm4 = diezmado(imagenbyn4, N, M, umb);
        imagen_diezm5 = diezmado(imagenbyn5, N, M, umb);


     % Figure 1: imagen original, imagen blanco y negro, imagen
diezmada
        figure
        subplot(311)
        imshow(imagen1)
        subplot(312)
        imshow(imagenbyn1)
        subplot(313)
        imshow(imagen_diezm1)
```

```
     % Figure 2: representación del cálculo de bordes con edge en
imagen
     % blanco y negro, imagen diezmada
%         BW1 = edge(imagenbyn);
%         BW2 = edge(imagen_diezm);
%         figure
%         subplot(211)
%         imshow(BW1)
%         subplot(212)
%         imshow(BW2)


     % Cálculo transformada de Hough
        segmentos1 = transform_hough(imagen_diezm1);
        segmentos2 = transform_hough(imagen_diezm2);
        segmentos3 = transform_hough(imagen_diezm3);
        segmentos4 = transform_hough(imagen_diezm4);
        segmentos5 = transform_hough(imagen_diezm5);


     % Figure 3: representación de los segmentos devueltos por la
transformada de Hough
%         figure
%         subplot(211)
%         imshow(imagenbyn1)
%         subplot(212)
%         imshow(imagen_diezm1)
%         xlabel('distance (km)')
%         hold on
%         for k = 1:length(segmentos1)
%             xy = [segmentos1(k).point1; segmentos1(k).point2];
%             plot(xy(:,1),xy(:,2), 'LineWidth', 1, 'Color', 'red')
%         end
%         dim = [0.5 0.8 0.1 0.1];
%
annotation('textbox',dim,'EdgeColor','none','String',[char(date3)],'Po
sition', [0.055 0.21 0.1 0.1],'FontSize',7);
%
annotation('textbox',dim,'EdgeColor','none','String',[char(date1)],'Po
sition', [0.055 0.185 0.1 0.1],'FontSize',7);
%
annotation('textbox',dim,'EdgeColor','none','String',num2str('0'),'Pos
ition', [0.124 0.17 0.1 0.1],'FontSize',10);
%
annotation('textbox',dim,'EdgeColor','none','String',num2str(dist_fibr
a/1000),'Position', [0.88 0.17 0.1 0.1],'FontSize',10);
%
annotation('textbox',dim,'EdgeColor','none','String',num2str(dist_fibr
a/2000),'Position', [0.502 0.17 0.1 0.1],'FontSize',10);
%
annotation('textbox',dim,'EdgeColor','none','String',num2str(dist_fibr
a/4000),'Position', [0.313 0.17 0.1 0.1],'FontSize',10);
%
annotation('textbox',dim,'EdgeColor','none','String',num2str(3*dist_fi
bra/4000),'Position', [0.691 0.17 0.1 0.1],'FontSize',10);
%
annotation('textbox',dim,'EdgeColor','none','String',num2str(dist_fibr
a/8000),'Position', [0.2185 0.17 0.1 0.1],'FontSize',10);
```

```
%
annotation('textbox',dim,'EdgeColor','none','String',num2str(3*dist_fi
bra/8000),'Position', [0.4075 0.17 0.1 0.1],'FontSize',10);
%
annotation('textbox',dim,'EdgeColor','none','String',num2str(5*dist_fi
bra/8000),'Position', [0.5965 0.17 0.1 0.1],'FontSize',10);
%
annotation('textbox',dim,'EdgeColor','none','String',num2str(7*dist_fi
bra/8000),'Position', [0.7855 0.17 0.1 0.1],'FontSize',10);


     % Procesado para obtener las líneas en un único sentido
        K =
3.6*(dist_fibra/size(imagenbyn1,2))*(size(imagenbyn1,1)/tiempo);
%constante de conversión de pixels a km/h
        [punto_ini1, punto_fin1] =
calcula(K,segmentos1,imagen_diezm1);
        [punto_ini2, punto_fin2] =
calcula(K,segmentos2,imagen_diezm2);
        [punto_ini3, punto_fin3] =
calcula(K,segmentos3,imagen_diezm3);
        [punto_ini4, punto_fin4] =
calcula(K,segmentos4,imagen_diezm4);
        [punto_ini5, punto_fin5] =
calcula(K,segmentos5,imagen_diezm5);


     % Figure 4: representación de las líneas procesadas
%         figure
%         subplot(211)
%         imshow(imagen_diezm1),title('Lines')
%         hold on
%         for k = 1:length(segmentos1)
%             xy = [segmentos1(k).point1; segmentos1(k).point2];
%             plot(xy(:,1),xy(:,2), 'LineWidth', 1, 'Color', 'red')
%         end
%
%         subplot(212)
%         imshow(imagen_diezm1),title('After processing')
%         hold on
%         for k = 1:size(punto_ini1,1)
%             xy = [punto_ini1(k,:); punto_fin1(k,:)];
%             plot(xy(:,1),xy(:,2), 'LineWidth', 1, 'Color', 'red')
%         end
%         dim = [0.5 0.8 0.1 0.1];
%
annotation('textbox',dim,'EdgeColor','none','String',[char(date3)],'Po
sition', [0.055 0.21 0.1 0.1],'FontSize',7);
%
annotation('textbox',dim,'EdgeColor','none','String',[char(date1)],'Po
sition', [0.055 0.185 0.1 0.1],'FontSize',7);
%
annotation('textbox',dim,'EdgeColor','none','String',num2str('0'),'Pos
ition', [0.124 0.17 0.1 0.1],'FontSize',10);
%
annotation('textbox',dim,'EdgeColor','none','String',num2str(dist_fibr
a/1000),'Position', [0.88 0.17 0.1 0.1],'FontSize',10);
%
annotation('textbox',dim,'EdgeColor','none','String',num2str(dist_fibr
a/2000),'Position', [0.502 0.17 0.1 0.1],'FontSize',10);
```

```
%
annotation('textbox',dim,'EdgeColor','none','String',num2str(dist_fibr
a/4000),'Position', [0.313 0.17 0.1 0.1],'FontSize',10);
%
annotation('textbox',dim,'EdgeColor','none','String',num2str(3*dist_fi
bra/4000),'Position', [0.691 0.17 0.1 0.1],'FontSize',10);
%
annotation('textbox',dim,'EdgeColor','none','String',num2str(dist_fibr
a/8000),'Position', [0.2185 0.17 0.1 0.1],'FontSize',10);
%
annotation('textbox',dim,'EdgeColor','none','String',num2str(3*dist_fi
bra/8000),'Position', [0.4075 0.17 0.1 0.1],'FontSize',10);
%
annotation('textbox',dim,'EdgeColor','none','String',num2str(5*dist_fi
bra/8000),'Position', [0.5965 0.17 0.1 0.1],'FontSize',10);
%
annotation('textbox',dim,'EdgeColor','none','String',num2str(7*dist_fi
bra/8000),'Position', [0.7855 0.17 0.1 0.1],'FontSize',10);


     % Procesado para obtener las líneas en los dos sentidos
        [punto_ini_alt1, punto_fin_alt1] =
calcula_dos_sentidos(K,segmentos1,imagen_diezm1);
        [punto_ini_alt2, punto_fin_alt2] =
calcula_dos_sentidos(K,segmentos2,imagen_diezm2);
        [punto_ini_alt3, punto_fin_alt3] =
calcula_dos_sentidos(K,segmentos3,imagen_diezm3);
        [punto_ini_alt4, punto_fin_alt4] =
calcula_dos_sentidos(K,segmentos4,imagen_diezm4);
        [punto_ini_alt5, punto_fin_alt5] =
calcula_dos_sentidos(K,segmentos5,imagen_diezm5);


%      % Figure 5: representación líneas en dos sentidos
%         figure
%         imshow(imagen_diezm)
%         hold on
%         for k = 1:length(punto_ini_alt)
%             xy = [punto_ini_alt(k,:); punto_fin_alt(k,:)];
%             plot(xy(:,1),xy(:,2), 'LineWidth', 1, 'Color', 'red')
%         end
%         dim = [0.5 0.8 0.1 0.1];
%
annotation('textbox',dim,'EdgeColor','none','String',[char(date3)],'Po
sition', [0.055 0.21 0.1 0.1],'FontSize',7);
%
annotation('textbox',dim,'EdgeColor','none','String',[char(date1)],'Po
sition', [0.055 0.185 0.1 0.1],'FontSize',7);
%
annotation('textbox',dim,'EdgeColor','none','String',num2str('0'),'Pos
ition', [0.124 0.17 0.1 0.1],'FontSize',10);
%
annotation('textbox',dim,'EdgeColor','none','String',num2str(dist_fibr
a/1000),'Position', [0.88 0.17 0.1 0.1],'FontSize',10);
%
annotation('textbox',dim,'EdgeColor','none','String',num2str(dist_fibr
a/2000),'Position', [0.502 0.17 0.1 0.1],'FontSize',10);
%
annotation('textbox',dim,'EdgeColor','none','String',num2str(dist_fibr
a/4000),'Position', [0.313 0.17 0.1 0.1],'FontSize',10);
```

```
%
annotation('textbox',dim,'EdgeColor','none','String',num2str(3*dist_fi
bra/4000),'Position', [0.691 0.17 0.1 0.1],'FontSize',10);
%
annotation('textbox',dim,'EdgeColor','none','String',num2str(dist_fibr
a/8000),'Position', [0.2185 0.17 0.1 0.1],'FontSize',10);
%
annotation('textbox',dim,'EdgeColor','none','String',num2str(3*dist_fi
bra/8000),'Position', [0.4075 0.17 0.1 0.1],'FontSize',10);
%
annotation('textbox',dim,'EdgeColor','none','String',num2str(5*dist_fi
bra/8000),'Position', [0.5965 0.17 0.1 0.1],'FontSize',10);
%
annotation('textbox',dim,'EdgeColor','none','String',num2str(7*dist_fi
bra/8000),'Position', [0.7855 0.17 0.1 0.1],'FontSize',10);


     % Cálculo de velocidades en tramos de dur_tramo kms en los
primeros
     % dist_max m
        dur_tramo = 3; %km
        dist_max = 24000; %m
        pixels_explorados = ceil(size(imagen_diezm1,2) * (dist_max) /
dist_fibra); %pixels
        R = pixels_explorados / ((dist_max)/1000); %pixels/km
        L = ceil(dur_tramo*R); %píxels/tramo
        num_tramos = ceil((dist_max/1000)/dur_tramo);

      % Descarte de puntos más allá de los dist_max m
        punto_fin1 = punto_fin1(punto_ini1(:,1)<pixels_explorados);
        punto_ini1 = punto_ini1(punto_ini1(:,1)<pixels_explorados);
        punto_fin2 = punto_fin2(punto_ini2(:,1)<pixels_explorados);
        punto_ini2 = punto_ini2(punto_ini2(:,1)<pixels_explorados);
        punto_fin3 = punto_fin3(punto_ini3(:,1)<pixels_explorados);
        punto_ini3 = punto_ini3(punto_ini3(:,1)<pixels_explorados);
        punto_fin4 = punto_fin4(punto_ini4(:,1)<pixels_explorados);
        punto_ini4 = punto_ini4(punto_ini4(:,1)<pixels_explorados);
        punto_fin5 = punto_fin5(punto_ini5(:,1)<pixels_explorados);
        punto_ini5 = punto_ini5(punto_ini5(:,1)<pixels_explorados);

      % Cálculo de velocidades de cada uno de los segmentos
        vel1 = K .* (punto_fin1(:,1)-punto_ini1(:,1)) ./
size(imagen_diezm1,1);
        vel2 = K .* (punto_fin2(:,1)-punto_ini2(:,1)) ./
size(imagen_diezm2,1);
        vel3 = K .* (punto_fin3(:,1)-punto_ini3(:,1)) ./
size(imagen_diezm3,1);
        vel4 = K .* (punto_fin4(:,1)-punto_ini4(:,1)) ./
size(imagen_diezm4,1);
        vel5 = K .* (punto_fin5(:,1)-punto_ini5(:,1)) ./
size(imagen_diezm5,1);

      % Cálculo de velocidades medias por tramos
        vel_media1 = velocidades(L, num_tramos, punto_ini1, vel1);
        vel_media2 = velocidades(L, num_tramos, punto_ini2, vel2);
        vel_media3 = velocidades(L, num_tramos, punto_ini3, vel3);
        vel_media4 = velocidades(L, num_tramos, punto_ini4, vel4);
        vel_media5 = velocidades(L, num_tramos, punto_ini5, vel5);
```

```
        vel_media_total = [vel_media1 vel_media2 vel_media3 vel_media4
vel_media5];
        vel_media = mean(vel_media_total,2);

        vel_media_min = 40;
        vel_media_max = 150;


    % Figure 6: representación diagrama de barras de las velocidades
    % por tramos. Azul -> entre el mínimo y el máximo. Rojo -> menor
que
    % el mínimo o mayor que el máximo

        ind = find((vel_media < vel_media_min | vel_media >
vel_media_max) & vel_media ~= 0);

        figure('units','normalized','outerposition',[0 0 1 1])
        subplot (211)
        for i = 1:num_tramos
            if (~isempty(find(i==ind(:))))
                b = bar((i-1)*dur_tramo + dur_tramo/2,vel_media(i),
'FaceColor', 'red', 'BarWidth', dur_tramo);
            else
                b = bar((i-1)*dur_tramo + dur_tramo/2,vel_media(i),
'FaceColor', 'blue', 'BarWidth', dur_tramo);
            end
            hold on
        end

        axis([0 dist_max/1000 0 vel_media_max])
        set(gca,'XTick',[dur_tramo/2:dur_tramo:dist_max/1000-
dur_tramo/2])
        xlabel ('distance (km)')
        ylabel ('speed (km/h)')

        subplot(212)
        imshow(imagen_total(:,1:pixels_explorados*M,:))
        xlabel('distance (km)')
        dim = [0.5 0.8 0.1 0.1];

annotation('textbox',dim,'EdgeColor','none','String',[char(date2)],'Po
sition', [0.03 0.352 0.1 0.1],'FontSize',7);

annotation('textbox',dim,'EdgeColor','none','String',[char(date1)],'Po
sition', [0.03 0.048 0.1 0.1],'FontSize',7);

annotation('textbox',dim,'EdgeColor','none','String',num2str('0'),'Pos
ition', [0.124 0.02 0.1 0.1],'FontSize',10);

annotation('textbox',dim,'EdgeColor','none','String',num2str(dist_max/
1000),'Position', [0.89 0.02 0.1 0.1],'FontSize',10);

annotation('textbox',dim,'EdgeColor','none','String',num2str(dist_max/
2000),'Position', [0.507 0.02 0.1 0.1],'FontSize',10);

annotation('textbox',dim,'EdgeColor','none','String',num2str(dist_max/
4000),'Position', [0.3155 0.02 0.1 0.1],'FontSize',10);

annotation('textbox',dim,'EdgeColor','none','String',num2str(3*dist_ma
x/4000),'Position', [0.6985 0.02 0.1 0.1],'FontSize',10);
```

```
annotation('textbox',dim,'EdgeColor','none','String',num2str(dist_max/
8000),'Position', [0.21975 0.02 0.1 0.1],'FontSize',10);

annotation('textbox',dim,'EdgeColor','none','String',num2str(3*dist_ma
x/8000),'Position', [0.41125 0.02 0.1 0.1],'FontSize',10);

annotation('textbox',dim,'EdgeColor','none','String',num2str(5*dist_ma
x/8000),'Position', [0.60275 0.02 0.1 0.1],'FontSize',10);

annotation('textbox',dim,'EdgeColor','none','String',num2str(7*dist_ma
x/8000),'Position', [0.79425 0.02 0.1 0.1],'FontSize',10);


    % Cálculo del número medio de vehículos en toda la loncha y
creación de textbox en la Figure 6

        %imagen_diezm = [imagen_diezm5; imagen_diezm4; imagen_diezm3;
imagen_diezm2; imagen_diezm1];
        %num_vehiculos = contar_vehiculos_totales(imagen_diezm);
        %dim = [0.5 0.8 0.1 0.1];
        %annotation('textbox',dim,'Color','red','String',['Average
number of vehicles = ', num2str(num_vehiculos)]);


    % Cálculo del número de vehículos por tramo, en tramos de
dist_tramo
    % km
        imagen_diezm = [imagen_diezm5; imagen_diezm4; imagen_diezm3;
imagen_diezm2; imagen_diezm1];
        dist_tramo = 0.5;%km
        dist_max = 24000; %m
        pix_totales = round(size(imagen_diezm,2) * (dist_max) /
dist_fibra); %solo nos fijamos en los primeros 24 km 25 km
        R = (size(imagen_diezm,2)/(dist_fibra/1000)); %pixels/km
        pix_tramo = round(dist_tramo*R); %píxels por tramo
        num_tramos = round((dist_max/1000)/dist_tramo);

        vehiculos_tramo = contar_vehiculos_tramo(pix_tramo,
num_tramos, imagen_diezm)


    % Figure 7: representación número vehículos por tramo (azul)
    % Descomentando, se puede comparar con los vehiculos en la loncha
anterior (verde)
        figure('units','normalized','outerposition',[0 0 1 1])
        subplot (211)
        %if (j == emp)
            for i = 1:num_tramos
                b = bar((i-1)*dist_tramo +
dist_tramo/2,vehiculos_tramo(i), 'FaceColor', 'blue', 'BarWidth',
dist_tramo);
                hold on
            end
        %else
           %for i = 1:num_tramos
            %    b = bar((i-1)*dist_tramo +
dist_tramo/2,vehiculos_tramo(i), 'FaceColor', 'blue', 'BarWidth',
dist_tramo);
            %   hold on
```

```
            %    b = bar((i-1)*dist_tramo +
dist_tramo/2,vehiculos_ant(i), 'FaceColor', 'green', 'BarWidth',
dist_tramo/3);
            %    hold on
           %end
        %end

        axis([0 dist_max/1000 0 20])
        set(gca,'XTick',[dist_tramo/2:dist_tramo:dist_max/1000-
dist_tramo/2])
        set(gca,'XTickLabelRotation',90)
        xlabel ('distance (km)')
        ylabel ('number of vehicles')
        %legend('Vehicles current frame','Vehicles previous frame')

        subplot(212)
        imshow(imagen_total(:,1:pix_totales*M,:))
        xlabel('distance (km)')
        dim = [0.5 0.8 0.1 0.1];

annotation('textbox',dim,'EdgeColor','none','String',[char(date2)],'Po
sition', [0.03 0.352 0.1 0.1],'FontSize',7);

annotation('textbox',dim,'EdgeColor','none','String',[char(date1)],'Po
sition', [0.03 0.048 0.1 0.1],'FontSize',7);

annotation('textbox',dim,'EdgeColor','none','String',num2str('0'),'Pos
ition', [0.124 0.02 0.1 0.1],'FontSize',10);

annotation('textbox',dim,'EdgeColor','none','String',num2str(dist_max/
1000),'Position', [0.89 0.02 0.1 0.1],'FontSize',10);

annotation('textbox',dim,'EdgeColor','none','String',num2str(dist_max/
2000),'Position', [0.507 0.02 0.1 0.1],'FontSize',10);

annotation('textbox',dim,'EdgeColor','none','String',num2str(dist_max/
4000),'Position', [0.3155 0.02 0.1 0.1],'FontSize',10);

annotation('textbox',dim,'EdgeColor','none','String',num2str(3*dist_ma
x/4000),'Position', [0.6985 0.02 0.1 0.1],'FontSize',10);

annotation('textbox',dim,'EdgeColor','none','String',num2str(dist_max/
8000),'Position', [0.21975 0.02 0.1 0.1],'FontSize',10);

annotation('textbox',dim,'EdgeColor','none','String',num2str(3*dist_ma
x/8000),'Position', [0.41125 0.02 0.1 0.1],'FontSize',10);

annotation('textbox',dim,'EdgeColor','none','String',num2str(5*dist_ma
x/8000),'Position', [0.60275 0.02 0.1 0.1],'FontSize',10);

annotation('textbox',dim,'EdgeColor','none','String',num2str(7*dist_ma
x/8000),'Position', [0.79425 0.02 0.1 0.1],'FontSize',10);


    pause(5)
    j = j+num_lonchas;
    %vehiculos_ant = vehiculos_tramo; % Para comparación en la
siguiente
    %iteración
```

```
end



%% Binarización de una imagen

function umb = abyn(original)
    imagen = imagen_bn(original);
    prueba = (1+imagen)/2; % '1's y '0's
    prueba(prueba < 0.2) = 0; % Eliminación ruido
    filt = medfilt2(prueba,[7 7]); % Filtrado de mediana 7x7
    umb = zeros(size(filt));
    umb(filt>=0.25)=1; % Umbralizado
end



%% Código de color de los waterfalls

function imagen_out = imagen_bn(imagen_in)
    imagen_out = zeros(size(imagen_in,1),size(imagen_in,2));
    for i = 1:size(imagen_in,1)
        R = find(imagen_in(i,:,1) == 255); % Pixeles con predominancia
de rojo
        G = find(imagen_in(i,:,2) == 255); % Pixeles con predominancia
de verde
        B = find(imagen_in(i,:,3) == 255); % Pixeles con predominancia
de azul
        imagen_out(i,R) = 1 - double(imagen_in(i,R,2))/510;
        imagen_out(i,G) = 0.45/229.5*(imagen_in(i,G,1)-
imagen_in(i,G,3));
        imagen_out(i,B) = -1 + double(imagen_in(i,B,2))/510;
    end
end



%% Diezmado de los waterfalls

function imagen_diezm = diezmado(imagenbyn, N, M, umb)

%N: Factor de diezmado dimension horizontal
%M: Factor de diezmado dimension vertical
%imagenbyn: Imagen original

x_aux = 1:N:size(imagenbyn,2);
y_aux = 1:M:size(imagenbyn,1);

tam_x = length(x_aux);
tam_y = length(y_aux);

imagen_diezm = zeros(tam_y,tam_x);

for i = 1:tam_x-1
    for j = 1:tam_y-1
        if (sum(sum(imagenbyn(1+(j-1)*M:j*M,1+(i-1)*N:i*N))) > umb)
```

```
            imagen_diezm(j,i) = 1;
        else
            imagen_diezm(j,i) = 0;
        end
    end
end

end
```

```
%% Cálculo de la transformada de Hough de imagen en blanco y negro
% para la obtención de líneas

function lines = transform_hough(BW)

    if (sum(BW(:))>800) % Tráfico denso
        [H,T,R] = hough(BW,'RhoResolution',3,'Theta',10:1:69);
    else % Poco trafico
        [H,T,R] = hough(BW,'RhoResolution',6,'Theta',10:1:69);
    end

    P  = houghpeaks(H,100,'threshold',ceil(0.01*sqrt(max(H(:)))));
    lines = houghlines(BW,T,R,P,'FillGap',6,'MinLength',25);
end
```

```
%% Procesado de las líneas de la transformada de Hough

function [punto_ini, punto_fin] = calcula(K,segmentos,imagen_diezm)

punto1 = [];
punto2 = [];
theta = [];
rho = [];
vel = [];
dist_pixel = [];

punto1_aux = zeros(length(segmentos),2);
punto2_aux = zeros(length(segmentos),2);
theta_aux = zeros(length(segmentos),1);
rho_aux = zeros(length(segmentos),1);

for k = 1:length(segmentos)
    punto1_aux(k,:) = segmentos(k).point1; %Punto inicial del segmento
(x,y)
    punto2_aux(k,:) = segmentos(k).point2; %Punto final del segmento
(x,y)
    theta_aux(k) = segmentos(k).theta;
    rho_aux(k) = segmentos(k).rho;
end

%ORIGEN DE EJE DE TIEMPOS: ARRIBA
dist_pixel_aux = (punto2_aux(:,1)-punto1_aux(:,1));
tiemp_pixel_aux = (punto1_aux(:,2)-punto2_aux(:,2));
vel_aux = K*dist_pixel_aux./tiemp_pixel_aux;
```

```
%Descarte de segmentos imposibles
    vel_min = 8;
    vel_max = 150;

    punto1 = punto1_aux;
    punto1(vel_aux < vel_min | vel_aux > vel_max, :) = [];
    punto2 = punto2_aux;
    punto2(vel_aux < vel_min | vel_aux > vel_max, :) = [];
    dist_pixel = dist_pixel_aux;
    dist_pixel(vel_aux < vel_min | vel_aux > vel_max) = [];
    tiemp_pixel = tiemp_pixel_aux;
    tiemp_pixel(vel_aux < vel_min | vel_aux > vel_max) = [];
    theta = theta_aux;
    theta(vel_aux < vel_min | vel_aux > vel_max) = [];
    vel = vel_aux;
    vel(vel_aux < vel_min | vel_aux > vel_max) = [];


punto_ini_aux = punto1;
punto_fin_aux = punto2;

% Aproximación: las líneas encontradas recorren toda la loncha, de
% principio a fin en tiempo
m = tiemp_pixel./dist_pixel;
x1 = punto_ini_aux(:,1);
y1 = punto_ini_aux(:,2);

% x_ini = ceil(x1 - y1.*tand(theta));
% x_fin = ceil(x1 + (size(imagen_diezm,1)-y1).*tand(theta));
x_ini = ceil(x1 - (size(imagen_diezm,1)-y1)./m);
x_fin = ceil(x1 + y1./m);



% Ordenación de los puntos de izquierda a derecha según donde empiezan
[x_ini_ord,index] = sort(x_ini(:,1));
x_fin_ord = zeros(length(x_fin),1);

for i = 1:size(x_ini,1)
    x_fin_ord(i) = x_fin(index(i),1);
end


% Eliminar segmentos del mismo vehículo
if (sum(imagen_diezm(:))>800) % Tráfico denso
    dist_min_if = 3; % Distancia mínima entre inicio/fin de segmentos
para considerar que se trata de vehículos distintos
else % Poco tráfico
    dist_min_if = 6;
end

divisor = 2;
multiplicador = 1;
i = 1;
while (i < length(x_ini_ord))

    dist_ini = abs(x_ini_ord(i)-x_ini_ord(i+1));
    dist_fin = abs(x_fin_ord(i)-x_fin_ord(i+1));
```

```
    if (dist_ini < dist_min_if & dist_fin < dist_min_if)
        x_ini_ord(i) = ceil((multiplicador*x_ini_ord(i) +
x_ini_ord(i+1))/divisor);
        x_fin_ord(i) = ceil((multiplicador*x_fin_ord(i) +
x_fin_ord(i+1))/divisor);
        divisor = divisor + 1;
        multiplicador = multiplicador + 1;

        x_ini_ord(i+1) = [];
        x_fin_ord(i+1) = [];
    else
        i = i+1;
        divisor = 2;
        multiplicador = 1;
    end
end

% Descarte de segmentos que empiezan en un vehículo y acaban en otro
% distinto
dist_min_dist = 12;
i = 2;
while (i<length(x_ini_ord))
    dist_ini_ant = abs(x_ini_ord(i)-x_ini_ord(i-1));
    dist_fin_ant = abs(x_fin_ord(i)-x_fin_ord(i-1));
    dist_ini_sig = abs(x_ini_ord(i)-x_ini_ord(i+1));
    dist_fin_sig = abs(x_fin_ord(i)-x_fin_ord(i+1));

    if ((dist_ini_ant < dist_min_dist & dist_fin_sig < dist_min_dist)
| (dist_ini_sig < dist_min_dist & dist_fin_ant < dist_min_dist))
        x_ini_ord(i) = [];
        x_fin_ord(i) = [];
    else
        i = i+1;
    end
end

x_ini_definitivo = x_ini_ord;
x_fin_definitivo = x_fin_ord;

% Ultima corrección para eliminar imposibles
dist_pixel_definitivo = x_fin_definitivo - x_ini_definitivo;
tiemp_pixel_definitivo = size(imagen_diezm,1);
vel_definitivo = K*dist_pixel_definitivo./tiemp_pixel_definitivo;
x_ini_definitivo(vel_definitivo < vel_min | vel_definitivo > vel_max)
= [];
x_ini_definitivo(x_ini_definitivo < 0) = 1;
x_fin_definitivo(vel_definitivo < vel_min | vel_definitivo > vel_max)
= [];

%EL EJE DE LAS ORDENADAS VA HACIA ABAJO

y_ini = zeros(length(x_ini_definitivo),1);
y_fin = zeros(length(x_fin_definitivo),1);
y_fin(1:end) = size(imagen_diezm,1);

punto_ini = [x_ini_definitivo y_fin];
punto_fin = [x_fin_definitivo y_ini];
```

```
end


%% Procesado de las líneas en los dos sentidos de la transformada de
Hough

function [punto_ini, punto_fin] =
calcula_dos_sentidos(K,segmentos,imagen_diezm)

punto1 = [];
punto2 = [];
theta = [];
rho = [];
vel = [];
dist_pixel = [];

punto1_aux = zeros(length(segmentos),2);
punto2_aux = zeros(length(segmentos),2);
theta_aux = zeros(length(segmentos),1);
rho_aux = zeros(length(segmentos),1);

for k = 1:length(segmentos)
    punto1_aux(k,:) = segmentos(k).point1; %Punto inicial del segmento
(x,y)
    punto2_aux(k,:) = segmentos(k).point2; %Punto final del segmento
(x,y)
    theta_aux(k) = segmentos(k).theta;
    rho_aux(k) = segmentos(k).rho;
end

%ORIGEN DE EJE DE TIEMPOS: ARRIBA
dist_pixel_aux = (punto2_aux(:,1)-punto1_aux(:,1));
tiemp_pixel_aux = (punto1_aux(:,2)-punto2_aux(:,2));
vel_aux = K*dist_pixel_aux./tiemp_pixel_aux;

%Descarte de segmentos imposibles
    vel_min = 8;
    vel_max = 150;

    punto1 = punto1_aux;
    punto1(abs(vel_aux) < vel_min | abs(vel_aux) > vel_max, :) = [];
    punto2 = punto2_aux;
    punto2(abs(vel_aux) < vel_min | abs(vel_aux) > vel_max, :) = [];
    dist_pixel = dist_pixel_aux;
    dist_pixel(abs(vel_aux) < vel_min | abs(vel_aux) > vel_max) = [];
    tiemp_pixel = tiemp_pixel_aux;
    tiemp_pixel(abs(vel_aux) < vel_min | abs(vel_aux) > vel_max) = [];
    theta = theta_aux;
    theta(abs(vel_aux) < vel_min | abs(vel_aux) > vel_max) = [];
    vel = vel_aux;
    vel(abs(vel_aux) < vel_min | abs(vel_aux) > vel_max) = [];


punto_ini_aux = punto1;
punto_fin_aux = punto2;

% Aproximación: las líneas encontradas recorren toda la loncha, de
% principio a fin en tiempo
```

```
m = tiemp_pixel./dist_pixel;
x1 = punto_ini_aux(:,1);
y1 = punto_ini_aux(:,2);

% x_ini = ceil(x1 - y1.*tand(theta));
% x_fin = ceil(x1 + (size(imagen_diezm,1)-y1).*tand(theta));
x_ini = ceil(x1 - (size(imagen_diezm,1)-y1)./m);
x_fin = ceil(x1 + y1./m);


% Ordenación de los puntos de izquierda a derecha según donde empiezan
[x_ini_ord,index] = sort(x_ini(:,1));
x_fin_ord = zeros(length(x_fin),1);

for i = 1:size(x_ini,1)
    x_fin_ord(i) = x_fin(index(i),1);
end


% Eliminar segmentos del mismo vehículo
if (sum(imagen_diezm(:))>800) % Tráfico denso
    dist_min_if = 3; % Distancia mínima entre inicio/fin de segmentos
para considerar que se trata de vehículos distintos
else % Poco tráfico
    dist_min_if = 6;
end

divisor = 2;
multiplicador = 1;
i = 1;
while (i < length(x_ini_ord))

    dist_ini = abs(x_ini_ord(i)-x_ini_ord(i+1));
    dist_fin = abs(x_fin_ord(i)-x_fin_ord(i+1));

    if (dist_ini < dist_min_if & dist_fin < dist_min_if)
        x_ini_ord(i) = ceil((multiplicador*x_ini_ord(i) +
x_ini_ord(i+1))/divisor);
        x_fin_ord(i) = ceil((multiplicador*x_fin_ord(i) +
x_fin_ord(i+1))/divisor);
        divisor = divisor + 1;
        multiplicador = multiplicador + 1;

        x_ini_ord(i+1) = [];
        x_fin_ord(i+1) = [];
    else
        i = i+1;
        divisor = 2;
        multiplicador = 1;
    end
end

% Descarte de segmentos que empiezan en un vehículo y acaban en otro
dist_min_dist = 12;
i = 2;
while (i<length(x_ini_ord))
    dist_ini_ant = abs(x_ini_ord(i)-x_ini_ord(i-1));
    dist_fin_ant = abs(x_fin_ord(i)-x_fin_ord(i-1));
```

```
    dist_ini_sig = abs(x_ini_ord(i)-x_ini_ord(i+1));
    dist_fin_sig = abs(x_fin_ord(i)-x_fin_ord(i+1));

    if ((dist_ini_ant < dist_min_dist & dist_fin_sig < dist_min_dist)
| (dist_ini_sig < dist_min_dist & dist_fin_ant < dist_min_dist))
        x_ini_ord(i) = [];
        x_fin_ord(i) = [];
    else
        i = i+1;
    end
end

x_ini_definitivo = x_ini_ord;
x_fin_definitivo = x_fin_ord;

% Ultima corrección para eliminar imposibles
dist_pixel_definitivo = x_fin_definitivo - x_ini_definitivo;
tiemp_pixel_definitivo = size(imagen_diezm,1);
vel_definitivo = K*dist_pixel_definitivo./tiemp_pixel_definitivo;
x_ini_definitivo(abs(vel_definitivo) < vel_min | abs(vel_definitivo) >
vel_max) = [];
x_ini_definitivo(x_ini_definitivo < 0) = 1;
x_fin_definitivo(abs(vel_definitivo) < vel_min | abs(vel_definitivo) >
vel_max) = [];

%EL EJE DE LAS ORDENADAS VA HACIA ABAJO

y_ini = zeros(length(x_ini_definitivo),1);
y_fin = zeros(length(x_fin_definitivo),1);
y_fin(1:end) = size(imagen_diezm,1);

punto_ini = [x_ini_definitivo y_fin];
punto_fin = [x_fin_definitivo y_ini];

end




%% Cálculo de velocidades medias por tramo en una loncha

function [vel_media] = velocidades(L,num_tramos,punto_ini,vel)

%L: numero de píxels por tramo
%num_tramos: número de tramos de x km en toda la fibra
%vel: velocidad de cada vehículo detectado

    tramos = zeros(num_tramos,1);
    vel_media = zeros(num_tramos,1);
    ya = 1;

    for i = 1:size(punto_ini,1)
        if (punto_ini(i,1)<1)
            punto_ini(i,1) = 1;
        end
        tramos(ceil(punto_ini(i,1)/L)) =
tramos(ceil(punto_ini(i,1)/L)) + 1;
    end
```

```
    for j = 1:num_tramos
        if (tramos(j) > 0)
            vel_media(j) = sum(vel(ya:ya-1+tramos(j))) / tramos(j);
        end
        ya = ya + tramos(j);
    end
end



%% Cálculo del número de vehículos en una loncha completa

function num_vehiculos = contar_vehiculos_totales(imagen_diezm)

    vehiculos_fila = zeros(1,size(imagen_diezm,1));

    for i = 1:size(imagen_diezm,1) % Recorrer todas las filas
        for j = 1:size(imagen_diezm,2) - 1 % Recorrer la fila entera
            if (imagen_diezm(i,j) == 0 & imagen_diezm(i,j+1) == 1) %
Se contabilizan cambios de negro a blanco
                vehiculos_fila(i) = vehiculos_fila(i) + 1;
            end
        end
    end

    num_vehiculos = round(mean(vehiculos_fila));


end



%% Cálculo del número de vehículos por tramo

function vehiculos_tramo = contar_vehiculos_tramo(pix_tramo,
num_tramos, imagen_diezm)

    vehiculos_columna = zeros(1,size(imagen_diezm,2));
    vehiculos_tramo = zeros(1,num_tramos);

    pix_tramo_aux = pix_tramo;

    for w = 1:num_tramos % Recorrer todos los tramos
        for i = 1:pix_tramo_aux % Recorrer todas las columnas de cada
tramo
            for j = 1:size(imagen_diezm,1)-1 % Recorrer la columna
entera
                if (imagen_diezm(j,(w-1)*pix_tramo_aux+i) == 0 &
imagen_diezm(j+1,(w-1)*pix_tramo_aux+i) == 1) % Se contabilizan
cambios de negro a blanco
                    vehiculos_columna((w-1)*pix_tramo_aux+i) =
vehiculos_columna((w-1)*pix_tramo_aux+i) + 1;
                end
            end
        end
```

```
        vehiculos_tramo(w) = round(mean(vehiculos_columna((w-
1)*pix_tramo_aux+1:w*pix_tramo_aux)));
    end
end
```