UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

TRABAJO FIN DE GRADO

DETECCIÓN DE ANOMALÍAS EN TIEMPO REAL

Ana Huélamo Vallejo
Tutor: Luis Alberto Caro Campos
Ponente: José María Martínez Sánchez

Julio 2014

DETECCIÓN DE ANOMALÍAS EN TIEMPO REAL

Ana Huélamo Vallejo
Tutor: Luis Alberto Caro Campos
Ponente: José María Martínez Sánchez

Video Processing and Understanding Lab
Departamento de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Julio 2014

Trabajo parcialmente financiado por el gobierno español bajo el proyecto
TEC2011-25995(EventVideo)

RESUMEN Y PALABRAS CLAVE

Resumen

Este trabajo tiene como objetivo la integración de un algoritmo de detección de anomalías en tiempo real en una plataforma destinada al trabajo con aplicaciones de vídeo vigilancia denominada DiVA. Con esta integración se pretende modificar la funcionalidad del algoritmo para que el mismo capaz de procesar imágenes recibidas de videocámaras en tiempo real sin comprometer el resto de su funcionamiento.

Por otro lado se quiere desarrollar una interfaz de gráfica que facilite el manejo de dicho algoritmo, tanto a usuarios cualificados en el manejo de algoritmos de tratamiento de vídeo, como a usuarios sin conocimientos previos del tema en cuestión. La interfaz deberá ajustarse a las necesidades propias del algoritmo y de los futuros usuarios.

Para llevara a cabo dicho trabajo se ha necesitado obtener conocimientos de varios campos previamente a la implementación de cada una de sus partes. En primer lugar se ha estudiado el algoritmo dado para su integración y todos los aspectos relativos del mismo para su integración en la plataforma y el desarrollo de la interfaz adecuada al mismo. De igual manera se han estudiado las distintas plataformas y herramientas necesarias para el desarrollo del conjunto.

Una vez completados los objetivos principales se pasó a realizar el estudio y evaluación de los resultados obtenidos mediante una serie de pruebas realizadas en entornos adecuados al funcionamiento de la aplicación.

Esta memoria es el pretende ser el reflejo del trabajo llevado a cabo durante los meses de duración del mismo a la vez que intenta ayudar al lector en la compresión de cada uno de los elementos que componen dicho trabajo.

Palabras clave

Detección de anomalías, detección de eventos, vídeo-vigilancia, procesado de vídeo, interfaz gráfica, evaluación.

ABSTRACT AND KEYWORDS

Abstract

This paper aims at integrating an anomaly detection algorithm real time on a platform designed to work with videosurveillance applications called DiVA. With this integration it intend to modify the functionality of the algorithm so that it can process images received from video in real time without compromising the rest of its operations.

On the other hand it want to develop a GUI to make easier the management of this algorithm, both skilled user in handling video processing algorithms, and user without knowledge of the subject matter. The GUI shall adjust to the own needs of both the algorithm and the future users.

To carry out this work, it's been necessary to obtain knowledge of several fields, before to start developing each of its parts. First of all, the algorithm has been studied and all of its parts for the integration. Same way, it's been studied all the different platforms and tools needed for the develop of the set.

Once all the main objectives were completed it proceeded to study and evaluation of the results through a set of test performed in appropiate enviroments.

This report is intended to be reflect of the work carried out during the duration months in same way it tries to help the reader in understanding of each ones of the elements of the work.

Keywords

Anomaly detection, event detection, videosurveillance, video processing, GUI, evaluation.

AGRADECIMIENTOS

En primer lugar quiero agradecer a Chema la oportunidad de dejarme participar en este trabajo y de formar parte del VPU. También agradecer toda la ayuda y el apoyo de mi tutor, Luis, a lo largo de todo el proceso, ya que sin él este trabajo no hubiera sido posible.

Gracias a todos los miembros del VPU por crear un ambiente de trabajo tan agradable y estar siempre dispuestos a ayudar, especialmente a Carlos, cuya ayuda y paciencia han sido inestimables.

A todo el colectivo de profesores por guiarnos y enseñarnos a lo largo de estos años y por darnos un trato cercano con el que todos nos hemos sentido muy cómodos, especialmente Jesús, Chema y Luis por su apoyo y ayuda los últimos años.

A mis compañeras de clase, Raquel, Miriam y sobretodo Nieves, el hacer de mi día a día más liviano y agradable. Sin olvidarme de Jorge, que gracias a su apoyo, su insistencia y su compañía me ha ayudado a llegar hasta aquí.

Por último, gracias a mi familia, especialmente a mi madre por animarme cuando más lo he necesitado.

Gracias a todos.

Ana Huélamo Vallejo.

Julio 2014

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

GLOSARIO DE ACRÓNIMOS

PTZ: Pan-Tilt-Zoom

DiVA: Distributed Video Analysis

CCTV: Closed Ciruit Television

VCR: Video Cassette Recorder

DVR: Digital Video Recorder

WLAN: Wireless Local Area Network

MPEG-2: Moving Picture Expert Group 2

H.264: MPEG-4

TCP: Transmission Control Protocol

UDP: User Datagram Protocol

RTP: Real-time Transport Protocol

RTSP: Real Time Streaming Protocol

NVR: Network Video Recorder

ONVIF: Open Network Video Interface Forum

PSIA: Physical Security Interoperability Alliance

VPU-Lab: Video Processing and Understanding Lab

BGS: Background Substraction

FIFO: First In First Out

BSD: Berkeley Software Distribution

QML: Qt Meta Language

FPS: Frames Per Second

1 INTRODUCCIÓN

1.1 Motivación

En la actualidad, el interés por la seguridad y el control personal mediante sistemas de vigilancia se encuentra a la orden del día y se observa como estos sistemas forman parte del día a día de muchas personas. El conflicto entre la seguridad y el respeto a la privacidad cada vez está más controlado y los sistemas de vídeo-vigilancia se encuentran cada día más asentados en la sociedad. En numerosos lugares se pueden observar diferentes tipos de cámaras con diferentes finalidades, control de tráfico, control anti-robos y demás aplicaciones, tanto en el sector público como en el privado.

La constante demanda de este tipo de sistemas ha dado lugar a la necesidad de implantar soluciones de apoyo al personal encargado de supervisar visualmente estos sistemas. En este punto se encuentran los sistemas de tratamiento de señal, que mediante distintos algoritmos, ofrecen un tratamiento de vídeo automático, previo a su visualización, de gran ayuda para el personal de vigilancia.

Es en estos sistemas automáticos donde se encuentra en la actualidad el auténtico interés por la investigación y la mejora de los servicios de vídeo vigilancia. En las últimas décadas, la constante investigación en este campo ofrece variadas y novedosas soluciones a estas cuestiones pudiendo realizar aplicaciones para un gran número de situaciones y necesidades.

La creación de sistemas de fácil e intuitivo manejo es igualmente interesante que los resultados ofrecidos por los mismos sistemas a la hora de atraer a un público interesado en este tipo de aplicaciones. Hay que tener en cuenta, que en el campo de la vídeo-vigilancia, los responsables del control de los sistemas destinados a estas aplicaciones no siempre cuentan con los conocimientos del tratamiento de vídeo o el trabajo con algoritmos, por este motivo, es igualmente importante la creación de aplicaciones para un público general.

1.2 Objetivos

El principal objetivo de este trabajo es desarrollar una aplicación con interfaz gráfica con las características de un sistema de vídeo vigilancia actual, que ofrezca unos resultados fiables y cuente con un diseño atractivo y de fácil manejo, ya sea para personal experimentado en el tratamiento de vídeo, como para personal no cualificado en este campo.

Más concretamente se quiere realizar una aplicación que funcione recibiendo imágenes de una cámara de vídeo, procese dichas imágenes, y sea capaz de localizar observaciones anómalas con el fin de alertar de forma visual al personal encargado del manejo de la aplicación.

Para ello se han establecido una serie de pautas necesarias para conseguir el objetivo final.

En primer lugar, se ha seleccionado un algoritmo de procesado de vídeo con tales fines, para su integración en sistema de vídeo-vigilancia distribuido, habiendo sido necesario profundizar en el conocimiento de dicho algoritmo. De esta manera se han identificado los aspectos necesarios a tener en cuenta en el manejo del mismo que deberán ser incluidos en la aplicación final, desde los resultados a mostrar, hasta las cuestiones que un usuario debe conocer y/o configurar para su correcto funcionamiento.

Ha sido necesario familiarizarse con el entorno de trabajo en el que se ha desarrollado la totalidad del trabajo y las herramientas a utilizar para el mismo con el fin de realizar la tarea de forma correcta y eficiente.

El algoritmo seleccionado ha sido integrado en una plataforma existente para su posterior uso como un sistema de vídeo vigilancia sin que los resultados se hayan visto afectados y cumpliendo con la necesidad de ejecución en tiempo real, necesaria para su aplicación en situaciones reales.

Para ello, se ha diseñado una interfaz de usuario que permita al usuario final un manejo fácil e intuitivo de la aplicación. Esta aplicación debe tener en cuenta los aspectos necesarios para la configuración del algoritmo, y permitir visualizar los resultados generados por éste.

Por último, se han realizado pruebas para asegurar la estabilidad de la aplicación, la fiabilidad de los resultados del algoritmo en un entorno real, y su aplicabilidad para el análisis en tiempo real.

En definitiva, como resultado final de este trabajo se pretende disponer de un algoritmo para la detección de anomalías integrado en una plataforma distribuida diseñada para sistemas de vídeo-vigilancia, que cuente con una interfaz de usuario que ayude al manejo del algoritmo general y cuyos resultados sean fiables y de utilidad al usuario final.

1.3 Estructura de la memoria

Esta memoria es el reflejo del trabajo llevado a cabo. En ella se dan a conocer los aspectos del trabajo teórico y práctico realizados para la finalización del mismo explicados de la siguiente manera:

- Capítulo 1. Introducción. Motivación y objetivos. En este primer capítulo se da una visión general del trabajo y se exponen los objetivos principales y la estructuración de la memoria.
- Capítulo 2. Estado del arte. Conocimiento del estado actual de soluciones integradas para el análisis de eventos en entornos de vídeo-vigilancia.
- Capítulo 3. Se da a conocer el algoritmo dado para su integración y se exponen sus características principales para entender su funcionamiento.
- Capítulo 4. Descripción del funcionamiento de la plataforma distribuida DiVA, destinada a la integración de algoritmos para la creación de sistemas de vídeo vigilancia.
- Capítulo 5. Descripción del desarrollo llevado a cabo por el estudiante a lo largo del transcurso del trabajo.
- Capítulo 6. Realización de pruebas y entorno seleccionado para las mismas.
- Capítulo 7. Conclusiones obtenidas y trabajo futuro. Exposición de las conclusiones extraídas del desarrollo total en el transcurso del trabajo y líneas generales para un trabajo futuro.

- Bibliografía

2 ESTADO DEL ARTE

En este capítulo se quiere poner en conocimiento el estado actual de sistemas de similares características a los utilizados para la realización de este trabajo con el fin de introducir al lector en la materia y asentar las bases para comprender posteriormente el funcionamiento del programa.

2.1 Sistemas de vídeo vigilancia

La implantación de sistemas de vídeo-vigilancia tal y como se conocen hoy en día comienza en la década de los 70, con la aparición de sistemas de almacenaje de vídeo sobre cintas [1]. Desde entonces, los sistemas en los que se basan han evolucionado de forma constante, desde sistemas puramente analógicos, hasta los sistemas inteligentes que conocemos en la actualidad, con distribución puramente digital, gracias a los constantes avances tecnológicos y al abaratamiento de los costes.

Concretamente, podemos distinguir entre varias generaciones de sistemas de vídeo vigilancia, de acuerdo a su integración en el mundo digital y sus características[2].

- **Primera generación**: circuitos cerrados de televisión (CCTV), conectados a través de cable coaxial a un multiplexor y a una grabadora de vídeo analógico VCR en soporte físico (cintas). En estos sistemas, la captura, la distribución y la grabación se realiza de forma totalmente analógica. La monitorización se realiza visualmente, ya sea en vivo o revisando material almacenado previamente.
- **Segunda generación**: circuitos cerrados de televisión (CCTV), conectados a través de cable coaxial a una digitalizadora/grabadora de vídeo (DVR). En estos sistemas, la digitalización facilita el almacenamiento en medios digitales, y la distribución directamente a ordenadores. En particular, al tratarse de vídeo digital, facilita la búsqueda de eventos de forma automática utilizando algoritmos de visión artificial.
- **Tercera generación**: sistemas donde la captura, la distribución y la grabación se realiza de forma completamente digital sobre redes IP. Esto permite descentralizar los distintos componentes, facilitando a su vez el uso de redes multi-sensor (multicámaras, sensores de movimiento y proximidad, audio, etc.).

En la **primera generación**, las captura y distribución se realiza de forma analógica y los resultados se graban directamente con un VCR sin compresión, lo que limita la cantidad de horas que se pueden grabar [3]. Una de las grandes desventajas de estos sistemas es que requieren intervención manual periódica para cambiar las cintas donde se graba el vídeo [4]. En instalaciones con múltiples cámaras, se utiliza un multiplexor para poder realizar la monitorización utilizando una única pantalla. La mayor limitación de los sistemas de esta generación se debe a la incapacidad de poder aplicar un procesamiento de la imagen que ayude a los usuarios con las tareas de vigilancia, ya que su capacidad de atención disminuye con el tiempo al visualizar información preveniente de múltiples cámaras [5].

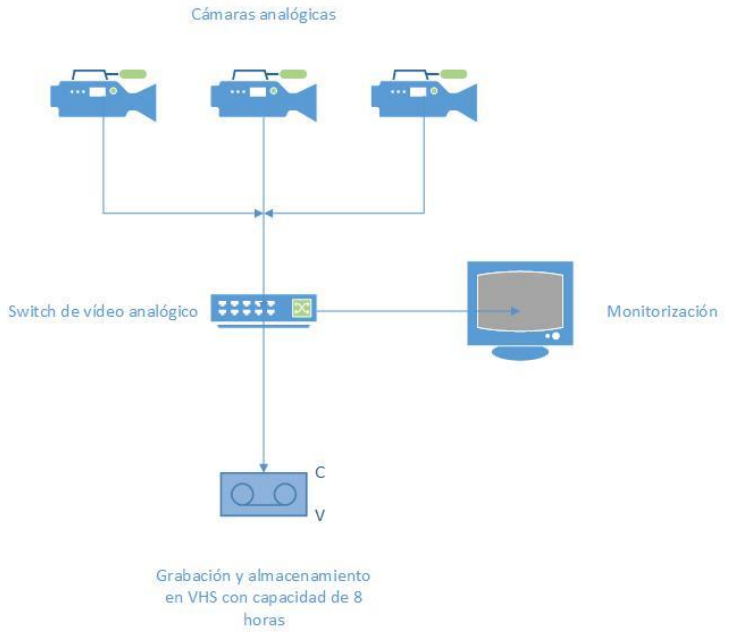


Figura 2.1: Sistema de vídeo-vigilancia de primera generación

En la **segunda generación**, las cámaras continúan siendo analógicas, sin embargo, como paso previo a la distribución de vídeo, se digitaliza la señal de vídeo al tiempo que se comprime en el módulo DVR. Esta digitalización y compresión permitió comenzar con el procesamiento de vídeo utilizando algoritmos de visión artificial, con el objetivo de apoyar la tarea del personal de vigilancia [3]. Sin embargo, el desarrollo de algoritmos de análisis se encuentra en etapas tempranas, limitándose a determinadas tareas que no siempre se pueden realizar en tiempo real debido a la capacidad de procesamiento de ordenadores, como reconocimiento facial y seguimiento de objetos. Al surgir esta generación, la capacidad de los discos duros era limitada, por lo que, aunque se pudiera grabar vídeos de distintas cámaras a calidad constante, la limitación del sistema se encontraba en la duración de las grabaciones [3].

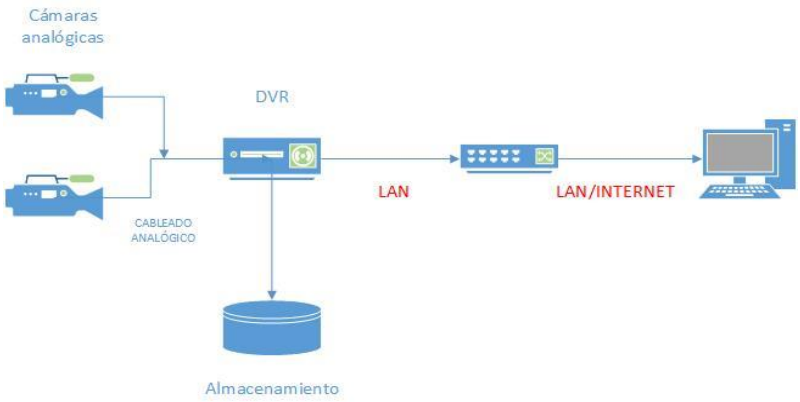


Figura 2.2: Sistema de vídeo-vigilancia de segunda generación

En **la tercera generación**, la digitalización de todos los componentes (captura, distribución, grabación, visualización y análisis) ha permitido pasar de sistemas centralizados a sistemas distribuidos escalables. Esto permite gestionar un mayor número de cámaras y de algoritmos sin que el conjunto del sistema se vea afectado. Los avances tecnológicos y el abaratamiento de los costes han permitido que las cámaras incorporen cada vez más funcionalidad:

cámaras PTZ (*pan-tilt-zoom*), digitalización, compresión y distribución en red (cámaras IP), trabajar a múltiples resoluciones bajo demanda, conectividad inalámbrica (WLAN), incorporación de módulos simples de análisis (sensor de movimiento, etc.). Por otra parte, las técnicas de detección de eventos en vídeo seguridad han evolucionado hacia el análisis de escenas complejas en tiempo real. Por ejemplo, podemos encontrar técnicas que realizan seguimiento de personas u objetos [6], [7], detección de objetos robados y abandonados [8], reconocimiento de actividades [8], [9], y detección de eventos anómalos [9], [10].

Esta última generación es la que nos encontramos principalmente en la actualidad.

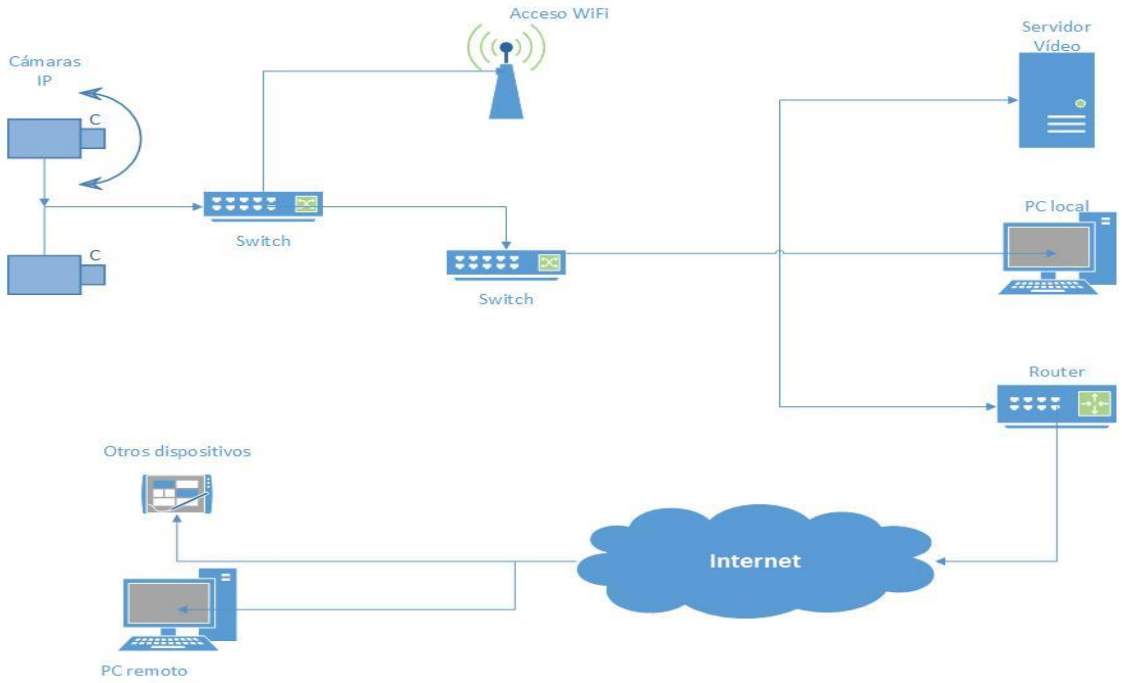


Figura 2.3: Sistema de vídeo-vigilancia de tercera generación

2.2 Arquitectura de sistemas distribuidos de vídeo vigilancia

Como se ha visto en el apartado anterior, la flexibilidad que proporcionan los sistemas distribuidos es fundamental, puesto que han permitido el despliegue de algoritmos de análisis de visión artificial, ya que permiten trabajar con programas pesados computacionalmente, aplicaciones multicámara y el acceso al sistema por parte de múltiples usuarios gracias a la concurrencia de los recursos. Por otra parte, la escalabilidad de los sistemas es necesaria en la actualidad debido al constante crecimiento de los mismos, mediante la incorporación de un mayor número de cámaras, servidores o usuarios.

En función de lo visto hasta ahora, podemos identificar los siguientes módulos de un sistema distribuido de vídeo vigilancia de tercera generación:

Captura: incluye las cámaras y otro tipo de sensores que puedan utilizarse (micrófonos, sensores de luz o movimiento, etc.). Las cámaras, como se ha visto en el apartado anterior, pueden ser analógicas o digitales. Si se trata de cámaras analógicas han de estar conectadas a un módulo que realice la conversión y compresión a vídeo digital. Existen varios formatos de codificación y compresión de vídeo, siendo MPEG-2 y H.264 los más extendidos. A su vez,

existen distintos formatos de encapsulación de vídeo tanto para su transmisión por redes de datos como para su almacenamiento.

Distribución: incluye todos los componentes responsables de la distribución de los servicios, como pueden ser cables *Ethernet* o cableados analógicos coaxiales así como los protocolos utilizados para dicha distribución.

Hardware: incluye todos los dispositivos de gestión de red, tales como *switches*, *routers*, y puntos de acceso WiFi.

Software: para que el personal de vigilancia pueda visualizar las cámaras desde un terminal cliente, o para que los algoritmos reciban la señal proveniente de las cámaras, se utiliza por lo general una arquitectura cliente-servidor, donde un servidor provee a los clientes de los fotogramas provenientes de las cámaras. Para ello, existen diversos protocolos de red para la transmisión de vídeo en tiempo real (RTP/RTSP). Dependiendo de la tecnología de la cámara, este servidor puede estar integrado en la cámara (cámaras IP), en la digitalizadora de vídeo (*network video recorder*, NVR), o en un ordenador conectado directamente a la cámara por interfaz local (USB, *FireWire*). Dependiendo de la aplicación y la tecnología de la cámara, es deseable que el protocolo permita controlar la resolución de la cámara, o los parámetros de la posición y zoom.

Análisis: Para la ejecución de algoritmos de visión artificial que analizan el vídeo para la detección de eventos, es necesario un ordenador con alta capacidad de procesamiento con el fin de que el análisis se realice a tiempo real. Para algoritmos más simples, es posible incorporar módulos de análisis directamente en una cámara IP .

Monitorización y gestión:

Hardware: ordenadores y monitores a través de los cuales se realice la monitorización. Más recientemente, se está extendiendo el uso de dispositivos móviles.

Software: en este caso, es deseable que el software cumpla dos funciones. Por un lado, permitir a los vigilantes seleccionar las cámaras que se desea visualizar, mostrando información relevante resultado de los algoritmos de análisis (alarmas, registro de eventos). Por otro, permitir gestionar distintos parámetros de las cámaras (resolución y *bitrate*, parámetros PTZ, etc.). En caso de que el sistema realice grabación de las cámaras, el software debe permitir seleccionar fácilmente el intervalo de tiempo que se desea visualizar.

Grabación:

Hardware: dispositivos de almacenamiento digital donde se graba el vídeo comprimido proveniente de distintas cámaras.

Software: bases de datos, donde se almacena las alertas generadas por los algoritmos de análisis, así como un registro de eventos.

2.3 Ejemplos de sistemas comerciales y estándares existentes

Como se ha visto en el apartado anterior, un sistema moderno de vídeo vigilancia involucra multitud de dispositivos (captura, red), soluciones software y protocolos. Algunos fabricantes, como AXIS o SONY, ofrecen soluciones completas, que van desde cámaras IP, capturadoras de vídeo analógico, dispositivos de red, hasta *software* de monitorización y gestión.

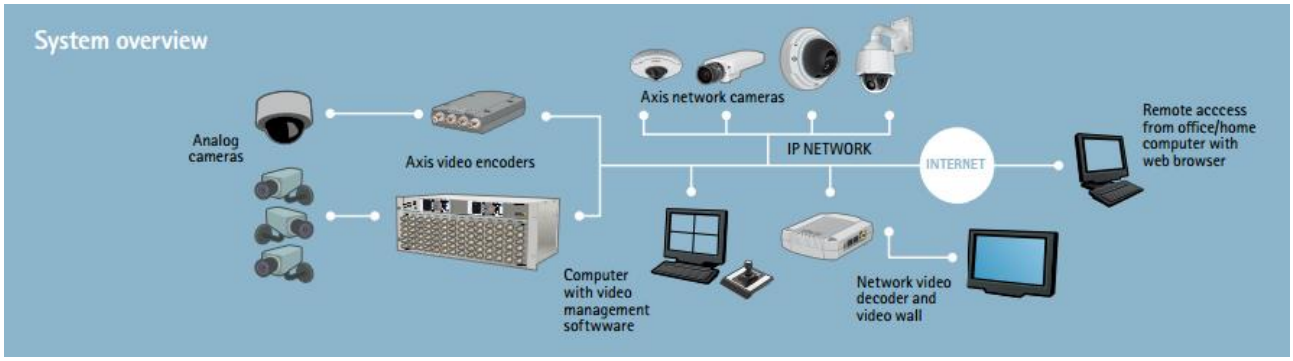


Figura 2.4: Ejemplo de solución completa ofrecido por Axis [11]

La variedad de soluciones en los distintos módulos de un sistema completo dificulta la interoperabilidad entre productos de distintos fabricantes. Más recientemente, existen propuestas de estandarización por parte de empresas del sector, con el fin de establecer estándares de intercambio de información entre los distintos servidores y clientes, así como la integración con sistemas de control de acceso. Como ejemplos podemos encontrar ONVIF[1] (*Open Network Video Interface Forum*) y PSIA[2] (*Physical Security Interoperability Alliance*), que actualmente compiten por definir estándares en las siguientes áreas:

Gestión cámaras IP (configuración, administración y visibilidad)

Control de *streaming* en vivo (códecs y control)

Análisis de vídeo y control de eventos (forma de almacenar la información)

Control estandarizado de cámaras PTZ

Almacenamiento de vídeo

Control de accesos

2.3.1 Introducción a DiVA

La Universidad Autónoma de Madrid dispone de una solución integrada para la captura, distribución y análisis de los datos enviados por las distintas cámaras con las que cuenta un sistema de vídeo vigilancia. Este sistema (DiVA, *Distributed Video Analysis Framework*), ha sido desarrollado por el grupo de investigación VPU-Lab en la Escuela Politécnica Superior, con el fin de proporcionar un entorno en el que implementar y probar aplicaciones de vídeo-seguridad.

Se trata de un sistema genérico, escalable y distribuido destinado la investigación y a la creación de prototipos y servicios de vídeo, y que posee las características que requieren los sistemas distribuidos de vídeo vigilancia de tercera generación.

[1] Open Network Video Interface Forum, http://www.onvif.org/

[2] Physical Security Interoperability Alliance, http://www.psialliance.org/

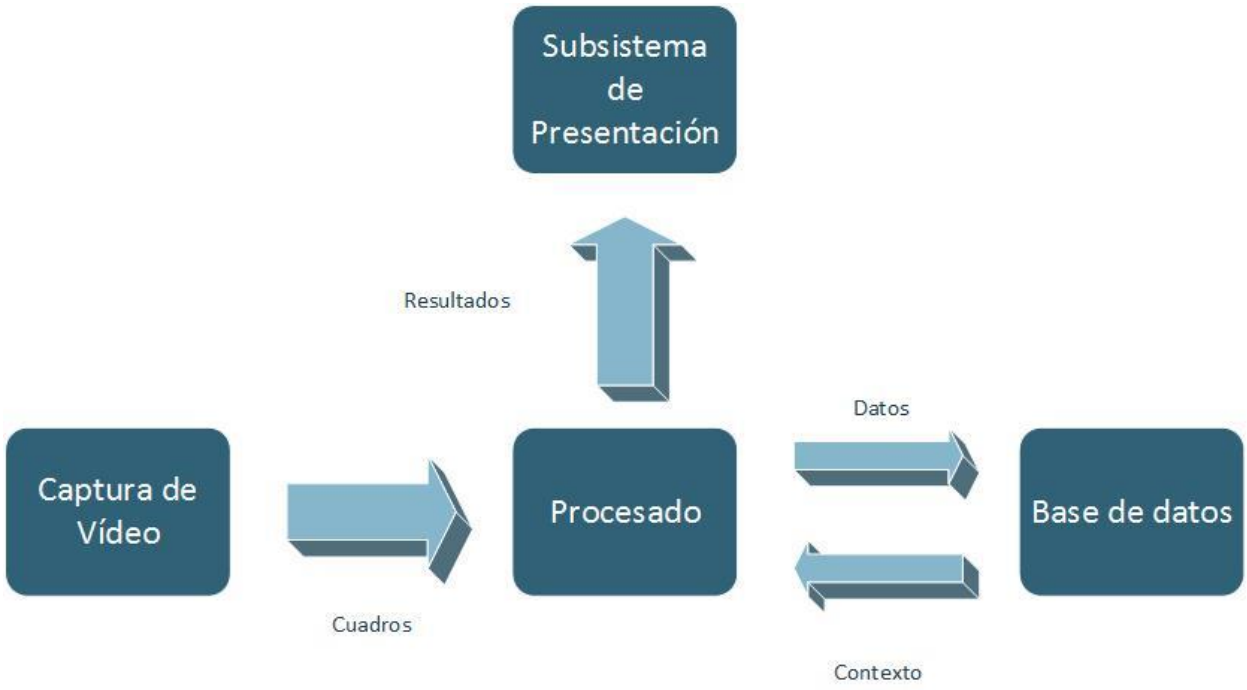


Figura 2.5: Subsistemas de DiVA [12]

Está basado en una arquitectura de cliente-servidor y contempla el trabajo con múltiples cámaras. Además, la información generada en cada módulo de trabajo es accesible para el conjunto del sistema mediante el uso de una base de datos. Los distintos módulos del sistema pueden estar conectados de diversas maneras, obteniendo así mayor flexibilidad.

Los dos principales criterios para la realización del sistema han sido el bajo coste computacional y la facilidad para agregar nuevos componentes. Los resultados experimentales de este sistema muestran un gran potencial.

Como se ha mencionado anteriormente, el principal objetivo de este trabajo es la integración de un algoritmo de detección de anomalías en la plataforma DiVA, y dotarlo de una interfaz gráfica apropiada para el mismo, con una funcionalidad que se ajuste a la de soluciones existentes en el mercado.

2.4 Ejemplos de interfaces gráficas

Para el desarrollo de este trabajo ha sido necesario documentarse en la situación del mercado actual, ya sea para conocer los algoritmos que se ofrecen, la arquitectura que se desarrolla, el software y el middleware necesario para la implementación del sistema, y también, el diseño de interfaces de usuario de los distintos productos comerciales. Para este apartado se ha realizado un análisis de varios sistemas fijándose en la interfaz de usuario de los mismos, ya que uno de los objetivos de este trabajo consiste en realizar la implementación de una interfaz gráfica que ayude al manejo del algoritmo. Con las conclusiones extraídas se han valorado los puntos clave y los elementos más necesarios para incluir en la propia interfaz gráfica y al mismo tiempo se han pretendido suplir las carencias que se han encontrado en los sistemas observados.

Dos de las compañías más asentadas en el mercado de la vídeo vigilancia son Sony y Axis, para lo cual se ha

recopilado información acerca de las soluciones ofertadas por estas compañías.

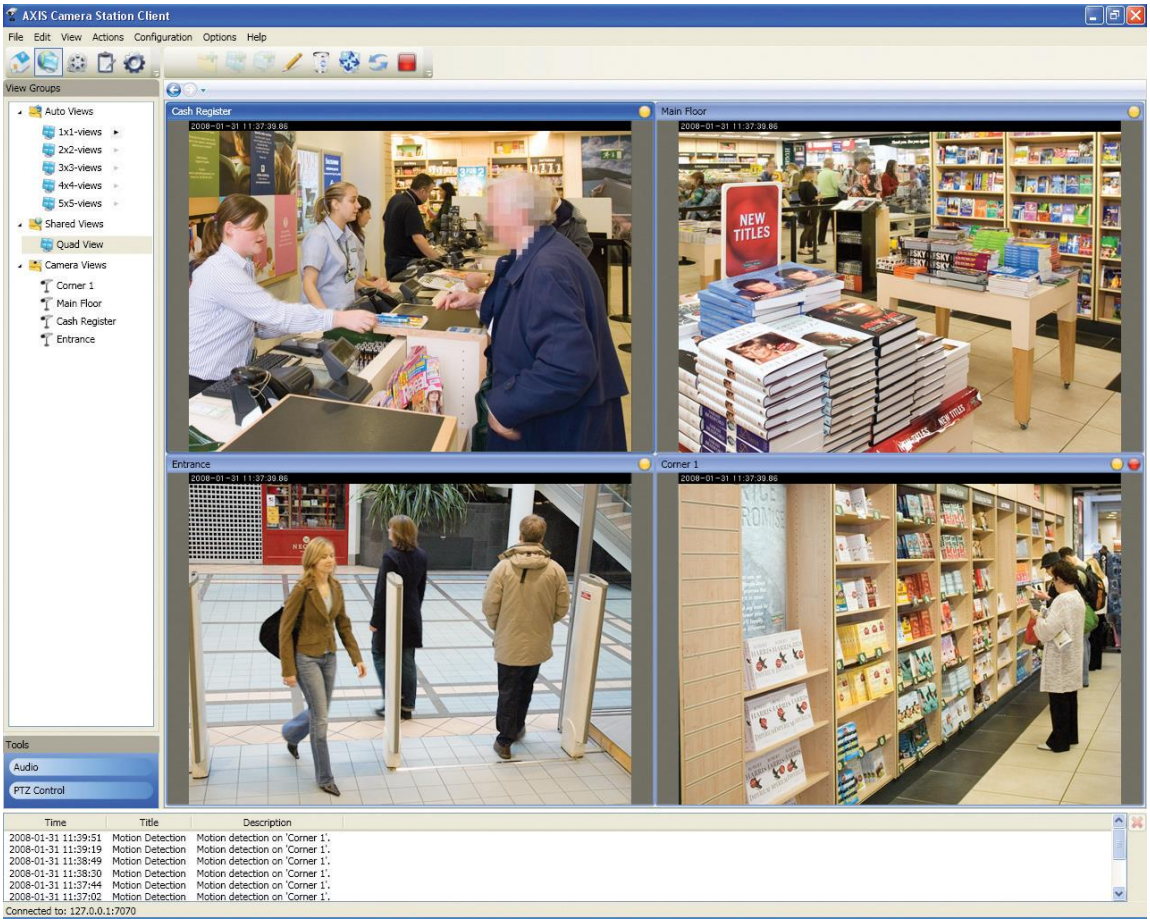


Figura 2.6: Solución software *de Axis [13]*

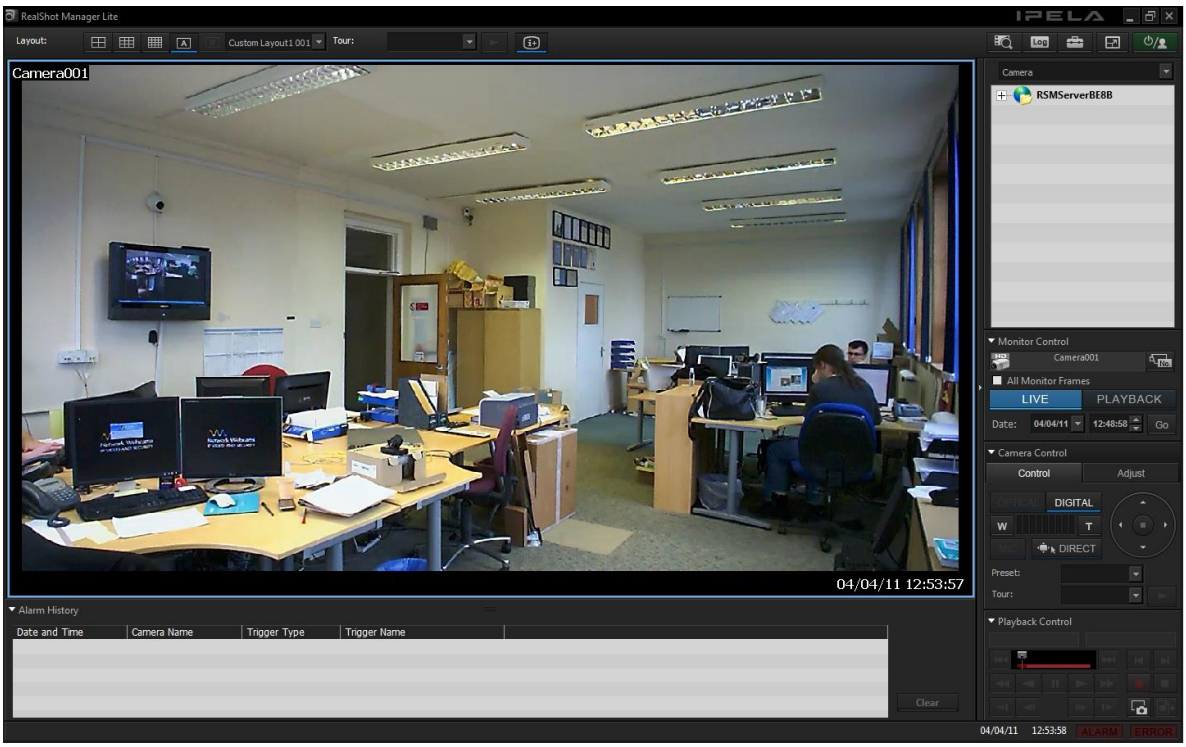


Figura 2.7: Solución software *de Sony [14]*

Como se puede ver en las imágenes, en todos los sistemas se da la opción de cambiar de cámara para observar distintas imágenes. En la imagen del sistema de Axis se puede ver cómo es posible visualizar distintas imágenes en la misma pantalla, en este caso, de distintas cámaras.

Ambos sistemas incluyen un apartado de alertas o avisos de los sucesos ocurridos donde, además, se acompaña una breve descripción de cada suceso. También se incluye en los dos sistemas herramientas para manejar el audio de las cámaras y la posición de las mismas si se trata de cámaras PTZ.

En las barras de herramientas en el de Axis se puede observar un botón que lleva al usuario a una ventana en la que se pueden realizar cambios en la configuración del programa y otras herramientas diversas.

Debido a las características del algoritmo con el que se va a trabajar, algunas de las características de estas interfaces será interesante implementarlas, mientras que otras de ellas no ayudarán al manejo del algoritmo e incluso serán incompatibles con el mismo. Más adelante, en sucesivos apartados y capítulos se expondrán estas cuestiones y las decisiones tomadas con respecto a la interfaz de usuario.

3 DETECCIÓN DE ANOMALÍAS EN SECUENCIAS DE VÍDEO

3.1 Introducción

En el campo de la visión artificial, la detección de eventos de interés pretende ser una ayuda para el personal de seguridad encargado de la visualización de escenarios controlados por cámaras. En la literatura podemos encontrar numerosos ejemplos de detección de eventos específicos, como la detección de aparcamiento en zonas restringidas [9], la detección de objetos abandonados [8] e interacciones sospechosas con cierto tipo de objetos [15]. Este tipo de técnicas, que se enmarcan dentro de la detección de eventos en secuencias de vídeo, por lo general requieren observar y modelar previamente el comportamiento que se persigue detectar. En la práctica, la amplia variabilidad de comportamientos observables en un escenario dado hace imposible que estas técnicas puedan aplicarse de manera general a escenarios genéricos. Más recientemente, podemos encontrar técnicas que siguen otra línea de trabajo, donde se observa y se modela el comportamiento "normal" en la escena, para detectar como anomalías aquellas observaciones que se desvían del comportamiento esperado, dependiendo del contexto [10], [16]. Para esto, generalmente se genera un modelo del comportamiento normal de forma estadística y se parte de dicho modelo para discriminar entre observaciones normales y anomalías.

En este apartado se ha adoptado una definición de anomalía genérica, que se utilizará en adelante para el resto del trabajo; sin embargo, hay que aclarar que esta no es la única manera de definir una anomalía, ya que dependiendo del contexto y la aplicación, la propia definición puede variar en gran medida [16].

3.2 Detección de anomalías. *Behaviour Subtraction*

Después de las características vistas de los detectores de eventos en general, y hablando de forma más específica, de los detectores de anomalías, en este apartado se pasará a hablar del algoritmo de detección automática de anomalías facilitado para la realización de este trabajo.

Este algoritmo [17], ha sido previamente implementado por los investigadores de VPU-Lab. Las operaciones de este algoritmo se realizan directamente en forma matricial, lo que lo hace idóneo para su implementación y utilización en situaciones de tiempo real. La definición de anomalía para este algoritmo es la mencionada en el apartado 2.5, donde se distingue como anomalía cualquier movimiento detectado que no siga unos patrones dados de normalidad.

Para la ejecución de este algoritmo, se asume que se trabaja con una cámara fija y una invariabilidad temporal de la actividad normal de la escena, es decir, que el movimiento considerado "normal" en la escena no varía con el tiempo.

En la Figura 3.1 se muestra un diagrama de bloques del algoritmo. Partiendo del flujo de *frames* proveniente de la cámara, este algoritmo identifica en primer lugar los objetos en movimiento, seguidamente genera una imagen que caracteriza el comportamiento de los objetos en movimiento en función de su tamaño (descriptor de objeto). Para

describir un evento, el algoritmo modela el número transiciones de cada píxel entre movimiento y estático en una ventana temporal (descriptor de evento). Por último, cada evento es clasificado como anómalo o normal mediante la comparación de los mismos con un modelo calculado previamente donde se plasma el comportamiento normal de la escena, generando de este modo un mapa binario de anomalías. Este modelo se calcula en una fase previa de entrenamiento. En la Figura 3.1 se ilustran las distintas etapas del algoritmo, partiendo del *frame* actual, y los descriptores correspondientes.

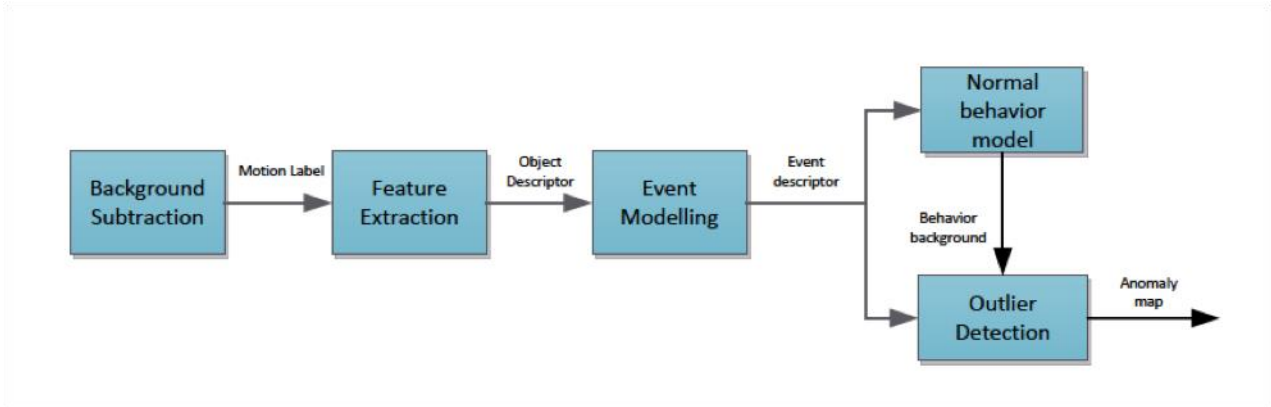


*Figura 3.1: Diagrama de bloques del algoritmo "*Behaviour Subtraction". Figura de [18]

En la primera etapa de **sustracción de fondo**, la principal finalidad es crear una imagen binaria donde se asigne para cada pixel una etiqueta con valor "estático" o "en movimiento". De esta manera se realiza una segmentación frente-fondo y se aíslan los pixeles de interés a la hora de buscar anomalías. Para la implementación de esta tarea, este bloque realiza un modelo de fondo de la escena, dicho fondo lo compara con los sucesivos frames para determinar que pixeles han cambiado con respecto al fondo, al mismo tiempo que con dichos frames actualiza el modelo para dotar al algoritmo de cierta robustez a cambios generales en la escena, como cambios en la iluminación, etc. Existen numerosas técnicas de sustracción de fondo [19]. Para este algoritmo se ha utilizado una técnica simple de promedio deslizante [19]. Adicionalmente, la implementación disponible incluye estrategias de inicio en caliente para inicializar el fondo cuando hay objetos en movimiento y de eliminación de sombras [20].

El segundo bloque, **extracción de características**, es el encargado de caracterizar los objetos utilizando las características que se deseen del objeto que se encuentra en movimiento. En un principio estas características pueden englobar, el tamaño, la dirección, la forma o la velocidad. Para el módulo incluido en el algoritmo que ahora se estudia, se utiliza como característica el tamaño de los objetos, por lo que es posible diferenciar entre objetos de diversos tamaños y discriminar aquellos que carezcan del interés para el usuario a la hora de poner en funcionamiento el algoritmo (por ejemplo, objetos muy pequeños que pueden tratarse de ruido en la máscara de movimiento). El hecho de extraer únicamente las características del tamaño de los objetos en este módulo se debe a la importancia de la ejecución en tiempo real de este tipo de algoritmos, ya que al tratarse de algoritmos para aplicaciones de vídeo vigilancia, el hecho de detectar sucesos en el momento que ocurren es de gran importancia.

El tercer bloque**, modelado de eventos**, los eventos son modelos mediante una cadena de Markov de dos estados como estáticos o en movimiento, dependiendo del valor de los píxeles de la máscara de movimiento extraída anteriormente. El descriptor de eventos describe el comportamiento de las transiciones entre los dos estados para cada píxel, en un periodo de tiempo. Esto se consigue acumulando en el tiempo (ventana temporal M) los descriptores de

tamaño.

Y por último, el bloque final corresponde a la creación de un **mapa de anomalías,** donde se indican los píxeles que pertenecen a eventos que se consideran anómalos. Una vez se tenga la imagen generada en el bloque anterior, esta se comparará con una imagen entrenada previamente a la que se ha hecho referencia al principio del apartado. Para realizar esta comprobación se realiza una resta entre ambas imágenes, y el resultado, a nivel de pixel, indicará que pixeles de la imagen se consideran anómalos, y que pixeles se consideran normales. Extrapolando esta información a la imagen de la cámara, se podrá visualizar el objeto o los objetos que están generando un aviso de anomalía en el sistema. La forma en la que se realice dicha resta determinará el tipo de anomalías que se están buscando, ya que puede que únicamente interese detectar movimiento anómalo, o también, detectar la ausencia de movimiento como un evento anómalo.

Una vez estudiado el funcionamiento del algoritmo, se han identificado los siguientes parámetros que regulan el funcionamiento del algoritmo, para dar la opción de modificarlos durante la ejecución del programa:

Módulo de sustracción de fondo:

- th: Umbral de sustracción de fondo para la máscara binaria.
- ro: Factor de actualización del modelo de fondo a partir del *frame* actual.
- cth1: Contador *HotStart*, número de *frames* para el inicializar un fondo estable cuando existe movimiento al inicializar el modelo.
- cth2: Contador de objetos estacionarios, número de *frames* para que los objetos estacionarios sean incorporados al modelo de fondo pasado un tiempo.

Módulo de descriptor de tamaño:

- alpha: Factor ponderador de ventana de análisis de blob (tamaño de blob) – tamaño de ventana espacial variable.
- N: Tamaño de la ventana de análisis para analizar el tamaño de los blobs.
- minN: Tamaño mínimo de la ventana cuadrada en caso de que al ponderar por alpha sea muy pequeño.

Módulo de descriptor de eventos:

- w: Número de *frames* de acumulación de movimiento. Un valor muy elevado implica perder como anomalía movimientos muy rápidos.

3.2.1 Escenarios posibles de aplicación

Una vez comprendido el funcionamiento del algoritmo, conviene conocer las aplicaciones reales del mismo para entender donde reside el interés de la realización de un sistema completo de vídeo vigilancia para el mismo. Al ser un algoritmo que permite definir que se considera como comportamiento normal en una escena conocida, se puede utilizar este algoritmo para un gran número de situaciones siempre y cuando se inicialice de manera correcta.

Figura 3.2: Ejemplo de aplicación para el control de tráfico. Figura de [18].

Entre algunas de sus aplicaciones se encuentra el control de tráfico. Pueden detectarse adelantamientos, abandonos en zonas de la calzada o incluso vehículos de gran tamaño.

Figura 3.3: Ejemplo de aplicación para control de tráfico de personas. Figure de [18].

Del mismo modo puede configurarse para controlar el tráfico de personas, ya sea en zonas interiores o exteriores, de manera que puede detectarse movimiento o abandonos en zonas no permitidas.

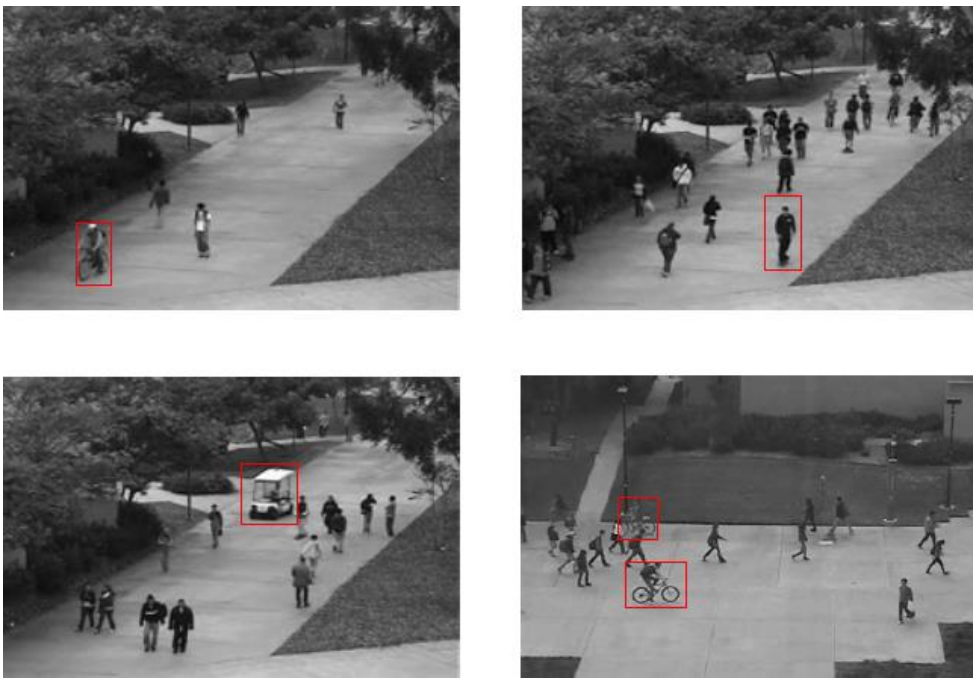


Figura 3.4: Ejemplo de aplicación para detección de objetos de tamaño anómalo [21]

Dado que uno de los módulos del detector tiene que ver con el tamaño de los objetos en movimiento a analizar, también se pueden detectar tráfico de vehículos u objetos de un tamaño mayor a una persona en zonas donde el tráfico es sólo permitido para las personas.

4 DiVA

En esta sección se pretende describir la plataforma DiVA. Dicha plataforma ha sido la diseñada para la integración de los algoritmos en un sistema común y distribuido para facilitar así la cooperación entre distintos algoritmos. Para realizar este capítulo se ha tenido en cuenta la documentación disponible en [22] y [12].

La plataforma está dotada de las siguientes funcionalidades:

Sistema escalable con una carga computacional distribuida.

Ejecución de operaciones a tiempo real.

Bajo consumo de recursos.

Control de comunicaciones.

Tiempo de ejecución reconfigurable.

4.1 Arquitectura del sistema

La plataforma está pensada para dos niveles de abstracción, uno físico y el otro lógico.

En el nivel físico se encuentra el *hardware* necesario para el funcionamiento del sistema, las cámaras y los PCs necesarios. Todos los elementos en este nivel estarán conectados mediante una red *Ethernet*.

Para hacer frente a las restricciones de ancho de banda y para mejorar el rendimiento del sistema, el nivel físico se divide a su vez en dos redes. La principal cuenta con un set de PCs interconectados entre sí por una red de *Gigabit Ethernet*. Estos PCs son los encargados de obtener las imágenes de las cámaras, ejecutar los algoritmos oportunos y almacenar los datos necesarios. El otro sistema se encuentra interconectado con una red distribuida en *100Baset Ethernet* a través de la red central. Las cámaras necesarias en cada aplicación se encontrarían conectadas, bien a uno de los PCs, bien a la red *Ethernet* si se trata de cámaras IP. Esta arquitectura cuenta con la principal ventaja de la flexibilidad del sistema.

La parte lógica está compuesta por cuatro capas independientes. Cada capa está diseñada de una manera modular y para un objetivo específico. Las capas son las siguientes:

Subsistema de captura: Obtiene el vídeo de múltiples fuentes y lo distribuye a todo el sistema *frame* por *frame*. Las *frames* se envían en formato JPEG o en un formato sin compresión con una marca de tiempo. Esta capa sigue un esquema cliente-servidor. Aquí se incluye un módulo que hace de interfaz entre las imágenes provenientes de una cámara local (USB, *Firewire*, o cámara IP), y la forma en la que se distribuyen la señal de vídeo en DiVA, junto con los protocolos de red necesarios.

Subsistema de procesado: Permite recibir las imágenes que proceden del subsistema de captura, y procesarlas

(análisis de vídeo), almacenarlas o enviarlas al subsistema de presentación. Durante el procesado, se puede utilizar el subsistema de base de datos tanto para obtener información necesaria, como para guardar la información sobre eventos detectados. Este subsistema está pensado de forma tal que pueda encadenar uno o varios módulos de análisis independientes (clase `DiVAAlgorithm`), mediante una arquitectura cliente-servidor.

Subsistema de presentación: Es el encargado de mostrar los resultados producidos por los módulos de análisis que componen el subsistema de procesado. Dependiendo de las necesidades de la aplicación, comprende varios niveles de presentación (volcado a fichero de vídeo, mostrar por pantalla, mostrar con interfaz gráfica, etc.). Se contempla la posibilidad de que un único cliente realice las tareas de procesado y presentación, además de que pueda ser realizado por distintos clientes conectados en modo cliente-servidor.

Subsistema de bases de datos: Este subsistema tiene dos finalidades. Por un lado, poder almacenar el resultado de los distintos módulos de análisis del subsistema de procesado (alertas, registro de eventos, seguimiento de objetos). Por otro lado, el de poder almacenar los distintos parámetros necesarios para la ejecución de distintos algoritmos, con el fin de que estos datos puedan ser distribuidos a distintos clientes.

4.2 Integración del algoritmo

En este trabajo de Fin de Grado se ha trabajado en la integración de un algoritmo de detección de anomalías en vídeo sobre la plataforma DiVA, con el fin de dotar a la aplicación de una interfaz gráfica similar a las existentes en soluciones comerciales. Para interactuar con los sistemas de captura, DiVA dispone en C++ de una clase genérica para algoritmos de análisis (`DiVAAlgorithm`). Esta clase cuenta con una plantilla para incorporar algoritmos de análisis ya existentes, que recibe los *frames* de vídeo provenientes de las cámaras del subsistema de procesado. En su implementación interna, el algoritmo se ejecuta en un hilo independiente al resto de la aplicación.

Para la integración de nuevos algoritmos en la plataforma DiVA de cara al sistema de presentación, se contemplan tres niveles de funcionamiento, cada uno de los cuales realiza una función específica y del mismo modo necesaria para el completo funcionamiento del programa final.

4.2.1 Nivel 1

El primer nivel consta de un preparamiento previo a la integración del algoritmo en la plataforma. En este nivel el algoritmo debe ser capaz de leer un vídeo, procesarlo y generar una salida, al mismo tiempo será necesario dotar al algoritmo de los métodos necesarios para la futura inserción. En este punto es imprescindible que el algoritmo se encuentre escrito en C++ y que posea una clase propia para dicho algoritmo, aunque dicha clase haga uso de otras clases distintas. Según la documentación, previo a la integración en DiVA, el algoritmo debe contar obligatoriamente con métodos que realicen las siguientes funciones:

Constructor. Construye la clase principal, crea los objetos necesarios e inicializa las variables del algoritmo.

Destructor. Destruye los objetos y libera la memoria del programa.

`void *process(IplImage *frame).` Dentro de esta función se desarrollará la funcionalidad del algoritmo.

`void showResults().` La función llamará a los métodos oportunos para la correcta visualización de los

resultados.

`void saveResults()`. La función guardará los resultados en el disco.

Métodos necesarios para leer y escribir los parámetros del algoritmo que se quieran modificar durante la ejecución del programa.

4.2.2 Nivel 2

En este nivel se encapsulará el algoritmo dentro de la clase `DiVAAlgorithm` para así poder beneficiarse de todas las funcionalidades propias de la plataforma DiVA. Al final de este nivel de la integración en la plataforma DiVA el algoritmo debe ser capaz de leer imágenes extraídas de un *frame server*, procesarlas y generar una salida.

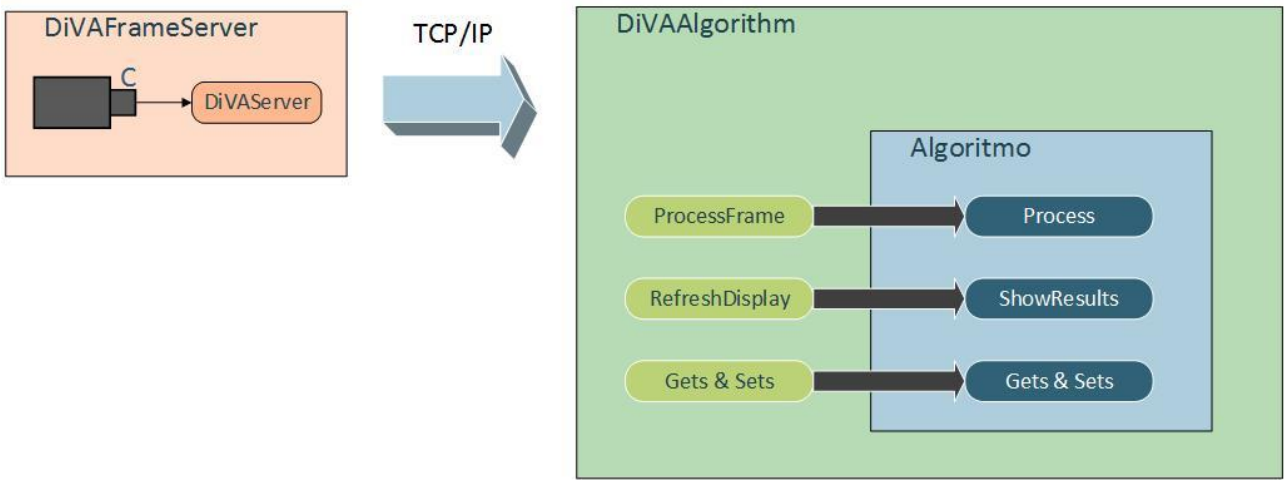


Figura 4.1: Integración DiVA nivel 2. Figura de [12].

Para llevar a cabo esta tarea será necesario crear una nueva clase para el algoritmo, diferenciada de la clase principal del algoritmo, a la que se hace referencia en el punto 4.2.1, que herede de DiVAAlgorithm. Dentro de esta nueva clase se podrán implementar los métodos que se crean oportunos para realizar que el algoritmo funcione plenamente.

De forma obligatoria para que el algoritmo funcione con las características de DiVA se tendrá que realizar un *override* de los siguientes métodos de `DiVAAlgorithm`:

`int *processFrame(DiVAImage *frame)`. Si la función `process(IplImage *frame)` del nivel anterior realiza las operaciones oportunas para que el algoritmo funcione, esta función únicamente debe llamar a la del nivel anterior.

`int refreshDisplay()`. Al igual que con `processFrame`, esta función deberá llamar a la del nivel anterior.

4.2.3 Nivel 3

En este nivel hay que dotar al conjunto generado previamente de una interfaz de gráfica que guíe al usuario en el funcionamiento y la ejecución del algoritmo para poder trabajar y observar los resultados de forma más cómoda e intuitiva.

En la documentación se propone tres submodos de funcionamiento para este nivel, cada uno de los cuales está

diseñado con unos fines específicos.

Modo desarrollador: Este modo está pensado para que se maneje por personal con nociones en el desarrollo y manejo de algoritmos y conocimiento previo del funcionamiento del algoritmo que se esté ejecutando en el programa. En este modo es necesario que se puedan mostrar, a voluntad del usuario, las imágenes de entrada y salida del algoritmo, así como las imágenes intermedias que el mismo utilice para generar la salida. También se propone que en la ventana principal esté visible un cuadro de manejo de los parámetros que posea el algoritmo, de esta manera la modificación de los parámetros durante la ejecución y la visualización de los efectos que esto provoca sobre los resultados podrá observarse de manera clara y automática.

Modo cliente: Este modo, a diferencia del anterior explicado, está pensado para usuarios sin noción alguna en el desarrollo y manejo de algoritmos. Para esta versión se recomienda dejar despejado en la medida de lo posible la ventana principal para no interferir en la observación de los resultados. Tampoco será necesario dar opción a visualizar las distintas imágenes intermedias que utilice el algoritmo, pero sí se recomienda mantener, en una ventana a parte, un cuadro para la modificación de los distintos parámetros del algoritmo.

Aplicación de clientes ligera: El objetivo de esta aplicación es crear un programa con un algoritmo cuya única función consista en la visualización del resultado final de la aplicación del algoritmo. Se pretende con esto que el programa con la ejecución real del algoritmo se lleve a cabo en un PC independiente, el cuál enviará los resultados al cliente, dejando que la máquina del mismo consuma muchos menos recursos.

5 DESARROLLO

En este capítulo se pretende dar una visión clara del trabajo llevado a cabo durante los meses duración del mismo.

Como se menciona en capítulos anteriores, el objetivo principal de este trabajo es la integración del algoritmo descrito en el capítulo 3 en la plataforma DiVA . Para ello, se ha seguido el proceso descrito en la sección 4.2, adaptando el código del algoritmo inicial a las especificaciones requeridas por DiVA.

5.1 Punto de partida

Al inicio del trabajo, se contaba con una versión del algoritmo implementada por el grupo de investigación VPU-Lab en el lenguaje C++. Inicialmente, el código consiste en un programa principal con un bucle que carga fotogramas desde un fichero de vídeo y realiza las operaciones necesarias para obtener el mapa de anomalías. Adicionalmente, se incluyen clases auxiliares para generar las imágenes intermedias necesarias previas al resultado final. El código de partida se encuentra estructurado en clases de la siguiente manera:

- `Main.cpp`: Inicialización de variables, lectura de fichero de vídeo, y bucle principal que contiene el grueso de operaciones del algoritmo. Opcionalmente, se puede ir guardando el mapa de anomalías o cualquier imagen intermedia en un fichero de salida.
- `BGS.cpp`: Clase para el modelo de fondo (*Background Subtraction*): este modelo se actualiza con nuevo *frame*, para adaptarse incrementalmente a cambios en la escena.
- `SizeDescriptor.cpp`: Funciones auxiliares para generar la matriz con el descriptor de tamaño. Utiliza de manera auxiliar las funciones contenidas en `BlobExtractor.cpp` para extraer de la máscara de movimiento las regiones de movimiento (análisis de componentes conexas [23]).
- `EventDescriptor.cpp`: Genera el descriptor de eventos a partir del modelo descrito en el capítulo 3. Ya que el descriptor de eventos acumula los últimos descriptores de tamaño, se mantienen éstos en una estructura de cola FIFO (*first-in, first-out*).

5.2 Entorno de trabajo y librerías disponibles

Para la implementación del algoritmo, se han utilizado principalmente métodos de la librería **OpenCV**[3] para el tratamiento de las imágenes. OpenCV[3] es un sistema de código abierto con licencia BSD pensado para uso académico y comercial. Cuenta con interfaces para C, C++, Phyton y Java con librerías para los sistemas operativos principales.

[3] www.opencv.org

Inicialmente se pensó para mantener una gran eficiencia computacional y está fuertemente focalizado a las aplicaciones en tiempo real. La librería de OpenCV[3] cuenta con más de 2500 algoritmos optimizados donde además se incluye un extenso estado del arte de cada uno para mayor información para el usuario.

Para la plataforma **DiVA**, se han utilizado las librerías disponibles en VPU-Lab. En particular, se ha trabajado principlamente con las clases `DiVAAlgorithm` para la encapsulación de un algoritmo de análisis de vídeo, y con la interfaz de conexión a servidores de *frames* compatibles con DiVA (`DiVAFrameServer`).

Por último, para la realización de la interfaz gráfica se han utilizado las herramientas facilitadas por la librería **Qt**[4]. Esta librería permite el desarrollo de aplicaciónes multiplataforma con interfaces gráficas, utilizando los lenguajes C++, QML, CSS y JavaScript. Para este trabajo, se ha utilizado el lenguaje C++ con las librerías de Qt[4].

Como herramienta principal para el desarrollo del trabajo, desde los primeros pasos, hasta la creación de la interfaz gráfica, se ha realizado en **Microsoft Visual Studio**. Adicionalmente, para la edición de los elementos de la interfaz gráfica, se ha empleado la utilidad **QT**[4] **Designer**.

Al inicio de este trabajo no se tenía conocimiento alguno del manejo de lenguajes orientados a objetos, en particular de C++. Por este motivo, se han dedicado las primeras semanas de trabajo al estudio de los algoritmos de interés en OpenCV[3] en C++, de esta manera se iniciaba la familiarización con C++. Para este proceso se desarrollaron pequeños algoritmos de tratamiento de imagen en C++ utilizando la librería de OpenCV[3].

5.3 Desarrollo nivel 1. Preparación del algoritmo

El paso siguiente consiste en la preparación del algoritmo para su posterior integración en DiVA, según se indica en el apartado 4.2.1. Como se ha remarcado anteriormente, el algoritmo en su implementación inicial se encontraba ordenado en diferentes clases en C++ y ya ejecutaba recibiendo como entrada un vídeo y una imagen, procesándolos y generando una salida.

Como paso inicial, ha sido necesario aislar la funcionalidad del algoritmo siguiendo el esquema planteado por la documentación de DiVA, recomienda para su preparación previa a la inserción. Se entiende como los mismos, los métodos `processFrame()` y `showResults()`.

Seguidamente, se han implementado los métodos para lectura y escritura para cada uno de los parámetros que es posible o conveniente cambiarlos durante la ejecución para modificar los resultados de la misma. En la sección 3.2, se identifican los parámetros alterables que dominan el funcionamiento del algoritmo. Para esta tarea, se ha estudiado el algoritmo para decidir cuáles de los parámetros son susceptibles de ser alterables por el usuario durante la ejecución, y cuáles deben quedar en valores fijos o implican reiniciar el algoritmo. Para esto se han realizado pruebas prácticas del algoritmo sobre secuencias cortas (disponibles en VPU-Lab) para observar los cambios que se producen por la modificación de los distintos parámetros durante la ejecución. Como resultado, se han implementado los siguientes

[4] www.qt-project.org

métodos:

- Parámetro th, métodos getTh y setTh. El método setTh controla que los valores del parámetro th se encuentren entre [0,255], ya que este parámetro indica un umbral de valores de gris en dicho rango.
- Parámetro ro, métodos getRo y setRo. El método setRo controla que los valores del parámetro ro se encuentren entre [0,1], ya que este parámetro indica un valor de actualización entre dos imágenes expresado en porcentaje.
- Parámetro cth1, métodos getCth1 y setCth1. Éste parámetro puede tener cualquier valor entero positivo.
- Parámetro cth2, métodos getCth2 y setCth2. Éste parámetro puede tener cualquier valor entero positivo.
- Parámetro alpha, métodos getAlpha y setAlpha. El método setAlpha controla que los valores del método alpha se encuentren entre [0,1].
- Parámetro N, métodos getN y setN. En un principio, el parámetro N puede obtener cualquier valor entero positivo.
- Parámetro minN, métodos getMinN y setMinN. En un principio, el parámetro minN puede obtener cualquier valor entero positivo.
- Parámetro ventana temporal W. En este caso, este parámetro controla directamente el tamaño de la cola FIFO a la hora de acumular los descriptores de tamaño. El aumentar el tamaño de esta cola implica que durante un período de tiempo, el descriptor generado por esta clase no se puede calcular. El disminuir el tamaño implica simplemente descartar elementos.

En este nivel de desarrollo, se ha probado la ejecución del algoritmo con entrada y salida en ficheros de vídeo. Para la salida, se visualizan las imágenes intermedias del algoritmo en una única imagen. Para generar esta vista multi-imagen, se ha utilizado la función `GridDisplay` facilitada por VPU-Lab.

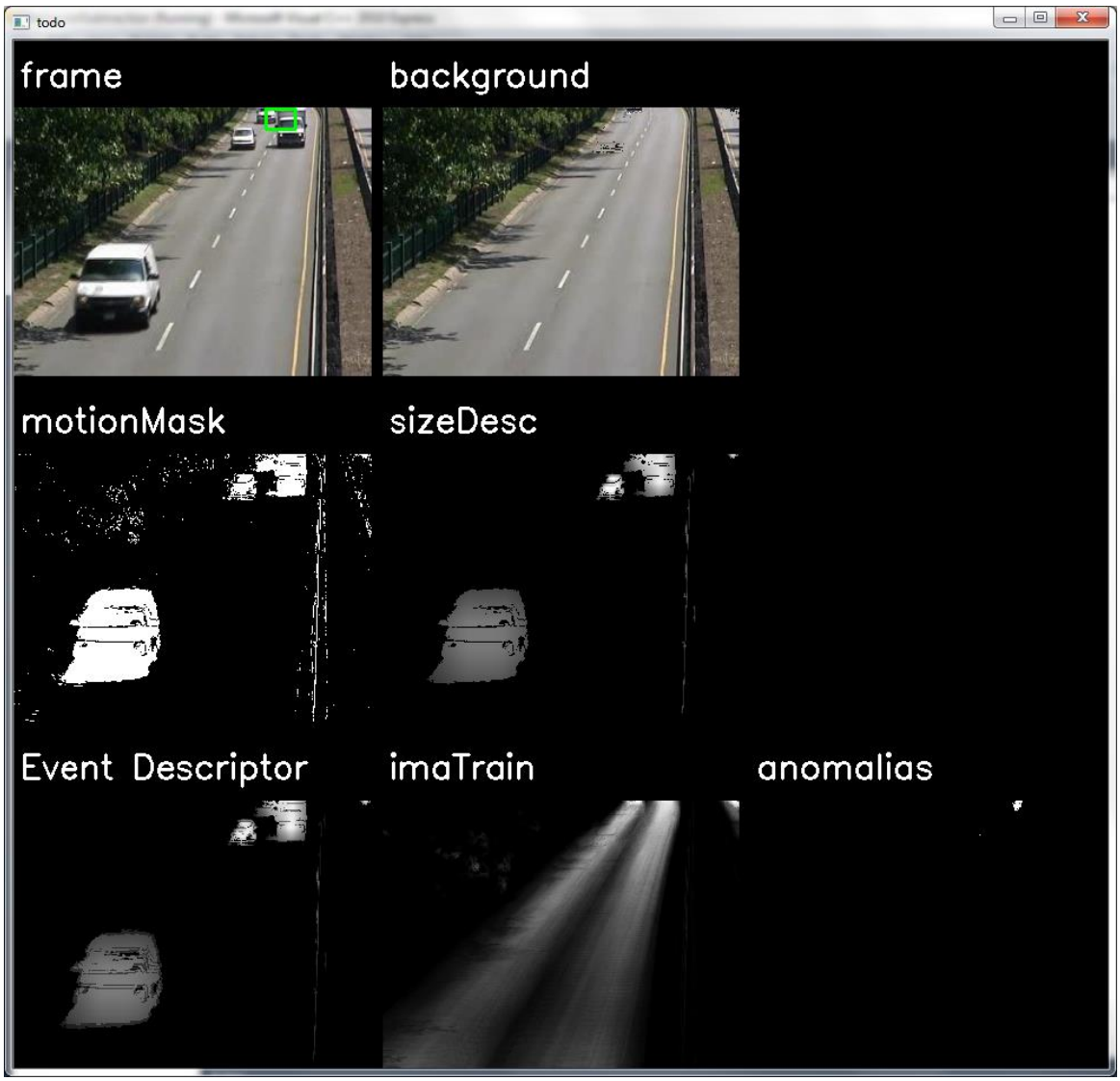


Figura 5.1: Resultado de la integración en el nivel 1

5.4 Desarrollo nivel 2. Encapsulación del algoritmo en DiVAAlgorithm

Como se ha descrito en la sección 4.2.2, el objetivo de la integración a este nivel es incorporar la funcionalidad del algoritmo en una subclase de `DiVAAlgorithm`, para poder recibir y procesar imágenes provenientes de las cámaras conectadas al sistema.

Siguiendo el manual, se creó una nueva clase, `DiVABehaviourSubtraction`, la cual hereda métodos de `DiVAAlgorithm`. Al tratarse de un objeto de una clase que hereda de `DiVAAlgorithm`, el constructor de esta nueva clase tiene que tener una estructura determinada, con los parámetros necesarios para conectarse a la cámara.

Dentro del constructor, se inicializan todas las variables necesarias para el funcionamiento del algoritmo. En el código original, estas operaciones se realizaban en el cuerpo de la función `main()`. Para esto, se han añadido como

miembros de la clase las variables de los tipos necesarios. En la Figura 5.2, se muestra un diagrama UML de la clase `DiVABehaviourSubtraction`, los miembros que se han añadido y los métodos que se han implementado. Adicionalmente, se han añadido los métodos get y set para alterar los parámetros que se describen en el apartado 3.2.

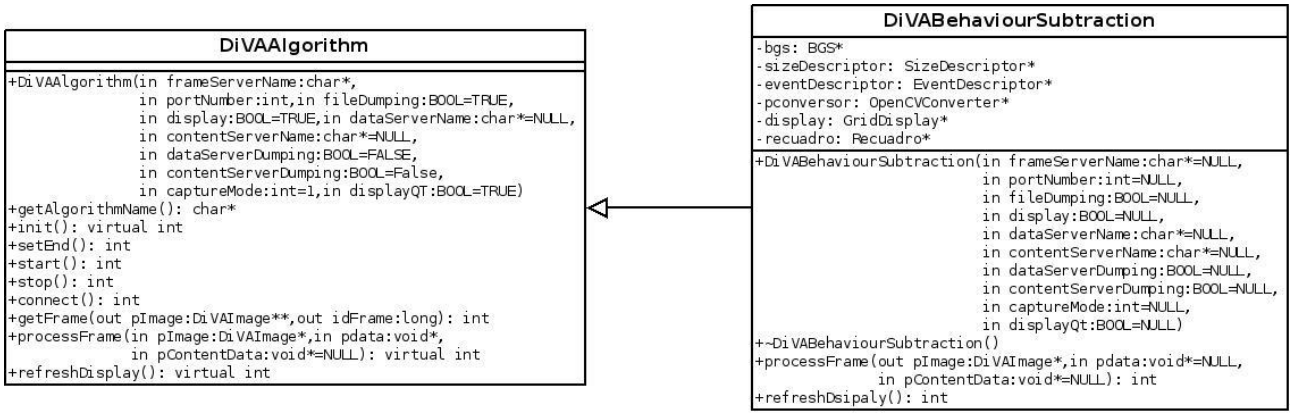


Figura 5.2: Diagrama UML de herencia de la clase de DiVA con DiVAAlgorithm

Para ejecutar el algoritmo, es necesario crear un nuevo programa donde se cree un objeto de la nueva clase, con la información de la cámara a la que nos vamos a conectar. Es decir, cada objeto de la clase `DiVABehaviourSubtraction` va asociado a una información de cámara distinta. Una vez llamemos al método `start()`, internamente DiVA ejecutará la función `ProcessFrame()` por cada *frame* entrante, actualizando donde sea necesario las imágenes intermedias, los modelos de fondo y de eventos, y el mapa de anomalías. En su funcionamiento interno, DiVA se asegura de ejecutar el contenido de `ProcessFrame()` en un hilo distinto al del proceso principal. Esto nos permite no bloquear la interfaz de la aplicación, ya sea línea de comandos o gráfica.

Una vez implementado el algoritmo en nivel 2, se ha procedido a probar el correcto funcionamiento del mismo con una serie de secuencias cortas disponibles en VPU. En particular, se ha desarrollado una pequeña aplicación en línea de comandos que ejecuta el algoritmo conectándose con una cámara, y ofreciendo por línea de comandos la opción de alterar parámetros. Se ha comprobado satisfactoriamente el efecto de alterar los parámetros durante la ejecución del programa.

5.5 Desarrollo nivel 3. Interfaz gráfica

5.5.1 Tareas iniciales

El siguiente paso es la implementación de una interfaz gráfica que permita al usuario trabajar con el algoritmo, para lo que se han contemplado varios modos de utilización. El modo más completo (**desarrollador**) completa ofrecer al usuario la posibilidad de alterar los parámetros para ayudar a determinar los parámetros óptimos para una escena dada y comprender mejor el funcionamiento del algoritmo. El modo más simple (**cliente**), se ha pensado para clientes finales con la finalidad de dirigir la atención del usuario a dónde han ocurrido las anomalías, sin entrar en la funcionalidad interna del algoritmo.

En el apartado se dan ejemplos de interfaces gráficas presentes en el mercado y se hace hincapié en los elementos comunes y dispares de las mismas. Para un primer diseño de la interfaz se utilizó la información obtenida en este

estudio previo, y además se atendió a las necesidades del algoritmo. Se ha determinado que el programa ha de contar con los siguientes elementos:

- Control de cámaras (cargar cámara desde servidor, iniciar/detener análisis)
- Zona de control de parámetros de algoritmo (puede hacerse en ventana independiente, o en un panel lateral)
- Diálogo para cargar y gestionar servidores de cámaras
- Zona de visionado de vídeo. Para este caso, es interesante poder ver, además de los *frames* provenientes de la cámara, las imágenes intermedias del algoritmo.
- Zona de alertas: registro de eventos (anomalías detectadas, u otros)
- Barra de estado. Para información varia: servidor al que estamos conectados, tasa de análisis (en frames por segundo)

Como paso previo a la implementación, se ha diseñado una primera aproximación a la interfaz gráfica en forma de bocetos, utilizando la herramienta balsamiq[5]. Dichos bocetos se muestran en la Figura 5.3, la Figura 5.4 y la Figura 5.5.

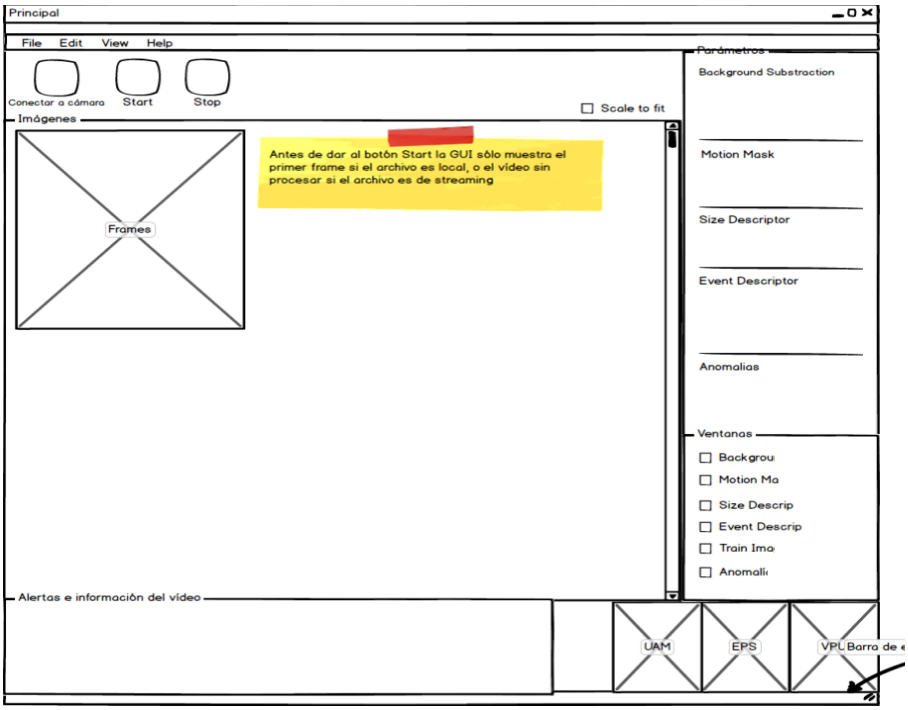


Figura 5.3: Primera aproximación de la ventana principal

[5] http://balsamiq.com/

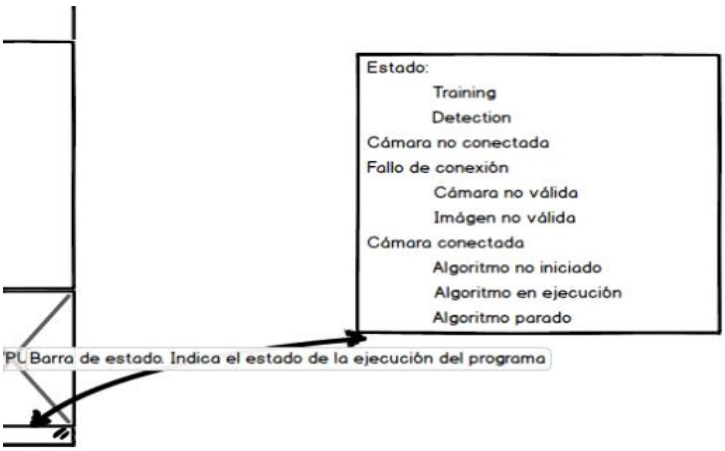


Figura 5.4: Primera aproximación de la barra de estado

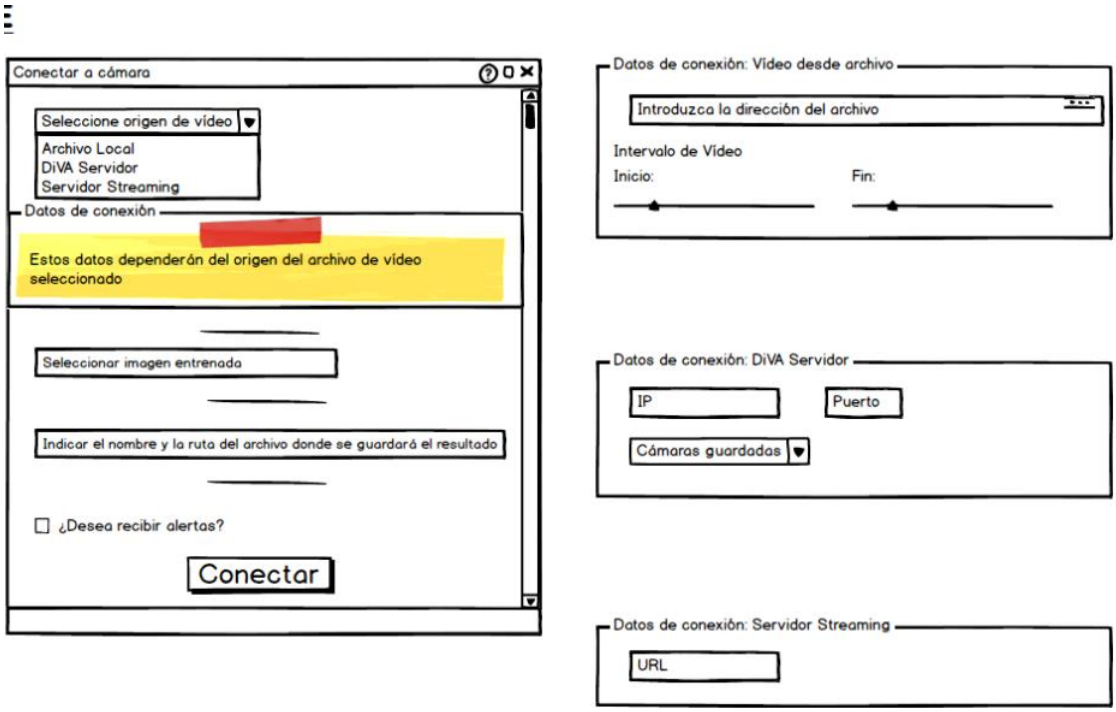


Figura 5.5: Primera aproximación de las ventanas secundarias

5.5.2 Desarrollo e implementación

A medida que se ha ido avanzando en la implementación, se han tomado distintas decisiones respecto al boceto inicial. Por un lado, se ha decidido prescindir de un recuadro para visualizar eventos. Esto es debido a que las anomalías se detectan a nivel de píxel, y en un momento dado pueden existir números grupos aislados de píxeles indicando anomalías, debido al posible ruido en la imagen y umbrales utilizados. Generar una alerta por cada grupo supondría un elevadísimo número de alertas con respecto al número real de anomalías detectadas.

Por otra parte, para simplificar el desarrollo, en lugar de implementar en dos programas aislados las interfaces de cliente y desarrollador, se ha decidido implementar una única interfaz que por defecto funcione en modo cliente,

mostrando únicamente la vista de la cámara, con las posibles anomalías recuadradas. Sobre esta misma interfaz, se ha adaptado el programa para mostrar una ventana separada con las opciones para alterar los parámetros del algoritmo, así como la posibilidad de alterar la vista actual para mostrar, además de la vista de la cámara, las imágenes intermedias.

Algunos de los elementos necesarios para la interfaz gráfica pueden considerarse comunes a distintos algoritmos que puedan incorporarse a la plataforma DiVA. Es por esto que durante la realización de este trabajo, paralelamente se han coordinado esfuerzos para armonizar el diseño de estos elementos. En particular:

- Diálogo para conectar a la cámara
- Ventana estándar de Ayuda/Acerca de
- *Widget* para el visionado de varias cámaras o varias imágenes simultáneamente en la misma área.

Debido a las características del algoritmo, éste puede funcionar de dos maneras distintas. En la primera de ellas, el algoritmo detecta las anomalías presentes en una escena a partir de un *frame server* y una imagen previamente entrenada. Un segundo modo auxiliar es necesario para poder generar esta imagen entrenada para una escena en concreto.

El algoritmo en su estado inicial no contaba con la implementación necesaria para poder generar la imagen entrenada (también llamada fondo de comportamiento, como se ha visto en el apartado 3.2). Así pues, se ha desarrollado también la opción de crear una imagen de entrenamiento a partir de los datos de un *frame server*.

En el modo de ***detección***, serán necesarios los siguientes datos de entrada: los datos del servidor de imágenes y la imagen entrenada con el movimiento de la escena. Se ha decidido que para el modo de ***entrenamiento***, únicamente sea necesario introducir al algoritmo los datos del servidor de imágenes. En cualquier caso, como la información de la cámara es necesaria para que el algoritmo empiece a ejecutarse, se ha decidido que cuando el usuario inicie manualmente el algoritmo, salte un cuadro de diálogo pidiendo al mismo que seleccione el modo de funcionamiento en el que desea trabajar y, dependiendo de esta decisión, se le pedirá la información correspondiente. Para ello se ha desarrollado la siguiente ventana que se muestra al iniciar el algoritmo (Figura 5.6).

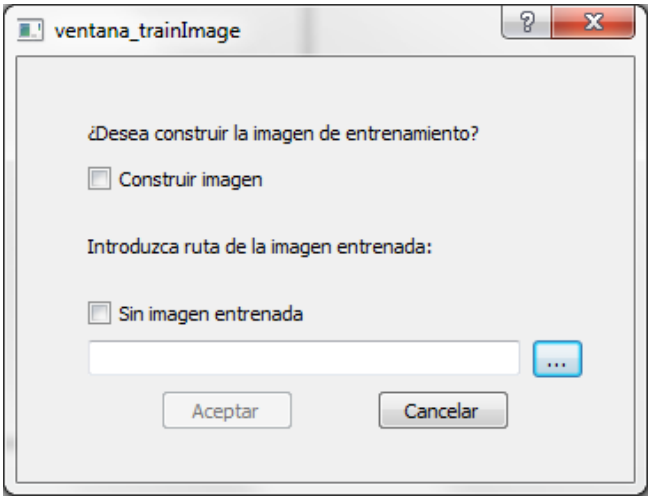


Figura 5.6: Ventana de introducción de la imagen entrenada

Con la segunda opción, seleccionando la pestaña “Construir imagen”, se altera la apariencia de la ventana para pedir al usuario una ubicación para almacenar la imagen de entrenamiento.

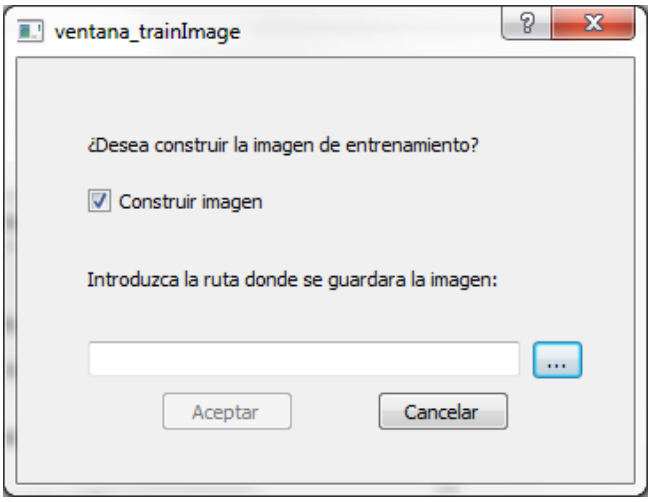


Figura 5.7: Ventana de introducción de la imagen entrenada. Opción para construir imagen

5.5.3 Diseño final y manejo de la interfaz

A continuación se describen el diseño final de la interfaz de la aplicación principal, así como su flujo de funcionamiento, justificando las decisiones que se hayan tomado con respecto al diseño inicial. En la Figura 5.8, se puede observar la ventana principal del algoritmo antes de haber iniciado la ejecución del mismo.

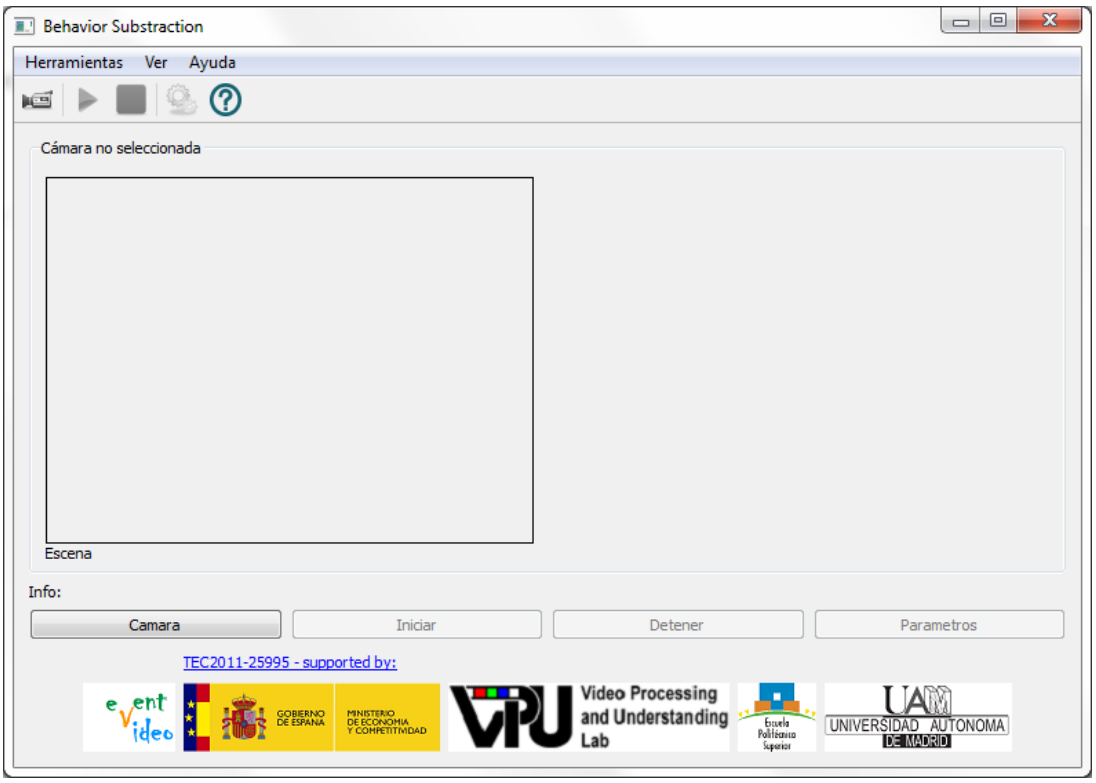


Figura 5.8: Ventana principal

Al ser necesario especificar los datos de conexión de un *frame server* antes de iniciar el algoritmo, se ha decidido bloquear los botones del resto de las opciones de la interfaz para guiar de manera más intuitiva al usuario en el manejo de la misma.

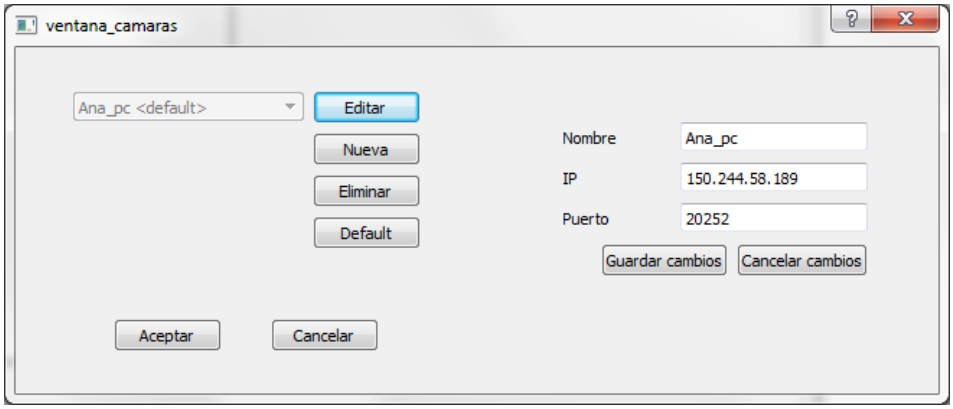


Figura 5.9: Ventana configuración de cámaras

Después de darle a la opción de conectar con la cámara se lanza la ventana de conexión de cámaras de similares características a la ventana proporcionada en la plantilla. Sin embargo, esta ventana ha sufrido modificaciones con respecto a la de la plantilla, puesto que se decidió incluir la opción de "cancelar cambios" en los datos de una cámara cuando se pulsan los botones de "Editar" o "Nueva". De esta manera, si el usuario presiona uno de estos botones por error podrá volver a la ventana oritinal sin ocasionar ningún cambio en la configuración de los datos de las cámaras.

Una vez seleccionada la cámara con la que se trabajará la interfaz permite una opción más, iniciar el algoritmo, y

contúa permitiendo cambiar los datos de la cámara.

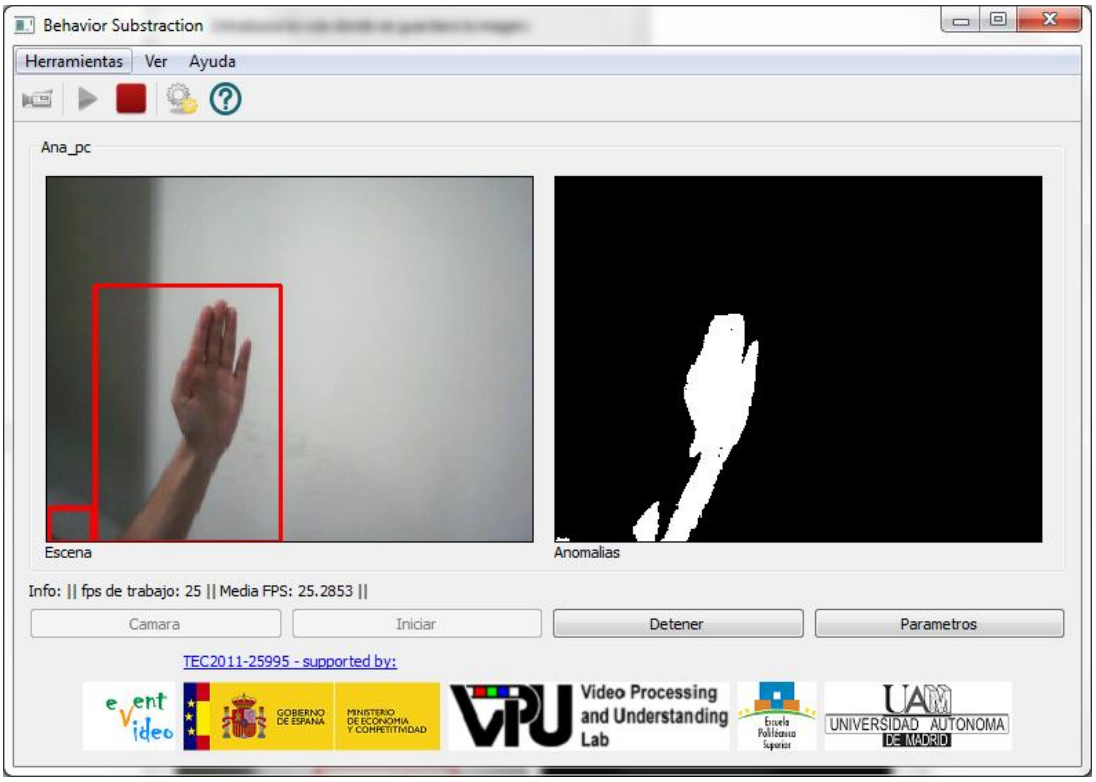


Figura 5.10: Programa en ejecución

Dependiendo del modo de trabajo elegido (detección o entrenamiento), el programa arrancará mostrando dos imágenes por defecto. En ambos casos, la primera imagen que se muestra es el último *frame* recibido por la cámara. Para el modo entrenamiento, la segunda imagen muestra la imagen de entrenamiento que se está generando. Para el modo detección, se muestra el mapa de anomalías.

En la ventana principal se puede observar una línea con una breve información del estado de ejecución del algoritmo referente al número de *frames* por segundo al que se encuentra trabajando en cada instante y la media desde el inicio de la ejecución de cada iteración del algoritmo.

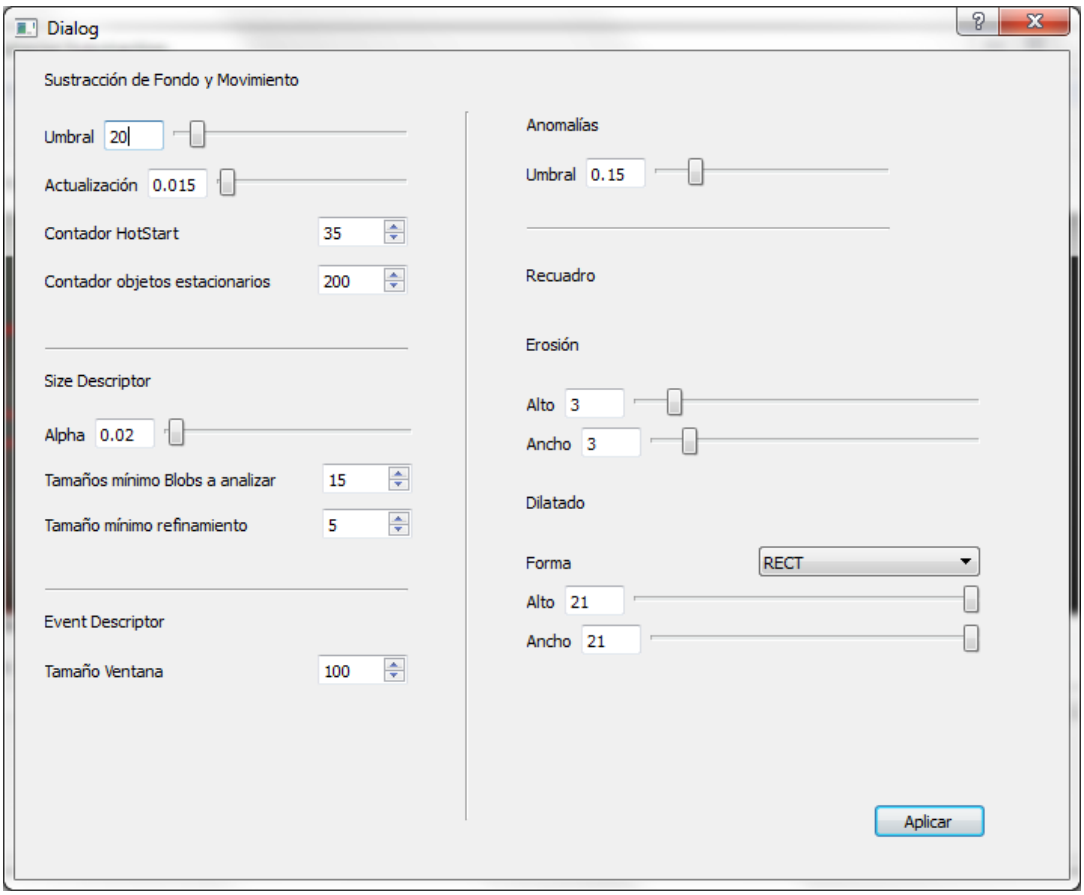


Figura 5.11: Ventana de configuración de parámetros

Para la interfaz gráfica también ha desarrollado una ventana de configuración de parámetros, en la cual, como se puede observar en la imagen, se pueden modificar todos los parámetros indicados en el apartado 4.2 y además todos los parámetros referentes al recuadro que se genera sobre los objetos que se consideran anómalos en la imagen del *frame* actual de la cámara.

La interfaz permite la visualización de todas las imágenes intermedias generadas por el algoritmo mediante su selección en la pestaña "Ver" de la barra de herramientas de la ventana principal:

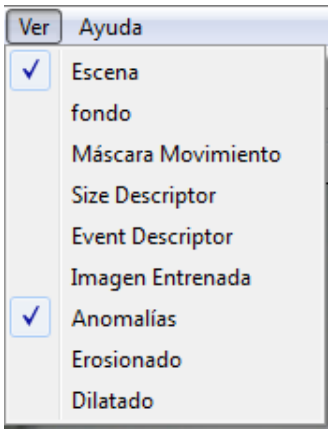


Figura 5.12: Menú de selección de imagen

También cuenta con una ventana de "Ayuda" donde se puede encontrar una breve explicación de cada uno de los parámetros modificables en la ventana de configuración:

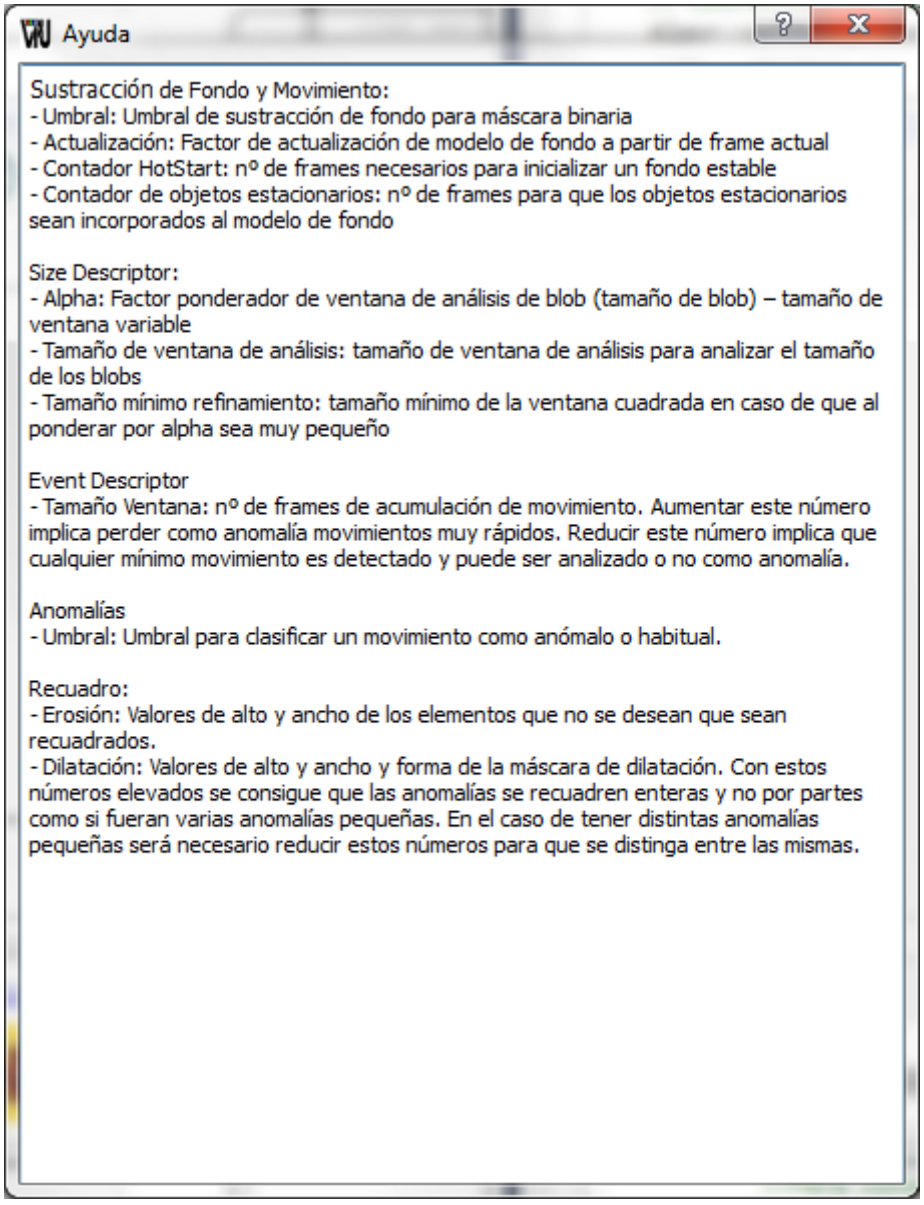


Figura 5.13: Ventana de ayuda

Dado que uno de los objetivos de este trabajo es facilitar a los usuarios la visualización de movimiento anómalo en una escena donde se mantiene un proceso de vídeo vigilancia, también se creyó conveniente añadir un cambio más al funcionamiento original del algoritmo. Una vez obtenida la imagen binaria de las anomalías encontradas, se procedió a realizar una erosión al mapa de anomalías (x) con un elemento estructurante (b_1) seguido de un dilatamiento con un elemento estructurante distinto (b_2), con la intención de obtener una imagen en la que las anomalías de un tamaño menor a uno seleccionado no sean detectadas y las anomalías de un tamaño mayor a ese se recuadren en la imagen que se recibe directamente de la cámara, la imagen de la escena. Con esta metodología se consigue que las anomalías se recuadren en la medida de lo posible con un único recuadro, y que estas no se dividan en varios recuadros.

$$Erosión: y_1 = x \ominus b_1$$

$$Dilatación: y_2 = y_1 \oplus b_2$$

El programa permite seleccionar el tamaño de ambos elementos estructurantes, y la forma del segundo.

6 PRUEBAS Y RESULTADOS

6.1 Introducción

Una vez finalizada la implementación, se han diseñado una serie de pruebas para probar el funcionamiento de la interfaz gráfica, así como del funcionamiento del algoritmo en un entorno real. Para ello, se ha hecho hincapié en los siguientes aspectos:

- **Estabilidad de la interfaz gráfica**. Se ha puesto especial hincapié en la estabilidad de la aplicación, para su correcto funcionamiento independiente del orden en el que usuario realice las operaciones. Se ha buscado de esta forma dotar a la aplicación de la robustez suficiente necesaria para su uso en un escenario real de vídeo vigilancia.
- **Funcionamiento del algoritmo**. La interactividad de la interfaz gráfica y las medidas tomadas para poder alterar el valor de los parámetros, permite a los investigadores y desarrolladores determinar de manera eficiente los valores óptimos de los parámetros para cada escenario particular. En el estado inicial del algoritmo, este trabajo resultaba especialmente tedioso ya que requería ejecutar una nueva instancia para cada combinación distinta de parámetros. Esta tarea ha requerido un estudio a fondo del funcionamiento del algoritmo y sus imágenes intermedias previas a la generación de la máscara de anomalías.
- **Aplicación en tiempo real**. Muchas de las técnicas en el estado del arte para análisis de vídeo, si bien ofrecen resultados aceptables, son computacionalmente complejas, lo que dificulta el análisis en tiempo real para resoluciones de vídeo aceptables. En este trabajo se ha evaluado la aplicabilidad del algoritmo para la monitorización de un escenario en tiempo real.

6.2 Entorno de pruebas

Durante el desarrollo de este trabajo, se han realizado pruebas del correcto funcionamiento del algoritmo para cada nivel de integración descrito en el capítulo 4. Esto ha permitido aislar el correcto funcionamiento del algoritmo y los métodos implementados tras su encapsulación en la clase `DiVAAlgorithm`, previo al desarrollo de la interfaz gráfica. Para la realización de estas pruebas se ha tenido como principal entorno los ordenadores y cámaras disponibles en VPU-Lab. Para los aspectos relativos únicamente a la interfaz gráfica, se han realizado pruebas utilizando un `FrameServer` conectado a una webcam local.

Se ha trabajado con dos entornos de vídeo para realizar las pruebas. En primer lugar, se ha utilizado un `FrameServer` para una *webcam* conectada a un ordenador en el propio laboratorio de trabajo. En este primer entorno se han ido probando todos los cambios realizados en el algoritmo, desde su correcta ejecución como la validez de los resultados obtenidos. Para ello, se ha verificado que el valor de los parámetros y la generación de imágenes intermedias

fuera correcta y consecuente con el funcionamiento del algoritmo. Se han comparado los resultados obtenidos con los ya disponibles en VPU para el mismo algoritmo.

El entorno del laboratorio es interesante por dos motivos, cada uno referente a un ámbito de comprobación de errores. El primer motivo es la facilidad de acceso a la cámara mencionada. Debido a que la misma se encontraba instalada en el ordenador donde se realizaron las pruebas, se pudo disponer de la misma en todo momento, pudiendo controlar lo que sucedía en la escena. Al tener imágenes de un entorno totalmente controlado, ha sido más sencillo realizar la verificación de las distintas etapas del algoritmo. Estas pruebas iniciales han permitido en etapas posteriores aislar las pruebas de ejecución de la interfaz gráfica del funcionamiento del algoritmo.

Por otra parte, se ha establecido un segundo entorno de trabajo, con una cámara PTZ colocada en el hall de la Escuela Politécnica Superior de la UAM. En la Figura 6.1 se muestra un fotograma de ejemplo de la vista elegida.

Figura 6.1: Vista de la cámara para el entorno de pruebas

Este entorno ha sido el elegido para realizar la demostración en el evento EventVideo ya que se encontraba en una zona con tráfico de personas moderado, no sujeto a cambios sustanciales de iluminación muy frecuentes y por contar con tres zonas diferenciadas, el pasillo central, la zona de información y el pasillo secundario a uno de los baños de la escuela. Esto permite visualizar en la imagen entrenada de comportamiento el tráfico de personas en estas tres zonas.

Este segundo entorno de pruebas ha sido elegido y utilizado después de comprobar la estabilidad y robustez de la interfaz gráfica en un entorno controlado. Bajo este segundo escenario principalmente se han comprobado comprobaron los resultados propios del algoritmo y la configuración de parámetros ideal para dicho escenario.

6.3 Pruebas en entorno real

Para la comprobación del algoritmo en este entorno ha sido necesario generar una imagen entrenada para posteriormente realizar la detección en vivo. Para esta tarea se ha realizado el siguiente procedimiento:

- Grabar secuencias cortas que incluyan solamente el movimiento “normal” de la escena, con los que se genera

la imagen entrenada.

- Grabar secuencias cortas en donde se aprecien algunas anomalías. Para esto han participado voluntarios del grupo de investigación.
- Para observar la robustez de algoritmo, se han grabado escenas que incluyen cambios de iluminación que ocurren habitualmente en esta vista de la cámara.
- Realizar la detección, utilizando la imagen generada en la fase de entrenamiento.

Para grabar los vídeos se ha contado con la ayuda de los miembros voluntarios en VPU-Lab. Para el vídeo de movimiento normal los participantes realizaron un movimiento de idas y venidas continuo, durante varios minutos, por el pasillo central y la zona de información, ya que se presupone que estas son las zonas habituales de movimiento *normal*. En la Figura 6.2 puede observarse un fotograma de estas secuencias de entrenamiento.

Figura 6.2: Vista de la secuencia utilizada para la imagen de entrenamiento

Para el vídeo de movimiento normal y anómalo, se pidió a los participantes moverse con libertad por todas las zonas de la escena. De esta manera se pretende comprobar que el algoritmo detecta con éxito patrones de movimiento anómalo y al mismo tiempo no detecta ninguna anomalía en patrones de movimiento normal. Se incluye un ejemplo en la Figura 6.3.

Figura 6.3: Vista de la secuencia utilizada para las pruebas de detección

También ha sido de gran importancia realizar grabaciones donde no hubiera movimiento alguno, para observar la evolución de los cambios de iluminación puntuales en la escena. Estos cambios de iluminación aparecen en los cristales o superficies reflectantes y también en zonas donde la luz natural o artificial incide de forma excesiva (zonas opuestas a ventanas o a focos de luz). Estos vídeos se han grabado con la finalidad de generar imágenes entrenadas con los cambios de iluminación que sufre la escena en las horas que se quiere probar el funcionamiento del algoritmo, ya que pueden resultar en la detección de falsas anomalías en la escena. Esto pone en evidencia sensibilidad del algoritmo a cambios de iluminación de cierta intensidad.

Todos los vídeos han sido grabados a una resolución de 640x480 a 16fps y con similares condiciones de iluminación.

Una vez compiladas las secuencias de vídeos necesarias, se procede a eliminar *frames* defectuosos debido a problemas observados en la cámara, como *frames* totalmente en negro. Una vez realizada esta tarea, se crea un nuevo vídeo a partir de continuas repeticiones del original, de esta manera se consigue una secuencia de vídeo con la duración suficiente para una correcta generación de la imagen de entrenamiento.

Después de obtener el vídeo con las características necesarias se generaron varias imágenes con distintas configuraciones de los parámetros del algoritmo con el fin de comprobar que configuración era la idónea para la escena y las condiciones de la misma, en particular en lo relativo a la iluminación. Se generaron por un lado imágenes con el movimiento normal, y por otro lado imágenes con la acumulación de los cambios de iluminación puntuales. Una vez obtenidas imágenes de ambos tipos estas se combinaron, de forma que para cada par de imágenes se obtenía una única imagen que era consecuencia de realizar el máximo pixel a pixel entre ambas imágenes. De esta manera se ha conseguido una mayor robustez a los cambios de iluminación en la escena.

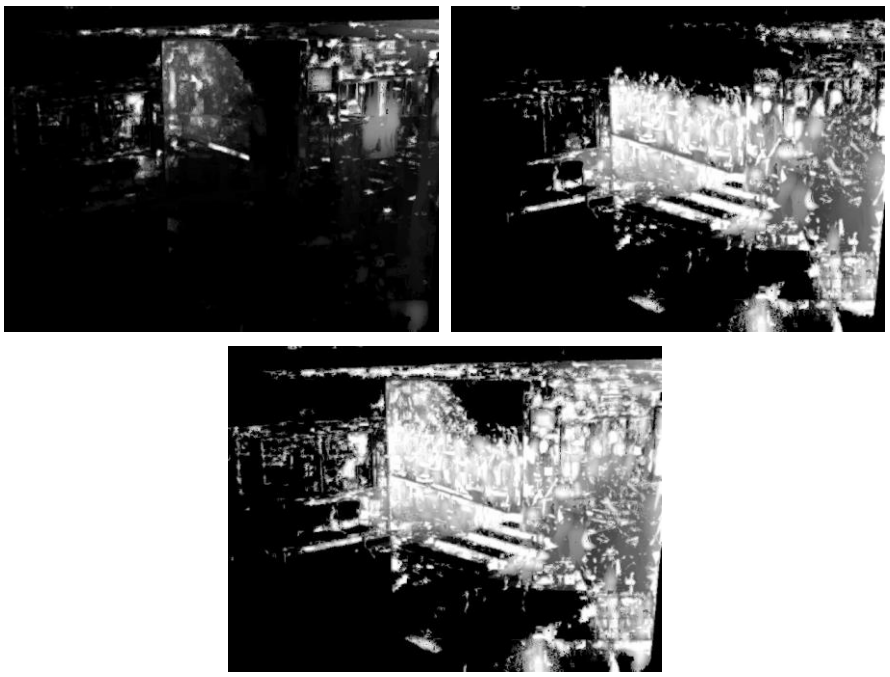


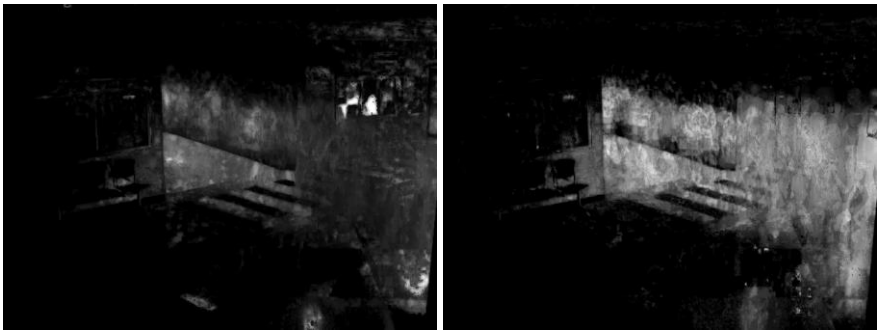
Figura 6.4: Imágenes entrenadas con el movimiento de la escena con distinta configuración de parámetros

Después de haber generado las imágenes necesarias, se realizaron repetidas pruebas con todas las imágenes obtenidas y diversas configuraciones de parámetros. Después de este procedimiento se determinó la configuración de parámetros con los que el algoritmo proporcionaba los mejores resultados para la escena escogida.

6.4 Resultados

En este apartado se exponen los resultados obtenidos y las conclusiones de las pruebas realizadas para preparar la demostración en EventVideo.

En primer lugar, se realizaron diversas imágenes de entrenamiento con los vídeos grabados para comprobar la eficacia de las mismas en la detección. El objetivo de esta práctica era encontrar la configuración de parámetros cuya imagen resultante analizara el vídeo con los mejores resultados, esto es, con la mínima ausencia de falsos positivos y falsos negativos.



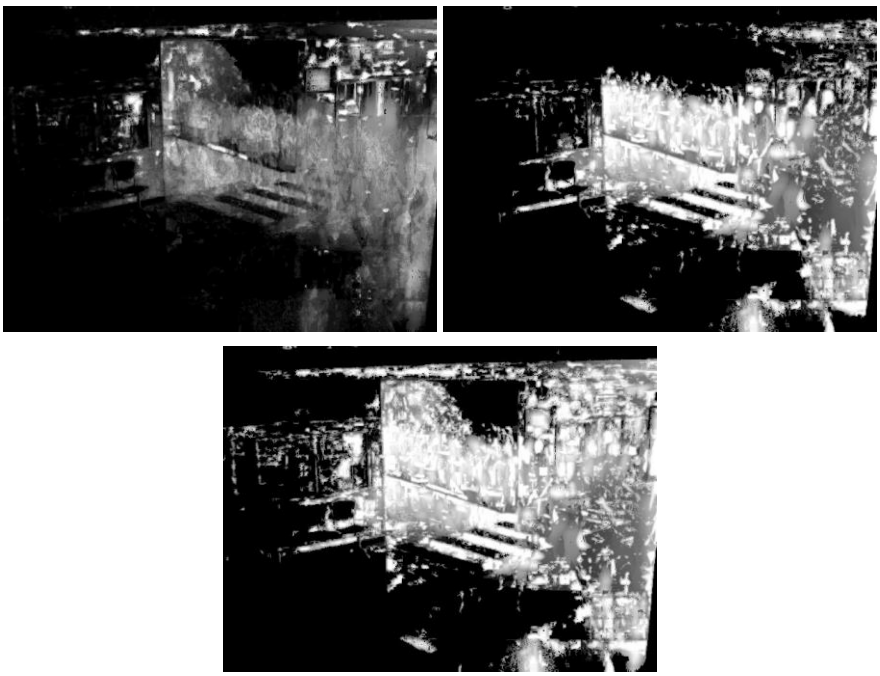


Figura 6.5: Imágenes utilizadas para realizar las pruebas del algoritmo

Finalmente se eligió la última imagen, puesto que era la más robusta frente a falsos positivos debidos a cambios puntuales de iluminación y al mismo tiempo no presentaba mayor índice de falsos negativos que el resto de las imágenes.

Después de sucesivas pruebas, se elige finalmente la siguiente configuración de parámetros para la búsqueda de anomalías en la escena en cuestión y con la imagen entrenada seleccionada:

- Umbral: 20
- Actualización: 0.015
- Contador HotStart: 35
- Contador de objetos estacionarios: 200
- Alpha: 0.02
- Mínimo tamaño de blobs a analizar: 5
- Tamaño mínimo de refinamiento: 1
- Tamaño de ventana: 5

Tras este estudio, se ha determinado que el parámetro más importante es la ventana de acumulación temporal para el descriptor de eventos (parámetro w, Tamaño de ventana), ya que esta acumulación es la que determina eventualmente si una determinada observación es una anomalía. Al utilizar un valor reducido para w en el entrenamiento, se fuerza la obtención de valores altos para el descriptor de eventos, lo que se traduce en una reducida posibilidad de obtener falsos positivos. Cabe destacar que esta estrategia es sólo aplicable cuando tanto el tamaño como la velocidad de los objetos es muy similar en toda la escena, como es el caso de la vista con la que se trabaja. Si se plantea otro escenario, por ejemplo

una cámara grabando a una autopista, el valor de w debería ser mayor, ya que los objetos se mueven a velocidades más alta. De esta forma, con la imagen extraída, se podrá detectar como anomalía un vehículo o un objeto que presenten un movimiento anormalmente lento para la vía o que se encuentren inmóviles.

También se eligieron valores pequeños del tamaño mínimo de refinamiento y el tamaño mínimo de blobs a analizar para que todos los destellos en los cristales y todos los pequeños cambios en los pixeles debidos a cambios en la iluminación se reflejaran en la imagen entrenada del mismo modo que refleja el movimiento de las personas. De este modo se evitan posibles falsos positivos o anomalías que no son de interés para la aplicación a realizar.

Después de obtener las imágenes se ha procedido a comprobar los resultados de las mismas con nuevos parámetros, en esta ocasión los valores del tamaño mínimo de refinamiento y el tamaño mínimo de blobs a analizar son ligeramente más elevados para eliminar los cambios en los pixeles de objetos muy pequeños, puesto que ya no es de interés analizar los cambios debidos a la iluminación.

A continuación se detallan algunas de las anomalías que se han probado en este escenario. En la Figura 6.6 se observa como el tránsito de una persona que se encuentra en la "zona permitida" de la escena no es detectado como anomalía. Por el contrario, en la Figura 6.7 podemos observar un comportamiento anómalo, al haberse quedado la persona detenida en una zona donde no suele haber tráfico de personas ("zona prohibida"). De la misma forma, una persona que se queda detenida o transita muy lentamente, provocaría también la detección de una anomalía.

Adicionalmente, en la Figura 6.7 se puede observar el resultado de la operación adicional que se ha desarrollado para mostrar con un único recuadro varios grupos de píxeles cercanos del mapa de anomalías.

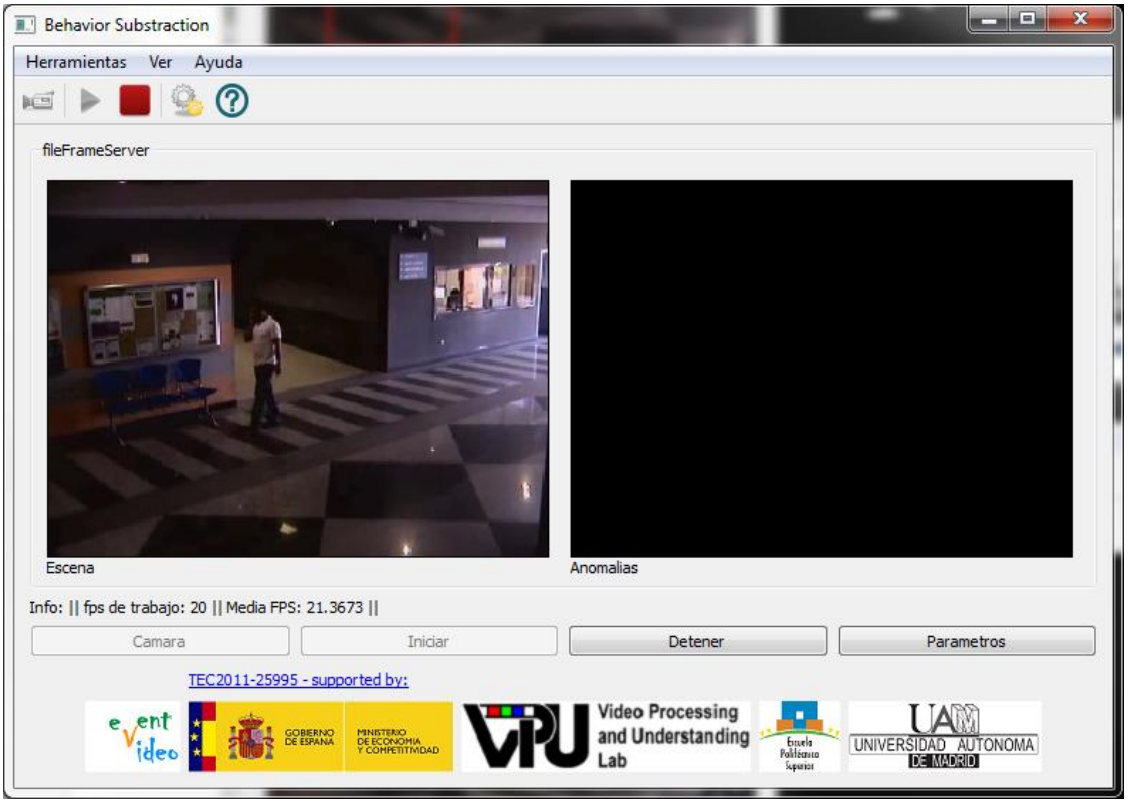


Figura 6.6: Ejemplo de funcionamiento sin detección de movimiento anómalo

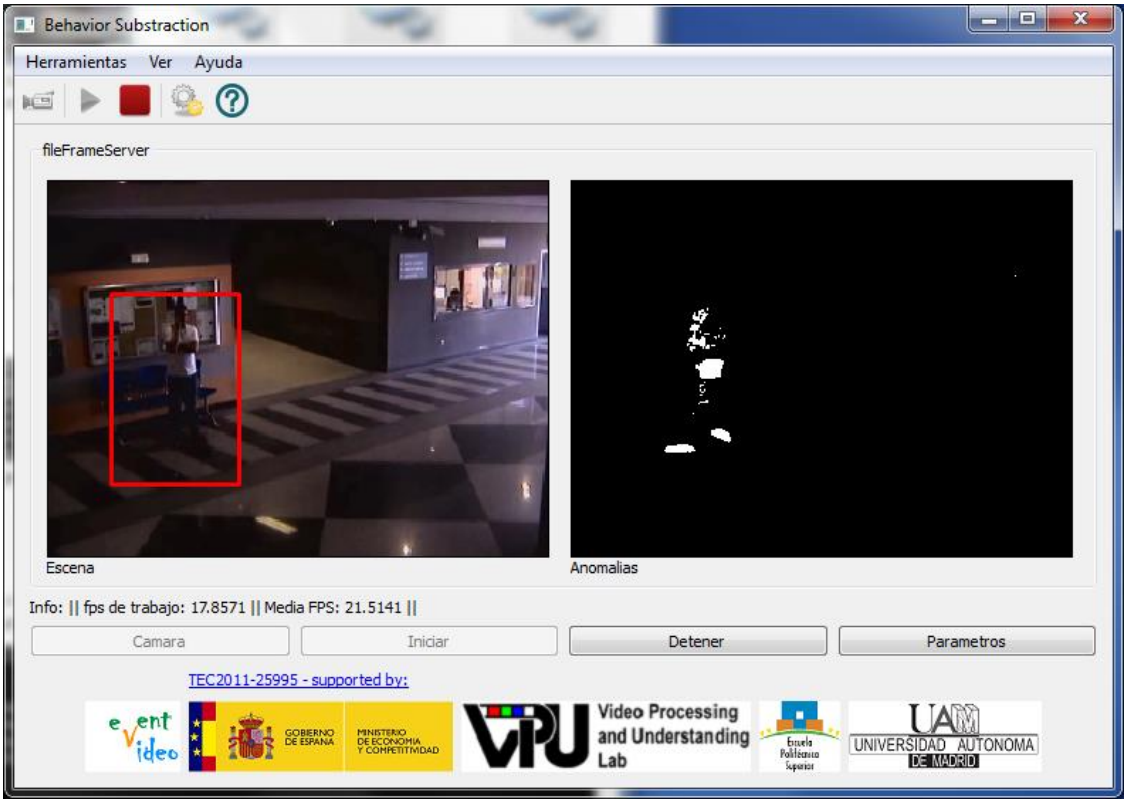


Figura 6.7: Ejemplo de funcionamiento con detección de movimiento anómalo

Por último se han realizado pruebas para comprobar la velocidad a la que el algoritmo es capaz de trabajar. Para ello, se han realizado pruebas utilizando *frames* a la resolución propia de la cámara (640x480), y a la mitad de resolución (320x240). En este último caso, se ha observado que la imagen conserva el suficiente nivel de detalle para seguir detectando correctamente las anomalías de las secuencias de prueba. Adicionalmente, se han realizado las mediciones de velocidad (en *fps*) para dos configuraciones típicas de la plataforma DiVA: con el frameServer ejecutándose en el mismo ordenador que el algoritmo (local), y con el *frame server* en otro ordenador (remoto).

Los resultados se consigan en la siguiente tabla:

	Frame Server Local		*Frame Server* Remoto	
	25fps	25fps	25fps	25fps
	5min	2,5min	5min	2,5min
Tamaño imagen	Secuencia aleatoria	Grabación pruebaAnomalias3.avi	Secuencia aleatoria	Grabación pruebaAnomalias3.avi
360x240	10,1455 fps	21,2835 fps	10,9726	22,6578
640x480	1,76712 fps	5,50821 fps	2,40813	5,63905

7 CONCLUSIONES Y LÍNEAS FUTURAS

7.1 Conclusiones

El objetivo final de este trabajo era obtener una aplicación de vídeo vigilancia que trabajara sobre la plataforma DiVA y contara con una interfaz de usuario adecuada a las características tanto del algoritmo principal, como las necesidades de los futuros usuarios de la misma.

Para cumplir con el objetivo fue necesario realizar un estudio del algoritmo dado para su integración en el conjunto del sistema. Este estudio, tanto teórico, como práctico, reveló los aspectos positivos del mismo para un sistema de vídeo vigilancia, y al mismo tiempo, aspectos donde es conveniente seguir trabajando para mejorar el conjunto del sistema.

Por un lado, el sistema cuenta con funcionalidades novedosas y con gran potencial para aplicaciones de vídeo vigilancia, ya que la detección de eventos anómalos es un campo que en la actualidad no se encuentra asentado en sistemas comerciales.

Por otro lado, el sistema cuenta con la limitación de basarse únicamente en la ausencia o no de movimiento para determinar si se ha encontrado una anomalía o no, sin especificar las características de dicho movimiento. Otra de las limitaciones del algoritmo es la baja robustez frente a cambios de iluminación, ya que es muy sensible a ligeros cambios en los valores de los píxeles en este sentido. El algoritmo tampoco cuenta con las técnicas necesarias para distinguir píxeles en escenas con baja iluminación, y esto compromete en ocasiones los resultados.

Tras las pruebas realizadas, se ha determinado que reducir la resolución de la imagen de entrada a un tamaño de 320x240 píxeles permite al algoritmo trabajar a tiempo real en situaciones que así lo requieran, sin llegar a perder detalle sobre las observaciones que se desean detectar.

En cuanto a la interfaz gráfica, gracias a las pruebas realizadas para la exposición de EventVídeo, se ha comprobado que cumple con las expectativas en el sentido de que facilitar al usuario el manejo del algoritmo y la comprobación de los resultados del mismo.

7.2 Trabajo futuro

Después de la experiencia obtenida con este trabajo, a continuación se exponen las líneas generales de trabajo futuro que se consideran más importantes.

En primer lugar, se considera interesante mejorar las capacidades y los resultados del algoritmo en los siguientes aspectos:

- Mejorar su robustez a cambios puntuales de iluminación.
- Mejorar su módulo de modelación de eventos para que se analicen características de los objetos además del

tamaño. De esta manera se podrán distinguir distintos tipos de eventos anómalos y no sólo aquellos correspondientes al movimiento o la ausencia del mismo de objetos en un rango específico de tamaño.

- Optimizar el algoritmo para que las mejoras antes mencionadas puedan implementarse en tiempo real.

En cuanto a la interfaz de usuario, hay aspectos en los que aún es interesante seguir trabajando para realizarla más atractiva al público y al mismo aumentar sus distintas aplicaciones:

- Realizar una interfaz de usuario que trabaje de forma remota, esto es, realizar una aplicación cuya ejecución principal pueda ejecutarse en un equipo independiente al equipo en el que se entren visualizando los datos, pero que a su vez siga permitiendo alterar los parámetros del algoritmo. Esto puede hacerse utilizando el subsistema de base de datos de DiVA.
- Incluir una sección de alertas en la ventana principal con la información que se disponga del resultado del algoritmo después de las mejoras mencionadas en esta sección.

A parte de las mejoras en el sistema que del que ya se dispone, se considera interesante seguir con la investigación en la línea del tratamiento de vídeo para aplicaciones de vídeo vigilancia, ya que, gracias al trabajo realizado estos meses se ha podido comprobar el gran potencial que ofrecen estos sistemas.

REFERENCIAS

[1] S. K. Pal, A. Petrosino, and L. Maddalena, *Handbook on Soft Computing for Video Surveillance*. CRC Press Taylor & Francis Group, 2012.

[2] V. Autores, *Intelligent Distributed Video Surveillance Systems*. IET Digital Library, 2006.

[3] Axis, Ed., *Axis Formación*. [Online]. Available: http://www.axis.com/es/products/video/about_networkvideo/evolution.htm. [Visitada: 01-Jun-2014].

[4] E. Fullerton, "The History of Video Surveillance," *sfactory.co.za*. [Online]. Available: http://www.sfactory.co.za/downloads/The%20History%20of%20Video%20Surveillance.pdf. [Visitada: 01-Jun-2014].

[5] M. Green, J. Reno, R. Fisher, and L. Robinson, "The appropriate and effective use of security technologies in U.S. schools: A guide for schools and law enforcement agencies series: Research report," *National Institute of Justice*, 1999.

[6] F. Bashir and F. Porikli, "Performance evaluation of object detection and tracking systems," *Workshop on Performance Evaluation of Tracking and Surveillance*, 2006.

[7] E. Maggio and A. Cavallaro, *Video Tracking: Theory and Practice*. Wiley, 2011.

[8] J. C. San Miguel and J. M. Martinez, "Robust unattended and stolen object detection by fusing simple algorithms," presented at the Advanced Video and Signal Based Surveillance, 2008. AVSS'08. IEEE Fifth International Conference on, 2008, pp. 18–25.

[9] S.-W. Joo and R. Chellappa, "Attribute Grammar-Based Event Recognition and Anomaly Detection," presented at the Computer Vision and Pattern Recognition Workshop, 2006. CVPRW '06. Conference on, pp. 107–107.

[10] O. P. Popoola and K. Wang, "Video-Based Abnormal Human Behavior Recognition—A Review," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 42, no. 6, pp. 865–878, 2012.

[11] Axis, "AXIS Product Guide," *axis.com*. [Online]. Available: http://www.axis.com/files/brochure/pg_video_51113_en_1303_lo.pdf. [Vistada: 01-Jul-2014].

[12] C. Sánchez Bueno, "Entorno de Desarrollo de Aplicaciones de Vídeo-Seguridad Multicámara," Escuela Politécnica Superior, Universidad Autónoma de Madrid, 2014.

[13] Axis, "Axis Camera Station: Product Description," *axis.com*. [Online]. Available: http://www.axis.com/products/cam_station_software/. [Vistada: 01-Jun-2014]

[14] J. Drinkwater, "Video Surveillance IP Cameras: Product Comparison," *networkwebcams.com*. [Online]. Available: http://www.networkwebcams.com/ip-camera-learning-center/2011/04/14/axis-sony-panasonic-ip-compact-camera-group-test/. [Visitada: 01-Jun-2014]

[15] J. C. SanMiguel, M. Escudero-Vinolo, J. M. Martinez, and J. Bescós, “Real-time single-view video event recognition in controlled environments,” presented at the Content-Based Multimedia Indexing (CBMI), 2011 9th International Workshop on IS -, 2011, pp. 91–96.

[16] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, Jul. 2009.

[17] P. M. Jodoin, V. Saligrama, and J. Konrad, “Behavior Subtraction,” *Image Processing, IEEE Transactions on*, vol. 21, no. 9, pp. 4244–4255, 2012.

[18] L. Caro-Campos, “Anomaly Detection in Video Sequences,” Escuela Politécnica Superior, Universidad Autónoma de Madrid, Madrid, 2013.

[19] M. Piccardi, “Background subtraction techniques: a review,” presented at the Systems, Man and Cybernetics, 2004 IEEE International Conference on, 2004, vol. 4, pp. 3099–3104.

[20] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, “Detecting objects, shadows and ghosts in video streams by exploiting color and motion information,” presented at the Image Analysis and Processing, 2001. Proceedings. 11th International Conference on, 2001, pp. 360–365.

[21] V. Mahadevan, W. Li, V. Bhalodia, and N. Vasconcelos, “Anomaly detection in crowded scenes,” presented at the Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, 2010.

[22] J. C. San Miguel, J. Bescós, J. M. Martinez, and A. Garcia, “DiVA: a Distributed Video Analysis framework applied to video-surveillance systems,” presented at the Image Analysis for Multimedia Interactive Services, 2008. WIAMIS'08. Ninth International Workshop on, 2008, pp. 207–210.

[23] L. Di Stefano and A. Bulgarelli, “A simple and efficient connected components labeling algorithm,” presented at the Image Analysis and Processing, 1999. Proceedings. International Conference on, 1999, pp. 322–327