Universitat Oberta de Catalunya

Desarrollo de un sistema de detección de obstáculos en vía en un paso a nivel de una red de tranvías

Autor: Javier González de la Cruz

Tutora: Verónica Vilaplana Besler

Profesor: David García Solórzano

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Desarrollo de un sistema de detección de obstáculos en vía en un paso a nivel de una red de tranvías*
Nombre del autor:	*Javier González de la Cruz*
Nombre del colaborador/a docente :	*Verónica Vilaplana Besler*
Nombre del PRA:	*David García Solórzano*
Fecha de entrega:	*01/2020*
Titulación o programa:	*Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación*
Área del Trabajo Final:	*Aplicaciones multimedia basadas en procesamiento de la señal*
Idioma del trabajo:	*Español*
Palabras clave	*Detección obstáculos, tranvía, aplicación*

Resumen

En el presente proyecto se desarrolla un sistema de detección de obstáculos, como pueden ser personas, vehículos, animales u objetos inertes, en un paso a nivel de una red de tranvías en zona urbana. El sistema, a través de una cámara de video fija que enfoca en todo momento al paso a nivel, detecta cualquier obstáculo que invada la vía, y que por tanto pueda suponer un riesgo para la seguridad en la normal circulación del tranvía, y lanza un aviso de riesgo de colisión en caso necesario.

Para la programación del software se utiliza la herramienta de programación *Matlab* y sus librerías de visión por computador y procesado de imagen. La aplicación realiza tareas de adquisición de la imagen de video, preprocesado de esta, delimitación del área de interés (ROI), eliminación de fondo y segmentación basada en movimiento.

El sistema se ha probado con vídeos grabados de las cámaras existentes en la plataforma de vía de Metro Ligero Oeste S.A., con buena iluminación y en condiciones climáticas favorables. Gracias a que la aplicación permite decidir la agrupación de píxeles mínima para detectar un objeto, se han detectado el 100% de los vehículos que pasan por la zona de detección.

El sistema es robusto bajo las condiciones antes citadas y en las grabaciones de prueba disponibles. No obstante, es adaptable a tecnologías más avanzadas de detección y segmentación de objetos, con lo que se puede adecuar en un futuro en función de las necesidades de la red tranviaria.

Abstract

This project is developing a system for detecting obstacles, such as people, vehicles, animals or inert objects, at a level crossing of a tramway network in an urban area. The system, by means of a fixed video camera that focuses at all times on the level crossing, detects any obstacle that invades the track, and that therefore may pose a risk to the safety of the normal tram traffic, and send a collision risk warning if necessary.

For programming the software is used the programming tool Matlab and its libraries of computer vision and image processing. The application performs tasks of video image acquisition, video image preprocessing, region of interest delimitation (ROI), background removal and movement-based segmentation.

The system has been tested with recorded videos of the existing cameras on the track platform of Metro Ligero Oeste S.A., with good lighting and in favorable weather conditions. Because the application allows you to decide the minimum pixel grouping to detect an object, they have been detected 100% of vehicles passing through the detection zone.

The system is robust under the above conditions and in the available test recordings. However, it is adaptable to more advanced detection and object segmentation technologies, which can be adapted in the future to the needs of the road network.

Dedicatoria

A mis padres y hermanos. Juan Carlos, lo hemos conseguido.

A mi mujer Susana y a mis hijos Asier e Ian. Sois mi alegría, mi aire y mi alimento.

Agradecimientos

Gracias a toda la gente que trabaja en Metro Ligero Oeste S.A. por las facilidades y consejos proporcionados para poder llevar a cabo este proyecto. Mis compañeros son gente maravillosa.
No quiero dejar pasar la oportunidad de dar las gracias a mí tutora en la UOC Meritxell Bosch García, por su profesionalidad, dedicación y ayuda todos estos años.

Abstract

This project is developing a system for detecting obstacles, such as people, vehicles, animals or inert objects, at a level crossing of a tramway network in an urban area. The system, by means of a fixed video camera that focuses at all times on the level crossing, detects any obstacle that invades the track, and that therefore may pose a risk to the safety of the normal tram traffic, and send a collision risk warning if necessary.

For programming the software is used the programming tool Matlab and its libraries of computer vision and image processing. The application performs tasks of video image acquisition, video image preprocessing, region of interest delimitation (ROI), background removal and movement-based segmentation.

The system has been tested with recorded videos of the existing cameras on the track platform of Metro Ligero Oeste S.A., with good lighting and in favorable weather conditions. Because the application allows you to decide the minimum pixel grouping to detect an object, they have been detected 100% of vehicles passing through the detection zone.

The system is robust under the above conditions and in the available test recordings. However, it is adaptable to more advanced detection and object segmentation technologies, which can be adapted in the future to the needs of the road network.

Keywords

Obstacles detecting, application, computer visión, system, tramway.

Resumen

En el presente proyecto se desarrolla un sistema de detección de obstáculos, como pueden ser personas, vehículos, animales u objetos inertes, en un paso a nivel de una red de tranvías en zona urbana. El sistema, a través de una cámara de video fija que enfoca en todo momento al paso a nivel, detecta cualquier obstáculo que invada la vía, y que por tanto pueda suponer un riesgo para la seguridad en la normal circulación del tranvía, y lanza un aviso de riesgo de colisión en caso necesario.

Para la programación del software se utiliza la herramienta de programación *Matlab* y sus librerías de visión por computador y procesado de imagen. La aplicación realiza tareas de adquisición de la imagen de video, preprocesado de esta, delimitación del área de interés (ROI), eliminación de fondo y segmentación basada en movimiento.

El sistema se ha probado con vídeos grabados de las cámaras existentes en la plataforma de vía de Metro Ligero Oeste S.A., con buena iluminación y en condiciones climáticas favorables. Gracias a que la aplicación permite decidir la agrupación de píxeles mínima para detectar un objeto, se han detectado el 100% de los vehículos que pasan por la zona de detección.

El sistema es robusto bajo las condiciones antes citadas y en las grabaciones de prueba disponibles. No obstante, es adaptable a tecnologías más avanzadas de detección y segmentación de objetos, con lo que se puede adecuar en un futuro en función de las necesidades de la red tranviaria.

Palabras clave

Detección obstáculos, aplicación, visión computador, sistema, tranvía.

Índice

Figuras y tablas

Índice de figuras

Índice de tablas

1. Introducción

1.1. Interés del proyecto. Justificación.

La visión humana es uno de los sentidos más utilizados y es un mecanismo de procesado de imágenes muy poderoso, capaz de detectar y analizar imágenes de manera muy precisa. Es por esta razón por la que desde la antigüedad ha fascinado a la raza humana y se ha estudiado su funcionamiento, hasta que hemos sido capaces de implementar algoritmos y sistemas capaces de simular, en cierta medida, su comportamiento. La visión artificial intenta emular la capacidad de las personas de ver una escena y entenderla.

La visión artificial es una extensión del procesado de imagen, una disciplina muy importante en la era actual, donde el mundo digital rodea todas las facetas de la vida, y es, además, una rama muy significativa de las Tecnologías de la Información y Comunicación (TIC). Las técnicas de visión artificial o visión por computador son utilizadas en campos tan diversos como la medicina, la seguridad vial, la astronomía, la robótica y muchas otras más. Son disciplinas complejas y en constante desarrollo.

Hoy en día existen innumerables aplicaciones y sistemas que la utilizan de una manera u otra, en el reconocimiento facial o de huellas dactilares, en la detección de obstáculos, en las retransmisiones deportivas, en el reconocimiento de patrones en la industria, etc. Tener unos conocimientos básicos y bien afianzados de algunas de las técnicas que se utilizan en estos campos es altamente recomendable para un ingeniero o un profesional de las TIC. En este proyecto se ponen en práctica muchas de las capacidades adquiridas durante el estudio de este grado en la UOC.

Por otro lado, el uso de este tipo de técnicas para desarrollar las llamadas “Smart Cities” o ciudades inteligentes, está cada vez más extendido. Con ello se consigue una gestión óptima de la ciudad y una alta capacidad de actuación en casos de catástrofes o situaciones de emergencia. Una parte muy importante para el buen funcionamiento de una ciudad es la gestión que se hace, tanto por parte de la administración como de las distintas empresas que puedan estar implicadas, en caso de un accidente en el casco urbano. Poder prevenir este tipo de incidentes es lo más deseable para todos, ya que se evitan daños físicos y materiales.

En el caso de la aplicación de este proyecto, se pretende evitar las colisiones del tranvía de Metro Ligero Oeste S.A. (MLO) [1] con todo aquello que pueda ser susceptible de invadir de alguna forma el carril por el que circula. Una colisión supone que la circulación tranviaria se interrumpa, con el consiguiente perjuicio para los viajeros, sin nombrar los daños ocasionados por esta colisión. Un sistema de visión artificial que se anticipe, en la medida de lo posible, a estas incidencias, supone un

gran adelanto en la seguridad de la circulación y un ahorro considerable para el ayuntamiento de la ciudad y para la empresa explotadora del servicio de transporte público.

Aunque en la actualidad existen numerosos sistemas y aplicaciones relacionadas con la visión artificial, todavía no se ven bastante implementadas en lo relativo al transporte urbano de pasajeros, con lo que este proyecto pretende solucionar en parte el problema de la seguridad y la gestión de este tipo de servicio público, ofreciendo mejores prestaciones al ciudadano.

Figura 1: Colisión vehículo-tranvía en un cruce urbano

1.2. Descripción del sistema

A lo largo de la historia, las ciudades han sufrido constantes cambios en sus infraestructuras, gestión de los recursos y, cómo no, en la movilidad de las personas. Muchas ciudades están apostando por la creación de espacios más verdes, calles peatonales, carriles para ciclistas y transporte público más sostenible y seguro, con el fin de convertirse en espacios más inteligentes y amigables para sus ciudadanos. El tranvía es un medio de transporte que cumple con todas estas expectativas, sin embargo, no existe el riesgo cero y son innumerables los accidentes que ocurren en las ciudades de toda Europa en las que este medio de transporte convive con el resto de vehículos.

La conducción de un tranvía en zonas urbanas se denomina “conducción a la vista”, lo que significa que el conductor debe saber anteponerse a los posibles riesgos que la circulación e interacción, en un mismo espacio, de trenes y otros vehículos, personas u objetos, supone. Debe disminuir la velocidad si se detecta un peligro de colisión o atropello, o incluso detener el tren si es necesario. El conductor tiene control total sobre los mandos del vehículo, sin que exista una conducción automatizada (exceptuando los sistemas de “hombre muerto” y de límite de velocidad). En la explotación de Metro Ligero Oeste existe señalización vial y tranviaria en los pasos a nivel, pero es inevitable que en muchos casos sucedan accidentes, ya sea por descuidos o negligencias de las personas, o por invasión de la vía por

parte de animales u objetos. A su vez, a la entrada de los túneles de los que dispone la línea en diversos puntos, la visibilidad del conductor disminuye y toda información que le pueda llegar del estado de la vía es bien recibida por este

La motivación principal de este proyecto es desarrollar un sistema de visión por computador capaz de detectar aquellos objetos que invadan el paso a nivel por donde circulan los trenes de Metro Ligero Oeste y que lance una señal de aviso al propio tren y al Puesto Central de Control (PCC) cuando esto pueda provocar una incidencia en la circulación, como puede ser cuando las señales estén en prohibición de paso para vehículos y peatones. De esta forma, el sistema servirá de apoyo para los sistemas ya existentes de seguridad en la circulación, evitando, en la medida de lo posible, incidentes que puedan ocasionar retrasos en el horario o daños materiales y personales.

El proyecto consta de una aplicación que detecta los obstáculos en vía de Metro Ligero Oeste o de cualquier explotación tranviaria que disponga de un sistema de videovigilancia en su infraestructura. En esta primera fase del proyecto, la aplicación no estará integrada en dicho sistema de videovigilancia, con lo que las imágenes no serán en tiempo real, sino grabaciones descargadas de la base de datos de Metro Ligero Oeste.

Dicha aplicación es capaz de seguir e implementar de forma correcta los siguientes pasos:

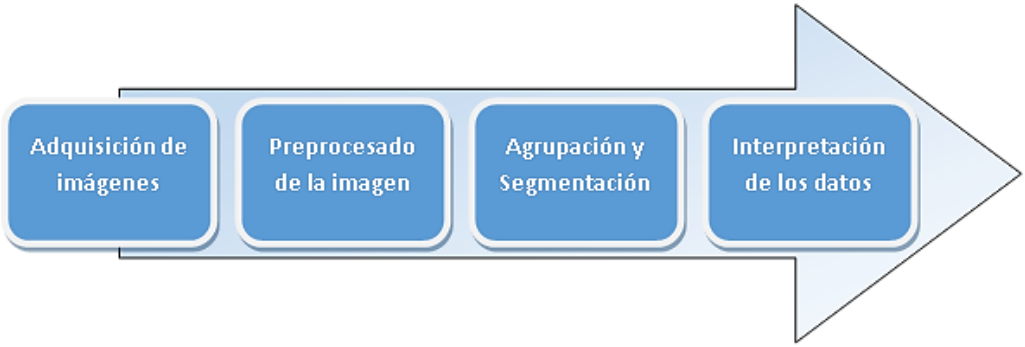

Figura 2: Pasos a implementar por la aplicación a desarrollar

- **Adquisición de imágenes**: En este primer paso se obtienen las imágenes de video que provienen de las cámaras de videovigilancia de la red de tranvías de Metro Ligero Oeste.
- **Preprocesado de la imagen**: Después de captar las imágenes de las distintas cámaras, la aplicación divide la secuencia de video en frames, para poder realizar las técnicas de procesado de imagen necesarias para el buen funcionamiento del sistema, es decir, eliminación del fondo, eliminación de ruido, mejoras de iluminación o saturación si fuera necesario, etc.
- **Agrupación y Segmentación**: En este paso se reagrupan los pixeles resultantes, después del filtrado, para poder segmentarlos. De esta manera se puede estimar que hay un objeto que no pertenece al fondo.

- **Interpretación de los datos obtenidos**: Una vez que se han realizado todas las operaciones sobre cada frame, la aplicación tendrá que evaluar los datos obtenidos, las señales recibidas y transmitir la información necesaria, con el fin de evaluar si un objeto está en movimiento y si este invade la zona de riesgo de colisión con el tranvía en un instante en el que no está permitido.

1.3. Objetivos del proyecto

Como se ha comentado anteriormente, el objetivo principal de este proyecto es obtener un sistema de visión por computador, integrado en el sistema de videovigilancia de la explotación tranviaria, que sea capaz de analizar una secuencia de imágenes procedente de una cámara, detectar aquellos objetos que invadan el paso a nivel por donde circulan los tranvías y que lance una señal de aviso al propio tren y al puesto de control cuando esto pueda provocar una incidencia en la circulación.

Con ello se pretende que:

- La aplicación sirva de apoyo para los sistemas ya existentes de seguridad en la circulación.
- Evitar incidentes que provoquen daños materiales y/o personales.
- Mejorar la seguridad del usuario y del trabajador del tranvía.

Junto con estos objetivos principales, se consiguen otros objetivos secundarios que van estrechamente ligados a estos:

- Evitar retrasos en los horarios de los trenes.
- Ahorrar costes de las instituciones y empresas implicadas.
- Obtener un registro de infracciones de los vehículos en los pasos a nivel.
- Mejorar los sistemas de transporte público actuales.
- Convertir las ciudades en espacios más inteligentes.

Así mismo, a través de los registros de las incidencias detectadas, ocasionen un accidente o no, se pueden obtener estadísticas que sirvan para tomar otras medidas preventivas en caso de ser necesario, como la colocación de badenes y de barreras antes de los pasos a nivel, señalización luminosa vertical y en suelo, etc.

1.4. Metodología y proceso de trabajo

Después de investigar y analizar las distintas librerías de visión artificial existentes, se decide realizar la aplicación con *MATLAB*. Si bien este software no es gratuito, su gran capacidad en procesado de imágenes y la cantidad de libros, seminarios web, tutoriales y ejemplos que existen sobre este lenguaje de programación aplicado a la visión por computador y procesado de imagen, son de gran ayuda para elegir el método de análisis que mejor se adapta al sistema a implementar. Además, se dispone de una versión de estudiante gratuita con la que se harán todas las pruebas hasta la finalización del proyecto.

Después de elegir el software a utilizar, se habla con los responsables de Metro Ligero Oeste para conocer su opinión sobre el proyecto y comprobar si el sistema podría ser útil para esta compañía, así como para que den su aprobación para utilizar grabaciones de vídeo de las cámaras del sistema de videovigilancia existente en sus instalaciones. Una vez obtenido su consentimiento (y para comprender mejor las necesidades que debe cumplir la aplicación) se hizo un estudio de las características de los distintos sistemas que componen la explotación, ya que el objetivo final es que el producto desarrollado esté integrado con todos ellos. La instalación de videovigilancia de Metro Ligero Oeste tiene una arquitectura física que se muestra en la imagen siguiente [3]:

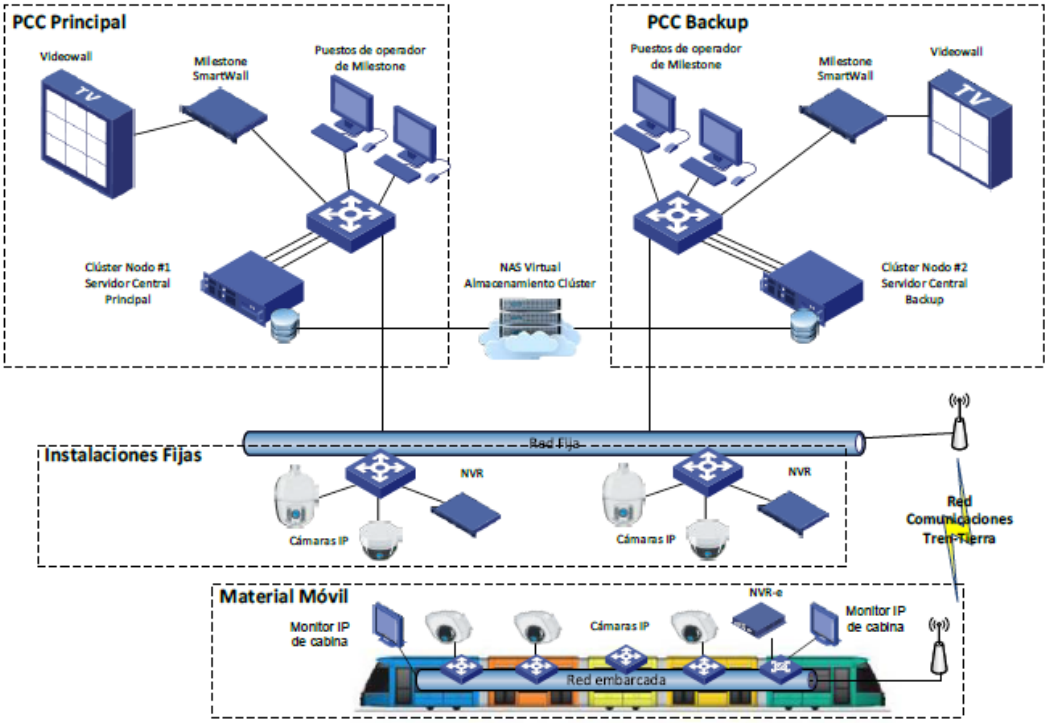

Figura 3: Arquitectura del sistema de videovigilancia de MLO

Donde tenemos los siguientes elementos por ubicación:

PCC Principal y PCC Backup:

- Un servidor central de CCTV para virtualización del software de la plataforma de gestión y operación centralizada de CCTV (Milestone XProtect Corporate y motor de base de datos SQL

Server) para implementación de topología de sistemas redundante basada en clúster de hipervisores (nodo principal).

- Dos puestos de operador con dos monitores para gestión, operación y control de la infraestructura de CCTV en instalaciones fijas y material móvil, donde está instalado el software de gestión y operación centralizada de CCTV (Milestone XProtect Smart Client).
- Dos descodificadores de vídeo para proyección de imágenes de CCTV en el videowall existente, donde está instalado el software cliente de descompresión-visualización (Milestone XProtect Smartwall).

Instalaciones fijas (paradas, cruces y túneles):

- Cámaras Full-HD tipo domo fijo, conectadas a la electrónica de red de la red fija existente y alimentadas mediante inyectores PoE+ externos con tendido de cableado basado en par trenzado de cobre.
- Un grabador de vídeo IP en red, conectado a la electrónica de red de la red fija existente para grabación local de las cámaras del emplazamiento. Poseen dos puertos 1 Gbps BaseTX por servidor, un puerto 10/100 BaseTX por servidor, cuatro salidas de vídeo digital HD y dos conexiones a suministro eléctrico estabilizado (UPS).

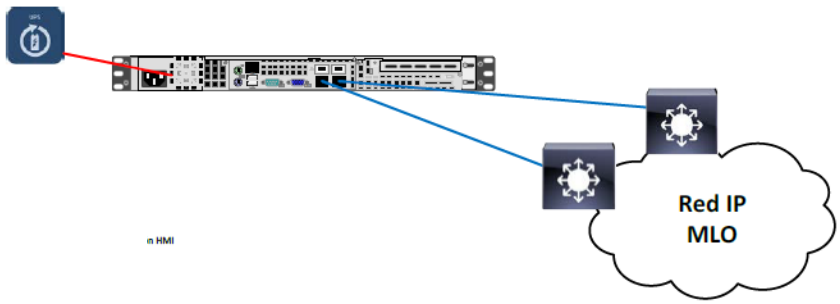

Figura 4: Grabador de video IP

El *switch* de comunicaciones existente en cada parada es el encargado de transmitir la señal de vídeo codificada hasta los Puestos de Control a través de la red existente. En el PCC se dispone de la señal de video y telemando de cada cámara para su integración en el Sistema de Control de Tráfico Distribuido, de modo que el puesto de Operación pueda, en cada momento, gestionar la cámara correspondiente.

La configuración de la parte de Material Móvil (trenes) no se describe, ya que no es de utilidad para este proyecto. Los avisos de alarma que envía la aplicación a los trenes en caso de detectar que un objeto ocupa la vía cuando no está permitido, se hacen a través del Sistema de Ayuda a la Explotación (SAE). Este sistema es el que se encarga de las comunicaciones PCC-tranvía y tanvía-tranvía, entre otras funcionalidades, como los mensajes sonoros de paradas, los teleindicadores, el posicionamiento del tren, distancia entre trenes en servicio, etc.

Por otra parte, está señalización ferroviaria de MLO, la cual está dividida en cinco enclavamientos que controlan la señalización de las líneas 2 y 3 (como pueden ser las balizas de posicionamiento, los circuitos de vía y los semáforos). Operan de forma independiente unos de otros, sin precisar el intercambio de información directa entre ellos, como se aprecia en la siguiente imagen [4]:

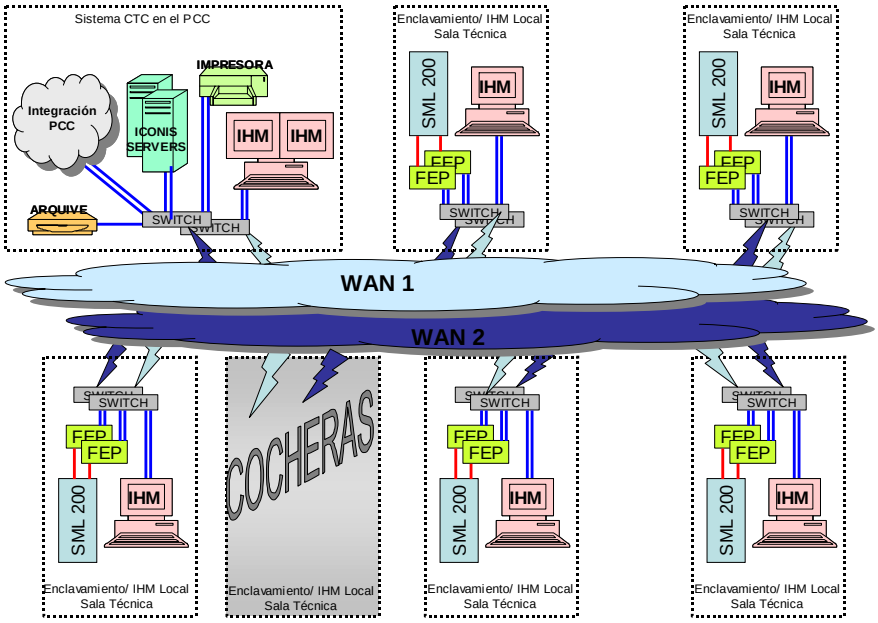

Figura 5: Sistema de enclavamiento de las líneas 2 y 3 de MLO

Cada enclavamiento cuenta de su puesto de mando local enlazado con el enclavamiento electrónico a través de los Gateways de comunicaciones que, trabajando en redundancia, desempeñan la función de servidor de datos, publicando el estado del enclavamiento a los sistemas de mandos locales y remotos de la línea. En cada cabina de enclavamiento, hay una red Ethernet que permite la comunicación entre el mando local, los Gateways locales y los sistemas de mantenimiento de equipos de cabina. Existe un enlace Ethernet que une todas las cabinas de enclavamiento de las líneas y el PCC, para permitir la interconexión de las redes locales de cada cabina de enclavamiento y permite a cualquier mando local/remoto alcanzar los Gateways del resto de enclavamientos de la línea.

Por lo tanto, para que la aplicación funcione de manera correcta, tendrá que recibir las señales que proceden del sistema de señalización viaria y tranviaria (a través del enlace Ethernet antes mencionado) y las imágenes de las cámaras de video ubicadas a lo largo del recorrido, para analizar y procesar dichas señales y visualizar y transmitir la información, en forma de aviso luminoso, acústico y de texto, tanto al PCC como al propio tren, en caso de ser necesario.

Una vez conocidos los requisitos a cumplir por la aplicación, se consensua con los Responsables de Línea de Metro Ligero Oeste, los cuales se encargan de la supervisión y análisis de la circulación de los tranvías, la elección de aquellas cámaras en pasos a nivel que sean más beneficiosas para el sistema.

Se eligen los pasos a nivel de las calles Nuevo Mundo y Prado del Espino, en Boadilla del Monte, y Ventorro del Cano, en Alcorcón, Madrid. El criterio llevado a cabo para esta elección es que el paso a nivel se visualice correctamente y que este sea problemático en cuanto a número de incidencias ocurridas en los últimos años.

Por último, se hace un trabajo de investigación sobre los métodos de detección y seguimiento de objetos en una secuencia de imágenes basados en visión artificial y procesado de la imagen. Existen numerosas técnicas para llevar a cabo estos procesos. Las tecnologías más modernas se basan en Machine Learning y Deep Learning, las cuales se sustentan en aprendizaje automático mediante el uso de algoritmos para organizar datos, reconocer patrones y hacer que las computadoras puedan aprender con esos modelos sin necesidad de reprogramación. Debido al tiempo limitado del que se dispone para realizar el proyecto y a que estos temas no se estudian durante el grado, se ha optado por realizar estas tareas mediante técnicas más básicas, pero de las que se posee un mayor conocimiento. De esta forma, se pueden resolver de manera más rápida y eficaz los problemas que puedan surgir durante el desarrollo e implementación de la aplicación.

1.5. Planificación

La planificación del proyecto está condicionada por el plan docente de la asignatura “TFG Aplicaciones multimedia basadas en procesamiento de la señal”. Esta asignatura sigue una evaluación continua y tiene marcadas cinco entregas en las siguientes fechas:

		FECHAS	
		INICIO	ENTREGA
NOMBRE	PEC1: Propuesta	19/09/2019	02/10/2019
	PEC2: Estado del Arte	03/10/2019	30/10/2019
	PEC3: Diseño e implementación	31/10/2019	18/12/2019
	PEC4: Memoria	19/12/2019	08/01/2020
	PEC5 A: Presentación	09/01/2020	19/01/2020
	PEC5 B: Defensa	20/01/2020	27/01/2020

Tabla 1: Entregas de la evaluación continua de la asignatura

- Propuesta: Durante este periodo se desarrolla la idea de propuesta y se elabora el documento correspondiente a la PEC1.
- Estado del arte: En esta fase se hace un trabajo de investigación sobre el tema del proyecto, se establecen los objetivos, el alcance, la metodología y la planificación de trabajo. Culmina con la entrega del documento de la PEC2.

- Diseño e implementación: En este periodo de tiempo se estudia los sistemas de videovigilancia y señalización de la explotación tranviaria, se diseña e implementa el producto a entregar, se realizan las pruebas pertinentes para comprobar su funcionamiento y sus limitaciones y se elabora la documentación del código utilizado. Al final, se hace entrega del documento de la PEC3.
- Memoria: En este documento se sintetiza el trabajo realizado y se muestra que se han alcanzado los objetivos propuestos.
- Se elaboran los distintos formatos para la presentación del TFG y se realiza la defensa del mismo.

Si bien el punto de partida para la planificación de las tareas del proyecto han sido las fechas estipuladas para las entregas de las distintas PEC, en ocasiones se ha tenido que modificar el calendario previsto, debido a la falta de disponibilidad de los responsables de Metro Ligero Oeste para mantener algunas reuniones necesarias para la correcta implementación de la aplicación.

La planificación definitiva se muestra en la siguiente página en forma de diagrama de Gantt, donde se puede apreciar cada tarea y su duración en días.

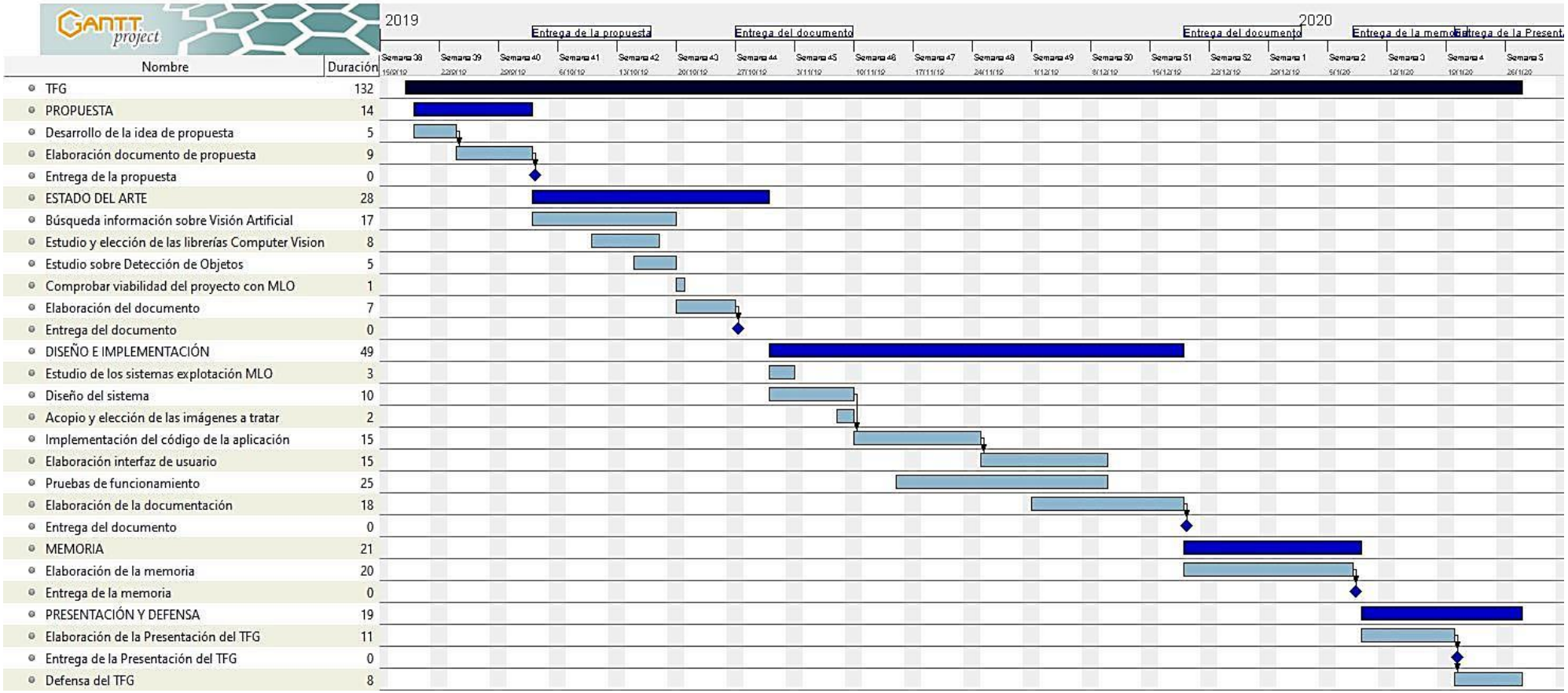

Figura 6: Diagrama de Gantt de la planificación del proyecto

1.6. Presupuesto

A continuación, se realiza un desglose del coste en función del personal y equipamiento necesario y de las horas destinadas a cada tarea.

Equipo humano

Para la realización del proyecto se necesita un Ingeniero de Telecomunicaciones con conocimientos de visión artificial.

Costes de tiempo

		PRECIO	CANTIDAD	SUBTOTAL
CONCEPTO	Estudio de la viabilidad de la aplicación en MLO	50€/h	2h	100€
	Estudio de los sistemas de videovigilancia y señalización de MLO	50€/h	6h	300€
	Estudio del sistema de detección de obstáculos que mejor se adapte a la explotación	50€/h	10h	500€
	Diseño del sistema	50€/h	30h	1500€
	Implementación del código de la aplicación	50€/h	120h	6000€
	Elaboración de la interfaz de usuario	50€/h	80h	3200€
	Integración y pruebas de funcionamiento	50€/h	100h	5000€
	Elaboración de la documentación	50€/h	60h	3000€
TOTAL		**19600 €**		

Tabla 2: Coste de tiempo

Hardware necesario

Se aprovecha toda la instalación de videovigilancia y señalización existente en MLO, con lo que no hay costes de instalación de cámaras, cableado, semáforos, etc. Sin embargo, es necesario añadir un puesto de operador con dos monitores para gestión, operación y control de la aplicación en el PCC. Actualmente se utilizan estaciones de trabajo de clase empresarial del fabricante Supermicro modelo 5039A-i y monitores LED del fabricante LG modelo 24MK400H-B.

		PRECIO	CANTIDAD	SUBTOTAL
CONCEPTO	Estación Supermicro 5039A-i	763€	1	763€
	Monitores LG 24MK400H-B	100€	2	200€
	Cables de red para señalización	10€	2	20€
	Cables de video	15€	2	30€
	Cables de alimentación	5€	3	15€
TOTAL		**1028 €**		

Tabla 3: Coste del hardware

Software necesario

Como ya se ha comentado anteriormente, la aplicación se ha desarrollado con *MATLAB,* con lo que es necesario que la estación de trabajo tenga instalado este software con licencia permanente.

		PRECIO	CANTIDAD	SUBTOTAL
CONCEPTO	MATLAB	2000€	1	2000€
	Image processing Toolbox	1000€	1	1000€
	Computer Vision Toolbox	1250€	1	1250€
TOTAL		**4250 €**		

Tabla 4: Coste del software

PRESUPUESTO TOTAL

		PRECIO	CANTIDAD	SUBTOTAL
CONCEPTO	Horas de trabajo	19600€	1	19600€
	Hardware	1028€	1	1028€
	Software	4250€	1	4250€
TOTAL		**24878 €**		

Tabla 5: Presupuesto total del proyecto

1.7. Estructura del resto del documento

Los capítulos restantes de esta memoria están estructurados de la siguiente forma:

- **Capítulo 2**. *Estado del Arte*: Se hace un análisis del estado actual de la visión por computador y de la detección de obstáculos, del software y hardware necesarios y de los campos de aplicación de estas técnicas.
- **Capítulo 3**. *Propuesta:* Se expone el proyecto desarrollado y se enumeran las especificaciones del mismo.
- **Capítulo 4**. *Diseño:* Se explican los detalles de la aplicación y su funcionamiento.
- **Capítulo 5**. *Implementación:* Explicación detallada de cómo está implementada la aplicación e instrucciones de instalación.
- **Capítulo 6**. *Demostración:* Se explican las instrucciones de uso de la aplicación y se muestran las pruebas de funcionamiento y los resultados obtenidos.
- **Capítulo 7**. *Conclusiones y líneas de futuro:* Describe las conclusiones personales y las líneas de futuro a seguir en las próximas fases del proyecto.

2. Estado del arte

La visión humana es uno de los sentidos más utilizados y es un mecanismo de procesado de imágenes muy poderoso, capaz de detectar y analizar imágenes de manera muy precisa. Es por esta razón por la que desde la antigüedad ha fascinado a la raza humana y se ha estudiado su funcionamiento, hasta que se ha logrado implementar algoritmos y sistemas capaces de simular, en cierta medida, su comportamiento, llegando, de esta forma, a la visión artificial.

En este capítulo se va a hacer un análisis de la situación de mercado de la visión artificial en general y de la detección de objetos en concreto, para tener una visión global del ámbito en el que se enmarca el proyecto.

2.1. Detección de objetos

La detección de objetos es una técnica de visión artificial mediante la cual se localizan objetos en una imagen o video digital. Existen numerosas técnicas para detectar objetos en una imagen. Una de ellas es mediante lo que se denomina detección de características, extracción y correspondencia. Con este método, se detectan características interesantes del objeto y se hace una correspondencia con características similares en la imagen donde se quiere buscar dicho objeto, estimando la ubicación del mismo. Actualmente existen algoritmos que permiten obtener los puntos de interés de los objetos de forma detallada, como pueden ser SURF o SIFT [5].

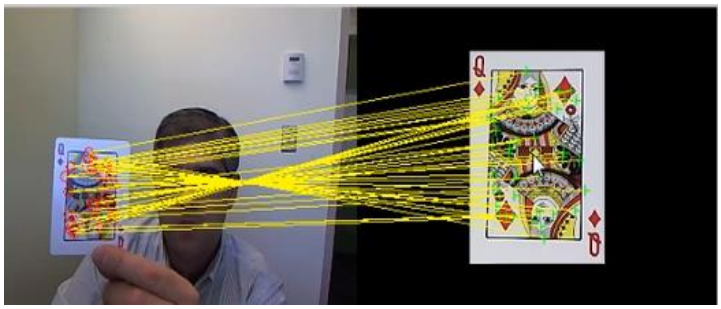

Figura 7: Técnicas de detección de características (Fuente: https://es.mathworks.com)

Técnicas más recientes, basadas en Machine Learning [6] y Deep Learning [7] se han convertido en las preferidas por los expertos para resolver problemas de reconocimiento de objetos, debido a que “aprenden” a identificar los objetos en las imágenes de forma automática. Se emplean modelos de redes neuronales convolucionales (CNN) las cuales aprenden a identificar las diferencias entre objetos mediante el análisis de miles de imágenes de entrenamiento.

Una CNN se puede entrenar desde cero, pero hay que recopilar un conjunto de datos etiquetados muy amplio y diseñar una arquitectura de red que aprenda las características y cree el modelo, lo que requiere una gran cantidad de datos de entrenamiento, por lo que lo más habitual y rápido es aprovechar un modelo de Deep Learning existente y ya entrenado, como AlexNet [8] o GoogLeNet [9], las cuales contienen miles de clases, e ir añadiendo datos nuevos de clases desconocidas por la red.

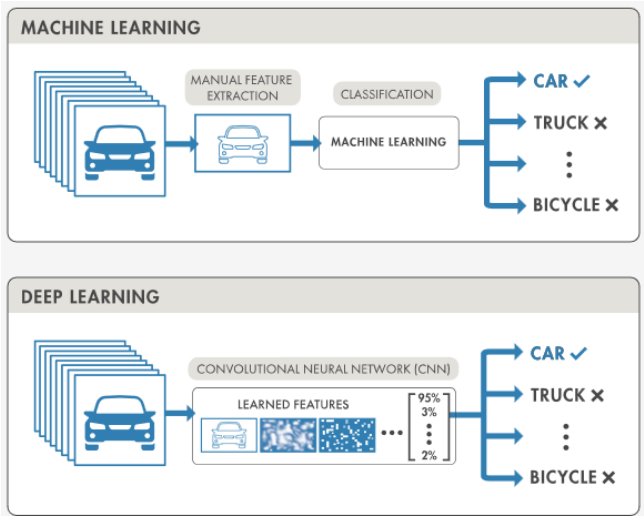

Figura 8: Técnicas Machine Learning y Deep Learning (Fuente: https://es.mathworks.com)

Existen otros métodos más básicos de detección de obstáculos, que en función de la aplicación, pueden proporcionar una solución sólida y llegar a ser igual de válidos que los vistos anteriormente, como pueden ser la coincidencia de plantillas, mapas de disparidad de píxeles o la segmentación de imágenes y análisis de componentes conectados, en los que se trabaja con propiedades simples de los objetos, como la forma, tamaño o color.

Para implementar la aplicación de detección de obstáculos de este proyecto, se va a utilizar la técnica de segmentación basada en movimiento, la cual permite la segmentación cuando tenemos un fondo estático y objetos en movimiento, como es el caso de nuestro sistema. Consiste, básicamente, en detectar los píxeles que se mueven en cada uno de los fotogramas. Esta técnica será explicada en detalle en el capítulo 5 del presente documento.

2.2. Software y hardware necesarios para la detección de objetos

En un sistema de visión por computador actual, se pueden distinguir básicamente las siguientes etapas o fases:

- **Captación:** Es el proceso a través del cual se obtiene una imagen o video.
- **Preprocesado:** En esta etapa se procesa la imagen para reducir o eliminar ruido, realzar detalles, añadir contraste o cualquier otro tipo de procesado.
- **Segmentación:** En donde se divide la imagen en varias partes que sean de interés.
- **Descripción:** Es el proceso mediante el cual se obtienen características para diferenciar un tipo de objeto de otro, como por ejemplo el tamaño y la forma.
- **Reconocimiento:** En esta fase se identifica a los diferentes objetos de una escena.
- **Interpretación:** Es el procedimiento que asocia un significado a un conjunto de objetos reconocidos.

Existen varios softwares que proporcionan librerías orientadas al procesado de la imagen y a la visión por computador, que aúnan las etapas antes descritas y con las que se consiguen excelentes resultados en la detección de objetos. A continuación, se detallan algunos de ellos.

MATLAB

Es un entorno matemático ideal para el procesado de imágenes digitales, ya que estas son, al fin y al cabo, matrices. Proporciona "cajas de herramientas" o *toolbox* como *Image Processing Toolbox* y *Computer Vision Toolbox*, si bien este último es un paquete más reciente y tiene funciones exclusivas para la visión artificial, procesado de video y visión 3D. También cuenta con las librerías de *Machine Learning* y *Deep Learning*. Con estas herramientas es posible llevar a cabo las etapas mencionadas y, en el caso de la visión 3D, es posible realizar la calibración de cámaras simples, estéreo y ojo de pez, la reconstrucción 3D y el procesamiento de nubes de puntos *lidar*, entre otras cosas.

OPEN CV [10]

OpenCV (Open Computer Vision) es una librería de software libre de visión por computador. Está escrita en C++ y C, tiene interfaces en Python, Java y Matlab y es compatible con los sistemas operativos Windows, Linux y MAC OS. La biblioteca contiene más de 500 funciones que abarcan muchas áreas de visión por computador y también de Machine Learning, que se centra en el reconocimiento y agrupamiento de patrones estadísticos.

INTEGRATING VISION TOOLKIT (IVT) [11]

Es una biblioteca de visión por computadora de código abierto, escrita en C++ y con una arquitectura orientada a objetos. Dispone de rutinas de procesamiento de imágenes y ofrece su propio kit de herramientas GUI multiplataforma (Windows, Linux, Mac OS). Posee técnicas de calibración de cámara única y estéreo. Es menos conocida y potente que Matlab y Open CV, aunque se espera que siga en desarrollo.

El hardware básico necesario de un sistema de visión por computador se compone de:

- **Cámaras**. Son las encargadas de capturar la imagen para su posterior procesado. En la actualidad, la mayoría de cámaras instaladas para sistemas de vigilancia son digitales, ya que simplifican el procesado posterior de la imagen. Estas poseen una lente convergente que proyecta la imagen sobre una superficie sensible a la luz denominada sensor de imagen.
- **Interfaz de entradas/salidas**. Sirve para establecer la comunicación entre los dispositivos externos (cámaras, sensores, señales, etc.) y la CPU de procesado de la imagen. Controla los elementos de entrada y salida para determinar las acciones a realizar antes y después del procesado de la señal.
- **CPU**. Registra, almacena y procesa las imágenes captadas por las cámaras y que le llegan a través de la interfaz de entradas/salidas.

2.3. Actualidad de la visión por computador y la detección de objetos

La visión por computador está muy presente en el día a día de las personas. En casi todas las tareas que se realizan a diario interviene un sistema que utiliza estas técnicas en mayor o menor medida.

2.3.1. Distintos campos de aplicación

Los campos en los que se aplica son amplios y variados. Por ejemplo, se pueden encontrar sistemas de este tipo que actúan con éxito en los siguientes ámbitos:

- **Industria**: Aplicaciones en supermercados, almacenes y fábricas de todo tipo para la lectura de datos (OCR, OCV, código de barras), verificación de orientación y posicionamiento de objetos, recuento de productos, sistemas de seguridad y vigilancia, etc.
- **Medicina**: Se emplean aplicaciones de visión por computador en el tratamiento y diagnóstico de melanomas, en al análisis de mamografías digitales, articulaciones y folículos, así como en robótica quirúrgica y visualización tridimensional de órganos.
- **Control de tráfico y Smart Cities**: Sistemas capaces de contar peatones y vehículos, detectar espacios de aparcamiento libres, identificar accidentes y avisar a los vehículos de emergencia para acortar los tiempos de acción, ajustar la potencia del alumbrado público al paso de vehículos, etc.

2.3.2. Éxitos en el ámbito del proyecto

Dentro del campo de la visión por computador orientada a la detección de objetos en la seguridad vial o en los medios de transporte, como el que abarca este proyecto, se pueden destacar las siguientes aplicaciones similares o que persiguen el mismo objetivo.

- **Sistema de detección de obstáculos en el paso a nivel** [12]. La empresa *MERMEC* dispone de un sistema de detección de obstáculos a nivel basado en la tecnología de infrarrojos láser y el principio de detección de distancias por luz, generando alarmas en función del tamaño del obstáculo detectado. De esta forma, se puede retrasar la apertura de la señalización hasta que la zona no esté libre de obstáculos.
- **Sistema sensor para la detección de objetos/obstáculos en puntos críticos de líneas férreas** [13]. El grupo de investigación de Ingeniería Electrónica Aplicada a Espacios Inteligentes y Transporte (GEINTRA) del Departamento de Electrónica de la Universidad de Alcalá, Madrid, ha desarrollado un sistema para detectar la presencia de cualquier tipo de objetos en puntos de interés del trazado ferroviario y envío al tren de información visual y señales de aviso ante la presencia de objetos. El dispositivo está constituido por un conjunto de cámaras ubicadas en el entorno de cada punto de interés, un sistema de iluminación infrarroja, un módulo de procesamiento de imágenes y un sistema inalámbrico de comunicaciones con el tren.
- **Detección de obstáculos en el entorno de carretera mediante visión por computador para ADAS** [14]. El Laboratorio de Sistemas Inteligentes (LSI) adscrito al departamento de Ingeniería de Sistemas y Automática de la Universidad Carlos III de Madrid, trabaja en el desarrollo de Sistemas Avanzados de Ayuda a la Conducción (ADAS) desde hace varios años. Disponen de un sistema que capta información tridimensional del entorno del vehículo mediante el empleo de dos cámaras que capturan imágenes desde dos puntos de vista diferentes. Con ello construyen un mapa de disparidad y detectan obstáculos próximos al vehículo, ya que etiquetan cada píxel del mapa de disparidad como obstáculo o calzada.

3. Propuesta

En el presente proyecto se desarrolla un sistema de detección de obstáculos, como pueden ser personas, vehículos, animales u objetos inertes, en un paso a nivel de una red de tranvías en zona urbana. El sistema, a través de una cámara de video fija, que enfoca en todo momento al paso a nivel, detecta cualquier obstáculo que invada la vía, y que por tanto pueda suponer un riesgo para la seguridad en la normal circulación del tranvía, y lanza un aviso de riesgo de colisión. En el caso que sea otro vehículo (coche, motocicleta, bicicleta, etc.) el que invada la vía, se lanza el aviso únicamente cuando la señalización vial esté en prohibición para dichos vehículos.

Como se ha comentado anteriormente, en esta fase del proyecto se trabaja con grabaciones de video procedentes de las cámaras del sistema de videovigilancia de Metro Ligero Oeste y se simulan los estados de la señalización, con el fin de evaluar el correcto funcionamiento de la aplicación. Las cámaras están localizadas en la glorieta de Nuevo Mundo y en Prado del Espino, situadas en Boadilla del Monte, y en el paso a nivel de Ventorro del Cano, Alcorcón, todas ellas de la línea ML3 de Metro Ligero Oeste, Madrid. El mensaje de alerta que envía el sistema será captado tanto por el Puesto Central de Control como por el tranvía que se aproxima a dicho cruce. De esta forma, el conductor estará debidamente informado con anterioridad suficiente y podrá actuar en consecuencia.

Para la programación del software se ha utilizado la herramienta de programación *MATLAB*, la cual permite crear este sistema gracias a que posee numerosos algoritmos de detección, extracción y coincidencia de características. *MATLAB* (abreviatura de Matrix Laboratory) es un software de cómputo numérico ampliamente utilizado en el entorno académico y en centros de investigación. Dispone de un lenguaje de programación propio y permite operaciones con matrices, la representación de datos e imágenes y la comunicación con otros programas en lenguajes diferentes, entre otras muchas cosas. Añadido a todo esto, posee cajas de herramientas (toolboxes) que almacenan una gran colección de funciones especializadas para distintas áreas, como puede ser la visión por computador, que es la que atañe a este proyecto.

La aplicación desarrollada tiene el nombre de “TramSafe System”. Si bien el sistema completo deberá componerse de software y hardware para su puesta en servicio, en este proyecto sólo se realiza la parte de software, dejando para más adelante (una vez terminado este TFG) la integración con los equipos que sean necesarios. La aplicación inicia las grabaciones de video y realiza un procesado inicial de la imagen, para eliminar el ruido, añadir contraste o cualquier otro tipo de preprocesado, para acto seguido identificar los objetos de la escena y hacer un seguimiento de ellos, con el fin de lanzar el mensaje de aviso en caso necesario. Una vez integrada en las instalaciones de la explotación

tranviaria, este software captará las imágenes emitidas en tiempo real por la cámara instalada en el exterior del cruce.

La principal ventaja del sistema de detección de obstáculos desarrollado en este proyecto respecto a otros sistemas similares (algunos ya vistos anteriormente en el apartado 2.3.2 de esta memoria) es que no necesita equipamiento adicional al que ya existe en la propia explotación, sino que aprovecha las cámaras del sistema de vigilancia de la circulación y las señales que provienen del sistema de señalización viaria y tranviaria del paso a nivel. Tampoco es necesario que las cámaras estén a la misma distancia del cruce ni con la misma orientación hacia el paso a nivel, ya que la región de interés (ROI) se delimita en cada cámara por separado, pudiendo estar en cualquier zona de la imagen.

El software desarrollado en este proyecto pretende abarcar las tareas de adquisición y análisis de video, procesamiento de la imagen, segmentación, identificación y seguimiento de objetos, áreas que en parte han sido vistas en las asignaturas de la Ingeniería de Tecnologías y Servicios de Telecomunicación.

3.1. Especificaciones del producto

Para que la aplicación desarrollada cumpla con los objetivos marcados y descritos en el apartado 1.3 de esta memoria, esta debe tener las siguientes características y prestaciones:

- Capacidad de procesar imágenes de video (grabadas o en tiempo real) y analizar cada uno de sus *frames.*
- Capacidad de delimitar la región de interés, donde se detectan los objetos, en cada una de las cámaras por separado, con formas geométricas diferentes y en cualquier zona de la imagen.
- Capacidad de detectar obstáculos de cualquier tamaño y forma.
- Capacidad de enviar alarmas visuales, acústicas y de texto en caso de detectar una incidencia en el paso a nivel.
- Disponer de una interfaz de usuario para gestionar el sistema de detección de obstáculos, con todas las funciones y servicios para que funcione correctamente.

4. Diseño

Una vez analizadas las características de los sistemas de videovigilancia y señalización de la explotación tranviaria MLO y las señales necesarias para que la aplicación funcione correctamente, se realiza una reunión con los Responsables de Línea y del PCC para consultarles las características y funcionalidades de la aplicación más interesantes para ellos, ya que son los usuarios finales del sistema. Estos destacan los siguientes aspectos:

- **Sencillez de la interfaz de usuario**. Desean que la interfaz tenga pocos botones y sea lo más intuitiva posible, ya que deben tomar decisiones en poco tiempo y manejan muchos otros sistemas a la vez.
- **Rapidez en el tiempo de ejecución**. Se necesita que la detección del obstáculo y el correspondiente envío de la alarma se haga en el menor tiempo posible.
- **Alarmas visuales y acústicas,** para no tener que estar pendiente de la pantalla de la aplicación en todo momento.
- **Posibilidad de guardar las imágenes de la infracción,** para tener una base de datos de incidencias. De esta forma, se podrán calcular estadísticas de infracciones (aunque no provoquen incidencias) y tomar medidas preventivas en caso necesario.
- **Datos de la cámara que ha registrado la incidencia.** Visualización de la ubicación de la cámara que ha detectado la incidencia, fecha y hora.

4.1. Diagrama de bloques del sistema

Con toda la información recabada, se procede a realizar un primer diseño de la aplicación. Para tener una idea más clara de cómo debe funcionar y tener una base para comenzar con la programación del código, se realiza un diagrama de bloques:

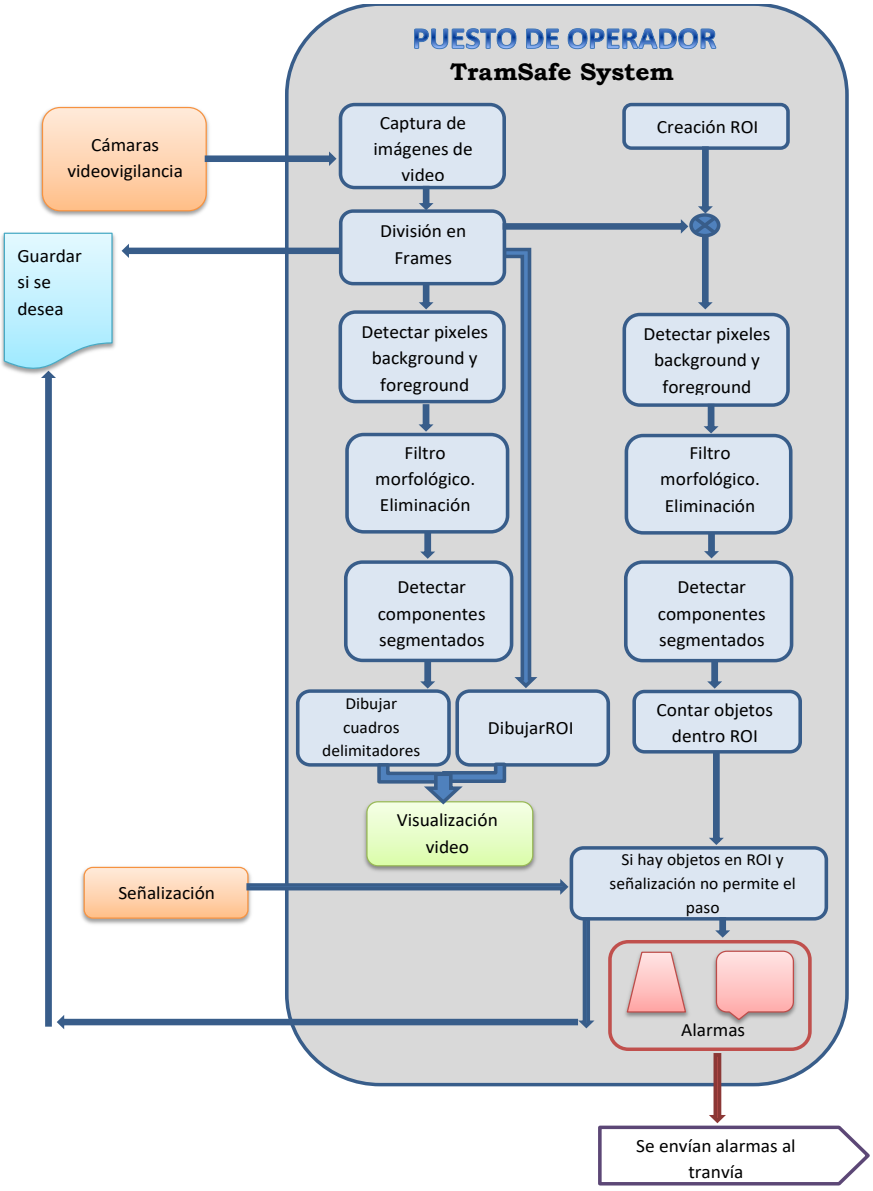

Figura 9: Diagrama de bloques de la aplicación

4.1.1. Descripción general de los elementos del diagrama de bloques

En el diagrama de bloques se pueden diferenciar los siguientes elementos o sistemas, descritos según el flujo de la señal, de arriba abajo, que aparece en la Figura 9:

- Captura de la imagen. Señal que proviene del sistema de videovigilancia de la infraestructura de MLO. Como se ha comentado anteriormente, en esta fase del proyecto, la señal es una grabación de video de una cámara seleccionada previamente.

- División de la imagen *frame* a *frame*, para tratar la secuencia de video en imágenes individuales.
- Creación de la región de interés (ROI), en la cual se detectarán los obstáculos y se enviarán las señales de alarma en caso de ser necesario. Se multiplica este ROI con el *frame,* para obtener una imagen con la zona externa del ROI en negro.
- Diferenciación de los píxeles que corresponden al fondo o al primer plano (*background y foreground)* para estimar qué píxeles cambian con el tiempo y, por tanto, están en movimiento.
- Filtro morfológico para eliminar ruido de la imagen.
- Reagrupación de píxeles para poder segmentarlos y diferenciar los objetos.
- Inserción de los cuadros delimitadores alrededor de los objetos detectados
- Dibujo de la ROI en el *frame,* para su visualización. De esta forma, se puede corregir su forma o ubicación en caso de que el usuario lo considere oportuno.
- Visualización completa de la imagen, con los cuadros delimitadores de objetos y ROI, para comprobar el funcionamiento de la aplicación.
- En caso de que la señalización no autorice el paso y haya objetos dentro de la ROI, creación y envío de alarmas.

4.2. Diseño de la interfaz de usuario

En la parte de interfaz gráfica, se desea realizar una aplicación visualmente sencilla, con pocos botones y con facilidad de uso, donde se puedan seleccionar y visualizar las distintas cámaras si se desea y en la que aparezcan las alarmas pertinentes si se ha producido una incidencia. El diseño inicial de la interfaz es el siguiente:

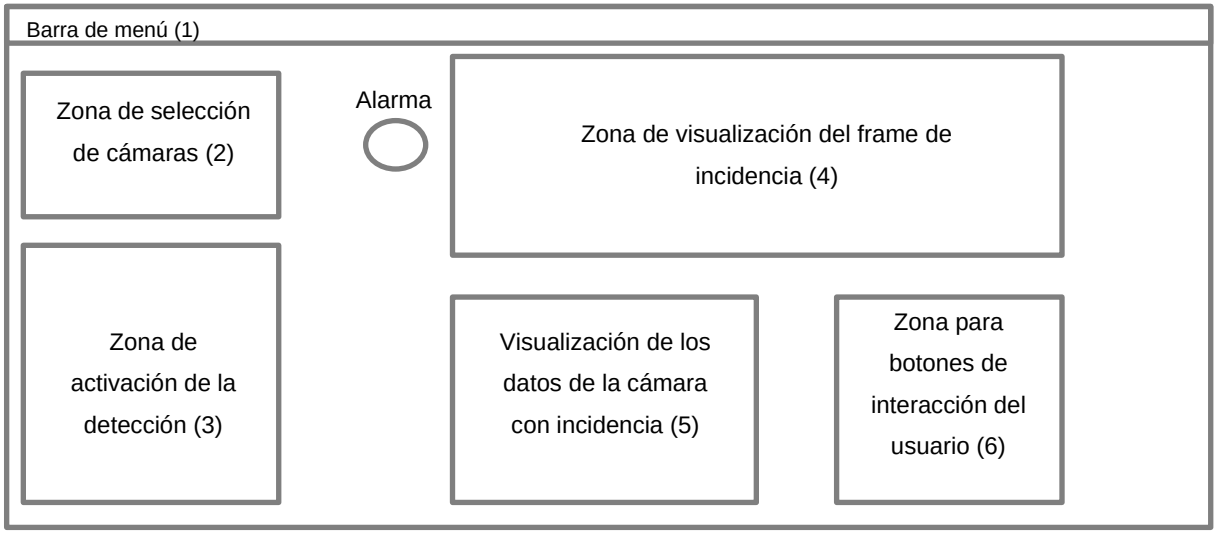

Figura 10: Diseño inicial de la interfaz de gráfica

4.2.1. Descripción general de los elementos de la interfaz de usuario

A continuación se describen de forma general las distintas partes de la interfaz gráfica, según la numeración que aparece en la figura 10:

1) Menú desplegable con las herramientas y opciones de la aplicación
2) Desde aquí se pueden seleccionar las distintas cámaras que componen el sistema de videovigilancia y, por tanto, del sistema de detección de obstáculos.
3) En esta zona se ubicará un conmutador que permite o no el paso del tranvía y de los vehículos, para simular el funcionamiento de la señalización y poder probar el funcionamiento de la aplicación.
4) Visualización de la imagen del momento de la incidencia. A su lado aparece la alarma visual.
5) En esta parte de la interfaz aparecerán los datos de la ubicación de la cámara que ha captado la incidencia y la fecha y hora del evento.

4.3. Entorno de desarrollo

Para programar todas las funciones que ejecutan cada uno de los elementos o sistemas vistos en el diagrama de bloques de la figura 9, se ha utilizado el entorno de programación *MATLAB R2019b* y dos de sus *Toolbox*.

***Image Processing Toolbox**:* Proporciona algoritmos y funciones para el procesamiento de imágenes, análisis, visualización y desarrollo de algoritmos, segmentación de imágenes, mejora de imágenes, reducción de ruido, etc.

***Computer Vision Toolbox**:* Proporciona algoritmos y funciones para diseñar sistemas de visión artificial, visión 3D, procesamiento de video, detección y seguimiento de objetos, así como detección de características, extracción y coincidencia.

La interfaz de usuario se ha realizado mediante *App Designer* [15], una utilidad de *MATLAB* para el desarrollo de interfaces y apps, la cual facilita mucho la tarea de diseño y reduce el tiempo requerido de programación. En este entorno, las vistas de código y diseño están vinculadas y los cambios que se realizan en una vista afectan de forma inmediata a la otra. Se pueden añadir devoluciones de llamada (*callbacks*) de componentes, así como interacciones con el ratón y el teclado personalizadas, que se ejecutan cuando un usuario interactúa con la app. Además, se genera automáticamente el código orientado a objetos que específica la distribución y el diseño.

5. Implementación

A partir de las especificaciones funcionales de la aplicación, se procede a implementar los módulos o sistemas descritos en los apartados 4.1 y 4.1.1 del presente documento. Se va a seguir el mismo orden que anteriormente para describir cómo trabaja cada módulo y qué algoritmos se utilizan.

5.1. Algoritmos principales del software

Captura de la imagen.

MATLAB dispone de lo que se llama “objetos de sistema”, que son objetos diseñados para trabajar con datos continuos, como un video. En este proyecto se usarán, entre otros, los objetos *VideoFileReader* para leer el video de entrada y *VideoPlayer* para la visualización del video.

Sus sintaxis de empleo son, respectivamente:

```
videoFReader = vision.VideoFileReader(Filename)
videoPlayer = vision.VideoPlayer
```

División de la imagen en *frames*

Para leer cada *frame*, se utiliza *step*, el cual llama al objeto de sistema y ejecuta el algoritmo. Dependiendo del objeto del sistema, puede devolver distintos argumentos de salida. Su sintaxis en el código de la aplicación es:

```
videoFrame = step(videoReader);
```

Creación de la región de interés

Existen varios métodos en *MATLAB* para dibujar estas regiones en una imagen determinada, pero se decidió hacer uso de técnicas básicas de procesado de imagen, como son las máscaras binarias, ya que permiten interactuar con ellas de varias maneras. Mediante la función poly2mask, es posible dibujar una máscara de ROI binaria sin una imagen asociada, indicando los vértices del polígono y el tamaño de la imagen binaria que se devuelve como argumento de salida.

Esta función tiene la expresión bw = poly2mask(xi, yi, m, n), siendo [xi,yi] las coordenadas de la región (vector numérico) y [m,n] el tamaño en píxeles de la imagen completa. Esta función establece con valor 1 los pixeles dentro de la región y con valor 0 los pixeles fuera de ella. De esta forma, al multiplicar pixel a pixel una imagen con está máscara binaria del mismo tamaño, obtenemos otra imagen en la que el polígono (que es la zona de interés) contiene esa misma región de la imagen original y el resto es negro, como muestra la siguiente imagen:

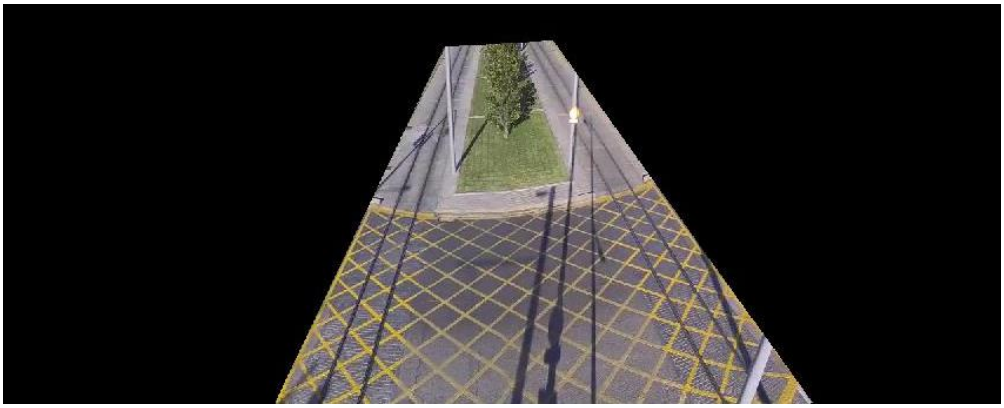

Figura 11: Creación de la zona de interés

Para cada una de las cámaras se ha estimado una ROI diferente, ya que no todas están orientadas de igual forma hacia el paso a nivel. Dicha región tiene unas coordenadas previamente definidas cuando se ejecuta la aplicación por primera vez, si bien es posible modificarla a posteriori por el usuario si así lo desea. Una vez obtenida esta imagen, si se detectan píxeles que no pertenecen al fondo en dicha ROI, se pueden lanzar avisos o alarmas en caso necesario.

Diferenciación de los píxeles que corresponden al fondo o al primer plano (*background y foreground)*

Para ello, se utiliza un algoritmo de eliminación del fondo mediante el objeto de sistema *ForegroundDetector*, el cual determina si los píxeles individuales son parte del fondo o del primer plano utilizando modelos mixtos Gaussianos [16]. Este estima que cada píxel de la escena está modelado por una mezcla K de distribuciones Gaussianas. La probabilidad de que cierto píxel tiene un valor de x_N en el momento en que N puede escribirse como:

$$p(x_N) = \sum_{j=1}^{K} w_j \eta(x_N; \theta_j)$$

donde w_K es el parámetro de peso del componente K^{th} gaussiano y $\eta(x_N; \theta_j)$ es la distribución Normal del componente K^{th}.

Su sintaxis en Matlab es detector = vision.ForegroundDetector(Name,Value) donde “Name” y “Value” son parejas de propiedades y su correspondiente valor, como puede ser el número de frames iniciales para entrenar el modelo o el número de modos Gaussianos utilizados. La sintaxis utilizada en el código de la aplicación es:

foregroundDetector = vision.ForegroundDetector('NumGaussians', 3,'NumTrainingFrames', 50);

Después, se leen los primeros *frames* (por ejemplo, 75) para que el sistema aprenda a detectar el fondo, mediante el siguiente bucle:

```
for i = 1:75
    videoFrame = step(videoReader); % Se lee el frame del video
    zonaInteres = bw .* videoFrame; % Se multiplica pixel a pixel la máscara binaria con el frame, para crear una imagen en la que la zona fuera de la región de interés es cero (negro).
    fondo = step(foregroundDetector,zonaInteres); % Se detecta el fondo de la zona de interés
end
```

Este algoritmo dejará con valor 0 (negro) los píxeles del fondo y con valor 255 (blanco) los píxeles que cambian con el tiempo. Lógicamente, en la zona fuera de la ROI, al ser todos los píxeles negros, estos no cambian. Obtendremos, pues, una imagen binaria como la siguiente, en la que se aprecian detalles que han cambiado en la ROI durante ese tiempo.

Figura 12: Detección de píxeles en movimiento

Filtro morfológico para eliminar ruido de la imagen

Después de aplicar el algoritmo antes descrito, puede haber ruido en la imagen que el sistema interprete que son objetos en movimiento, como se aprecia en la figura 12. Por lo tanto, se debe eliminar este ruido mediante un operador morfológico.

Las transformaciones morfológicas son operaciones simples basadas en la forma de la imagen, que normalmente se realizan en imágenes binarias. Necesita dos entradas, una es la imagen original y la segunda se llama elemento estructurante *(EE)*, cuyo tamaño es MxN (normalmente de 3x3) y el valor de sus coeficientes es 0 ó 1. Este se desplaza sobre la imagen píxel a píxel y la imagen filtrada recoge los valores calculados en cada punto.

Figura 13: Elementos estructurales más comunes

Para simplificar las siguientes explicaciones, se considerará un EE con todos los coeficientes de valor 1, valor del negro = 0 y valor del blanco = 1 en la imagen binaria. Las principales operaciones morfológicas son:

a) Erosión

El EE se desliza a través de la imagen. Un píxel en la imagen original se considerará 1 solo si todos los píxeles debajo del EE son 1, de lo contrario se pondrá a cero. Por lo tanto, los detalles claros se reducen o se borran si su tamaño es menor que el del EE. Es útil para eliminar pequeños ruidos blancos, si bien la imagen se oscurece.

b) Dilatación

Hace lo contrario que la erosión. Un píxel toma valor 1 si al menos un píxel debajo del EE es 1, con lo que la imagen se aclara. Sirve para reducir o eliminar los detalles oscuros, dependiendo del tamaño del EE.

c) Apertura

Consiste en realizar una erosión seguida de una dilatación, usando en ambos casos el mismo EE. La apertura borra detalles claros de pequeño tamaño manteniendo el resto de la imagen sin variaciones.

d) Cierre

Consiste en realizar una dilatación seguida de su erosión, también con el mismo EE. El cierre elimina pequeños detalles oscuros sin modificar el resto de la imagen.

Después de probar varias de estas de estas operaciones y ver cómo se comportan, se elige el operador de apertura y se aplica a la imagen obtenida en el paso anterior mediante el siguiente código, donde *strel* es un elemento de estructuración en forma de disco con un radio 1:

```
limpiar = imopen(fondo,strel('Disk',1))
```

Obteniendo la siguiente imagen libre de ruido.

Figura 14: Eliminación de ruido mediante operación morfológica de apertura

Reagrupación de píxeles

Una vez filtrada la imagen, se reagrupan los píxeles restantes para poder segmentarlos. Esta acción la realizamos mediante el análisis de componentes conectados y el objeto de sistema *BlobAnalysis*, cuya sintaxis es Hblob = vision.BlobAnalysis(Name,Value).

El análisis Blob calcula estadísticos de las regiones conectadas de una imagen binaria, con lo que podemos obtener su área, la caja delimitadora del objeto, el tamaño de los ejes, el perímetro, etc. A este objeto le podemos decir que filtre los objetos con área menor a un número de píxeles dado, con lo que aún filtramos más ruido si no lo hemos eliminado antes.

Para realizar esta operación, primero se declara el objeto de sistema y después se aplica a la imagen sin ruido, de la siguiente forma:

```
% Declaración del objeto de sistema blobAnalysis
blobAnalysis = vision.BlobAnalysis('BoundingBoxOutputPort', true, 'AreaOutputPort', false, 'CentroidOutputPort', false, 'MinimumBlobArea', 150);
% Se detectan los componentes conectados de la imagen sin ruido, con un área mínimo específico, y se calculan sus cuadros delimitadores, obteniendo la variable “caja”
caja = blobAnalysis(limpiar);
```

Obtenemos la variable “caja”, que es una matriz que contiene el número de segmentos encontrados y su coordenada. El valor mínimo del área del objeto detectado (150 en la expresión de arriba) está predefinido cuando se ejecuta la aplicación, si bien es posible modificarlo a posteriori por el usuario.

Inserción de los cuadros delimitadores alrededor de los objetos detectados

Para hacer un seguimiento visual del objeto detectado, se insertan cuadros que delimitan dicho objeto mientras está en movimiento, con la siguiente función:

```
result = insertShape(videoFrame, 'Rectangle', caja, 'Color', 'green', 'LineWidth', 2);
```

El ancho de la línea de la caja delimitadora (valor 2 en la expresión anterior) está predefinido al ejecutar la aplicación, si bien es posible modificar este valor a posteriori por el usuario.

Dibujo del ROI en el *frame*

Del mismo modo, se dibuja en el *frame* la misma forma del ROI que hemos creado antes para detectar los objetos en ella, solo que ahora es simplemente para tener información visual de dónde está ubicada. Esto se hace así para detectar si está orientada donde desea el usuario final y comprobar que el sistema está funcionando correctamente. Se hace con la misma función que anteriormente, solo que ahora aplicada a la variable *result* obtenida en el paso anterior.

```
result1 = insertShape(result,'FilledPolygon',area,'Opacity',0.4,'LineWidth',2, 'Color','red');
```

Visualización de la imagen

Se visualiza la imagen de video completa, con los cuadros delimitadores en los objetos detectados y la ROI, de la siguiente forma:

```
step(videoPlayer,result1);
```

Envío de alarmas

La aplicación tiene en cuenta, en todo momento, los objetos que están presentes en la ROI, mediante la función `numCars = size(caja, 1);`

Esta función indica el tamaño de la primera dimensión de un array, en este caso el número de segmentos encontrados en “caja”, por lo tanto, si hay un objeto en la ROI y la señalización no lo permite, se envía una señal de alarma, tanto visual como de audio y texto.

5.2. Implementación de la interfaz de usuario

Una vez obtenidos todos los pasos necesarios para la detección de objetos e implementadas las funciones necesarias en código *MATLAB*, se procede a crear la interfaz gráfica de la aplicación. La ventana principal se puede dividir en tres partes, tal y como se puede observar en la figura 15.

Figura 15: Interfaz gráfica de la aplicación

El recuadro de color azul se corresponde con la barra de menús de la aplicación. En la parte recuadrada en color rojo, el usuario puede seleccionar las cámaras existentes en las dos líneas de MLO, además de aparecer varios indicadores luminosos y un conmutador. En el área de color verde se visualizan las imágenes de las incidencias detectadas por las cámaras, la información de la cámara (ubicación, fecha y hora) y los botones de eliminar incidencia y guardar imagen, así como de configuración de los distintos parámetros.

5.2.1. Descripción de la interfaz

A continuación, se comentan por separado las funcionalidades de cada una de las partes principales que conforman la interfaz gráfica de la aplicación y mostradas en la anterior figura.

Barra de menú

En esta sección se encuentran las opciones de la aplicación dispuestas en menús desplegables, como se aprecia en las siguientes imágenes:

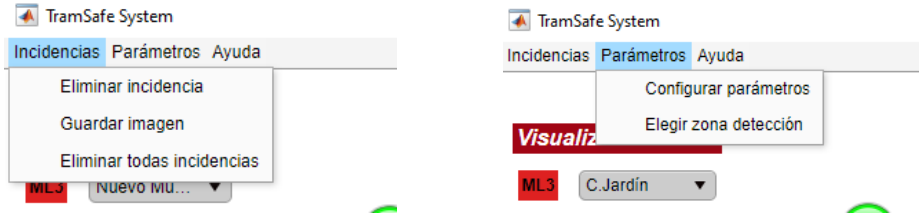

Figura 16: Menús desplegables

A través de estos menús se pueden realizar las mismas acciones que con los botones ubicados en la interfaz, así como cambiar la contraseña necesaria para realizar algunas acciones y acceder a un manual de usuario de la aplicación, el cual se abre en formato pdf.

Área de selección de cámaras

En esta parte de la interfaz, se pueden seleccionar las distintas cámaras existentes para visualizarlas en tiempo real y observar el seguimiento de los objetos, así como la zona de detección de obstáculos dibujada en rojo. De esta forma, se puede comprobar si el sistema está funcionando correctamente o si es necesario modificar la zona de detección, ya sea porque la cámara se ha movido con un golpe o debido al viento. Como ya se ha comentado anteriormente, en esta fase del proyecto las imágenes son grabadas y no en tiempo real, con lo que este menú servirá para iniciar el video de la cámara seleccionada, como se explicará más adelante.

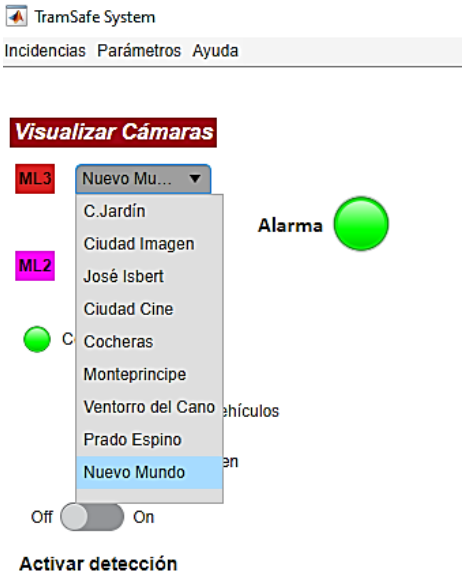

Figura 17: Menú desplegable de selección de cámaras

También están ubicados en esta área los siguientes indicadores luminosos:

- Alarma. Indica si se ha detectado alguna incidencia. Su color por defecto es verde, cambiando a rojo si hay alguna incidencia presente y que no se ha eliminado.
- Conectadas. Informa que todas las cámaras del sistema están conectadas y funcionando. Si alguna cámara está apagada por alguna razón, el indicador se pondrá en color rojo. Este indicador tendrá como señal de entrada la señal correspondiente del sistema de videovigilancia que funciona actualmente en MLO, con lo que ahora no tiene funcionalidad.
- Semáforo de vehículos y de tren. Estos indicadores muestran el estado de las distintas señales que rigen el tráfico en el paso a nivel y, por tanto, tendrán como entradas sus respectivas señales que provienen del sistema de enclavamiento descrito en el apartado 1.4 de este documento. Dependiendo del estado de estas señales, el sistema lanzará o no la señal de alarma por incidencia. Dicha alarma se lanzará, en principio, cuando el semáforo de vehículos esté en rojo y el de tren, en verde, lo que indica un riesgo de colisión. Debido a que la aplicación ahora no está integrada en el conjunto del sistema de videovigilancia, existe un conmutador ("Activar detección") para simular dicha combinación de señales y probar el funcionamiento del sistema.

Figura 18: Indicadores luminosos de señales y conmutador para pruebas

Área de incidencias

En esta zona se visualizan las incidencias detectadas.

- Imagen de incidencia. Se visualiza el *frame* del momento en el que se produce la invasión de la zona de detección de algún objeto cuando la señalización no lo permite.
- Cámaras con incidencias. Se muestra información de la cámara que ha registrado la incidencia (ubicación, fecha y hora).

Además, están ubicados los siguientes botones.

- Eliminar incidencia. Borra la última incidencia de la lista de información y la imagen del *frame*.
- Guardar imagen. Guarda la imagen de la infracción en la ubicación elegida por el usuario.

- Eliminar todas las incidencias. Se borran todas las incidencias detectadas (es necesario insertar contraseña para realizar esta acción).
- Configurar parámetros. Mediante este botón se pueden modificar distintos parámetros, como el grosor de la línea que delimita los objetos detectados o el conjunto de píxeles conectados, para detectar objetos de mayor o menor tamaño en la imagen (es necesario insertar contraseña para realizar esta acción).
- Elegir zona detección. Permite definir una nueva zona de detección (ROI) o modificar la existente, ya sea porque se quiere detectar una zona nueva o porque se ha movido la cámara y hay que adecuar la zona (es necesario insertar contraseña para realizar esta acción).

Todos estos botones tienen una ayuda visual o *tooltip* cuando se sitúa el cursor sobre ellos, explicando de forma resumida la función de dicho botón.

5.3. Instalación de la aplicación

La aplicación desarrollada se puede instalar en cualquier ordenador que tenga MATLAB instalado mediante el archivo *"TramSafeSystem. mlappinstal* ", el cual es un archivo de instalación de la app.

Para instalarlo, hay que abrir *MATLAB,* irse a la pestaña *APPS* y pinchar en *Install App,* como muestra la siguiente imagen.

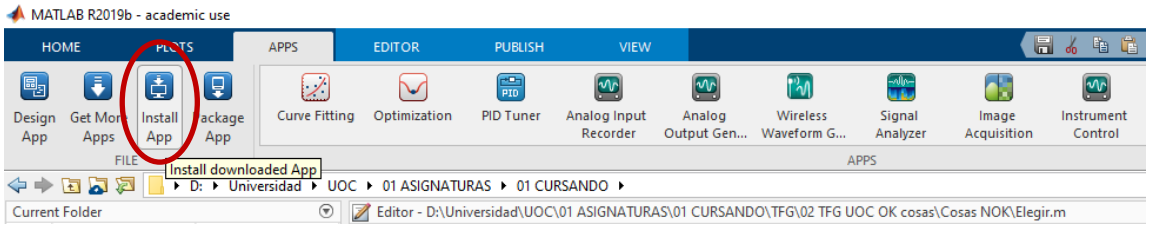

Figura 19: Pestaña de MATLAB para instalar la aplicación

Se abrirá una ventana de explorador para seleccionar, en la ubicación donde se encuentre, la aplicación que se quiere instalar. Una vez instalada, aparecerá la aplicación "TramSafe System" en el menú de aplicaciones disponibles para ser ejecutadas, junto con otras *apps* que ya vienen por defecto.

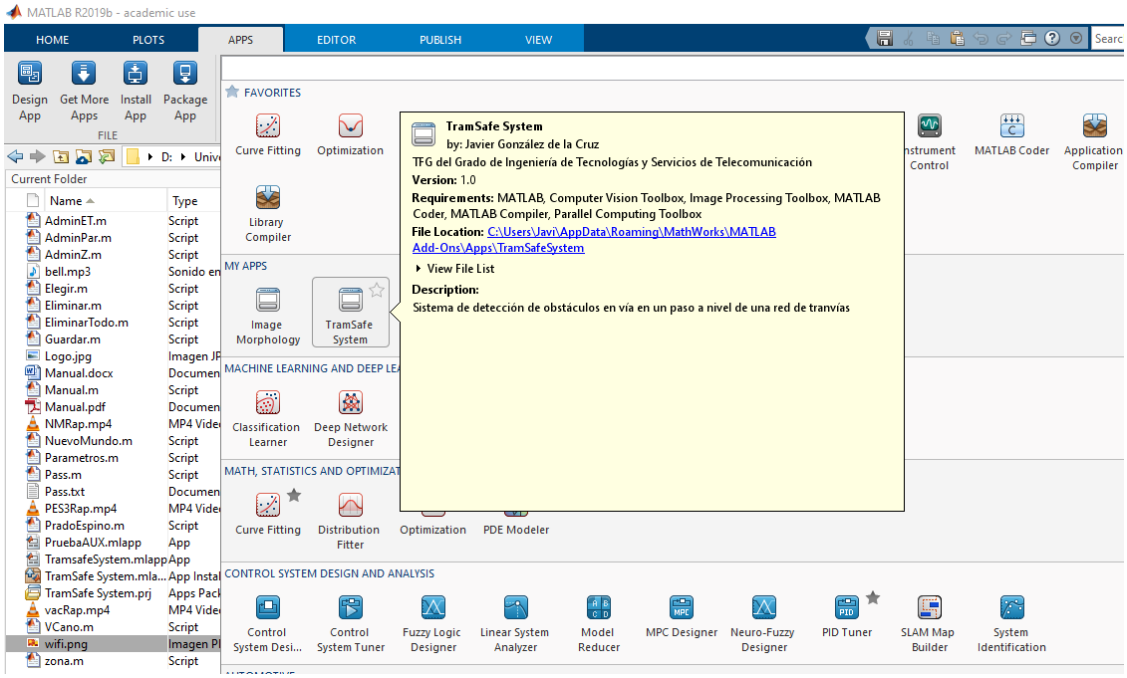

Figura 20: Conjunto de aplicaciones disponibles para ejecutar

6. Demostración

6.1. Instrucciones de uso

Una vez que se ejecuta la aplicación, se deben seguir los siguientes pasos para que esta funcione correctamente. Para que comience un video de prueba y, por tanto, las funciones que posibilitan la detección de obstáculos, se debe seleccionar una cámara de los menús desplegables de las líneas de MLO, tal y como indica la imagen:

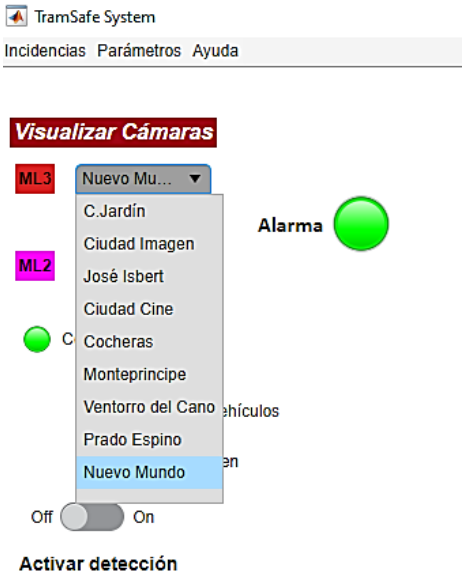

Figura 21: Menú desplegable de selección de cámaras

Si bien se pueden seleccionar cámaras de varias ubicaciones (para una fase futura del proyecto), **solo están disponibles los videos de las zonas de Ventorro del Cano, Prado del Espino y Nuevo Mundo**, como se ha indicado anteriormente. Esto es debido a que para descargar los videos de la base de datos de MLO, es necesario tener una autorización expresa del Director de Operaciones y del Director General, y estos solo han autorizado la descarga de estas tres zonas para la realización de este proyecto.

Por lo tanto, una vez que se arranca el programa, **este permanecerá inactivo hasta que no se seleccione una de estas tres cámaras**. La primera vez que se selecciona una cámara, el video tarda un tiempo en visualizarse debido a que los objetos de sistema deben inicializarse, como muestra la figura 20. Después, estos no se inicializan más y la operación es más rápida.

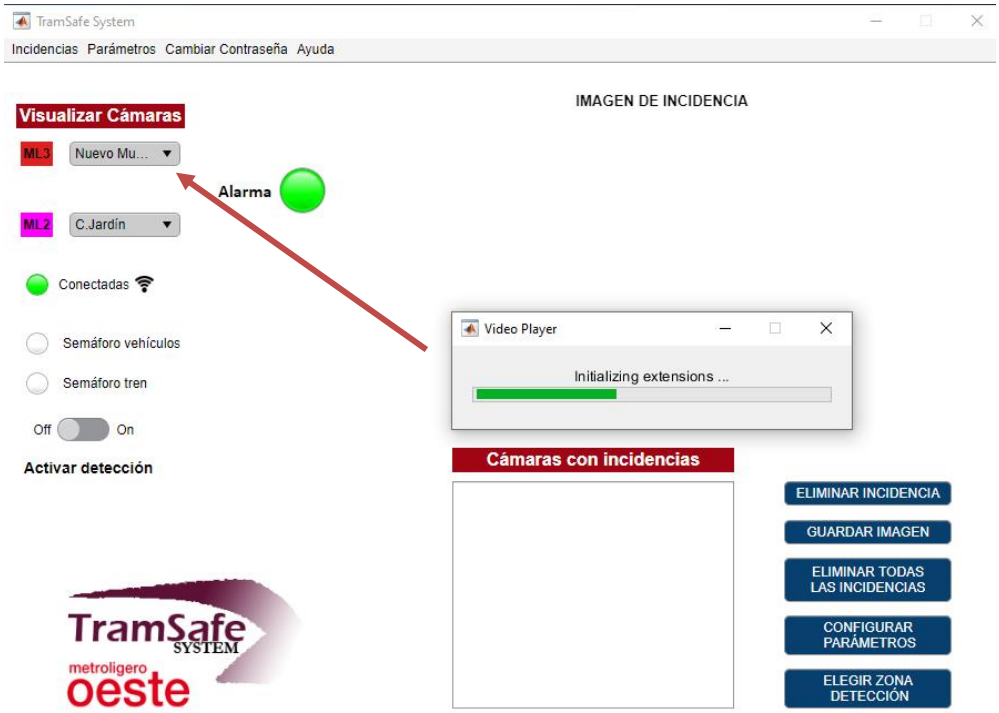

Figura 22: inicialización de los objetos de sistema al iniciar video de prueba

Una vez seleccionada la cámara, se abre una nueva ventana, donde se visualiza el video y comienza el funcionamiento del algoritmo de detección de obstáculos, para poder hacer pruebas mediante el conmutador de “Activar detección”.

Figura 23: Visualización del video de prueba de la cámara de “Nuevo Mundo”

Si algún objeto (en los videos disponibles pueden ser vehículos o personas) invade la zona de detección (marcada en rojo) cuando el conmutador se encuentra en “On”, sonará un aviso acústico y se visualizará el *frame* con la infracción y la información de la cámara, con fecha y hora, como muestra la figura 22.

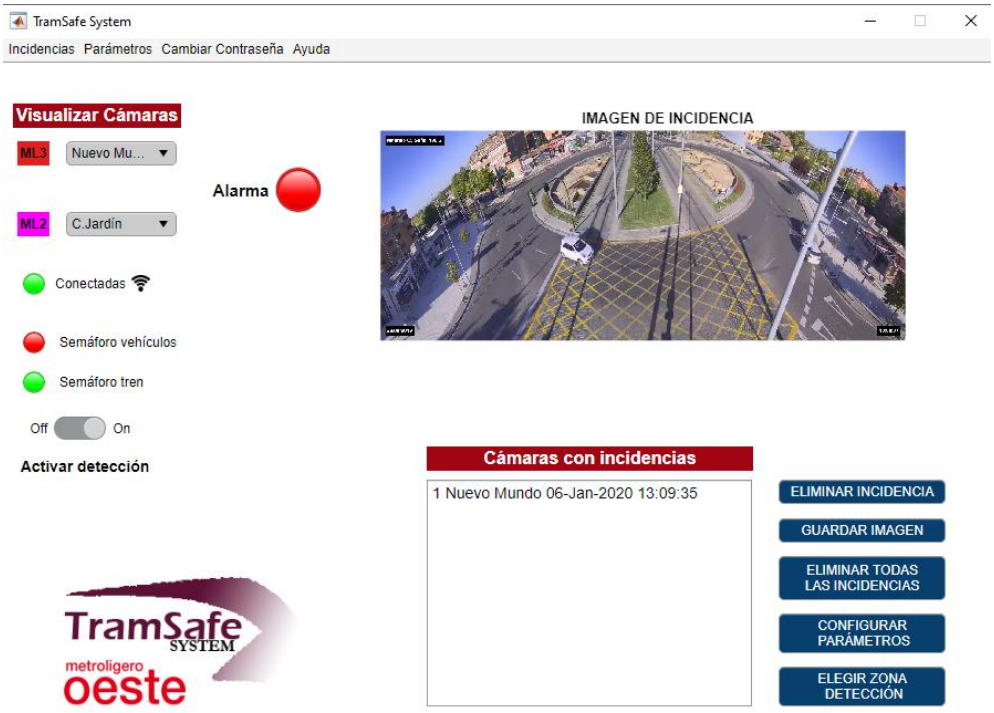

Figura 24: Captura de una incidencia en la cámara de “Nuevo Mundo”

Para detectar otro objeto, se deberá poner el conmutador en “Off” y de nuevo en “On”. Esto es así porque de lo contrario, la aplicación lanzaría una alarma y una captura de imagen cada *frame* y esto no es lo deseado. Una vez que la aplicación esté integrada en las instalaciones de MLO, se pondrá un retardo de varios segundos para la captura de incidencias, con el fin de evitar que salte una alarma por cada *frame.*

Una vez detectada la incidencia, se puede eliminar de lista o guardar la imagen de la infracción mediante los botones “Eliminar Incidencia” (o “Eliminar todas las Incidencias” si hay más de una) y “Guardar Imagen”, respectivamente. También se puede iniciar otro video o interactuar con los botones “Configurar Parámetros” y “Elegir Zona Incidencia”, descritos anteriormente, con lo que al abrir de nuevo la cámara afectada se visualizarán los cambios efectuados. Estas acciones se explican con más detalle en el siguiente apartado.

6.2. Pruebas realizadas y resultados

Se han realizado una serie de test de la aplicación para comprobar que detecta los objetos en movimiento correctamente y que muestra las distintas alarmas (cuando se activa la señalización simulada) en todos los videos disponibles. A su vez, se ha testeado cada una de las funcionalidades de la interfaz de usuario. A continuación, se exponen los resultados.

6.2.1. Pruebas de detección de objetos en los videos disponibles

En primer lugar, se selecciona el video de la cámara del paso a nivel de Nuevo Mundo, tal y como se ha explicado en las instrucciones de uso del apartado 6.1. Se abre el video en una nueva ventana para visualizarlo. Al comienzo de este video se ve un tranvía pasando por el paso a nivel, con lo que se deja el conmutador “Activar Detección” en posición “Off”, ya que no se desea que salten las alarmas. Efectivamente, el tranvía pasa sin que la aplicación muestre ninguna alarma. Así mismo, se observa que tanto el tranvía como otros vehículos que aparecen en la imagen y que están en movimiento, son detectados y aparecen con un recuadro de color verde alrededor de ellos, para poder hacer un mejor seguimiento de estos.

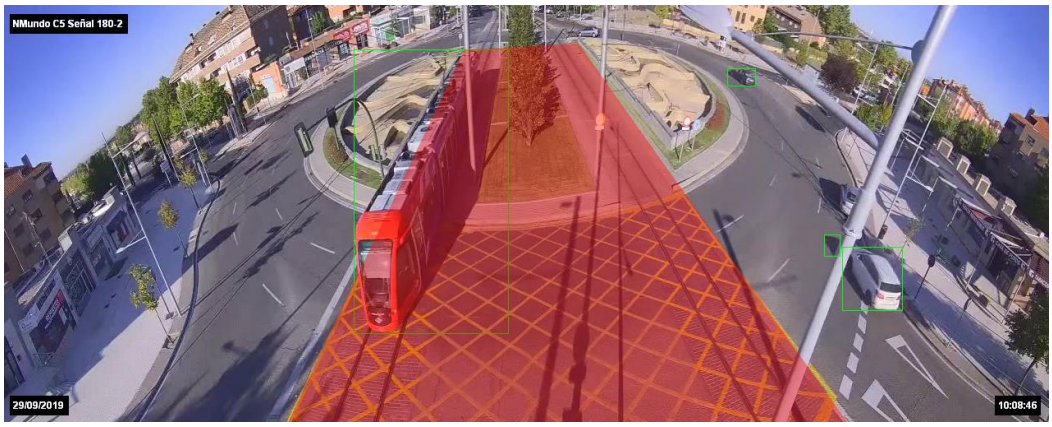

Figura 25: Imagen de la cámara del “Nuevo Mundo”

Instantes después, se observa que un coche atraviesa el paso a nivel, momento en que se pone el conmutador en “On” para ver si el sistema lo detecta correctamente. Se comprueba que lo detecta y que salta el sonido de alarma, muestra el frame del momento de la incidencia y muestra la información de la cámara, fecha y hora.

Figura 26: Visualización del momento de la incidencia

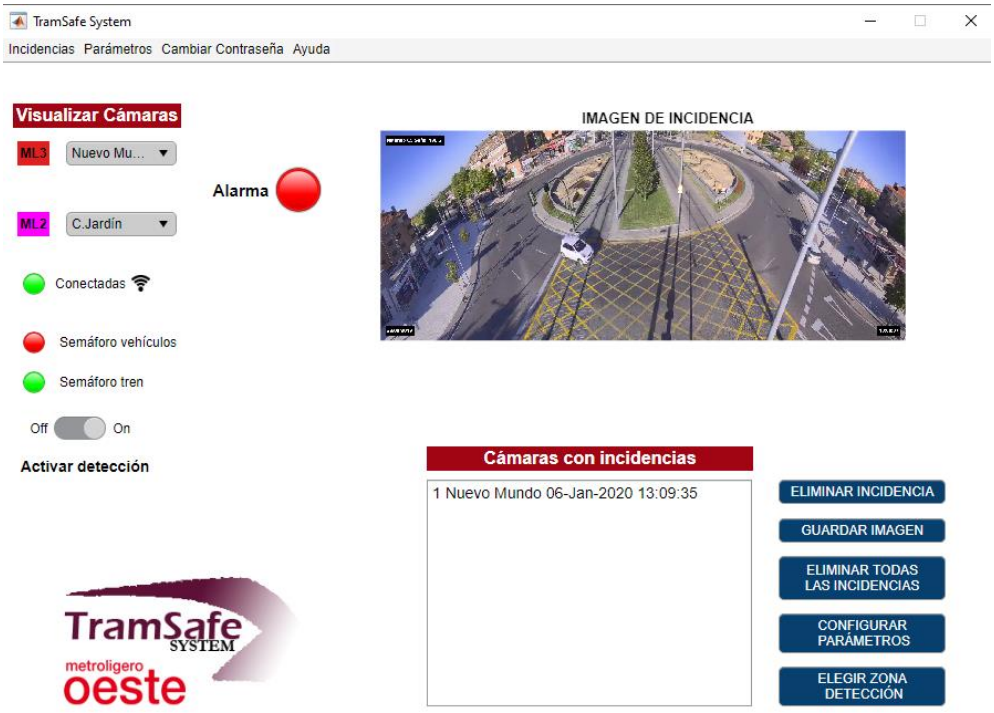

Figura 27: Captura de la incidencia detectada en “Nuevo Mundo”

Los tiempos de ejecución obtenidos son suficientemente rápidos (alrededor de 1 segundo entre que se detecta la incidencia y se muestran las alarmas por pantalla) para los tiempos que maneja la explotación MLO, ya que la señalización de esta infraestructura permanece cerrada para los vehículos de 12 a 30 segundos antes de que pase el tranvía, dependiendo del paso a nivel.

Se realizan las mismas pruebas en los otros dos vídeos disponibles de las cámaras de “Prado del Espino” y “Ventorro del Cano”, con idénticos resultados. Se puede observar como a medida que se detectan diferentes incidencias, si no se borran anteriormente, estas se van acumulando en la lista de información de las cámaras con incidencia.

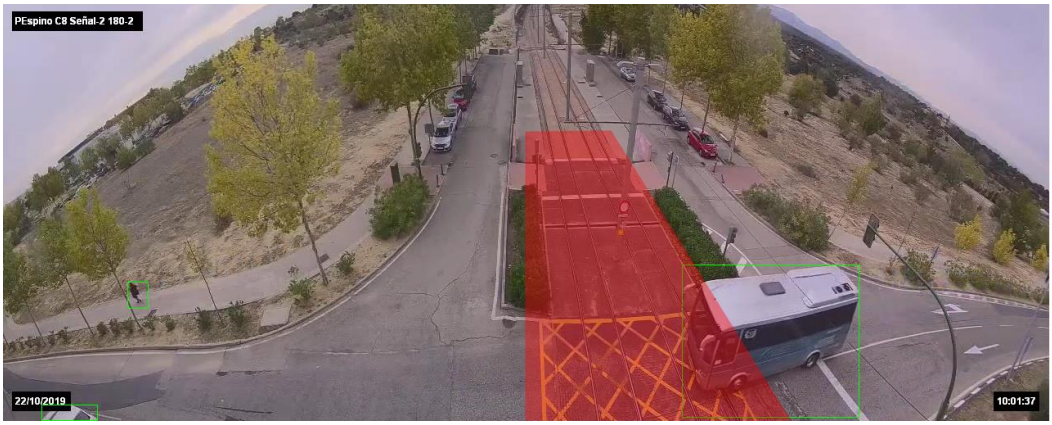

Figura 28: Incidencia en “Prado del Espino”

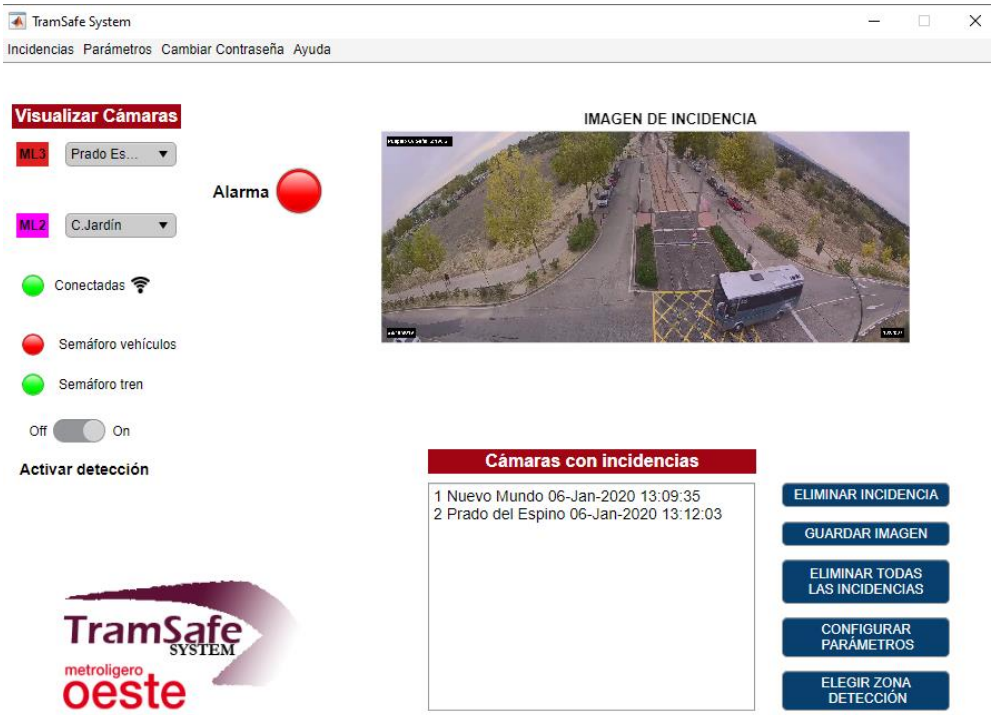

Figura 29: Captura de la incidencia detectada en “Prado del Espino”

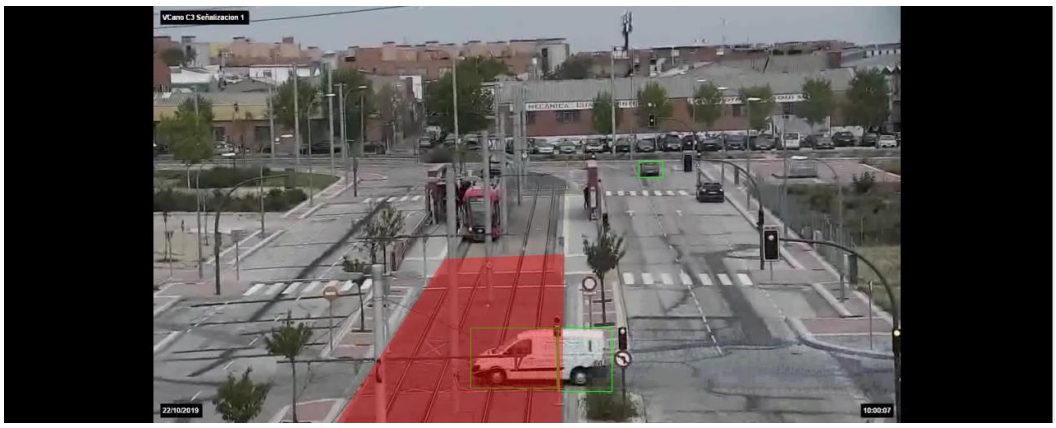

Figura 30: Incidencia en “Ventorro del Cano”

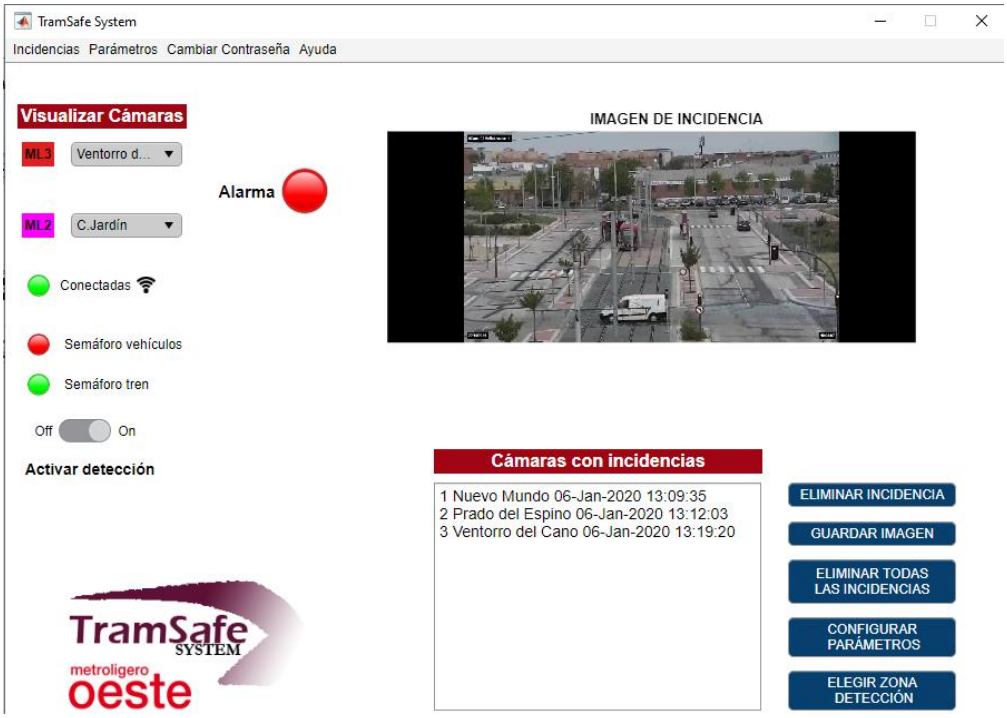

Figura 31: Captura de la incidencia detectada en “Ventorro del Cano”

6.2.2. Pruebas de las diferentes opciones de la aplicación

Una vez detectadas las incidencias de cada cámara, se selecciona el botón de “Eliminar Incidencia” para eliminar la última incidencia detectada, que corresponde con la del paso a nivel de “Ventorro del Cano”. Aparece una ventana emergente de confirmación de eliminación antes de borrarla.

Figura 32: Eliminar incidencia de “Ventorro del Cano”

Después, se prueba la opción de guardar imagen. Aparece una ventana emergente para guardar la imagen de la última incidencia de la lista que no ha sido borrada, en la ubicación y formato deseado por el usuario. En este caso, se guarda la imagen de la incidencia de “Prado del Espino”.

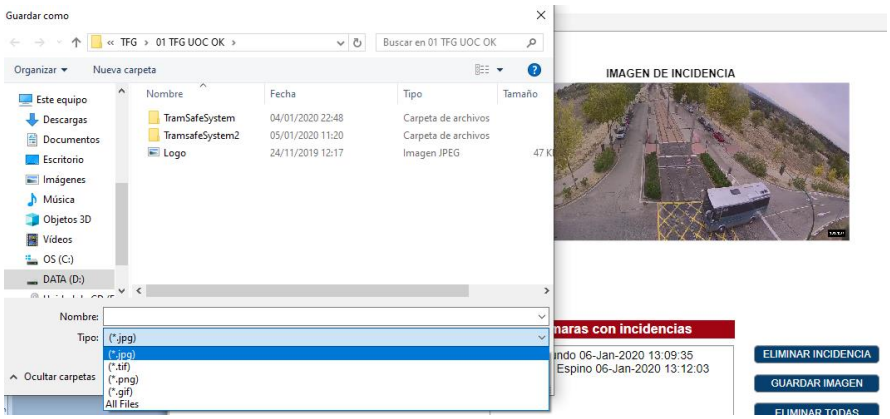

Figura 33: Guardar imagen de la incidencia

También se prueba la opción de borrar todas las incidencias. Se pide al usuario que inserte la contraseña para borrar todas las incidencias detectadas y que aparecen en la lista. Esto se hace así para evitar que se borren todas las incidencias por accidente y para que solo pueda borrarlas un usuario con privilegios.

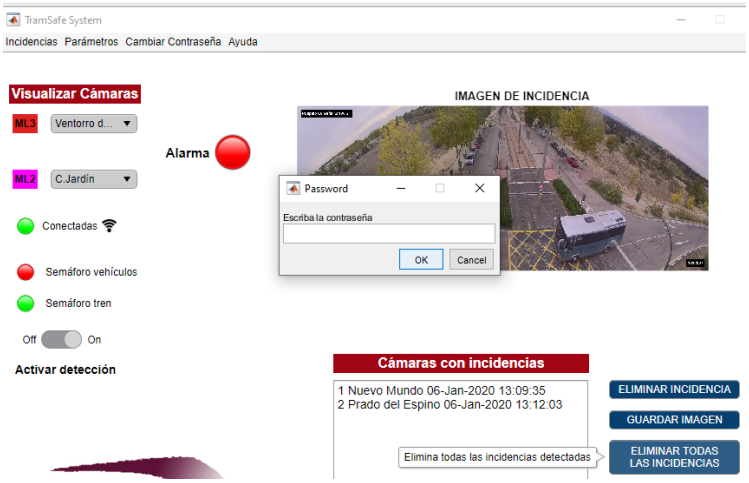

Figura 34: Eliminar todas las incidencias

Seguidamente, se prueba la opción de configurar parámetros, A través de esta acción, se pueden modificar los valores de los parámetros predefinidos al ejecutar la aplicación, como el grosor de la línea que delimita los objetos detectados o el conjunto de píxeles conectados, para detectar objetos de mayor o menor tamaño en la imagen. Al ejecutar la acción, aparece una ventana emergente para preguntar la contraseña. Si esta es correcta, aparece otra ventana donde se elige la cámara a la que se desea modificar los parámetros.

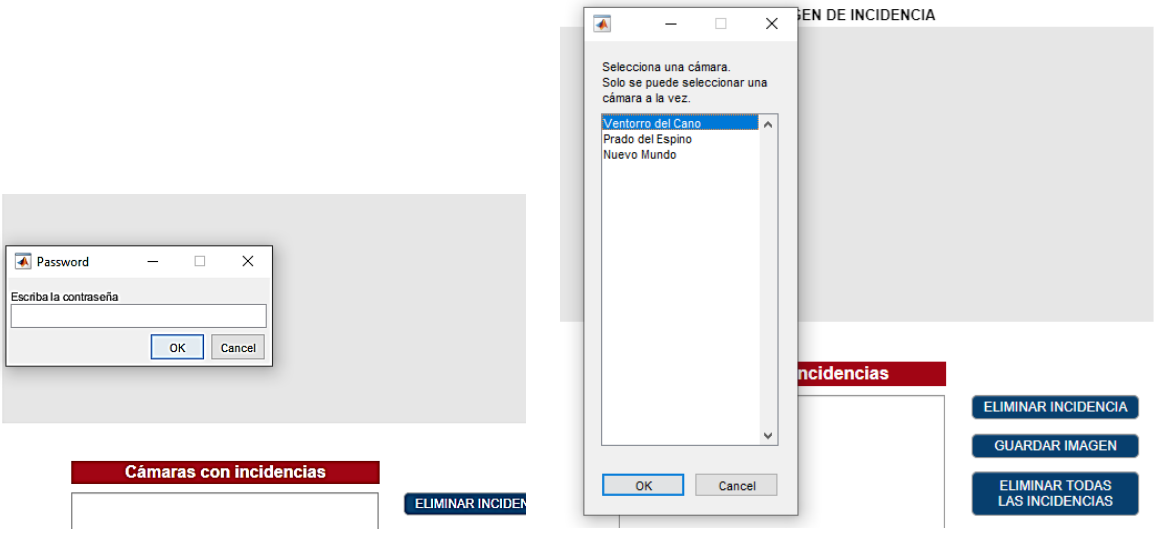

Figura 35: Elección de la cámara a modificar parámetros

Una vez seleccionada la cámara, aparece una ventana emergente para modificar los valores de los parámetros, con unos valores por defecto, que son los predefinidos al ejecutar la aplicación.

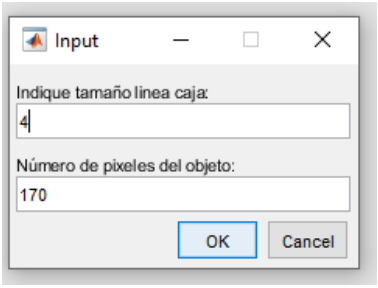

Figura 36: Elección de la cámara a modificar parámetros

El valor por defecto del tamaño de la línea de la caja delimitadora es 1. En la siguiente imagen se puede comprobar que al cambiar este valor a 4, el grosor cambia con respecto a anteriormente.

Figura 37: Modificación del grosor de la línea de la caja delimitadora del objeto

Después, se procede a probar la modificación de la zona de interés de detección de obstáculos en una cámara. Al igual que antes, al seleccionar esta opción, se pide contraseña (para que solo lo pueda modificar un usuario con privilegios) y cámara a la que se quiere modificar la ROI. Una vez seleccionada, aparece una nueva ventana con una imagen fija de la cámara y un cursor. A través de este cursor, se pincha en cada punto de los vértices de la región que se desea dibujar, hasta delimitarla completamente.

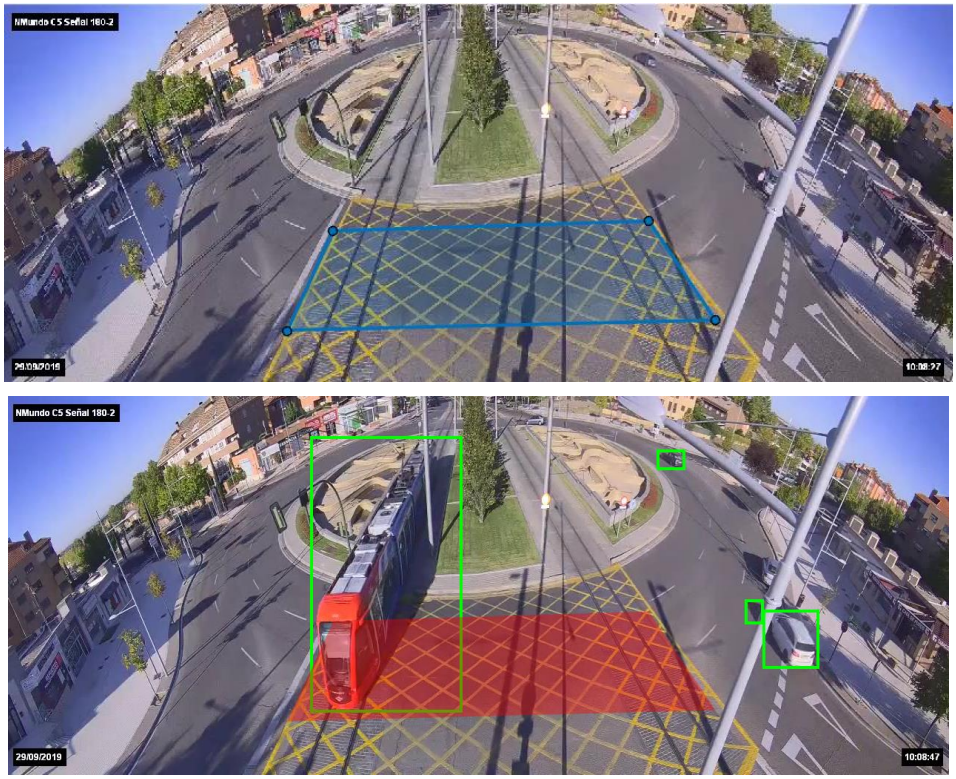

Figura 38: Modificación de la ROI en una cámara (arriba) y visualización posterior (abajo)

Al dibujar una nueva ROI, no solo se modifica en la imagen de visualización, sino también en la imagen binaria (figura 11) que sirve para detectar los objetos en movimiento que invaden dicha región, ya que ambas comparten parámetros. La figura 38 (abajo) muestra la imagen de la cámara de Nuevo Mundo con el grosor de línea de los cuadros delimitadores de los objetos y de la ROI modificados.

7. Conclusiones y líneas de futuro

Una vez concluidas las pruebas de funcionamiento del sistema, se extraen las siguientes conclusiones en función de los objetivos marcados al inicio del proyecto y las líneas a seguir en el futuro para mejorar la aplicación, hacerla más robusta y que pueda funcionar en cualquier explotación tranviaria y bajo cualquier circunstancia.

7.1. Conclusiones

Conclusiones extraídas a la vista de los resultados obtenidos a la finalización de las pruebas realizadas a la aplicación, los objetivos estipulados y las especificaciones iniciales del producto:

- Se cumple el objetivo de analizar una secuencia de video y detectar los objetos en movimiento que aparecen en ella, diferenciándolos del resto de la imagen.
- El sistema lanza correctamente una serie de señales de alarma cuando el objeto detectado invade la zona de interés de detección en el momento en que la señalización no lo permite.
- La aplicación desarrollada es capaz de detectar objetos de cualquier tamaño y forma, si bien no detecta un objeto en el momento en que este está parado. También es capaz de detectar dichos objetos en las condiciones de iluminación y condiciones climáticas de los videos, pero queda pendiente analizar imágenes en circunstancias adversas.
- Los tiempos de ejecución obtenidos son suficientemente rápidos para los tiempos que maneja la explotación MLO, alrededor de 1 segundo entre que se detecta el objeto y saltan las alarmas. No se sabe si estos tiempos son debidos a la propia implementación del código y los objetos de sistema utilizados o por la capacidad de procesamiento del ordenador en el que se han realizado las pruebas. No obstante, no hay que olvidar que la conducción de un tranvía es “conducción a la vista” y que la aplicación no está pensada para evitar colisiones inminentes.
- El sistema es capaz de adaptarse a la orientación de la cámara y estimar una región de interés de formas diferentes y en cualquier lugar de la imagen, detectando los objetos en movimiento que pasen por ella.
- La aplicación obtiene información de las incidencias detectadas, ofreciendo la posibilidad de guardar la imagen del momento de la infracción. Con ello se consigue tener una base de datos de incidencias para obtener estadísticas de infracciones y otras utilidades.
- La interfaz es simple e intuitiva, como se pedía por parte del usuario final, si bien es mejorable en otros aspectos.
- La aplicación funciona correctamente en ordenadores con *MATLAB* instalado, pero no se ha podido probar en ordenadores sin este entorno de trabajo.

- La planificación ha sido adecuada y se ha seguido sin demasiados problemas. Las reuniones mantenidas con los responsables de Metro Ligero Oeste no se hicieron en las fechas previstas inicialmente, pero ello no ocasionó ningún inconveniente ni retraso en el desarrollo del producto. No obstante, se intentó acaparar más grabaciones de video de distintas ubicaciones y con condiciones de iluminación diferentes, pero debido a la política de protección de datos de MLO no fue posible. Se decidió, entonces, dar prioridad a que fueran grabaciones de pasos a nivel conflictivos en cuanto número de incidencias, debido a que estas incidencias se dan más en hora punta de la mañana y de la tarde y, por tanto, es donde más partido se le puede sacar al sistema.

En general, se puede estimar que el sistema desarrollado cumple con los objetivos y especificaciones marcadas al inicio del proyecto.

7.2. Líneas de futuro

A continuación, se exponen algunas mejoras a implementar en futuras versiones del producto, para hacerlo más robusto, flexible, atractivo y funcional. Debido al tipo de sistema desarrollado, las mejoras y actualizaciones en futuras fases del proyecto son una parte muy importante de este.

- En cuanto a la técnica de detección de obstáculos, si bien el método de segmentación basada en movimiento utilizada en este proyecto cumple con los objetivos marcados, se deberán investigar métodos más complejos y que requieren más tiempo de implementación, como los basados en *Deep Learning y Machine Learning,* para comprobar si estos resultan más adecuados.
- Se deberá probar la aplicación en circunstancias de iluminación y climatología adversas, con el fin de comprobar el comportamiento del sistema y ver las dificultades que puedan surgir en cuanto a detección de los objetos. Por otra parte, se podrían hacer pruebas con cámaras de visión nocturna e incluso térmicas y ver los resultados obtenidos.
- Se deberán adoptar técnicas de detección de objetos inmóviles en ciertas cámaras del sistema de videovigilancia de MLO, como en las de las entradas a los túneles. De esta forma, se detectarán objetos olvidados o abandonados en mitad de la vía.
- Se deberán hacer pruebas de funcionamiento en ordenadores más rápidos y potentes, para comprobar si el tiempo de ejecución de la aplicación disminuye.
- En cuanto a la aplicación y la interfaz de usuario de esta, se buscará la manera de que funcione en ordenadores sin *MATLAB* instalado, para ahorrar costes y hacerla más flexible. Por otro lado, se analizarán otros diseños gráficos, para hacerla más atractiva visualmente.

Bibliografía

[1]: **Metro Ligero Oeste website**: https://www.metroligero-oeste.es/, consultado 03/01/2020.

[2]: **MathWorks (MATLAB) website**: https://es.mathworks.com/?s_tid=gn_logo consultado 18/12/2019.

[3]: **Documentación oficial inicial del proyecto de la explotación de Metro Ligero Oeste S.A.** *Diseño de sistema de videovigilancia.* Novation Security Systems, S.L. Rivas Vaciamadrid (Madrid).

[4]: **Documentación oficial inicial del proyecto de la explotación de Metro Ligero Oeste S.A.** *Requerimientos de comunicaciones del sistema de señalización. Alstom (2005) Madrid.*

[5]: **A.M. Romero and M. Cazorla.** *Comparativa de detectores de características visuales y su aplicación al SLAM*. (Setiembre 2009), Cáceres.

[6]: **Wikipedia website**: https://es.wikipedia.org/wiki/Aprendizaje_autom%C3%A1tico, consultado 15/10/2019

[7]: **Wikipedia website**: https://es.wikipedia.org/wiki/Aprendizaje_profundo, consultado 15/10/2019

[8]: **Learn OpenCV:** https://www.learnopencv.com/understanding-alexnet/, consultado 15/10/2019

[9]: **Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed,Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich**. Going Deeper with Convolutions, (2015)

[10]: **Open CV website**: https://opencv.org/, consultado 14/10/2019

[11]: **IVT website**: http://ivt.sourceforge.net/index.html, consultado 14/10/2019

[12]: **Mermec website**: http://www.mermecgroup.com/es/pageview2.php?i=1033&sl=1, consultado 03/01/2020

[13]: **Universidad de Alcalá we**bsite: https://www.uah.es/es/investigacion/servicios-para-empresas/Oferta-Cientifico-Tecnologico/Sistema-sensor-para-la-deteccion-de-objetos-obstaculos-en-puntos-criticos-de-lineas-ferreas./, consultado 04/01/2020

[14]: **Musleh, Basam; Armingol, José María; de la Escalera, Arturo**. *Detección de obstáculos en el entorno de carretera mediante visión por computador para ADAS*. Plataforma Tecnológica Española de la Carretera (PTC). (2015), Madrid.

[15]: **MathWorks (App Designer) website:** https://es.mathworks.com/help/matlab/app-designer.html?searchHighlight=app%20designer&s_tid=doc_srchtitle, consultado 10/10/2019

[16]: **Kaewtrakulpong, P. and R. Bowden**. *An Improved Adaptive Background Mixture Model for Realtime Tracking with Shadow Detection.* In Proc. 2nd European Workshop on Advanced Video Based Surveillance Systems, AVBS01, VIDEO BASED SURVEILLANCE SYSTEMS: Computer Vision and Distributed Processing (September 2001)

Danilo García Santillán, Iván *Visión Artificial y Procesamiento Digital de Imágenes usando Matlab*. Ibarra, (2008) Ecuador.

González, Rafael C.; Woods, Richard E. *Digital Image Processing*. Second Edition. Upper Saddle River, Nueva Jersey: Prentice Hall. (2002)

González, Rafael C.; Woods, Richard E.; Eddins, Steven L. *Digital Image Processing Using MATLAB*®. Upper Saddle River, Nueva Jersey: Prentice Hall. (2004)

José Antonio Morán Moreno (coordinador); Sílvia Pujalte Piñán, Verónica Vilaplana Besler. *Procesado de imagen - Recursos de aprendizaje: Codificación del sonido y de la imagen* [Recurs electrònic]. Barcelona : UOC, 2009. Recursos en línea.

Anexos

Anexo A: Glosario

A continuación, se muestra una serie de abreviaturas que aparecen en algún momento en el texto del documento y su significado.

CCTV: Circuito Cerrado de Televisión (Closed Circuit Televisión)
CNN: Red Neuronal Convolucional (Convolutional Neural Network)
CPU: Central Processing Unit
EE: Elemento estructurante
HD: Alta Definición (High Definition)
LED: Light Emitting Diode
MLO: Metro Ligero Oeste
PCC: Puesto Central de Control
PEC: Prueba de Evaluación Continua
PoE: Power over Ethernet
ROI: Región de interés (Region Of Interest)
SAE: Sistema de Ayuda a la Explotación
SQL: Structured Query Language
TIC: Tecnologías de la Información y Comunicación
UOC: Universitat Oberta de Catalunya

Anexo B: Entregables del proyecto

El presente proyecto se compone de los siguientes entregables:

- Software, generado con *MATLAB y App Designer*, del sistema de detección de objetos
- Manual de usuario, que explica las funciones de cada objeto de la interfaz gráfica y su funcionamiento.
- Archivos necesarios para la instalación y correcto funcionamiento de la aplicación, como pueden ser videos, código de las funciones y otros archivos que componen el sistema.
- Documentación, en formato html, de las distintas funciones que componen el conjunto del software.

Anexo C: Código de las funciones del sistema de detección de objetos

A continuación, se expone el código generado de las funciones más importantes del sistema desarrollado.

Iniciar Video y sistema de detección

```
% Se lee el video y se almacena en el objeto de sistema videoReader
videoReader = vision.VideoFileReader('NMRap.mp4');
% Objeto de Sistema para visualizar video
videoPlayer = vision.VideoPlayer;

% Se crea objeto de sistema. Algoritmo de eliminación del fondo. Crea máscara binaria

foregroundDetector = vision.ForegroundDetector('NumGaussians', 3,'NumTrainingFrames',
50);

foregroundDetector2 = vision.ForegroundDetector('NumGaussians',
3,'NumTrainingFrames', 50);

%Leemos los primeros 75 fotogramas para que el sistema aprenda o detecte el fondo
%Lo hacemos dos veces para que detecte los objetos en toda la pantalla y no solo en la
%región de interés

% Coordenadas de la región de interés
app.xNM;
app.yNM;
% Se crea la máscara de la región de interés y se pasa a single para operar
% con ella
bw = single(poly2mask(app.xNM,app.yNM,512,1280));

for i = 1:75
    app.videoFrame = step(videoReader);
    foreground = step(foregroundDetector,app.videoFrame);
end

for i = 1:75
    app.videoFrame = step(videoReader);

    zonaInteres = bw .* app.videoFrame; % Se crea una imagen en la que la zona fuera
de la región de interés es cero (negro)
    fondo = step(foregroundDetector2,zonaInteres);
end

% Se realiza filtrado morfológico para eliminar ruido
cleanForeground = imopen(foreground, strel('Disk',1));
limpiar = imopen(fondo,strel('Disk',1));

%Análisis de componentes segmentados. Objeto de sistema BlobAnalysis
%Agrupar pixeles en movimiento para poder segmentar objetos en movimiento
blobAnalysis = vision.BlobAnalysis('BoundingBoxOutputPort', true, ...
    'AreaOutputPort', false, 'CentroidOutputPort', false, ...
    'MinimumBlobArea', app.pixelNM);

blobAnalysis2 = vision.BlobAnalysis('BoundingBoxOutputPort', true, ...
    'AreaOutputPort', false, 'CentroidOutputPort', false, ...
    'MinimumBlobArea', app.pixelNM);
```

```
% Se recorre todo el video hasta el final
while  ~isDone(videoReader)

    % Leemos el video frame a frame
    app.videoFrame = step(videoReader);

    zonaInteres = bw .* app.videoFrame;


    % Se detectan los pixeles del fondo de ambas imágenes
    foreground = step(foregroundDetector,app.videoFrame);

    fondo = step(foregroundDetector2,zonaInteres);

    % Filtro morfológico para eliminar ruido
    cleanForeground = imopen(foreground, strel('Disk',1));

    limpiar = imopen(fondo, strel('Disk',1));

    % Se detectan los componentes conectados con un área mínimo específico
    % y se calculan sus cuadros delimitadores

    bbox = blobAnalysis(cleanForeground);

    caja = blobAnalysis2(limpiar);

    % Área para dibujar, en la visualización de la imagen, la región de
    % interés. De esta forma vemos si el sistema funciona y detectamos
    % si la cámara se ha descolocado de su posición original

    app.areaNM;

    % Se dibujan cajas de color verde alrededor de los objetos detectados
    result = insertShape(app.videoFrame, 'Rectangle', bbox, 'Color', 'green',
'LineWidth', app.lineaNM);
    result1 =
insertShape(result,'FilledPolygon',app.areaNM,'Opacity',0.4,'LineWidth',2,
'Color','red');


    % Se cuentan los objetos en la región de interés
    numCars = size(caja, 1);

    % Si se detecta un objeto y el semáforo está en rojo, salta alarma, se
    % muestra la foto del instante y la información de fecha y hora
    if (numCars >= 1)

       if (app.sem == 1)

            app.foto{app.i} = app.videoFrame;
            app.Alarma.Color = 'red';
            sound(app.audio,app.Fs);
            imshow(app.foto{app.i},'Parent',app.UIAxes);
             app.sem = 0;
             date = datestr(datetime('now'));
             app.lista = {app.input};
             app.input = "Nuevo Mundo ";
             app.lista{app.i} = app.i + " " + app.input + date;
             app.existe = 1;

            if app.i ==1
                app.Camaras.Value = [app.lista{app.i}];
```

```
            else

                app.Camaras.Value = [app.Camaras.Value ; app.lista{app.i}];

            end
            app.i = app.i + 1;
        end

    end

    % Visualización de la cámara elegida
    step(videoPlayer,result1);

  end
```

Eliminar incidencia

```
%Esta función elimina la última captura del frame con la incidencia y la información
%de la cámara, fecha y hora
if app.existe == 1

app.i = app.i-1;

if app.i >= 2

            app.f = app.foto{app.i-1};
            imshow(app.f,'Parent',app.UIAxes);
            app.Camaras.Value(app.i,:) = [];

            else
                app.Camaras.Value = '';
                imshow(app.UIAxes.BackgroundColor,'Parent',app.UIAxes);
                app.Alarma.Color = 'green';
                app.existe = 0;
            end
     else
                msgbox('No hay incidencia que borrar','Error de acción','Error');
end
```

Eliminar todas las incidencias

```
%Esta función elimina todas las capturas de frames con incidencia y la información de
%todas las cámaras, fecha y hora
%Se pone a 1 el índice de la lista de frames, se borra todo, se pone en verde la señal
%de alarma y a cero el indicador de incidencias

app.i = 1;

  app.Camaras.Value = '';
  imshow(app.UIAxes.BackgroundColor,'Parent',app.UIAxes);
  app.Alarma.Color = 'green';
  app.existe = 0;
```

Guardar imagen de la incidencia

```
%Llamar a interface windows que permite guardar un fichero en la ubicación elegida.
%Se guarda en el formato elegido por el usuario

if app.existe == 1
    imagen = app.foto{app.i-1};
            [fichero, p, formato]= uiputfile({'*.jpg';'*.tif';'*.png';'*.gif';'*.*'},
'Guardar como');
            save=[p fichero];
            switch formato
                 case 1
                          imwrite(imagen,save,'jpg' )
                 case 2
                          imwrite(imagen,save,'tif')
                 case 3
                          imwrite(imagen,save,'png')
                 case 4
                          imwrite(imagen,save,'gif')
            end
            else
                msgbox('No hay imagen para guardar','Error de acción','Error');
            end
```

Cambiar parámetros

```
%Función que permite al usuario elegir el grosor de la línea delimitadora de las cajas
%de los objetos y de la agrupación de pixeles en cada cámara
%Se muestra un cuadro de diálogo donde el usuario inserta los valores de grosor de línea
%delimitadora del objeto y la agrupación de pixeles

switch app.indx
                case 1
                    prompt = {'Indique tamaño linea caja:','Número de pixeles del
objeto:'};
                    dlgtitle = 'Input';
                    dims = [1 35];
                    definput = {'1','300'};
                    answer = inputdlg(prompt,dlgtitle,dims,definput);

                    % Se pasan los caracteres insertados a números, para poder
                    % operar con ellos
                    user_val = str2num(answer{1})
                    app.lineaVC = user_val;
                    user_val = str2num(answer{2})
                    app.pixelVC = user_val;

                case 2
                    prompt = {'Indique tamaño linea caja:','Número de pixeles del
objeto:'};
                    dlgtitle = 'Input';
                    dims = [1 35];
                    definput = {'1','150'};
                    answer = inputdlg(prompt,dlgtitle,dims,definput);

                    % Se pasan los caracteres insertados a números, para poder
                    % operar con ellos
                    user_val = str2num(answer{1})
                    app.lineaPE = user_val;
```

```
                    user_val = str2num(answer{2})
                    app.pixelPE = user_val;

                case 3
                    prompt = {'Indique tamaño linea caja:','Número de pixeles del
objeto:'};
                    dlgtitle = 'Input';
                    dims = [1 35];
                    definput = {'1','170'};
                    answer = inputdlg(prompt,dlgtitle,dims,definput);

                    % Se pasan los caracteres insertados a números, para poder
                    % operar con ellos
                    user_val = str2num(answer{1})
                    app.lineaNM = user_val;
                    user_val = str2num(answer{2})
                    app.pixelNM = user_val;

            end
```

Determinar o modificar ROI

```
%Función que permite elegir la zona de interés de detección de obstáculos
%Se leen los primeros frames de la cámara deseada y visualizamos el último, dibujamos
%el polígono en la posición deseada y guardamos las coordenadas para elegir la zona de
%interés

switch app.indx
                case 1
                    videoR = vision.VideoFileReader('vacRap.mp4');
                    for i = 1:10
                     videoF = step(videoR);
                    end
                    imshow(videoF);
                    h = drawpolygon;
                    app.xVC = [h.Position(1,1) h.Position(2,1) h.Position(3,1)
h.Position(4,1)];
                    app.yVC = [h.Position(1,2) h.Position(2,2) h.Position(3,2)
h.Position(4,2)];
                    app.areaVC = [h.Position(1,1) h.Position(1,2) h.Position(2,1)
h.Position(2,2) h.Position(3,1) h.Position(3,2) h.Position(4,1) h.Position(4,2)];
                    close;

                case 2
                    videoR = vision.VideoFileReader('PES3Rap.mp4');
                    for i = 1:10
                     videoF = step(videoR);
                    end
                    imshow(videoF);
                    h = drawpolygon;
                    app.xPE = [h.Position(1,1) h.Position(2,1) h.Position(3,1)
h.Position(4,1)];
                    app.yPE = [h.Position(1,2) h.Position(2,2) h.Position(3,2)
h.Position(4,2)];
                    app.areaPE = [h.Position(1,1) h.Position(1,2) h.Position(2,1)
h.Position(2,2) h.Position(3,1) h.Position(3,2) h.Position(4,1) h.Position(4,2)];
                    close;

                    case 3
                    videoR = vision.VideoFileReader('pruebaRap.mp4');
```

```
                    for i = 1:10
                     videoF = step(videoR);
                    end
                    imshow(videoF);
                    h = drawpolygon;
                    app.xNM = [h.Position(1,1) h.Position(2,1) h.Position(3,1)
h.Position(4,1)];
                    app.yNM = [h.Position(1,2) h.Position(2,2) h.Position(3,2)
h.Position(4,2)];
                    app.areaNM = [h.Position(1,1) h.Position(1,2) h.Position(2,1)
h.Position(2,2) h.Position(3,1) h.Position(3,2) h.Position(4,1) h.Position(4,2)];
                    close;
            end
```