UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

PROYECTO FIN DE CARRERA

Ingeniería de Telecomunicación

PRESERVACIÓN DE PRIVACIDAD DE PERSONAS EN VÍDEO SEGURIDAD

Jaime Mateo Herrero

Octubre 2015

PRESERVACIÓN DE PRIVACIDAD DE PERSONAS EN VÍDEO SEGURIDAD

AUTOR: Jaime Mateo Herrero

TUTOR: José M. Martínez

Video Processing and Understanding Lab

Escuela Politécnica Superior

Universidad Autónoma de Madrid Septiembre de 2015

Trabajo parcialmente financiado por el gobierno español bajo el proyecto TEC2014-53176-R (HAVideo) (2015 - 2017)

Resumen

En este Proyecto Fin de Carrera se estudia la implementación de un módulo de preservación de privacidad para los actuales sistemas de vídeo vigilancia. El imparable despliegue de estos sistemas en muchos escenarios de nuestro día a día invita al desarrollo de técnicas que permitan ocultar los rasgos personales de los individuos involucrados en secuencias de seguimiento o *tracking,* siempre y cuando se pueda recuperar la secuencia original para su posible uso forense tras la correspondiente autorización judicial.

En primer lugar se realiza un estudio exhaustivo del estado del arte de las técnicas existentes que permiten ocultar los rasgos personales, tanto las reversibles como las que no lo son. Además, se presenta un escenario en el cuál se puede integrar el submódulo de privacidad. Una vez detalladas todas las técnicas, se elige aquella que mejor respeta los límites de privacidad, reversibilidad y seguridad.

A continuación, se programa un algoritmo que caracteriza la técnica elegida. Dicho algoritmo es evaluado con diferentes detectores de personas para estudiar su respuesta y se plantean alternativas para mejorar el funcionamiento del mismo. Una vez obtenidos los mejores resultados posibles, se integra dicho algoritmo en una aplicación que permite detectar personas tras la inclusión de la técnica seleccionada de privacidad y observar en tiempo real las características visuales del algoritmo mediante una serie de funcionalidades básicas.

Palabras clave

Visión artificial, privacidad, vídeo seguridad, detección de personas, *scrambling*, detector HOG, detector Latent SVM, RGB, YCrCb.

Abstract

This Master Thesis Project consists on studying the implementation of a privacy preserving module for the existing video surveillance systems. The unstoppable growth of these systems in many scenarios of our daily life implicate the development of techniques that permit hiding personal features of those individuals involved in tracking video sequences, providing that original sequence can be restored for forensic use after corresponding judicial authorization.

Firstly an exhaustive study of state of the art is performed. Existing privacy preserving techniques, those reversible and those that are not, are detailed. Furthermore, a scenario in which this privacy module can be integrated in is also presented. After this, the technique that respects privacy, reversibility and coding losses' boundaries is selected.

Later on, an algorithm that performs the selected technique is programmed. This algorithm is evaluated with several person detectors in order to study the behavior, introducing alternatives to improve its performance as well. Once best possible results are achieved, this algorithm is integrated in an application that detects persons after introducing the chosen privacy technique and shows visual real-time algorithm's features using some basic functions.

Key words

Computer vision, privacy, video surveillance, people detection, *scrambling*, HOG detector, Latent SVM detector, RGB, YCrCb.

Agradecimientos

Quiero agradecer este Proyecto Fin de Carrera al tutor del mismo José María Martínez y a los trabajadores del laboratorio VPULab que me han ayudado a llevar a cabo esta labor, especial gracias a Fulgencio y a Álvaro.

También quiero agradecer a la Universidad Autónoma de Madrid haberme dado la oportunidad de formarme como Ingeniero con los mejores profesores de la Escuela Politécnica Superior. También por otorgarme una beca Erasmus para cursar un año académico en la Universidad Tecnológica de Eindhoven, a la cuál también agradezco su acogida y todo el conocimiento y buenas experiencias que me aportó.

Este Proyecto también va dedicado a familiares y amigos cercanos, quienes me han apoyado desde el minuto uno. Gracias a mi abuela Gregoria, a mi madre Alicia y a mi hermana Irene.

Jaime Mateo Herrero

Septiembre 2015

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

1.1 Motivación

La bajada del precio de las vídeo cámaras y sistemas de almacenamiento de datos en la actualidad ha llevado a un despliegue imparable de dichos sistemas en todos los ámbitos de nuestro entorno diario (hogar, trabajo, transporte público…) [1]. Esta situación plantea diversos debates sobre el derecho a la privacidad dado que no se han aplicado hasta el momento técnicas de preservación, estando la mayoría de ellas aún en período de desarrollo o investigación.

En el caso de los circuitos cerrados de televisión (CCTV) se han desarrollado en los últimos años numerosas técnicas de procesamiento de vídeo como son el seguimiento de personas o la identificación de las mismas. Dichas aplicaciones son de vital importancia para detectar posibles actos criminales o conductas denunciables. Es el turno ahora de desarrollar algoritmos que permitan filtrar los rasgos personales que se consideren privados o que puedan llevar a casos de abuso o discriminación, evitando lo que algunos ya relacionan con la utopía de la novela de George Orwell "1984".

Pero, ¿qué consideramos privacidad? ¿Hasta qué punto es necesario preservarla? Unos pensarán que no es necesario que las cámaras filtren los rasgos que les identifican si nunca llegarán a realizar un acto delictivo o se verán involucrados en una situación que en un futuro pueda resultar embarazosa o perjudicial para su imagen. Sin embargo, una vez sus movimientos y acciones son monitorizados y almacenados, éstos pueden ser objeto de robo y uso fraudulento. Lo ideal sería desarrollar un sistema en el que los rasgos identificativos sean preservados automáticamente y puedan ser revelados cuando las personas involucradas lo requieran o cuando lo solicite, por ejemplo, un juez o forense en caso de necesidad de identificación de los sujetos involucrados en eventos sospechosos de delito.

1.2 Objetivos

En este Proyecto Fin de Carrera se realizará un estudio exhaustivo de las técnicas que permiten filtrar los rasgos personales en una secuencia de vídeo. Para ello, en primer lugar, se debe detectar mediante segmentación u otras técnicas la región (o regiones) de interés R.O.I. (Region Of Interest) de nuestra secuencia y a partir de ahí filtrarla para su imposible reconocimiento sin la autenticación correspondiente.

Es necesario mantener un compromiso entre la privacidad de la secuencia y el resultado de los algoritmos utilizados. Hay numerosas técnicas de protección de privacidad pero no todas ellas permiten recuperar la secuencia inicial porque degradan la señal original y, por lo tanto, son irreversibles. También se deberá tener en cuenta una serie de factores como puede ser la complejidad de implementación, la carga computacional o el grado de seguridad de cara a elegir la técnica más eficiente.

1.3 Organización de la memoria

- **Estudio del estado del arte**

Se estudia el estado de los sistemas de preservación de privacidad desarrollados por la comunidad científica hasta el momento y, en concreto, todos los esfuerzos realizados en el ámbito de las técnicas de preservación. Se enumeran los fallos y aciertos y se plantean los retos a los que se deben hacer frente a partir de ahora. Todas las técnicas son caracterizadas según su complejidad y características (degradación de señal, reversibilidad, coste computacional…), de cara a hacer una comparativa. Finalmente se elige aquella que se adecua más al marco de este PFC.

- **Desarrollo de algoritmos**

En esta fase del trabajo se desarrolla el algoritmo que realiza la técnica elegida para analizar su comportamiento y eficiencia. En primer lugar se explica en qué plataforma se desarrolla y qué lenguaje de programación se utiliza. Además, se analizan diferentes aproximaciones del algoritmo, aportando varias modalidades de cara a realizar una evaluación exhaustiva.

- **Evaluación de una aplicación**

La evaluación del algoritmo consiste en comprobar hasta qué punto la técnica elegida es compatible con los actuales sistemas de detección de personas. Para ello, la metodología de evaluación pondrá en práctica un detector de personas antes y después de introducir el módulo de privacidad. Se analizan los resultados obtenidos llegando a una conclusión sobre la validez del algoritmo de privacidad así como sus limitaciones o incompatibilidades.

- **Aplicación**

Finalmente se desarrolla una aplicación para visualizar el funcionamiento final del algoritmo. Para ello, se llevará a cabo una modularidad del código del algoritmo que permita integrarlo en la plataforma de desarrollo de la aplicación. Una interfaz gráfica con una serie de funcionalidades básicas es implementada para un posible uso por parte del cliente de la aplicación.

2 Estado del arte

2.1 Introducción

En este capítulo se recoge una amplia visión del estado actual de las técnicas de preservación de privacidad. En el apartado 2.2 se presenta un prototipo de sistema de vídeo seguridad que incluye un módulo de privacidad. Se describen las diferentes fases que todo sistema de estas características debe tener y se enumeran los objetivos que se deben alcanzar.

En la sección 2.3 se analizan las técnicas irreversibles de preservación de privacidad. Estas técnicas son utilizadas en muchas aplicaciones por su simplicidad y buenos resultados. Sin embargo, para el caso de los sistemas de vídeo seguridad son necesarias técnicas reversibles, siendo éstas estudiadas en la sección 2.4.

2.2 Sistemas de preservación de privacidad

Todo sistema de preservación de privacidad en vídeo seguridad cuenta con las siguientes etapas:

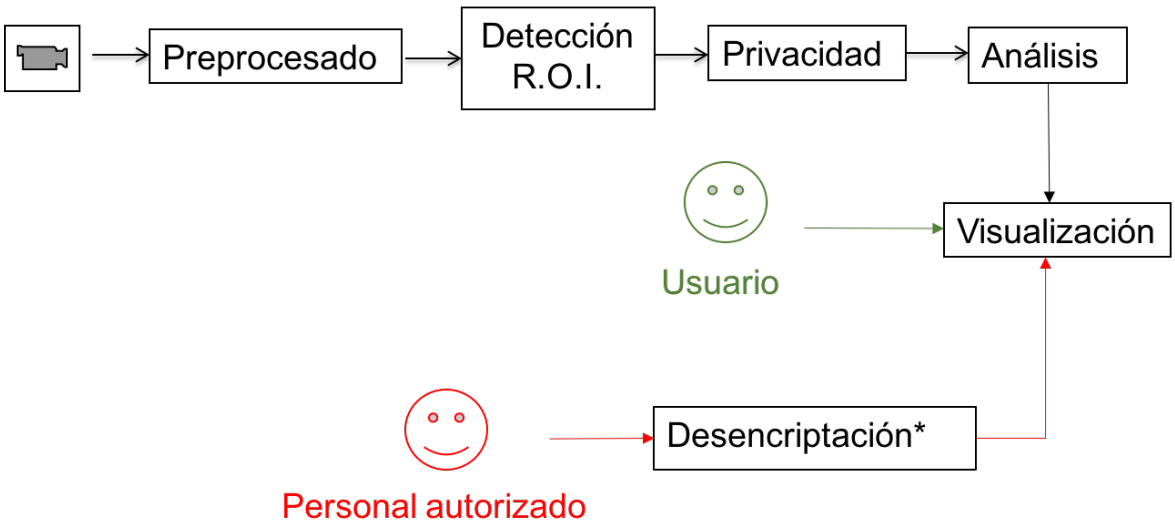

Figura 2.1 Diagrama de bloques de prototipo de preservación de privacidad

- Preprocesado: consiste en aquellas técnicas que permitan mejorar la eficiencia de los

algoritmos utilizados en etapas posteriores.

- Detección de R.O.I.: existen numerosas técnicas de detección de R.O.I. "Background subtraction" [2] es un algoritmo que permite modelar el fondo de la escena en estado de reposo. Cuando tengamos una secuencia de vídeo, se comparará cada una de ellas con dicho modelo y cualquier diferencia será considerada como movimiento o R.O.I. Cuando la R.O.I. no sea un objeto en movimiento (persona estática), se utilizarán técnicas de detección de personas basadas en reconocimiento facial o corporal.

- Privacidad: una vez detectada nuestra R.O.I. se aplicará un algoritmo que permita preservar la privacidad de la misma. Para mantener el compromiso entre privacidad e información resultante tras el procesado, se elegirá una técnica que permita la recuperación de la señal filtrada en caso de ser solicitada.

- Análisis: en este módulo se introducen técnicas que convierten al sistema de video seguridad en uno inteligente: detección de personas, detección de robos, detección facial… En este PFC se utiliza un detector de personas para comprobar cómo afecta la técnica de preservación de privacidad a la secuencia de vídeo.

- Visualización: en la aplicación real, el usuario visualiza la secuencia de vídeo con los resultados de la técnica de preservación de privacidad. Las personas que aparecen en la secuencia resultan irreconocibles para dicho usuario.

- Desencriptación: en caso de que la técnica de preservación sea reversible, existe una etapa en la cuál una persona autorizada puede recuperar la secuencia de vídeo original mediante la desencriptación de la técnica. Para ello es necesaria una clave o *"key"*. Los resultados de la visualización en este caso serán los de la secuencia de vídeo original, pudiendo identificar así a las personas que aparecen en el vídeo.

En este PFC no se estudia la gestión de datos privados (Privacy Data Management) [2]. Sin embargo, ésta es una de las fases más importantes de todo sistema de seguridad ya que es la encargada de la preservación y gestión de la información privada. Esta información debe ser almacenada (en caso de utilizar técnicas que lo permitan) para su posterior recuperación, siendo un agente mediador intermedio quien determine si la correspondiente solicitud cuenta con la autenticación correcta.

En [14] se enumeran las características que todo sistema fiable preservación de privacidad en vídeo seguridad debe tener:

- Alta precisión en la detección de las ROI (si el submódulo de detección tiende a detectar falsas ROIs o no las detecta, el sistema tendría que pasar a ser supervisado

por una persona, perdiendo así la privacidad).

- Reversibilidad en el método de protección (si no es reversible se pierde el sentido de la seguridad en muchos escenarios).
- Funcionar en tiempo real para evitar el almacenamiento de frames no protegidos.
- Asegurar la información protegida mediante técnicas de criptografía y protocolos fiables.

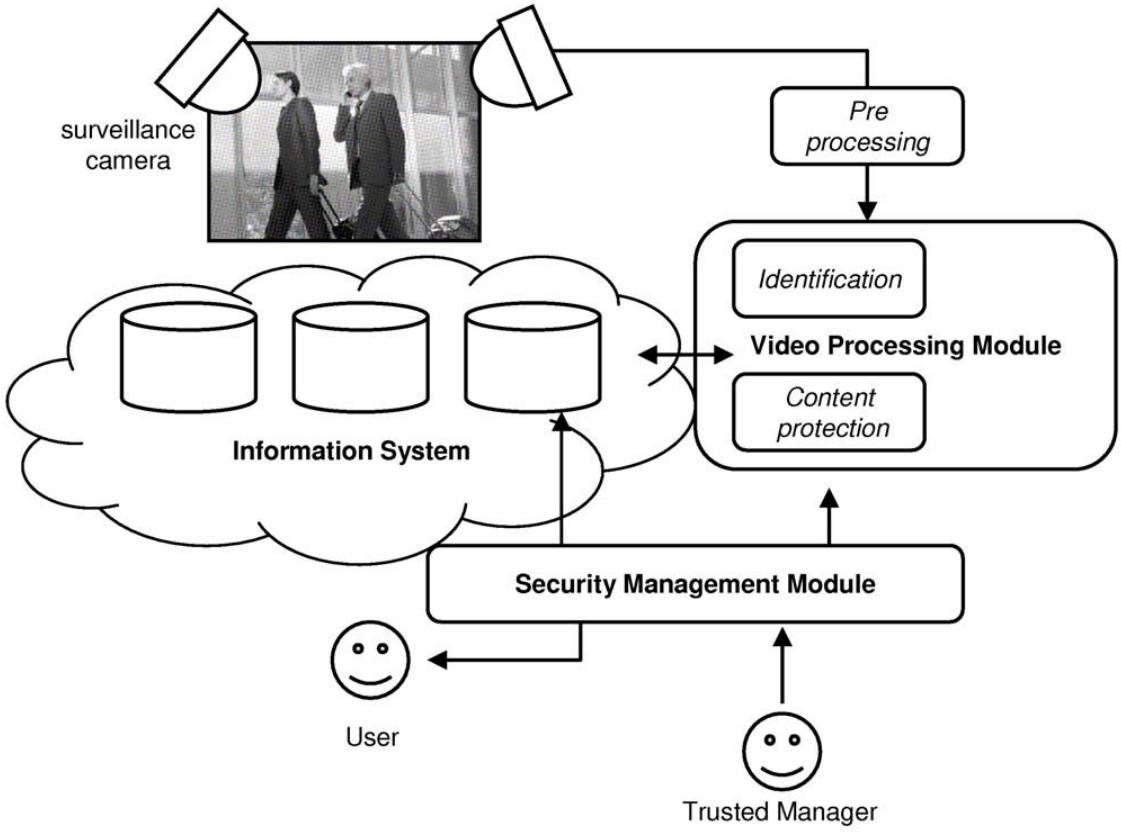

Figura 2.2: Ejemplo de un sistema de vídeo seguridad con un módulo de privacidad y otro de gestión segura de la información [14].

En los próximos apartados se estudian las técnicas más conocidas dentro del campo de preservación de privacidad, tanto las irreversibles (sección 2.2) como las reversibles (sección 2.3). A continuación, se analiza el caso de los sistemas RFID (sección 2.4) y su propuesta para englobar las técnicas de preservación en un escenario concreto. Por último, se llegará a una conclusión sobre la técnica que mejor cumple las exigencias enumeradas arriba para su posterior evaluación (capítulo 4) e integración en una aplicación (capítulo 5).

2.3 Técnicas irreversibles de preservación de privacidad

Las técnicas que permiten ocultar ciertos detalles en una imagen han sido objeto de estudio exhaustivo en los últimos años. Existen numerosas formas de alterar el contenido del *frame* para no mostrar aquello que queremos ocultar. En este primer apartado se analizan aquellas técnicas que ofrecen resultados con un notable grado de privacidad pero que, sin embargo, deterioran la imagen de manera que es imposible recuperar su estado original.

2.3.1 Obfuscation

Consiste en reducir el nivel de detalle en regiones de la imagen de manera que las personas no pueden ser reconocidas aunque su comportamiento permanece inalterado [3]. Existen numerosas técnicas que van desde el uso de cajas negras o *pixelados* hasta quitar el mismo objeto de la imagen o reemplazarlo [2]. Las tecnologías de preservación de privacidad desarrolladas hasta hace poco se han centrado principalmente en estas técnicas sin llegar ha proponer una solución definitiva. El principal escollo de las mismas es el hecho de destruir la naturaleza de la imagen y sus características, limitando su utilidad para muchas aplicaciones u objetivos.

2.3.1.1 Cartooning

En [4] Erdélyi, Winkler y Rinner proponen una solución para el MediaEval Privacy Task 2013 que consiste en aplicar un efecto *"cartooning"* de manera que la identidad del sujeto queda preservada mientras que la información sobre su comportamiento se mantiene. El efecto se aplica a la ROI en el frame de la imagen siguiendo los dos pasos siguientes:

1. Se aplica un filtro Mean Shift con una ventana espacial de radio 20 y una ventana de color de radio 40. Esto hace que la imagen sea más lisa reduciendo así el número de colores dando como resultado formas que se asemejan a un dibujo animado.

2. Para mejorar el resultado, una copia de los contornos de la imagen original son copiados en el frame procesado. Esto se consigue aplicando una máscara de gradiente del detector de bordes Sobel. La imagen resulta así menos borrosa y más parecida a un dibujo animado donde los contornos han sido remarcados.

(a) Original. (b) Modified. (c) Original. (d) Modified.

Figura 2.3: Resultados visuales de la técnica Cartooning. (a) y (c) son las imágenes originales y (b) y (d) muestran el resultado de aplicar la técnica. Se puede observar el efecto de dibujo animado tras aplicar la máscara que enfatiza los contornos de la imagen [4].

2.3.1.2 Pixelation

La aplicación deliberada de la pixelación es una técnica ampliamente conocida que consiste en maximizar el tamaño de los píxeles de una imagen o una porción de la misma de manera que resulten visibles al ojo humano. Esta técnica se utiliza habitualmente para "emborronar" los rostros de personas en medios de comunicación.

En [5] Boyle, Edwards y Greenberg utilizan un filtro de pixelado que produce un efecto "mosaico" en la imagen. Para ello se realiza un resampleado de la imagen que consiste en dividir la imagen en rectángulos de la misma área formando una cuadrícula. Los píxeles dentro de una misma celda son modificados al valor medio de todos los píxeles originales. Este filtro es computacionalmente simple y puede ser aplicado en secuencias de vídeo en tiempo real.

2.3.1.3 Blurring

Por otro lado, en [5] también se estudia el efecto de aplicar un emborronado. Los resultados muestran un efecto suavizado en las regiones de la imagen donde se utiliza. El filtro en forma de caja aplica a los píxeles un cambio de valor tomando la media de los píxeles vecinos.

Este filtro es muy utilizado hoy en día en diferentes aplicaciones. Google Street View, por ejemplo, tiene un sofisticado software que detecta caras y matrículas de coche para emborronarlas mediante esta técnica. A pesar de todo, el software no es perfecto y todavía

hay casos en los que hay que solicitar que apliquen la técnica en imágenes donde no se han detectado las regiones de interés.

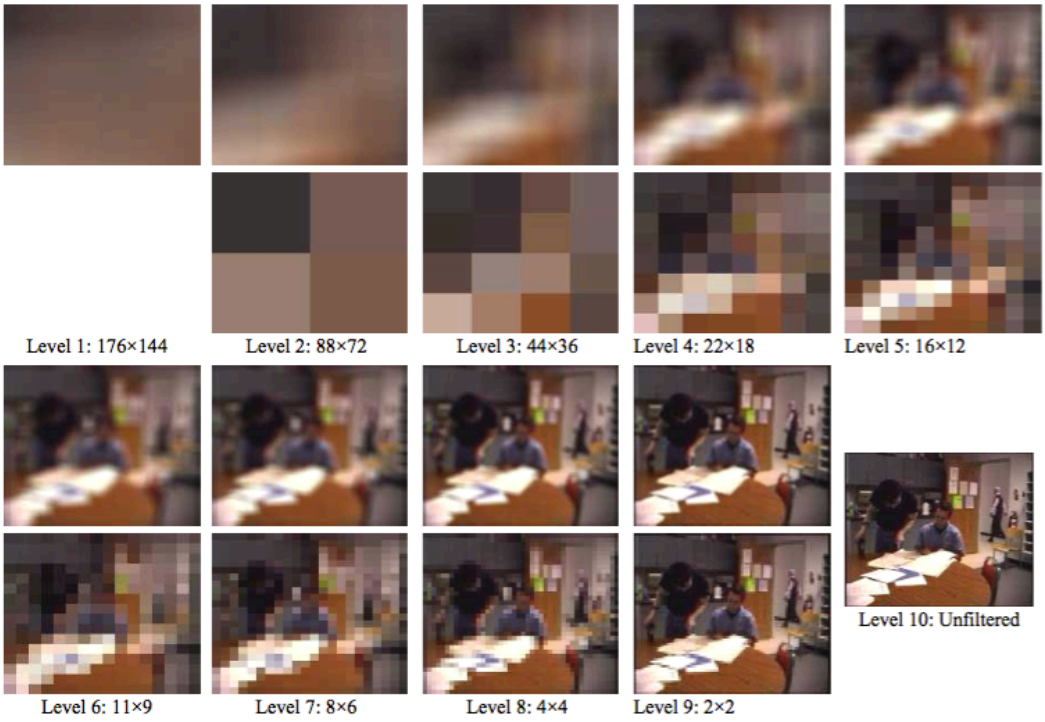

Figura 2.4: Ejemplo los filtros de pixelado (arriba) y emborronado (abajo) con nueve niveles de grado (de mayor a menor tamaño de las cajas) [5].

2.3.2 Blanking

Una forma de no dejar rastro de información sobre una persona (y su comportamiento) en una imagen consiste en quitar esa misma región dejando un cuadro negro [3]. Esta técnica se conoce como *blanking* y aunque promete una preservación absoluta de la privacidad, esto la convierte en una técnica útil para aplicaciones demasiado básicas de vídeo vigilancia o detección de intrusos ya que la única información que conserva es la presencia de una persona en el frame y su posición. A pesar de esto, *blanking* es una de las técnicas más utilizadas en preservación de la privacidad en vídeo seguridad.

Esta técnica enlaza con la de Video Inpainting (capítulo 2.4.1.) ya que es posible restaurar esa zona preservada colocando un fondo o background dando la sensación de que no hay o ha habido una persona en el frame.

Figura 2.5: Ejemplo de detección de persona (a), así como la diferencia entre la aplicación de la técnica *"blanking"* (b) y la técnica *"pixelation"* (c) [3].

2.3.3 Abstraction

Esta técnica reemplaza las regiones de interés con cajas o, en caso de personas, con avatares, figuras o siluetas [3]. Dependiendo del tipo de abstracción, la identidad, el comportamiento o ambas características de la persona pueden ser preservadas o no.

En [6] Maniry, Acar y Albayrak plantean ocultar la identidad de la persona reemplazando el cuerpo por una silueta definida mediante los contornos en movimiento. Utilizan una adaptación del filtro "foreground" utilizado por O'Gorman en [7]. Cada píxel que se encuentra en el contorno del individuo se cambia a un determinado color y el fondo del primer frame es utilizado para cubrir el interior de la región que se quiere preservar:

Figura 2.6: Ejemplo de abstracción utilizando un filtro *"foreground"* [7].

En [8] y [9] Pavel Korshunov y Touradj Ebrahimi utilizan un filtro para ocultar los rasgos faciales remplazándolos por una representación gráfica:

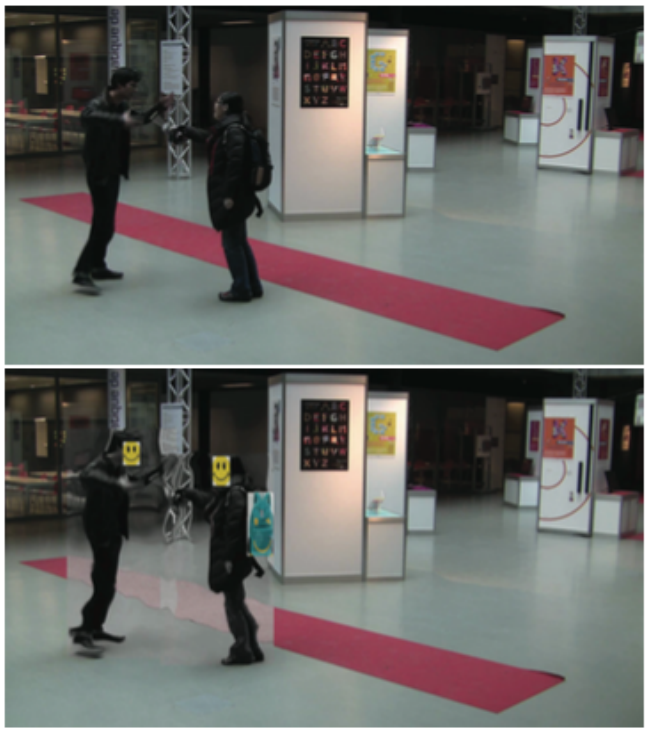

Figura 2.7: Ejemplo de la técnica de abstracción [8] y [9].

2.4 Técnicas reversibles de preservación de privacidad

2.4.1 Encriptación

Las técnicas de preservación de privacidad que incorporan en alguna de sus fases un módulo de encriptación son las más robustas y las que generan más confianza de cara a la seguridad de los datos protegidos. La encriptación es el proceso de codificar el mensaje de manera que sólo el personal autorizado pueda leer la información [11]. En el caso de la vídeo seguridad, la encriptación se utiliza para codificar los códigos que determinan los cambios en un determinado frame. Estos códigos o valores numéricos se conocen como *"seeds"* y son los que generan una serie de números aleatorios (pseudo-random generation) que cambian las características o propiedades de la imagen para hacer irreconocible la región a preservar. Las técnicas que se estudian a continuación utilizan un módulo de encriptación.

2.4.1.1 Video Inpainting

Video Inpainting consiste en quitar una parte del frame y rellenar el hueco dejado mediante técnicas de interpolación de la imagen [10]. Los beneficios potenciales de esta técnica son notables aunque en la actualidad presenta un alto grado de complejidad computacional. Esta técnica es reversible dado que utiliza encriptación (sección 2.4.2.) para preservar esa ROI separada del frame.

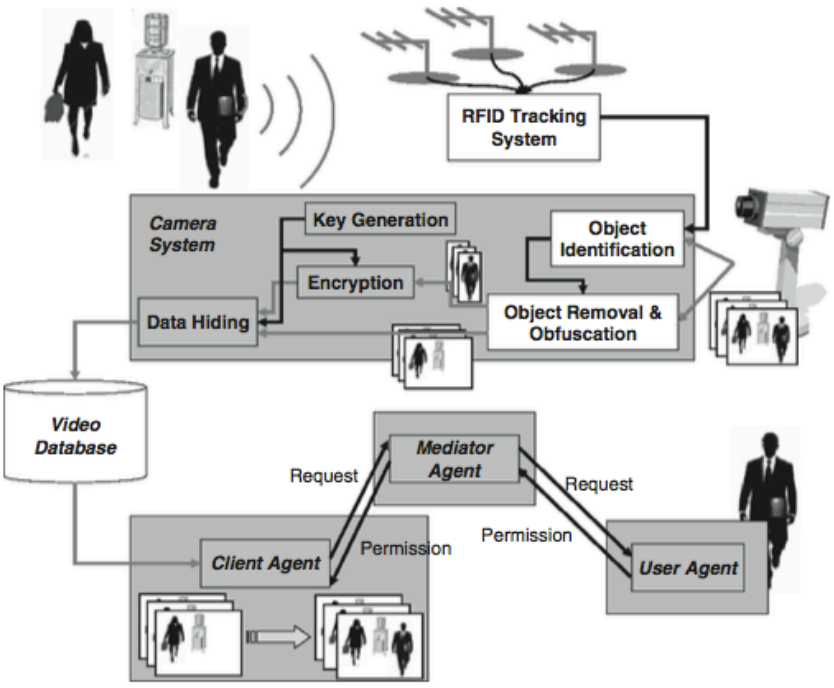

Figura 2.8: Esquema de un sistema de seguridad integrado con preservación de privacidad mediante Video Inpainting [2].

La técnica de *Video Inpainting* elimina del frame la región a ocultar mediante técnicas de *background substraction* y segmentación. A continuación, en un módulo trabaja con diferentes algoritmos de interpolación para la restauración del hueco dejado por la parte extraída para preservar. En otro módulo esta ROI se encripta y se embebe en cada frame de manera que el decodificador pueda reconstruir la imagen original cuando le llegue el bitstrem, siempre y cuando se tenga acceso a la clave para desencriptar.

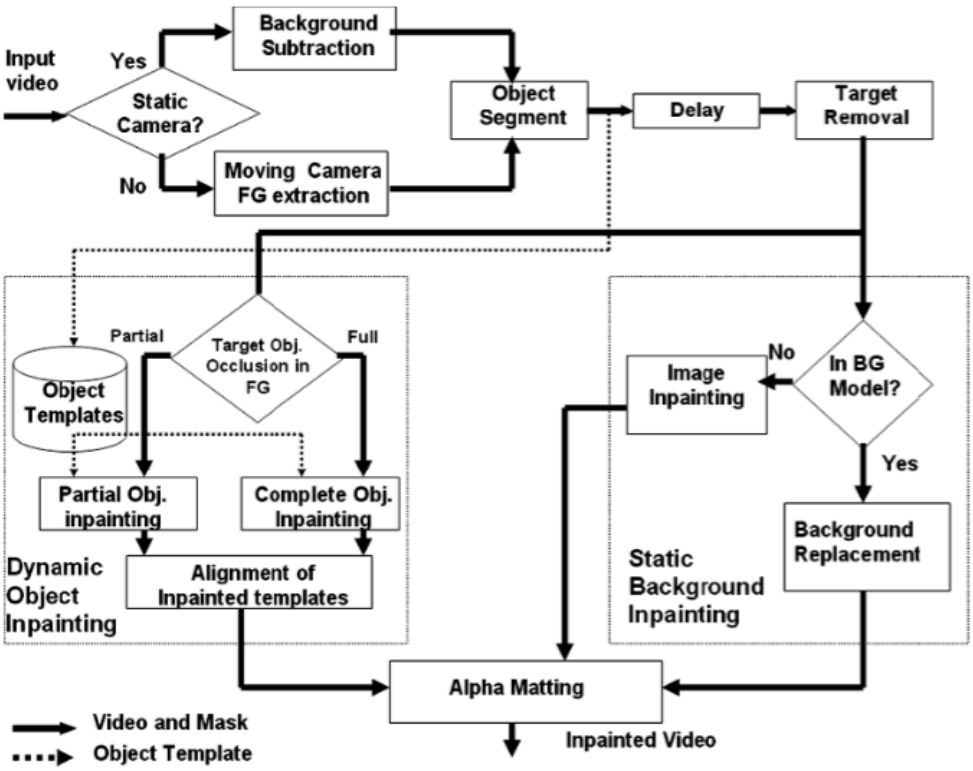

Figura 2.9: Diagrama de bloques del sistema de Video Inpainting [2].

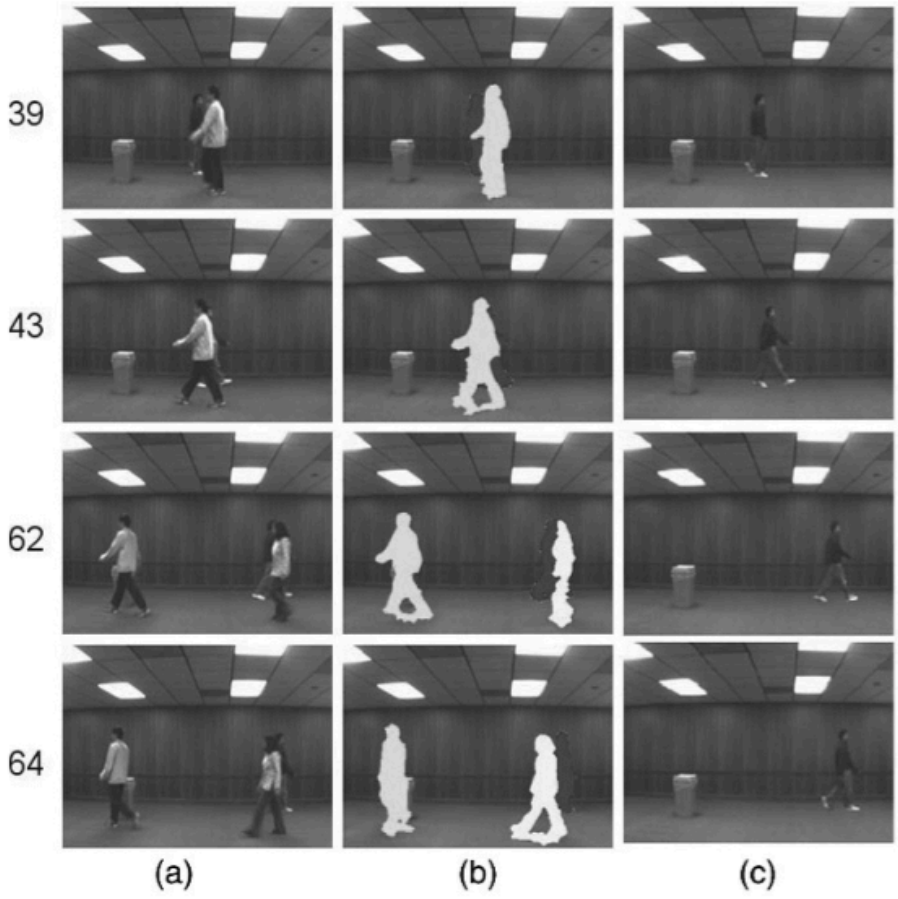

Figura 2.10: Resultado de aplicar el algoritmo de Video Inpainting. (a) Esta columna muestra cuatro frames del vídeo de entrada. (b) muestra el resultado del tracking y la segmentación de los individuos. (c) elimina del frame las personas cuya privacidad se desea preservar mientras que se restaura el hueco que dejan [2].

2.4.1.2 Face Morphing

Pavel y Touradj plantean en [12] una solución al problema de la privacidad en los sistemas de vídeo vigilancia. El método ofrece reversibilidad y seguridad y se basa en una técnica de *morphing* convencional. Para ello, la imagen deformada será interpolada en un punto intermedio entre la imagen original y la imagen objetivo, ajustando las intensidades de los píxeles de manera que en el resultado final los rostros faciales iniciales resulten irreconocibles. La imagen original puede además ser recuperada aplicando la técnica inversa de transformación, dado que se conocen los valores de interpolación e intensidad. Una vez llevada a cabo esta reversibilidad, las técnicas de reconocimiento facial siguen funcionando bien aunque la distorsión facial haya sido intensa.

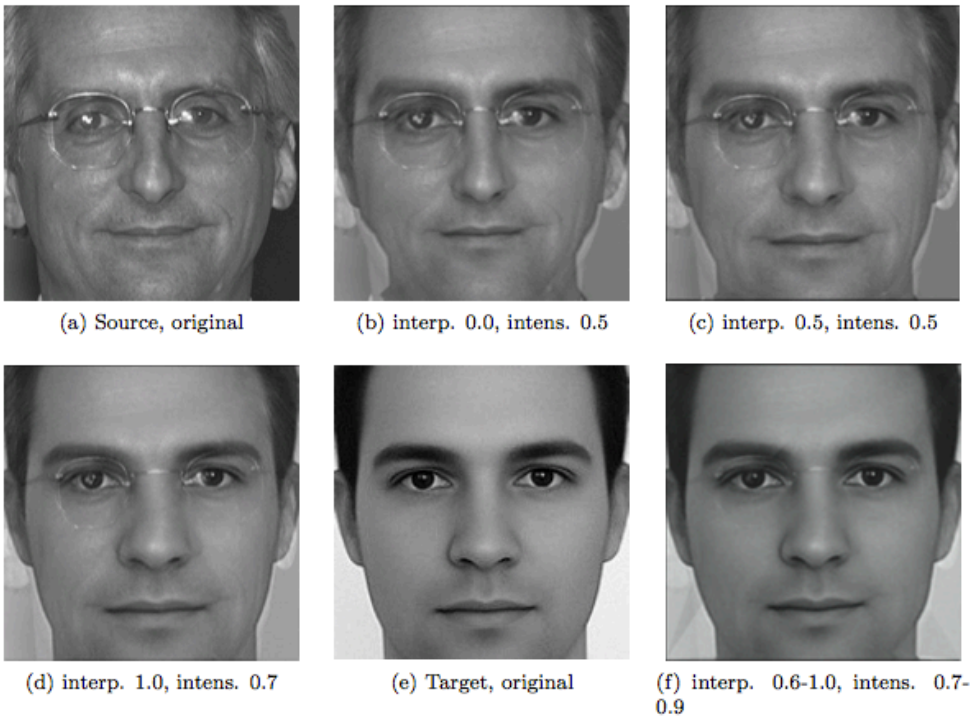

Figura 2.11: Ejemplo de deformación facial o Face Morphing con diferentes grados de intensidad y niveles de interpolación [12].

Como *Face Morphing* se aplica a los valores de los píxeles, el método es independiente de la fase de compresión. Además, la seguridad puede depender de una clave secreta que determine el número aleatorio o *seed* que genera los grados de intensidad, los niveles de interpolación y los puntos usados para la triangulación. El algoritmo funciona de la siguiente manera:

- Automáticamente selecciona los puntos clave de la imagen original (source) y la objetivo (target) detectando los ojos, nariz, boca…
- Para cada par de puntos correspondientes se determina uno intermedio usando un determinado valor de interpolación.
- Divide las imágenes usando la triangulación Delaunay [13] y los puntos que determinan los vértices del triángulo.
- Encuentra las coordenadas de los píxeles en la imagen final después de aplicar interpolación a la imagen original y a la objetivo.

- Para cada uno de esos píxeles, calcula su peso como la suma de intensidades entre los píxeles correspondientes de la imagen original y la objetivo. Los pesos se calculan con los valores de las intensidades aplicando la siguiente fórmula:

  If = (1 wi)Is + wiIt,

  donde If es la intensidad final, Is la de la imagen original e It la de la objetivo.

Para recuperar el rostro original, los puntos clave y el rostro objetivo deben conocerse. El algoritmo de recuperación es el mismo que el de la operación morphing con la única diferencia de que el rostro fuente es estimado, mientras que el deformado y el objetivo son conocidos. La seguridad del método queda resuelta mediante la encriptación de los puntos clave: niveles de interpolación y grados de intensidad de los triángulos deformados.

2.4.1.3 Scrambling

Las técnicas de *Video Scrambling* se utilizan para prevenir el acceso desautorizado a secuencias de vídeo. Su funcionamiento se basa en alterar los coeficientes de la DCT (o cualquier otra transformada) realizando cambios de signo, rotaciones aleatorias, cambios de bloques, etc [15]. Todo ello controlado mediante una clave o "key" que genera de forma aleatoria un número que determina si se realiza un cambio en el coeficiente y de qué tipo. La clave que genera la alteración es encriptada para que sólo la persona autorizada conozca la secuencia aleatoria que permite recuperar el frame original.

Las alteraciones pueden implementarse en la secuencia de vídeo transformada (*transform domain*) o en la secuencia de vídeo original (*spatial domain*). El problema de trabajar las técnicas de scrambling en el segundo de los casos es que los coeficientes pierden las propiedades estadísticas de la secuencia de vídeo, dificultando así su posterior compresión y reduciendo drásticamente la eficiencia. Además, aplicar scrambling en el dominio temporal facilita los ataques que aprovechan la correlación espacial y temporal características del vídeo [16].

Una vez decidido trabajar en el dominio frecuencial, se considera además hacerlo en el marco de la compresión 8x8 de la DCT dado que es el más utilizado por los principales estándares (JPEG, MPEG-4, H.264…) que se utilizan en vídeo seguridad [15]. La estructura de una secuencia de vídeo particionada se representa a continuación:

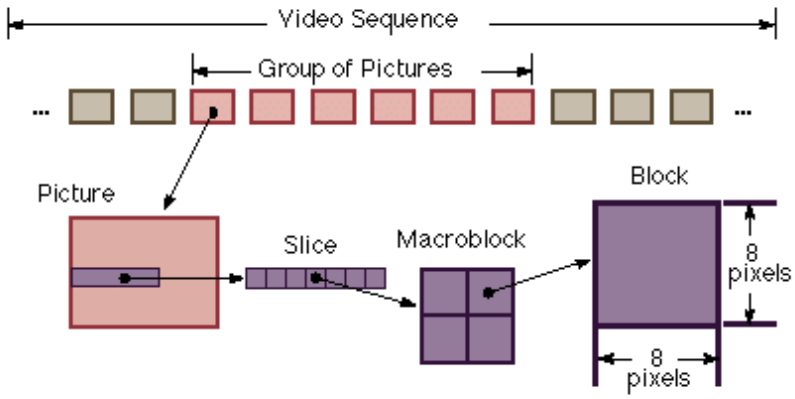

Figura 2.12: Esquema de una secuencia de vídeo (obtenida en http://www.iem.thm.de/telekom-labor/zinke/mk/mpeg2beg/beginnzi.htm)

Dicha secuencia está formada por varios *Group of Pictures*. Los tres tipos de imágenes que conforman dicho grupo según el estándar MPEG son imágenes tipo *I* (*intra pictures*), tipo *P* (*predicted picture*) y tipo *B* (*bidirectional predicted picture*). Dichas imágenes o *pictures* están formadas a su vez por *slices* rectangulares divididades en *macroblocks* (16x16). Cada uno de estos *macroblock* contiene 4 *blocks* (de 8x8) de luminancia, uno de crominancia azul y otro de crominancia roja.

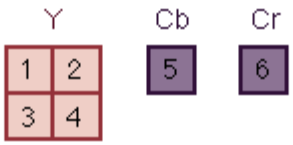

Figura 2.13: Componentes de un macroblock (obtenida en http://www.iem.thm.de/telekom-labor/zinke/mk/mpeg2beg/beginnzi.htm)

Son estos bloques 8x8 a los que se les aplica la DCT (*Discrete Cosine Transform*), obteniendo una matriz con la energía de la señal concentrada en los primeros coeficientes. Una vez cuantificados estos coeficientes, se les aplicará la técnica de scrambling a estudio.

139	144	149	153	155	155	155	155
144	151	153	156	159	156	156	156
150	155	160	163	158	156	156	156
159	161	162	160	160	159	159	159
159	160	161	162	162	155	155	155
161	161	161	161	160	157	157	157
162	162	161	163	162	157	157	157
162	162	161	161	163	158	158	158

235.6	-1.0	-12.1	-5.2	2.1	-1.7	-2.7	1.3
-22.6	-17.5	-6.2	-3.2	-2.9	-0.1	0.4	-1.2
-10.9	-9.3	-1.6	1.5	0.2	-0.9	-0.6	-0.1
-7.1	-1.9	0.2	1.5	0.9	-0.1	0.0	0.3
-0.6	-0.8	1.5	1.6	-0.1	-0.7	0.6	1.3
1.8	-0.2	1.6	-0.3	-0.8	1.5	1.0	-1.0
-1.3	-0.4	-0.3	-1.5	-0.5	1.7	1.1	-0.8
-2.6	1.6	-3.8	-1.8	1.9	1.2	-0.6	-0.4

Figura 2.14: Ejemplo de una tabla de cuantificación de luminancia (izquierda) y una matriz 8x8 de la DCT cuantificada (obtenida en http://web.ece.ucdavis.edu/cerl/ReliableJPEG/Cung/jpeg.html)

En el próximo capítulo se analizan las diferentes alteraciones que se pueden llevar a cabo sobre los coeficientes para preservar la información de los bloques de la R.O.I.

2.5 Conclusiones

En este capítulo se han presentado las técnicas que en la actualidad están siendo implementadas o estudiadas con el objetivo de preservar la privacidad de las personas en secuencias de vídeo seguridad. Los resultados visuales de todas ellas garantizan que la región cuya información se desea ocultar resulta irreconocible una vez se le aplica la técnica. Sin embargo, como ya se mencionó en la introducción, hay una serie de exigencias que todo sistema de seguridad debe cumplir para garantizar la privacidad, esto es, reversibilidad, trabajar en tiempo real y máxima seguridad.

Descartando así todas las técnicas irreversibles cuyas aplicaciones quedan fuera del marco de este Proyecto, se elige continuar este trabajo con la técnica de *scrambling* por las siguientes razones:

I. Es una técnica reversible por lo que puede aplicarse en el ámbito de la vídeo vigilancia.

II. Su coste computacional no es demasiado elevado como en el caso de la técnica *Video Inpainting*. Esto asegura poder trabajar en mayor o menor medida en tiempo real.

III. Se le puede exigir máxima seguridad mediante la encriptación de la secuencia aleatoria que determina los cambios en los coeficientes.

En los próximos capítulos se estudia cómo desarrollar un algoritmo de *scrambling* que pueda trabajar con secuencias de vídeo y los resultados que determinan su viabilidad en términos de integración en otras aplicaciones de detección de personas.

3 Desarrollo

3.1 Introducción

Una vez decidida la técnica de preservación de privacidad a utilizar en el sistema de vídeo seguridad comienza la etapa de desarrollo que ocupa este capítulo. En una primera sección 3.2 se explica en qué consiste el algoritmo de *scrambling* que preserva la privacidad de forma reversible mediante la encriptación de los rasgos faciales haciéndolos irreconocibles. A continuación, en la sección 3.3 se explica brevemente el código en C++ que trabaja la funcionalidad de la técnica. Por último, se presentan dos alternativas diferentes dentro del programa de cara a la evaluación. La primera consiste en una aproximación del algoritmo al espacio de color RGB (sección 3.4) y la segunda al espacio YCrCb (sección 3.5).

3.2 Descripción general del algoritmo

En [15] y [16] se propone cambiar el signo de los coeficientes de la DCT en función de un PRNG (*Pseudo Random Number Generator*). A partir de un valor inicial o *seed* se genera una secuencia de bits de unos y ceros aleatoria que determina qué coeficientes cambian de signo y cuáles no. Dicho valor inicial o *seed* se cifrará mediante una *key* y se embeberá en el stream para poder deshacer el *scrambling* en el decodificador. Además, dado que el *scrambling* se hace en el dominio transformado y antes de la formación del *bitstream* codificado, cualquier decodificador convencional puede deshacer el *scrambling* del *bitstream* codificado como si fuese uno sin alterar [17].

Por otro lado hay que considerar que los coeficientes DC son siempre positivos por lo que no se puede plantear una alteración aleatoria de cambio de signo como tal. Sin embargo, en [17] se plantea “cambiar el signo” según un determinado umbral que podría ser la mitad del máximo valor que pudiera tomar el coeficiente DC.

Aunque aplicar el scrambling a los coeficientes AC proporciona buenos niveles de seguridad, se puede además cifrar los coeficientes DC alterando el valor cuantificado. En cualquier caso, la eficiencia de compresión no desciende de manera significativa así como la complejidad computacional [17].

Por último, cabe destacar que en este algoritmo se ha optado por aplicar el *scrambling* sólo a la parte superior de la persona (rostro facial en mayor o menor medida). Dado que la evaluación (capítulo 4) consiste en comprobar que las técnicas de detección de persona

siguen funcionando una vez se ha realizado el *scrambling*, se entiende que cuanto menor es la región con scrambling mejores resultados se obtendrán en este sentido.

3.3 Implementación del algoritmo

En esta sección se ha implementado un algoritmo que sigue las fundamentaciones teóricas de la técnica del Scrambling. La implementación de dicho algoritmo se ha llevado a cabo utilizando el lenguaje de programación C++ y la librería OpenCV (http://opencv.org/). Previamente se evaluó el resultado visual de una función del algoritmo de *scrambling* en Matlab para comprobar el grado de privacidad trabajando con bloques DCT 8x8:

Editor - /Users/jaimemateo/Documents/MATLAB/pruebaScrambling/scrambling.m

DCT_Imagen.m | scrambling.m | roi.m

```
1   function [dctScram] = scrambling( bloque )
2   %Aplica un scrambling de signo a los coeficientes AC
3
4 -  PSRG=rand(8,8);
5 -  PSRG=uint8(PSRG);
6
7 -  [filas,columnas] = size(bloque);
8 -  bloquescr=zeros(filas,columnas);
9
10 - for i=1:columnas
11 -     for j=1:filas
12 -         if i == 1 && j==1
13 -             bloquescr(i,j)=bloque(i,j);
14 -         elseif PSRG(i,j) == 1
15 -             bloquescr(i,j)=-bloque(i,j);
16 -         elseif PSRG(i,j)~=1
17 -             bloquescr(i,j)=bloque(i,j);
18
19 -         end
20 -     end
21 - end
22
23 - dctScram=bloquescr;
24
```

Figura 3.1: Función de Matlab que lleva a cabo el scrambling de un bloque de coeficientes DCT 8x8

(a)

(b)

Figura 3.2: Imagen original (a) y resultado visual de aplicar la función de *scrambling* en Matlab (b).

Una vez comprobados los resultados visuales en Matlab, se lleva a cabo la implementación del algortimo completo en OpenCV. Open Computer Vision es una librería distribuida bajo la licencia BSC y que es de acceso libre para usos académicos y comerciales. Trabajo con C++, C, Phyton y Java y soporta Windows, Linux, Mac OS, iOS y Android. OpenCV fue diseñada para aplicaciones en tiempo real por lo que la eficiencia cumputacional es su principal característica [25]. Dicha librería es escogida para la implementación del algoritmo por su amplio uso dentro del laboratorio.

En primer lugar, el algoritmo obtiene los frames de una determinada secuencia de vídeo para aplicarles por separado el detector de personas a evaluar (HOG y Latent SVM en este proyecto). A continuación, y para cada uno de los frames, se ejecutan tres funciones: *predetection*, *scrambling* y *postdetection*. La primera de ellas sirve para evaluar el detector

sin aplicar el *scrambling*, guardando los valores de las R.O.I.s obtenidas (coordenadas y puntuación) en un primer análisis de la secuencia.

A continuación, la función *scrambling* repite el proceso anterior añadiéndole el módulo de privacidad. Dicho módulo está formado por varias funciones que utilizan la transformada DCT y su inversa para aplicar el *scrambling*. La R.O.I., cuya privacidad queremos preservar, es en este caso el rasgo facial e identificativo de la persona por lo que se guarda en una variable la parte superior de la caja detectada. En esta primera aproximación, dicha R.O.I. se descompone en las componentes RGB o YCrCb para la aplicación de la técnica de scrambling. La función DCT recibe por separado dichas componentes y le aplica la transformación a bloques 8x8. Para cada uno de esos bloques se aplica la técnica de *scrambling* explicada en el punto 3.1.1.

Una vez los coeficientes de la DCT han sido modificados, se procede a aplicar la IDCT para obtener el resultado final del frame con el scrambling. Para ello, se realiza una copia de los nuevos valores de cada componente (RGB o YCrCb) de la R.O.I. con el scrambling en el frame original. Finalmente se unen las tres componentes para visualizar los resultados finales. Se puede consultar el cuerpo de las funciones *process* y *scrambling* (caso HOG) del algoritmo en el Anexo A.

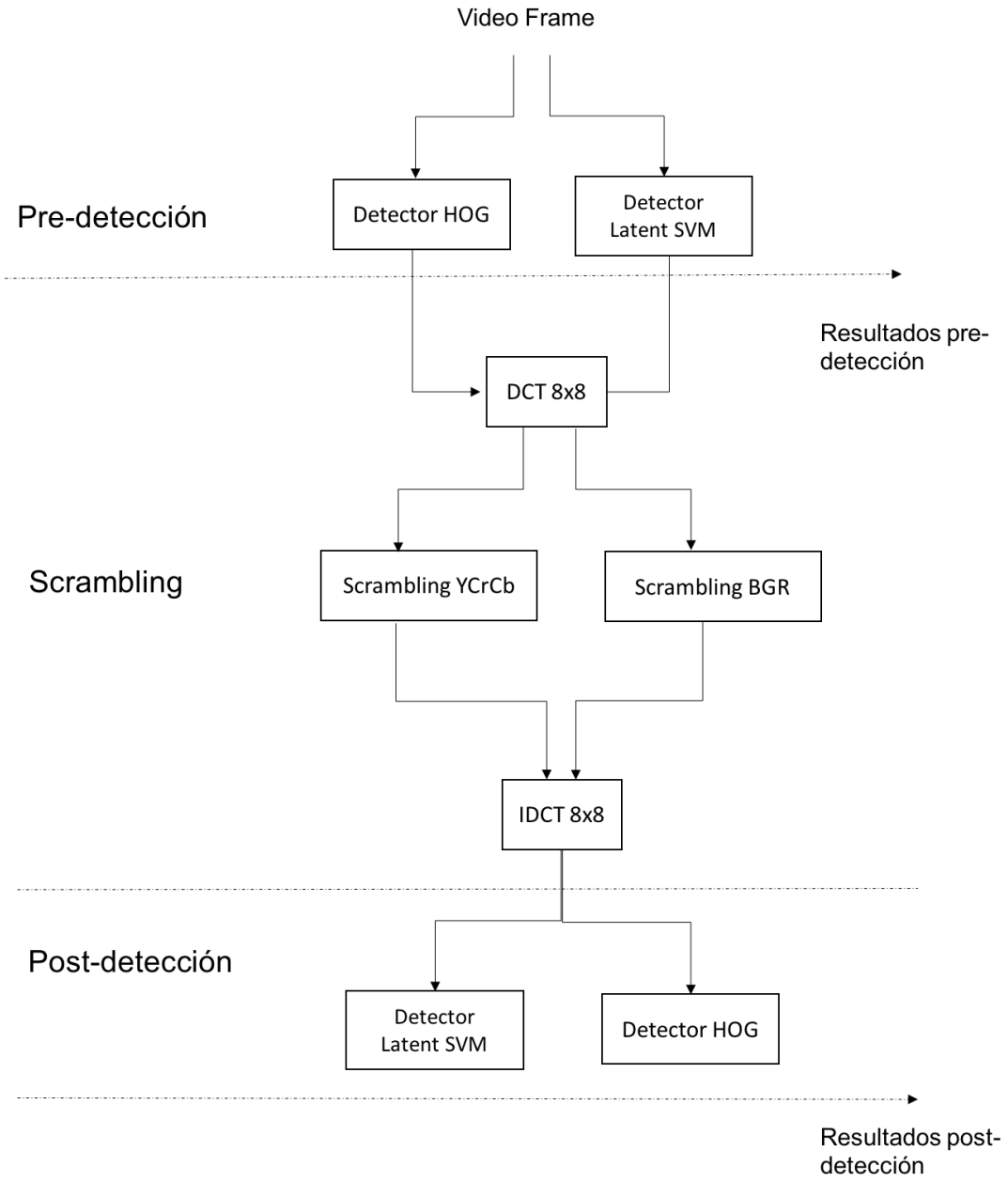

Figura 3.3 Diagrama de bloques del desarrollo del algoritmo

3.4 Primera aproximación: HOG detector

El Histograma de Gradientes Orientados (HOG) es un descriptor de características utilizado en visión artificial y en tratamiento de imágenes para la detección de personas [20]. El método consiste en evaluar los histogramas de los gradientes orientados de la imagen teniendo en cuenta que los objetos pueden ser caracterizados por la distribución de las intensidades de dichos gradientes [21]. Para ello, se divide la imagen en pequeñas regiones o "*cells*", acumulando cada una de ellas un histograma de los gradientes sobre los píxeles de

la imagen, esto es la magnitud y dirección de variación de color o intensidad. El histograma otorga información característica de las diferentes formas que aparecen en la imagen.

En una primera aproximación, se utiliza dicho detector para evaluar el algoritmo de scrambling. La librería OpenCV contiene estructuras que permiten definir un descriptor HOG y utilizarlo para detectar personas en un determinado frame. La función *detectMultiScale* devuelve las coordenadas (x, y, ancho y alto) del contorno detectado así como una puntuación o "score". Dicha puntuación se utilizará para la posterior evaluación.

3.4.1 Scrambling componentes RGB

Dentro de los espacios de color (modelos de color matemáticos), el RGB es uno de los más utilizado. La plataforma OpenCV contiene funciones que permiten descomponer una imagen a color en las componentes RGB. Una vez hecho esto, se aplica el *scrambling* a cada color por separado. Los resultados visuales muestran un alto grado de preservación de privacidad, siendo prácticamente imposible reconocer los rasgos faciales del individuo.

Figura 3.4: Scrambling en las tres componentes RGB

Dado que el scrambling se practica a cada canal por separado, también se estudia el comportamiento del algoritmo aplicando la técnica a dos canales o a sólo un canal. Los resultados visuales (Figura 3.6) muestran cómo la aplicación a un solo canal no es suficiente de cara a mantener un alto grado de privacidad:

Figura 3.5: Scrambling en las componentes GR

Figura 3.6: Scrambling en la componente G

3.4.2 Scrambling componentes YCrCb

Otra posibilidad a la hora de aplicar el scrambling es hacerlo a las componentes YCrCb. Este espacio de color separa por un lado la luminancia (Y) de la crominancia (diferencia de azul Cb, diferencia de rojo Cr). En este apartado se estudia el comportamiento del algoritmo cuando los tres canales son alterados por el scrambling o cuando sólo ocurre con la luminancia (Y). Los resultados visuales muestran una ligera diferencia entre los dos casos por lo que se considerarán ambos para la evaluación de resultados.

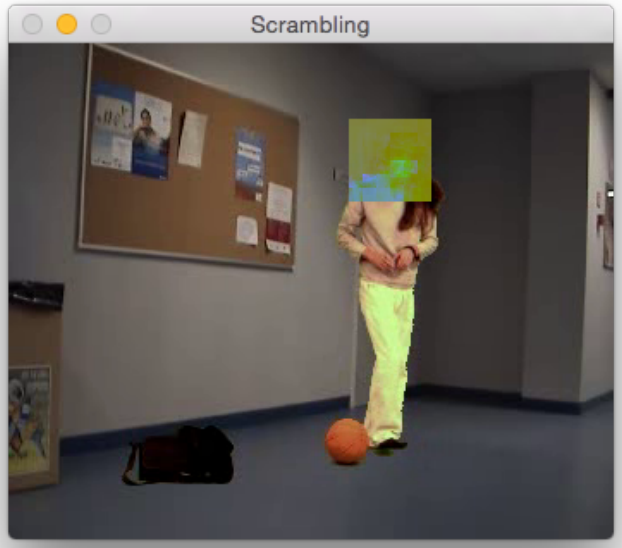

Figura 3.7: Scrambling en las componentes YCrCb

Figura 3.8: Scrambling en la componentes Y

3.5 Segunda aproximación: Latent SVM detector

El detector Latent SVM utiliza un modelo de persona basado en partes [22]. Se fundamenta en la idea del detector HOG original y la utiliza para modelar cada una de las partes deformables del cuerpo en las que se divide la detección. El modelo en el que se basa este detector lo define un filtro raíz similar al filtro Dalal-Triggs [21] más un conjunto de filtros de partes y sus modelos de deformación asociados.

La puntuación del método en una posición determinada de la imagen se calcula como la puntuación del filtro raíz en esa ubicación más la suma de puntuaciones de cada parte menos el coste de deformación (desviación respecto a la posición relativa a la raíz). Por su lado, los filtros de partes trabajan al doble de resolución espacial con respecto al filtro raíz por lo que consiguen características más precisas en múltiples escalas.

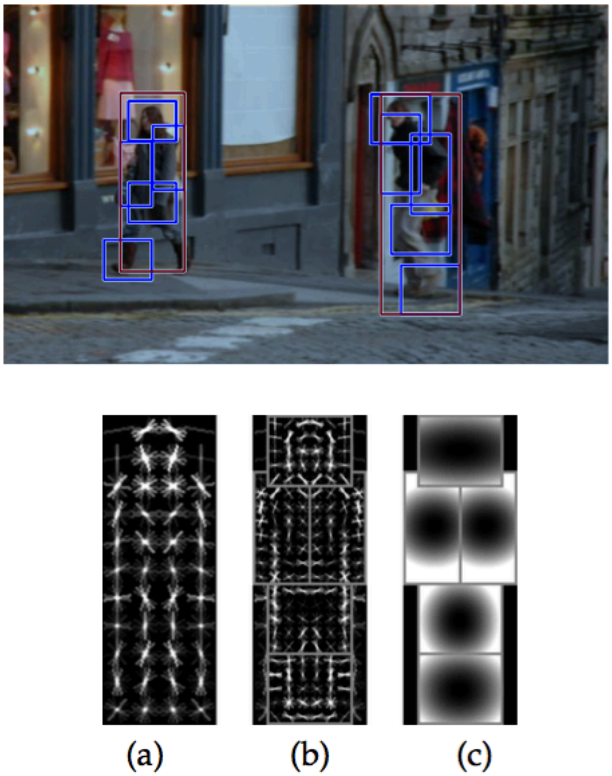

Figura 3.9: Resultados visuales obtenidos aplicando el modelo de persona. En (a) se muestra el filtro raíz, en (b) los filtros por partes y en (c) un modelo espacial para la localización de cada parte relativa a la raíz [21].

Al igual que en el caso del detector HOG, con Latent SVM se comprobará el funcionamiento del algoritmo con dos modelos de color, RGB e YCrCb. Los resultados visuales de aplicar el scrambling son idénticos (la técnica no cambia) a los obtenidos con HOG detector pero la evaluación analizada en el siguiente capítulo muestra claras diferencias entre un detector y otro.

3.6 Conclusiones

En esta sección se ha explicado el desarrollo de la técnica de preservación de privacidad elegida tras el estudio del Estado del Arte de la sección 2. La función de *scrambling* que se evalúa en la siguiente sección se desarrolla primero en Matlab para una rápida visualización de su comportamiento con una imagen de prueba (Figura 3.2). A continuación se programa el algoritmo completo utilizando la librería OpenCV. El diagrama de la Figura 3.3 recoge las diferentes funciones y etapas del algoritmo. En el Anexo 1 se puede consultar el algoritmo completo. Por último, en las secciones 3.4 y 3.5 se exponen las diferentes modalidades de *scrambling* a evaluar en la siguiente sección.

4 Evaluación

4.1 Introducción

En este capítulo se evalúa la respuesta del algoritmo de *scrambling* utilizando dos detectores (HOG y Latent). En la sección 4.2 se explica en qué consiste la evaluación y cómo se lleva a cabo. El dataset utilizado para la evaluación se presenta en la sección 4.3. Una vez obtenidos los resultados, se analizarán en primer lugar aquellos que arrojan el detector HOG (sección 4.4) y a continuación los del detector Latent (sección 4.5). Finalmente se hará una comparación entre ambos resultados y se analizarán las diferencias llegando a una conclusión.

4.2 Metodología

La evaluación del algoritmo de scrambling consiste en ver hasta qué punto las aplicaciones existentes de detección de personas, objetos, eventos, etc. siguen funcionando una vez hemos modificado el frame con la técnica de scrambling. En el marco de este PFC, se utiliza la detección de personas (detectores HOG y Latent) como medio para evaluar el algoritmo. Como se muestra en la Figura 3.2, después de ejecutar el algoritmo obtenemos dos ficheros de datos: por un lado las puntuaciones o *"scores"* que deja el detector de personas (HOG o Latent) antes de aplicar el *scrambling* (resultados pre-detección) y las puntuaciones después de aplicar el scrambling y volver a ejecutar el detector de personas (resultados post-detección).

Estos ficheros de datos se obtienen para cuatro secuencias de vídeo diferentes (Sequence1, Sequence2, Sequence3, Sequence4) que se introducen en el apartado siguiente. Los ficheros deben recibir el siguiente nombre para su posterior análisis en Matlab:

SequenceXpreABC.txt (resultados pre-detección)

SequenceXpreABC.txt (resultados post-detección)

Donde X es el número de secuencia y ABC los canales del modelo de color en los que se aplica el scrambling (BGR, GR, Y, YCrCb).

Los datos dentro del fichero deben seguir además la siguiente estructura para poder interpretarse correctamente con el script de Matlab:

./Videos/SequenceXpre/inYYYYY.jpg; (A,B,C,D):E;

donde X es el número de secuencia, YYYYY es el número del frame, A, B, C y D son las coordenadas de la ROI y E la puntuación o *"score"*.

```
./Videos/Sequence1pre/in00000.jpg;
./Videos/Sequence1pre/in00001.jpg;
./Videos/Sequence1pre/in00002.jpg;
./Videos/Sequence1pre/in00003.jpg;
./Videos/Sequence1pre/in00004.jpg; (36, 126, 114, 282):2.22507e-308;
./Videos/Sequence1pre/in00005.jpg;
./Videos/Sequence1pre/in00006.jpg;
./Videos/Sequence1pre/in00007.jpg;
./Videos/Sequence1pre/in00008.jpg;
./Videos/Sequence1pre/in00009.jpg; (33, 125, 112, 283):2.22507e-308;
./Videos/Sequence1pre/in00010.jpg;
./Videos/Sequence1pre/in00011.jpg;
./Videos/Sequence1pre/in00012.jpg;
./Videos/Sequence1pre/in00013.jpg; (33, 125, 112, 283):2.22507e-308;
./Videos/Sequence1pre/in00014.jpg;
./Videos/Sequence1pre/in00015.jpg; (316, -20, 388, 123):2.22507e-308;
./Videos/Sequence1pre/in00016.jpg; (316, -20, 388, 123):2.22507e-308;
./Videos/Sequence1pre/in00017.jpg; (315, -22, 389, 126):2.22507e-308;
./Videos/Sequence1pre/in00018.jpg;
./Videos/Sequence1pre/in00019.jpg;
./Videos/Sequence1pre/in00020.jpg;
./Videos/Sequence1pre/in00021.jpg;
./Videos/Sequence1pre/in00022.jpg;
./Videos/Sequence1pre/in00023.jpg;
./Videos/Sequence1pre/in00024.jpg;
./Videos/Sequence1pre/in00025.jpg;
./Videos/Sequence1pre/in00026.jpg;
./Videos/Sequence1pre/in00027.jpg;
./Videos/Sequence1pre/in00028.jpg;
./Videos/Sequence1pre/in00029.jpg;
./Videos/Sequence1pre/in00030.jpg; (311, 63, 389, 218):0.0792913, (313, -14, 389, 138):0.0792913;
./Videos/Sequence1pre/in00031.jpg; (307, 58, 392, 227):2.22507e-308;
./Videos/Sequence1pre/in00032.jpg; (307, -7, 390, 159):2.22507e-308, (302, 49, 396, 237):2.22507e-308;
./Videos/Sequence1pre/in00033.jpg; (307, -7, 390, 159):2.22507e-308, (302, 46, 396, 234):2.22507e-308;
./Videos/Sequence1pre/in00034.jpg;
./Videos/Sequence1pre/in00035.jpg; (304, 73, 393, 250):0.033786;
./Videos/Sequence1pre/in00036.jpg; (299, 71, 394, 262):1.0728;
./Videos/Sequence1pre/in00037.jpg; (315, -13, 387, 131):2.22507e-308, (284, 41, 399, 270):0.3799;
./Videos/Sequence1pre/in00038.jpg; (314, -16, 389, 135):2.22507e-308, (269, 45, 380, 267):2.22507e-308;
./Videos/Sequence1pre/in00039.jpg; (312, -13, 389, 140):0.00900605, (298, 120, 402, 327):2.22507e-308, (267, 47, 376, 265):0.
00900605;
```

Figura 4.1: Ejemplo de fichero de texto obtenido en el algoritmo con los resultados. Los números entre paréntesis son las coordenadas de la ROI y el último número la puntuación.

Una vez obtenidos todos los ficheros de datos, estos se analizan en Matlab junto con el Ground Truth. Este fichero contiene las coordenadas de todas las detecciones correctas (personas en este caso) de cada uno de los *frames* de la secuencia. De esta forma se calcula la puntuación de acierto antes del scrambling y después del mismo.

4.3 Métricas

Para analizar los resultados obtenidos en la evaluación es necesario contar con métricas y fórmulas que determinen si dichos resultados son o no los esperados. En esta sección se explica el fundamento de estas métricas.

Las curvas ROC enfrentan la fracción de verdaderos positivos entre todos los positivos (*True Positive Rate* -TPR-, *Sensitivity* o *Recall*) contra los falsos positivos de todos los negativos

(*False Positive Rate* -TPR- o *1-Specificity*) [23]. Se utiliza para esta evaluación la curva 1-*Presicion/Recall*, cuyos puntos se calculan mediante las fórmulas de *Recall* y *Precision*:

$$Precision = \frac{\#TruePositivePeopleDetections}{\#TruePositivePeopleDetections + \#FalsePositivePeopleDetections}$$

$$Recall = \frac{\#TruePositivePeopleDetections}{\#TruePositivePeopleDetections + \#FalseNegativePeopleDetections}$$

Por último se calcula el área bajo la curva (AUC), que determina la efectividad del algoritmo siendo uno el mayor valor posible. La evaluación se realiza para cada secuencia completa en primer lugar y por último para el conjunto de todo el dataset.

4.4 Dataset

Si bien es cierto que el dataset es muy importante a la hora de llevar a cabo una evaluación, en el caso de este Proyecto Fin de Carrera no es algo tan crucial dado que lo que se quiere comprobar no es tanto lo bueno que es el algoritmo si no si siguen funcionando las técnicas de detección una vez se ha aplicado el scrambling. La elección del dataset se ha basado por lo tanto en considerar uno con un número de ROIs reducido y que generase altas puntuaciones con los detectores HOG y Latent.

En un primer momento se optó por evaluar el dataset PDbm por ser más completo y acercarse más a las secuencias de vídeo seguridad. Sin embargo, los resultados obtenidos con el detector HOG de OpenCV no eran demasiado buenos incluso antes de aplicar el scrambling, por lo que se optó por otro dataset que permitiese partir de valores más óptimos:

ALGORITMO HOG	**N=1000**	**AUC pre-mean**	**AUC post-mean**	**% Caída**
Scrambling 8x8 BGR		0,2248	0,2283	-1,53

Tabla 4.1: Resultados para la secuencia PETS2006 con detector HOG

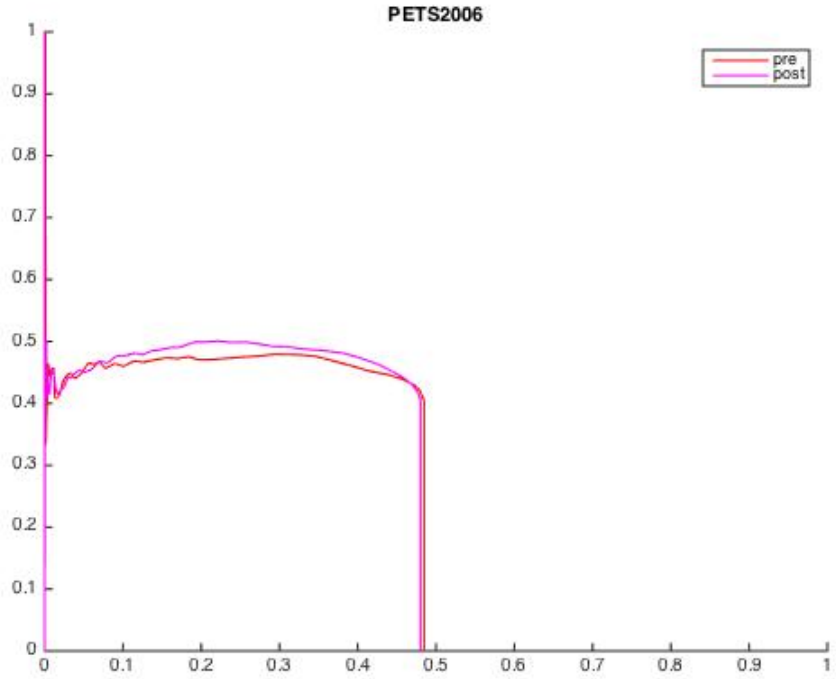

Figura 4.2: Curva ROC de la secuencia PETS2006 del dataset PDbm. Se puede observar cómo cae el Recall por el comportamiento del detector HOG.

En la siguiente tabla se recogen algunas características del dataset (número de frames y número de ROIs en la secuencia) elegido. También se pueden ver algunos *frames* de ejemplo más abajo:

Nombre de la secuencia	**Número de frames**	**Número de ROIs**
Sequence1	193	1
Sequence2	349	2
Sequence3	897	2
Sequence4	193	1

Tabla 4.2: Características del dataset

Figura 4.3: Ejemplo de un frame de *Sequence1*

Figura 4.4: Ejemplo de un frame de *Sequence2*

Figura 4.5: Ejemplo de frame de *Sequence3*

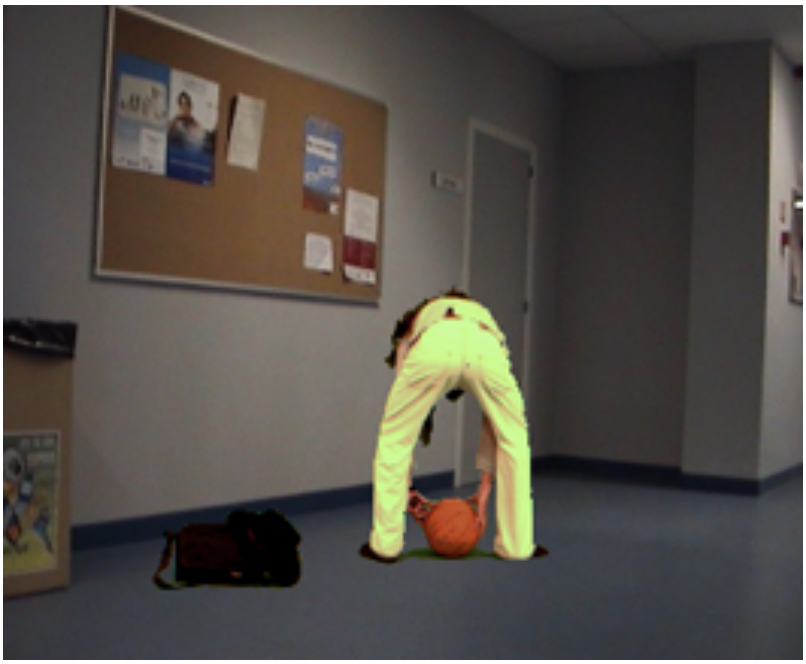

Figura 4.6: Ejemplo de frame de *Sequence4*

4.5 Resultados HOG detector

En esta sección se recogen las tablas con los resultados de las cuatro secuencias por separado y por último la media de la evaluación completa. Para el caso de las secuencias por separado se han utilizado 1000 iteraciones y para la media 100 iteraciones.

Cada fila de la tabla corresponde a uno de los tipos de *scrambling* presentados en la sección 3.4 y en las columnas se recogen los valores antes de aplicar el *scrambling* (AUC pre-mean) y tras aplicarlo (AUC post-mean). La última columna muestra la caída en la puntuación, determinando cuánto afecta el *scrambling* en el detector HOG.

ALGORITMO HOG	**N=1000**	**AUC pre-mean**	**AUC post-mean**	**% Caída**
Scrambling 8x8 BGR	0,8663		0,8388	3,28
Scrambling 8x8 GR	0,8663		0,8503	1,88
Scrambling 8x8 YCrCb	0,8663		0,8458	2,42
Scrambling 8x8 Y	0,8663		0,8447	2,56

Tabla 4.3: Resultados obtenidos para el Sequence1 del dataset con detector HOG

ALGORITMO HOG	**N=1000**	**AUC pre-mean**	**AUC post-mean**	**% Caída**
Scrambling 8x8 BGR	0,7639		0,7304	4,59
Scrambling 8x8 GR	0,7639		0,7775	-1,75
Scrambling 8x8 YCrCb	0,7639		0,7227	5,70
Scrambling 8x8 Y	0,7639		0,7505	1,79

Tabla 4.4: Resultados obtenidos para el Sequence2 del dataset con detector HOG

ALGORITMO HOG	N=1000	AUC pre-mean	AUC post-mean	% Caída
Scrambling 8x8 BGR	0,6106		0,5984	2,04
Scrambling 8x8 GR	0,6106		0,6109	-0,05 *
Scrambling 8x8 YCrCb	0,6106		0,6108	-0,03 *
Scrambling 8x8 Y	0,6106		0,6141	-0,57 *

Tabla 4.5: Resultados obtenidos para el Sequence3 del dataset con detector HOG. (*) Los valores negativos indican una mejora en el éxito de las detecciones tras el *scrambling*.

ALGORITMO HOG	N=1000	AUC pre-mean	AUC post-mean	% Caída
Scrambling 8x8 BGR	0,8555		0,8477	0,92
Scrambling 8x8 GR	0,8555		0,8348	2,48
Scrambling 8x8 YCrCb	0,8555		0,872	-1,89 *
Scrambling 8x8 Y	0,8555		0,8573	-0,21 *

Tabla 4.6: Resultados obtenidos para el Sequence4 del dataset con detector HOG. (*) Los valores negativos indican una mejora en el éxito de las detecciones tras el *scrambling*.

ALGORITMO HOG	N=100	AUC pre-mean	AUC post-mean	% Caída
Scrambling 8x8 BGR	0,7754		0,7537	2,88
Scrambling 8x8 GR	0,7754		0,7685	0,90
Scrambling 8x8 YCrCb	0,7754		0,7615	1,83
Scrambling 8x8 Y	0,7754		0,76	2,03

Tabla 4.7: Resultados obtenidos para la media de todas las secuencias del dataset con detector HOG

De estos resultados se entiende que las caídas en el funcionamiento del detector HOG una vez se introduce el módulo de privacidad con la técnica *scrambling* son despreciables en muchos casos. La mayor caída se produce cuando el *scrambling* se aplica a los tres canales del espacio de color BGR, siendo la caída media de todas las secuencias evaluadas para este caso en concreto de tan sólo el 2,88%.

Los resultados también arrojan el mejor escenario en el que utilizar el *scrambling*. Cuando éste se aplica tan sólo a dos canales (GR en este caso), la caída media de todas las secuencias tras la evaluación es de un 0,9%, casi un punto por debajo del siguiente mejor caso (el de *scrambling* YCrCb).

En definitiva, se puede concluir que el detector HOG se comporta bien ante la técnica de *scrambling* en una sencilla evaluación, pudiendo detectar a las personas del dataset tras incluir el módulo de privacidad y con un porcentaje de caída en las detecciones bastante bajo.

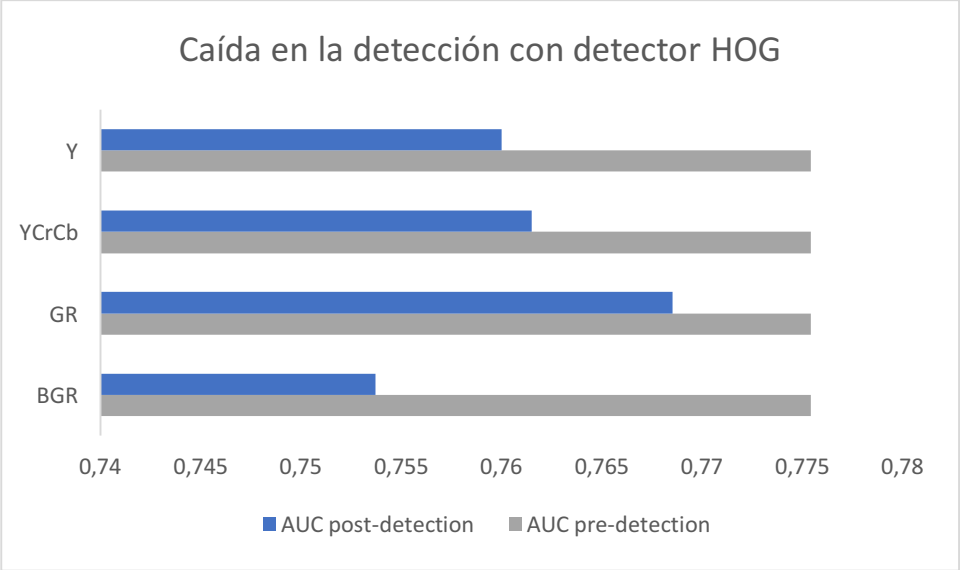

Figura 4.7: Gráfico de resultados en la caída de la detección tras aplicar scrambling con un detector Latent

4.6 Resultados Latent SVM detector

Al igual que en la sección anterior, a continuación se recogen las tablas con los resultados de las cuatro secuencias a evaluar en esta ocasión con el detector Latent SVM. Con en el caso del detector HOG, se han utilizado 1000 iteraciones para las secuencias individuales y para la media 100 iteraciones.

Se vuelven a evaluar los tipos de *scrambling* presentados en la sección 3.4 recogidos en las filas de la tabla. Las columnas recogen los valores antes de aplicar el *scrambling* (AUC pre-mean) y tras aplicarlo (AUC post-mean). La última columna muestra la caída en la puntuación, determinando cuánto mejora o empeora la detección con Latent tras introducir el *scrambling*.

ALGORITMO Latent	N=1000	AUC pre-mean	AUC post-mean	% Caída
Scrambling 8x8 BGR	0,827		0,2549	224,44
Scrambling 8x8 GR	0,827		0,4551	81,72
Scrambling 8x8 YCrCb	0,827		0,4614	79,24
Scrambling 8x8 Y	0,827		0,1743	374,47

Tabla 4.8: Resultados obtenidos para Sequence1 del dataset con detector Latent

ALGORITMO Latent	N=1000	AUC pre-mean	AUC post-mean	% Caída
Scrambling 8x8 BGR	0,876		0,5232	67,43
Scrambling 8x8 GR	0,876		0,6213	40,99
Scrambling 8x8 YCrCb	0,876		0,4043	116,67
Scrambling 8x8 Y	0,876		0,4289	104,24

Tabla 4.9: Resultados obtenidos para Sequence2 del dataset con detector Latent

ALGORITMO Latent	N=1000	AUC pre-mean	AUC post-mean	% Caída
Scrambling 8x8 BGR	0,4463		0,2173	105,38
Scrambling 8x8 GR	0,4463		0,2276	96,09
Scrambling 8x8 YCrCb	0,4463		0,1593	180,16
Scrambling 8x8 Y	0,4463		0,1665	168,05

Tabla 4.10: Resultados obtenidos para Sequence3 del dataset con detector Latent

ALGORITMO Latent	N=1000	AUC pre-mean	AUC post-mean	% Caída
Scrambling 8x8 BGR	0,8085		0,1074	652,79
Scrambling 8x8 GR	0,8085		0,5329	51,72
Scrambling 8x8 YCrCb	0,8085		0,2617	208,94
Scrambling 8x8 Y	0,8085		0,4604	75,61

Tabla 4.11: Resultados obtenidos para Sequence4 del dataset con detector Latent

ALGORITMO Latent	N=100	AUC pre-mean	AUC post-mean	% Caída
Scrambling 8x8 BGR	0,7393		0,2755	168,35
Scrambling 8x8 GR	0,7393		0,4719	56,66
Scrambling 8x8 YCrCb	0,7393		0,3217	129,81
Scrambling 8x8 Y	0,7393		0,3079	140,11

Tabla 4.12: Resultados obtenidos para la media de todas las secuencias del dataset con detector Latent

Los resultados que se pueden observar en las tablas anteriores muestran unas caídas muy acusadas cuando el detector utilizado tras la utilización de *scrambling* es el Latent SVM. Si se atiende a la media de todas las secuencias del dataset (Tabla 4.12), se confirma que el *scrambling* GR es la modalidad que mejores resultados ofrece, aunque en este caso la caída es del 56,66%.

Para el resto de escenarios, estas caídas llegan a alcanzar el 168,35% en el caso del espacio de color BGR, que se confirma como la peor modalidad de *scrambling* posible (también ocurre en la sección anterior con el detector HOG).

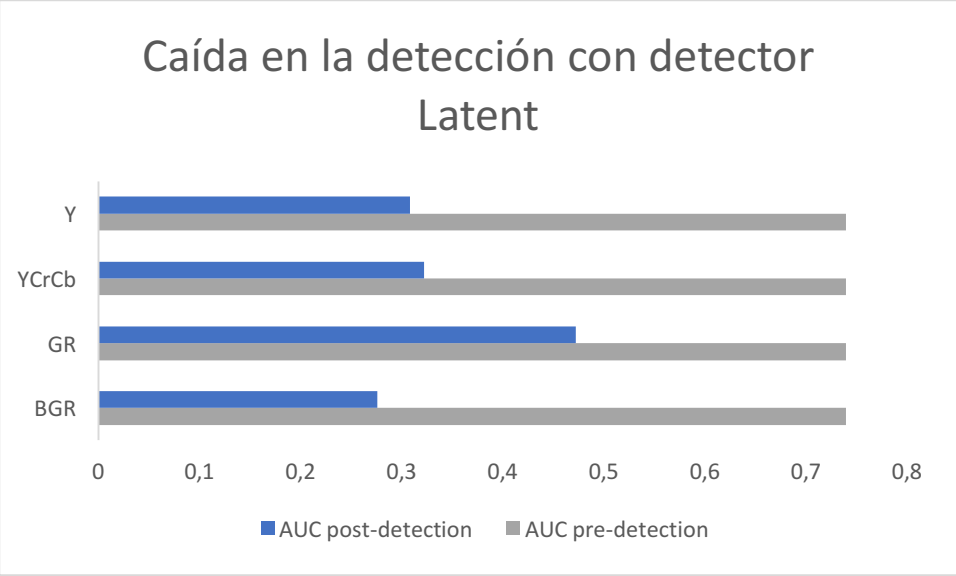

Figura 4.8: Resultados en la caída de la detección tras aplicar scrambling con un detector Latent

4.7 Conclusiones

En esta sección se ha desarrollado el sistema de evaluación que permite comprobar hasta qué punto la técnica de *scrambling* resulta de interés a la hora de aplicarla en un sistema de preservación de privacidad.

En primer lugar se ha introducido la metodología de esta evaluación (sección 4.2), que consiste en analizar si las técnicas de detección de personas siguen funcionando después de utilizar el *scrambling*. A continuación se han detallado las métricas y el dataset utilizados en la evaluación (sección 4.3 y 4.4 respectivamente). Por último se han presentado los resultados que arrojan los dos detectores utilizados para la evaluación: HOG detector (sección 4.5) y Latent SVM (sección 4.6).

Teniendo en cuenta el funcionamiento de los detectores HOG y Latent, los resultados anteriores tienen una lectura coherente. Mientras que el detector HOG utiliza un modelo de detección basado en los gradientes y las formas dentro de un determinado *frame,* Latent SVM utiliza por su parte un modelo de persona basado en partes. Al aplicar el *scrambling* únicamente a la parte superior de la ROI, lo que corresponde a la cara o cabeza de la persona, no impide al detector HOG detectar el resto del cuerpo y confirmar que lo que se está analizando es una persona. Sin embargo, en el caso de Latent SVM, el hecho de aplicar el *scrambling* supone no detectar la parte superior de la ROI y por consiguiente la persona al completo.

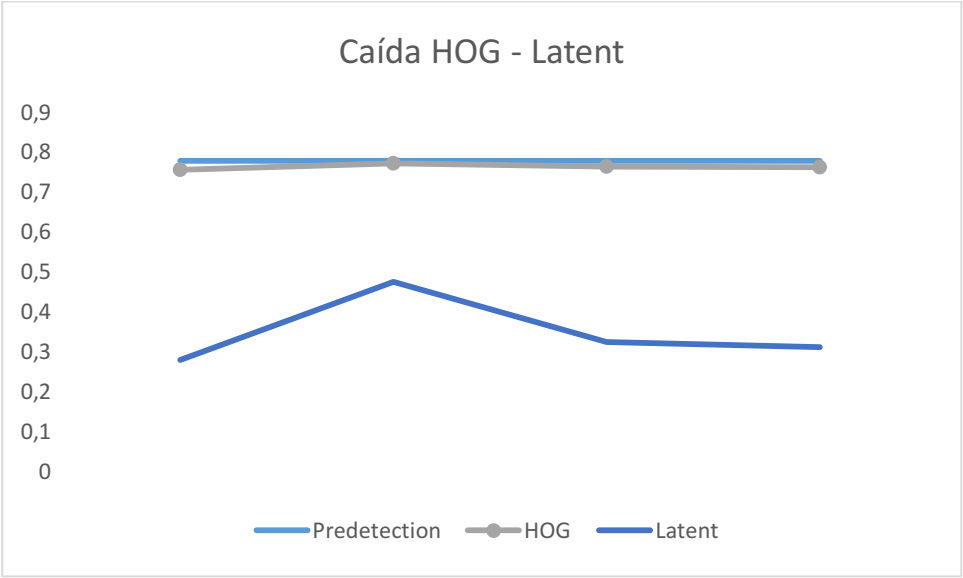

Figura 4.9: Comparativa en la caída de post-detección tras aplicar la técnica de *scrambling* con detector HOG y detector Latent

El objetivo de este PFC es encontrar una técnica de preservación de privacidad que respete el correcto funcionamiento de aplicaciones como la detección de personas, por ello Latent

SVM (y cualquier detector de persona que utiliza un modelo basado en partes) resulta incompatible con el algoritmo de privacidad basado en *scrambling*. Sin embargo, aunque esto queda fuera del estudio de este PFC, sí que podrían funcionar correctamente siempre y cuando en su configuración no se tenga en cuenta esa región que penaliza (parte superior o cabeza) a la hora de aplicar el modelo basado en partes.

Por todo esto la combinación detector HOG y *scrambling* es elegida a la hora de integrarla en una aplicación de preservación de privacidad que se expone en la siguiente sección.

5 Aplicación

5.1 DiVA

La plataforma DiVA (*Distributed Video Analysis*) desarrollada en el VPULab de la Escuela Politécnica Superior de la Universidad Autónoma de Madrid permite diseñar y desarrollar algoritmos de procesamiento para múltiples fuentes de vídeo. Trabaja además con esquemas de procesamiento secuenciales o en paralelo y cuenta con interfaz gráfica para visualizar resultados a tiempo real [23].

Los principales criterios de la aplicación son los siguientes:

- Escalabilidad: tiene que ser flexible para incluir módulos adicionales así como algoritmos de procesamiento o aplicaciones de bases de datos. Además, debe ser soportar fácilmente el uso de nuevas fuentes de vídeo (USB, IP,…) o protocolos de vídeo (MPEG2, h264).

- Eficiencia: debe realizar las operaciones sin un incremento del coste computacional significante.

- Generalidad: todas las funcionalidades tienen que ser desarrolladas usando herramientas y operaciones simples. No se recomienda por lo tanto incluir aplicaciones que dependan de otro software o código.

- Tolerancia a los fallos: tiene que incluir mecanismos para detectar y corregir fallos durante el tiempo de ejecución.

La plataforma DiVA cuenta con las siguientes características:

- Entorno distribuido para investigación, prototipado y demostración de sistemas de análisis visuales con soporte multi-cámara.

- Diseño modular y multi-hilo para procesamiento a nivel de *frame.*

- Modo asíncrono basado en modelo cliente-servidor.

- Flujo de trabajo dinámico (secuencial o en paralelo) y actualizado (escalabilidad on-line) basado en información semántica.

DiVA cuenta con cuatro submódulos que realizan funciones diferentes: Captura de Datos, Procesamiento, Presentación y Bases de Datos.

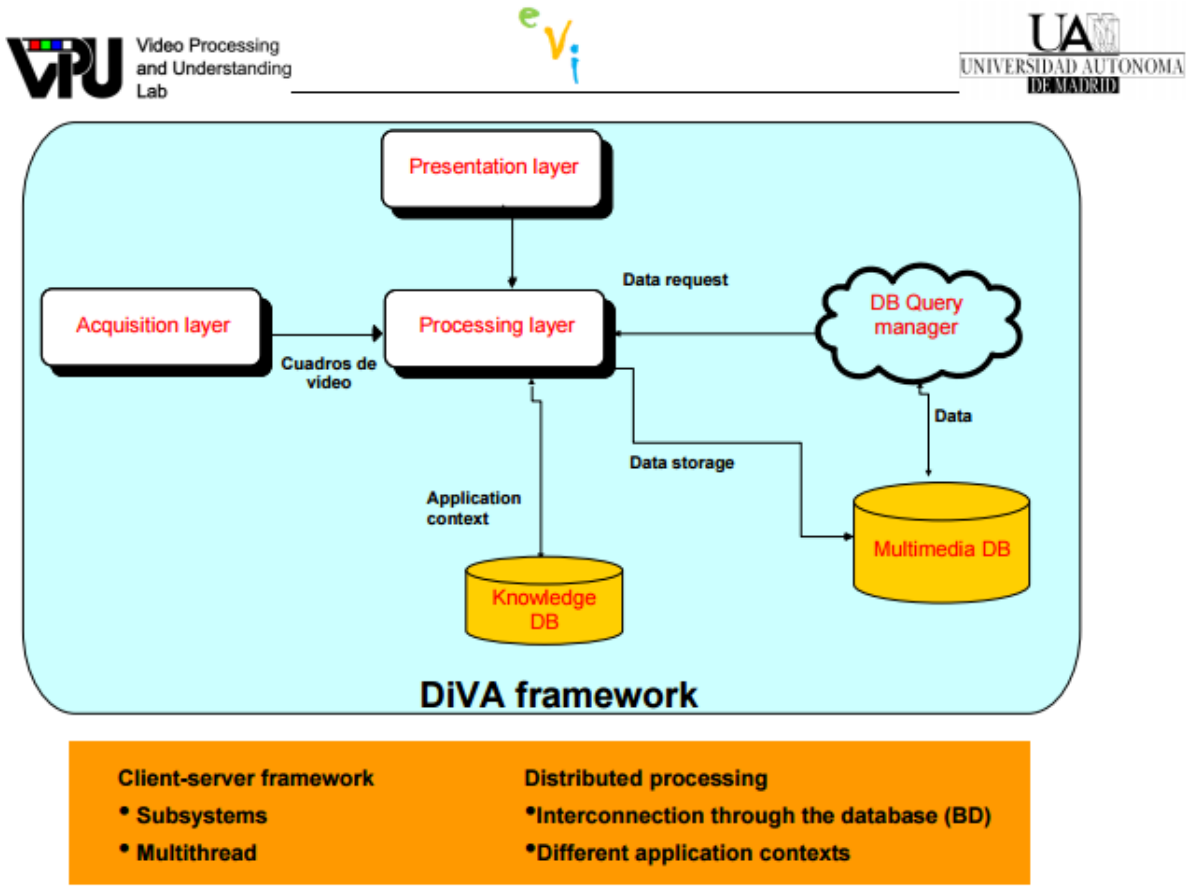

Figura 5.1: Arquitectura de la plataforma DiVA con los diferentes submódulos [23].

5.2 Desarrollo

En esta sección se explica el proceso por el cuál el algoritmo programado en OpenCV en la sección 3 y evaluado en la sección 4 pasa a formar parte de la aplicación DiVA. Para ello, es necesario remodelar el algoritmo ya que éste parte de un funcionamiento puramente secuencial. La conversión del código del algoritmo a una clase que se pueda encapsular dentro del entorno DiVA comienza con la modulación del código. Las funciones *process.cpp*, *predetection.cpp*, *scrambling.cpp* y *postdetection.cpp,* antes dentro del propio *main.cpp,* están ahora en archivos separados para poder utilizarlas dentro de nuestra nueva clase. Para la preparación del algoritmo se siguen las instrucciones de la Guía para la integración de algortimos en DiVA [26].

5.2.1 Preparación del algoritmo

La preparación del algoritmo comienza creando la clase *detector.cpp* en C++ que recoge las siguientes características:

- Un constructor que inicialice los parámetros con unos valores por defecto.
- Un destructor que se encargue de liberar los recursos, en este caso, las ventanas en las que se muestran los resultados.
- Un método público encargado de procesar cada *frame* con la sintaxis:

  *void * process (IplImage *frame);*

- Un método público que muestre los resultados:

  *void *showresults();*

- Métodos set y get para configurar los parámetros:

  *int setParam(int parámetro, int *valor_int)*

  *int getParam(int parámetro, int *valor_int)*

Los parámetros del algoritmo son los siguientes:

- Parámetro 1: determina si se aplica el scrambling (case: POSTSCRAM) o solamente la detección (case: PRESCRAM). Queda definido de la forma:

  #define SCRAM

- Parámetro 2: determina el espacio de color al que se aplica el scrambling. Queda definido de la forma:

  #define COLOR

Se puede consultar en el Anexo B el cuerpo de la clase *detector.cpp*

5.2.2 Encapsulamiento

El encapsulamiento del algoritmo en la plataforma DiVA se lleva a cabo mediante la clase DiVA_detector desarrollada en C++ y que incluye las siguientes funciones:

- Petición de datos a través de un cliente que se conecta al servidor de información.
- Método *refreshDisplay* de presentación de resultados. Consiste básicamente en llamar a la función *showResults* de la clase *detector.cpp*
- Método de procesado de imágenes *processFrame.*En un primer momento convierte la imagen recibida en formato DiVAImage al lenguaje OpenCV. Posteriormente llama a la función *process* de la clase *detector.cpp.*
- Métodos *setParam* y *getParam* que sirven como interfaz para modificar los parámetros.

Se puede consultar la clase DiVA_detector y sus métodos en el Anexo C.

En la última etapa del encapsulamiento se crea un demostrador para visualizar los resultados. Dicho demostrador crea un objeto de la clase derivada de DiVAAlgorithm que implementa el algoritmo pasándole los parámetros solicitados por línea de comandos.

A continuación se llama a la función *init* para que inicie las conexiones con los servidores y comenzar a ejecutar el algoritmo con la función *start*. Para detener el algoritmo basta con llamar al método stop puslando la tecla ESC. Una vez hecho esto, el destructor libera los recursos y termina la ejecución.

5.2.3 Aplicación e interfaz gráfica

5.2.3.1 Qt

Qt es una libreria multiplataforma ampliamente usada para el desarrollo de aplicaciones con interfaz gráfica de usuario o GUI (Graphic User Interface) [27]. Qt es desarrollada como un software libre y de código abierto bajo la comunidad Qt Project donde colaboran empresas como Nokia o Digia.

Qt utiliza el lenguaje de programación C++ de forma nativa. En la librería se establece una jerarquía de clases y herencias enfocada a la programación de interfaces gráficas aunque también contiene módulos para la gestión de tareas tales como bases de datos o aplicaciones multihilo [28].

La librería Qt es compatible con el código de la plataforma DiVA y por eso se elige utilizarla para el desarrollo de la interfaz gráfica. También ayuda que cuente con una extensa comunidad de desarrolladores que ofrecen ayudar para la resolución de problemas durante el trabajo.

5.2.3.2 Implementación

La librería Qt introduce en la herramienta Visual Studio una funcionalidad que permite crear una interfaz gráfica sin necesidad de escribir código. *Qt Designer* implementa en el código los elementos que son añadidos a la interfaz generando el fichero *ui_diva_detector_qt.h*.

Para el desarrollo de esta interfaz se ha partido de un modelo plantilla desarrollado en el VPULab al cuál se le ha añadido el parámetro del *color mode* mediante el cuál se puede ajustar el tipo de *scrambling* que queremos aplicar de los estudiados en la sección 3. En la interfaz de este proyecto los elementos son los siguientes:

- *display*: para la visualización de los resultados de la secuencia de vídeo
- *pushIniciar* - Iniciar: para iniciar la aplicación
- *pushDetener* – Detener: pulsar para terminar la ejecución y cerrar la aplicación
- *PushPrivacidad* – Privacidad: pulsar para activar o desactivar la Privacidad

- *camName*: indica la configuración de cámara que estamos utilizando
- *menuBar*: menú superior que cuenta con las secciones Herramientas y Ayuda.
- *toolsBar*: menú con iconos y acceso directo a las funcionalidades Iniciar, Detener, Cámara y Configuración de Parámetros

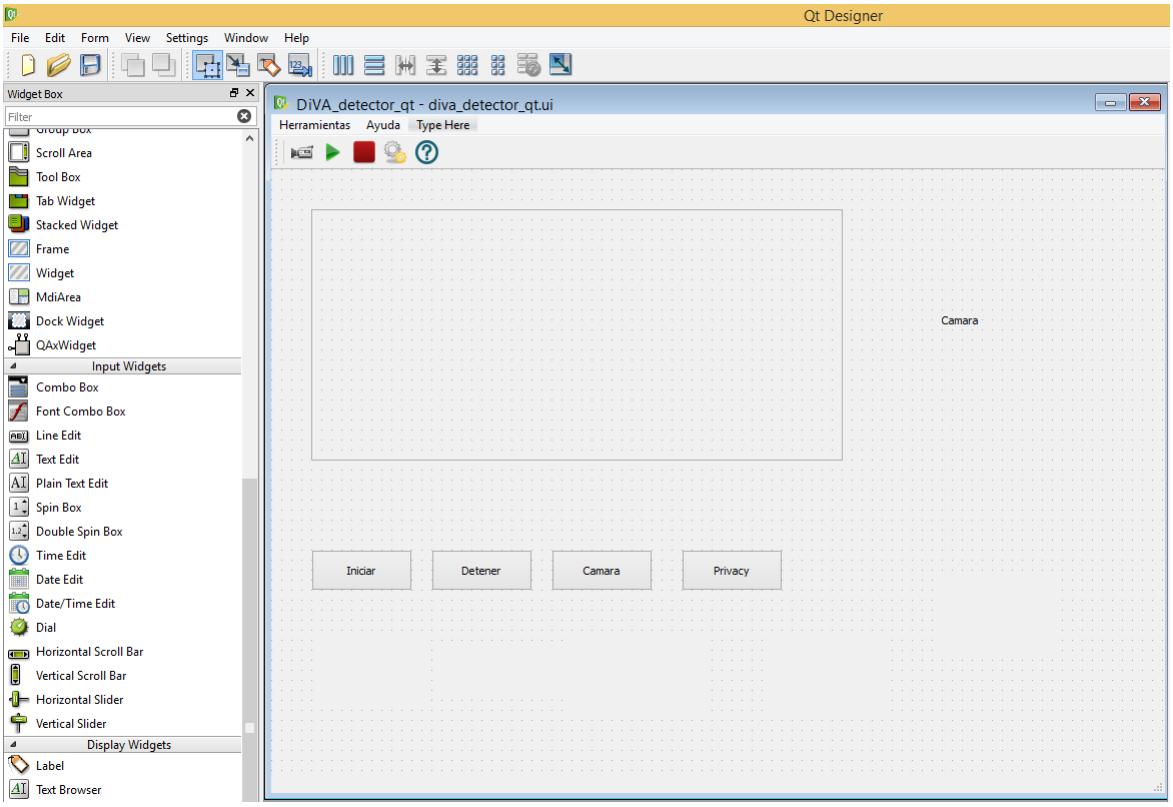

Figura 5.2: Interfaz de Qt Designer

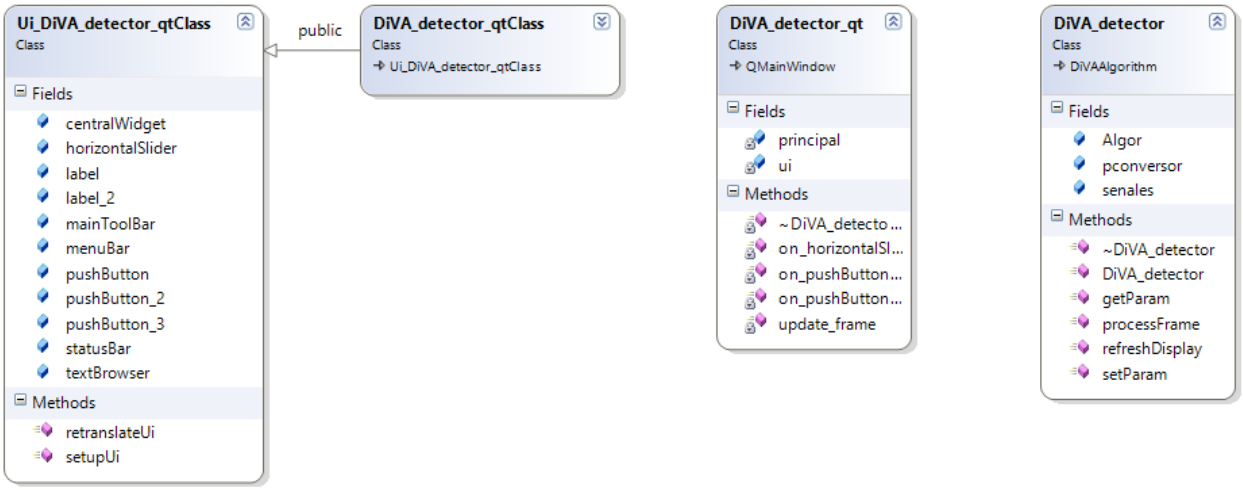

Figura 5.3: Diagrama de clases con sus métodos y atributos después de introducir la herramienta Qt en nuestro algoritmo. De izquierda a derecha: clase User Interface (UI), clase DiVA_detector_qt y clase DiVA_detector

De cara a visualizar los resultados correctamente en la interfaz, es necesario realizar una serie de cambios en el código. Dichos cambios corresponden a las conexiones entre *signals* y *slots*. Por un lado es necesario modificar el método *refreshDisplay* de *DiVA_detector* para que emita la señal de visualizar el vídeo. Esta señal está conectada con el slot de actualizar el frame (*update_frame*) mediante la siguiente instrucción del *slot* `void on_pushButton_clicked();`:

```
connect(principal->senales,
SIGNAL(refresh_display(QPixmap)),this,SLOT(update_frame(QPixmap)));
```

De esta manera, cada vez que se pulse el botón *Start*, se emitirá una señal que tendrá como consecuencia la visualización de la secuencia de vídeo. Se deben realizar además las conexiones para todos los botones de la interfaz que abren nuevas ventanas o generan realizan diferentes funcionalidades. Al modelo desarrollado en el VPULab se le ha añadido la funcionalidad de activar o desactivar el algoritmo de privacidad mediante un botón en la interfaz principal:

```
void DiVA_detector_qt::on_pushPrivacidad_clicked(){

	if (this->principal->Algor->scram == 0)
		this->principal->Algor->scram = POSTSCRAM;
	else
		this->principal->Algor->scram = PRESCRAM;
```

```
        connect(principal->senales,
SIGNAL(refresh_display(QPixmap)),this,SLOT(update_frame(QPixmap)));
}
```

En la sección de parámetros se ha añadido un nuevo campo para configurar el *color mode.* Las cuatro opciones son los *scrambling* aplicados a los diferentes canales de color (RGB, GR, YCrCb, Y).

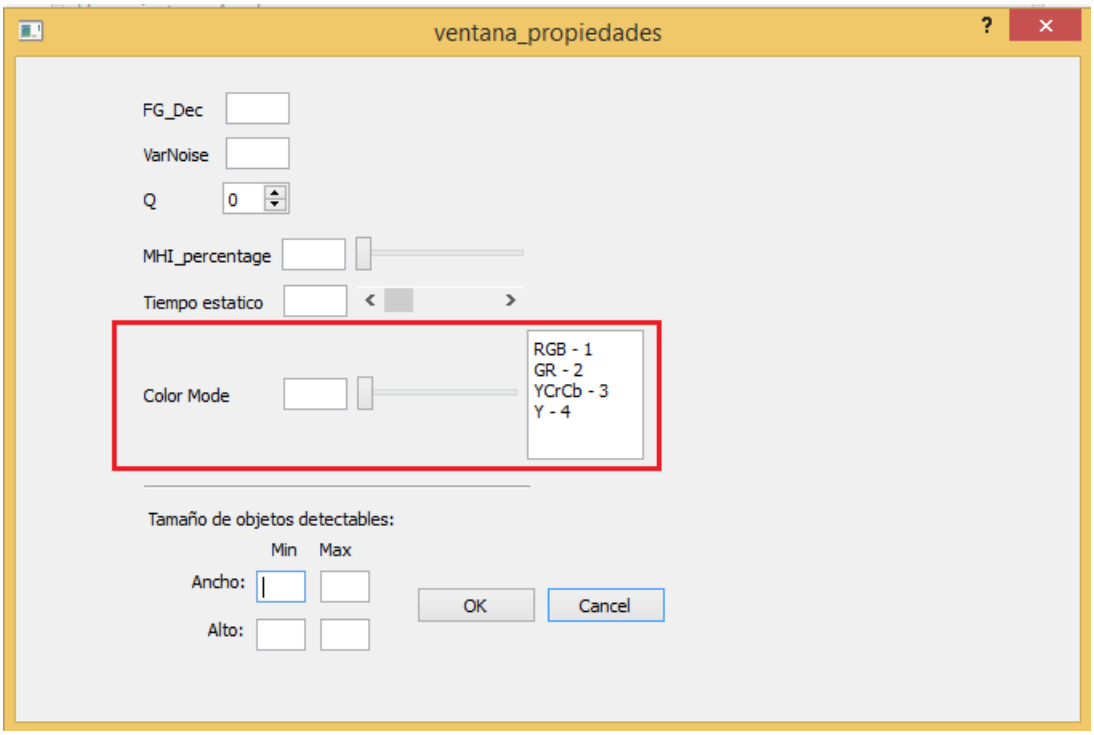

Figura 5.4: Parámetro de modelo de color utilizado con el *scrambling*

5.3 Demostradores

Para la visualización del resultado del algoritmo de la aplicación se han realizado dos modalidades de Interfaz de Usuario que dan lugar a dos aplicaciones o demostradores. La primera de ellas corresponde a una aplicación orientada a cliente en la que se puede comprobar el funcionamiento del algoritmo de privacidad en tiempo real. La otra aplicación está orientada a un desarrollador que quiera visualizar en tiempo real cómo funciona la detección antes de aplicar el *scrambling* y después del mismo utilizando una doble pantalla en la aplicación.

5.3.1 Resultado visual aplicación cliente

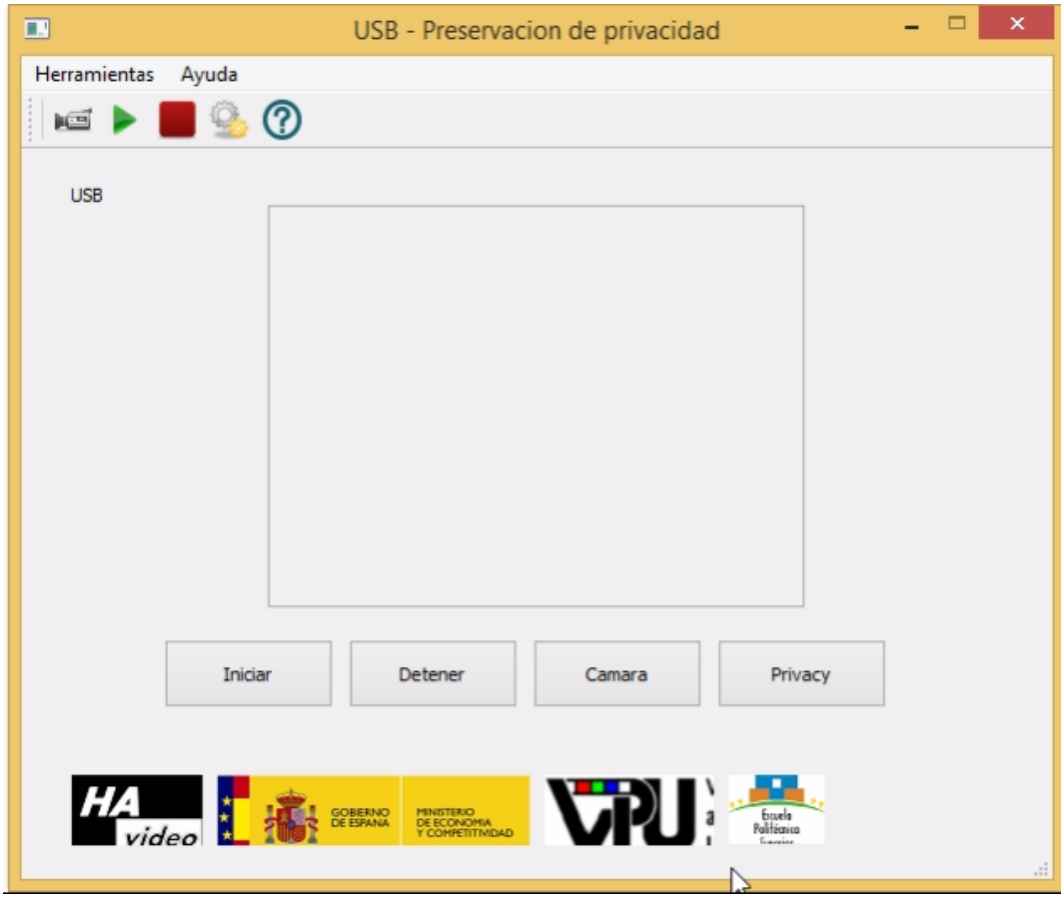

Figura 5.5: Aspecto inicial de la aplicación de cliente. Contiene los elementos detallados en el apartado 5.2.3.2

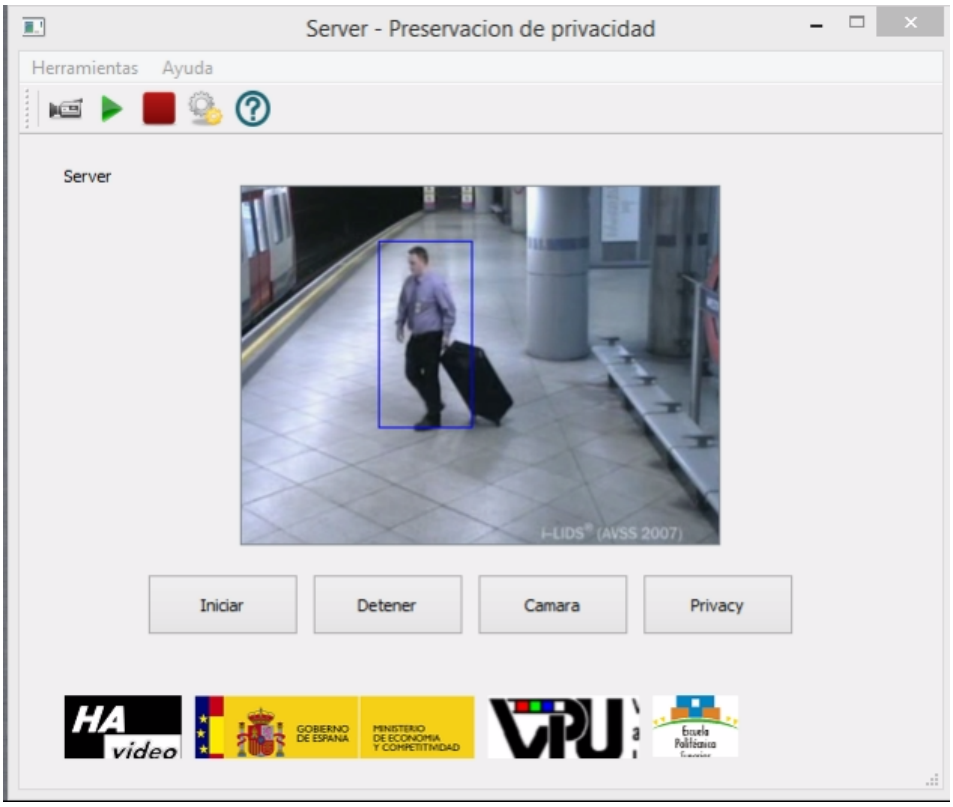

Figura 5.6: Ejemplo de frame con la detección antes de realizar el *scrambling*

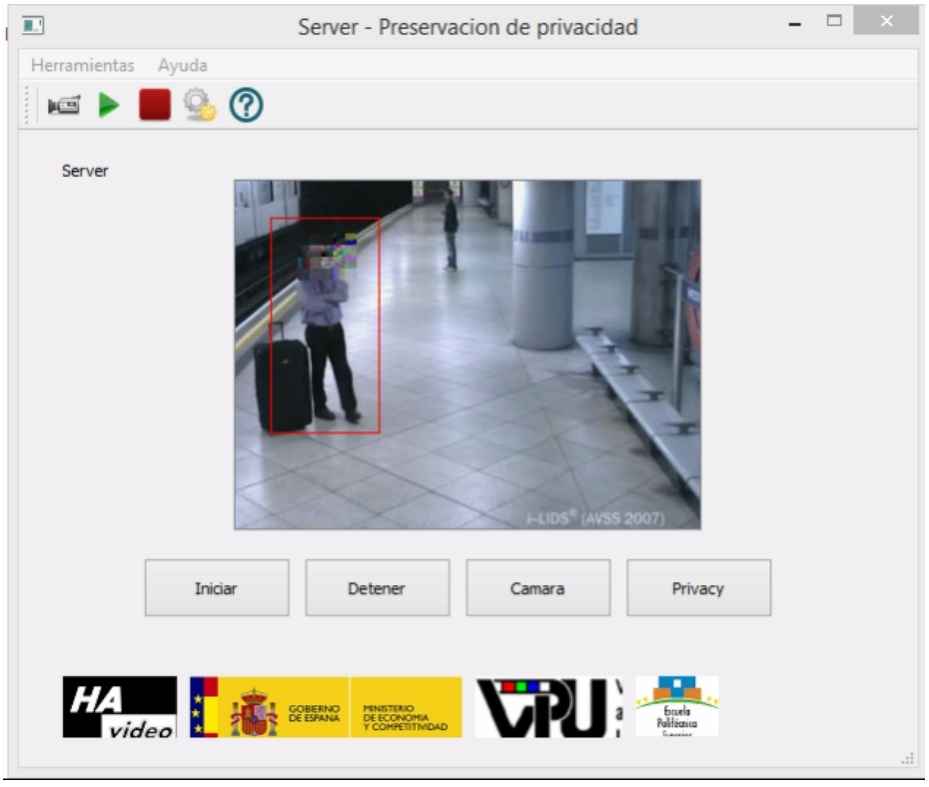

Figura 5.7: Ejemplo de frame con la detección después de realizar el *scrambling*

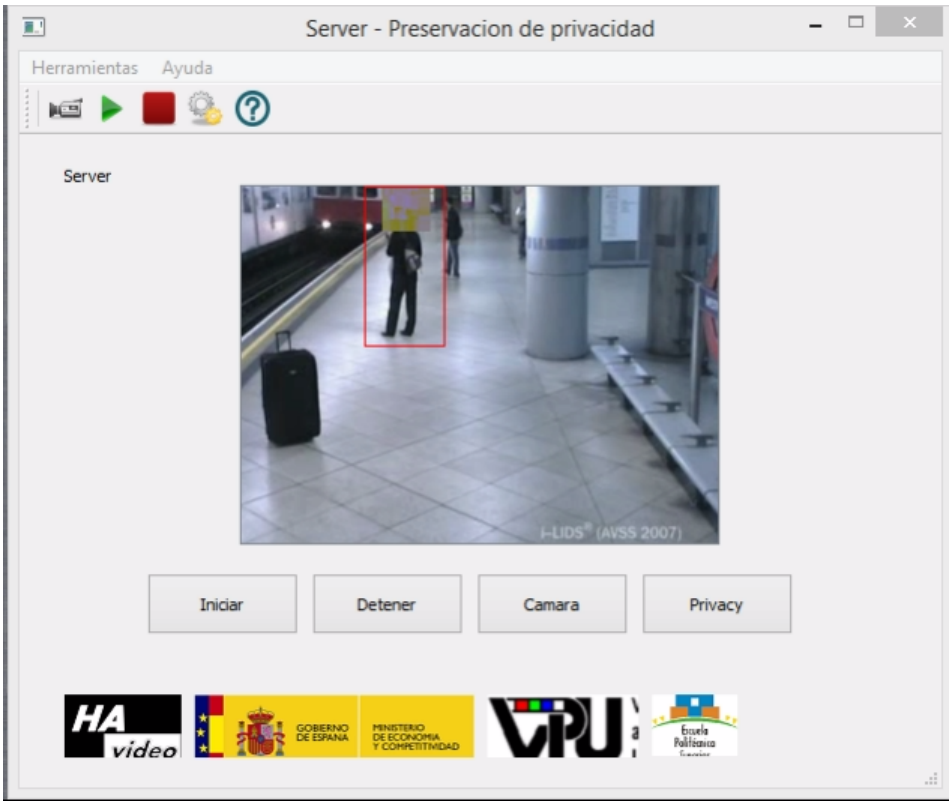

Figura 5.8: Ejemplo de frame con la detección después de realizar el *scrambling*, en este caso al canal Y

5.3.2 Resultado visual aplicación desarrollador

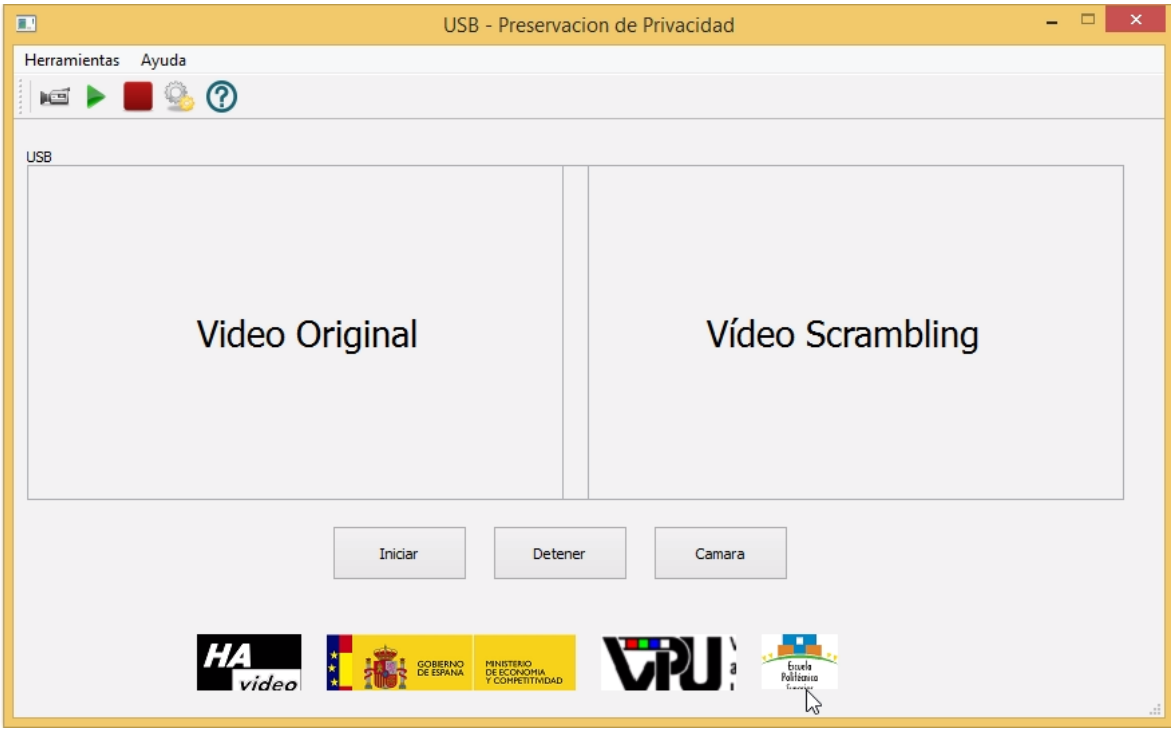

Figura 5.9: Aspecto inicial de la Interfaz de Usuario de la aplicación desarrollador

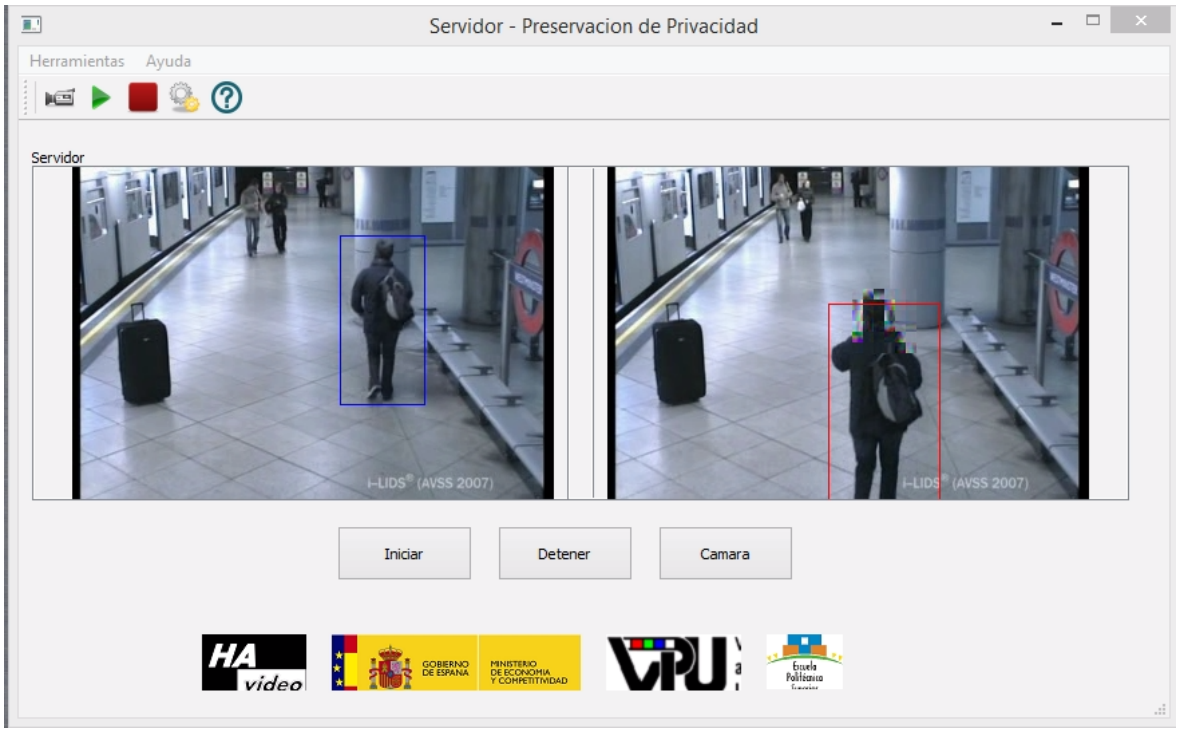

Figura 5.10: Ejemplo de frame con detección sin *scrambling* (izquierda) y con *scrambling* (derecha)

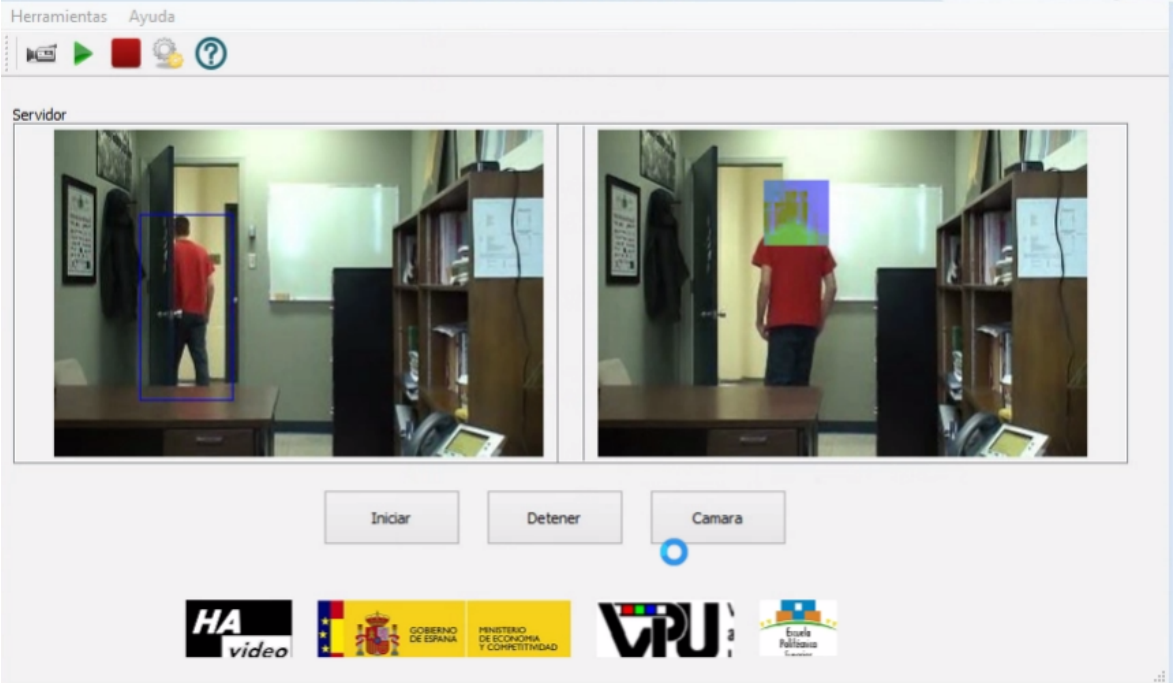

Figura 5.11: Ejemplo de frame con detección sin *scrambling* (izquierda) y con *scrambling* (derecha), en este caso con el parámetro de modelo de color YCrCb

5.4 Conclusiones

En el capítulo 5 se ha desarrollado un aplicación básica para visualizar los resultados de la técnica de preservación de privacidad.

En un primer apartado se introdujo la herramienta a utilizar, en este caso la plataforma DiVA. En un segundo apartado se detalla cómo se ha adaptado el algoritmo de scrambling desarrollado en la sección 3 y evaluado en la 4 para que sea compatible con la herramienta DiVA. Por último, se implementa una Interfaz Gráfica de Usuario utilizando la librería Qt para visualizar el funcionamiento del algoritmo en una pequeña aplicación que permite controlar una serie de parámetros.

6 Conclusiones y Trabajo futuro

6.1 Conclusiones

En este PFC se ha realizado un estudio exhaustivo sobre la problemática de preservar la privacidad en los actuales sistemas de vídeo seguridad. Existen numerosas técnicas que permiten ocultar los rasgos identificativos de las personas en secuencias de vídeo, sin embargo, no todas ellas son de utilidad a la hora de integrarlas en aplicaciones que utilicen, por ejemplo, un detector de personas.

En una primer lugar, se han estudiado estas técnicas así como su funcionamiento, sus características y sus limitaciones. La característica fundamental que requiere una técnica de preservación de privacidad es la reversibilidad ya que resulta poco práctico ocultar la identidad de una persona si luego es imposible recuperar esa imagen original, sobre todo cuando se habla de sistemas de vídeo seguridad. Se plantea así el escenario real en el cuál se puede recuperar dicha secuencia después de que en caso de delito una autorización judicial lo requiera. De esta forma, únicamente la persona autorizada puede ver la identidad de la persona en la secuencia. Es en este punto donde se descartan todas esas técnicas irreversibles que degradan la señal de vídeo de manera irreversible.

Dentro de las técnicas reversibles se elige aquella que mejor respeta los límites de compresión, carga computacional y nivel de privacidad. La técnica *scrambling* es la que lo consigue guardando ese compromiso con los requerimientos.

Una vez elegida la técnica de preservación de privacidad, se desarrolla un algoritmo que por un lado permite visualizar el funcionamiento del mismo y por otro permite sacar unos resultados de cara a una posterior evaluación. El algoritmo se desarrolla utilizando diferentes alternativas de *scrambling* (BGR, GR, YCrCb, Y) de cara a realizar una comparativa y determinar cuál es la que se comporta mejor en la evaluación. Tras un primer prototipo en Matlab, el desarrollo completo del algoritmo se realiza en C++ utilizando las librerías de OpenCV para el tratamiento de vídeo.

La evaluación consiste en determinar si la técnica de *scrambling* afecta a las apliaciones ya existentes en el ámbito de la vídeo seguridad. En este PFC se analiza la detección de personas con y sin el módulo de privacidad del *scrambling.* Se evalúa en primer lugar con el detector HOG y posteriormente con Latent así como todas las versiones de *scrambling* del algoritmo por separado (*scrambling* BGR, GR, YCrCb y Y). El dataset y los programas de evaluación utilizados pertenecen al VPULab. Los resultados demuestran que la técnica de *scrambling* aplicada a sólo dos canales (GR por ejemplo) obtiene los mejores resultados, es decir, afecta

insignificativamente a la posterior detección de personas. Las otras alternativas también se comportan bien al respecto. Sin embargo, hay una clara diferencia a la hora de utilizar un detector u otro. En el caso del detector HOG los resultados son como los descritos antes pero en el caso del detector Latent SVM, el aplicar la técnica *scrambling* arruina la detección de personas tras incluir este módulo de privacidad. Eso se debe a que dicho detector utiliza un modelo basado en partes que detecta la cabeza, los brazos, las piernes… en definitiva compone un modelo de persona a partir de sus extremidades. Al aplicar el *scrambling* al rostro facial, el modelo de cabeza resulta alterado de tal forma que no permite al detector modelar por completo a la persona, fallando en la detección.

Por último, aprovechando la plataforma DiVA diseñada en el VPULab, se desarrolla una aplicación que permite visualizar el resultado del algoritmo de preservación de privacidad. Mediante una serie de funcionalidades básicas (aplicar el módulo de privacidad o no y elegir qué tipo de *scrambling* se aplica) se integra el algoritmo utilizando una interfaz gráfica Qt que permite su utilización por parte de un usuario.

6.2 Trabajo futuro

La preservación de privacidad de personas en el ámbito de la vídeo seguridad es a día de hoy un tema de investigación muy prematuro. Las pocas técnicas que existen para conseguir ese compromiso entre privacidad y seguridad tienen muchas limitaciones y algunas son inservibles para ciertas aplicaciones. Es por ello que el margen de mejora y aportación es enorme.

Este PFC sirve como una primera aproximación a la temática, introduciéndola en las actuales herramientas (evaluación en Matlab, aplicación DiVA…) que se utilizan en el VPULab. Se ha concluido que es posible combinar módulos de privacidad como el estudiado aquí con aplicaciones ya existentes sin que afecten a su funcionamiento. La solución que se ofrece aquí tiene, a pesar de los resultados, muchas cosas que pulir.

En primer lugar, el algoritmo de *scrambling* debe ser mejorado para que no haya incompatibilidades cuando dos regiones de interés están juntas y se solapan. Cuando esto ocurre, como se puede ver en las demostraciones, no tiene en cuenta que haya dos ROIs y sólo aplica el scrambling a una de ellas por lo que su funcionalidad no es del todo correcta.

Es conveniente también realizar una evaluación más exhaustiva del algoritmo, tomando datasets más complejos y con ROIs más complejas. Esto requiere el uso de detectores más potentes que los HOG y LatentSVM, cuyas puntuaciones de detección se acerquen más al Ground Truth y se pueda determinar mejor hasta qué punto el módulo de privacidad afecta

a las técnicas de detección. Con los detectores utilizados en este PFC, hay falsos positivos y ROIs que no se detectan (cuando están en los bordes, por ejemplo), algo que no debe ocurrir en un sistema fiable de vídeo seguridad. También es interesante evaluar el algoritmo con aplicaciones más complejas como la detección de robos.

Como se apunta en las conclusiones de la evaluación [4.7], los detectores que utilizan modelos basados en parte no se comportan bien tras el *scrambling*. Sería interesante por lo tanto preconfigurar dichos modelos de forma que no penalicen cuando la parte correspondiente a la cabeza está "ocluída" por el *scrambling*.

Otra línea de investigación es la de estudiar más en profundidad esas técnicas reversibles que en este PFC se descartan por, en la actualidad, contar con limitaciones. *Video Inpainting* ofrece una solución interesante a la problemática, pero para que resulte útil hay que optimizar la técnica reduciendo su coste computacional de manera que pueda trabajar en tiempo real.

La encriptación y la seguridad de las *"keys"* para recuperar la secuencia original es un tema que no se ha tratado en este PFC pero que es igual de importante que los resultados de la técnica en sí. De nada sirve tener una técnica que oculte los rasgos faciales a la perfección si luego es sencillo desencriptar la secuencia sin autorización.

Por último, la aplicación desarrollada en este PFC también se puede mejorar haciendo una interfaz con más funcionalidades como, por ejemplo, seleccionar la persona cuyo rostro se quiere descifrar dejando los demás con el *scrambling*. El algoritmo aquí desarrollado realiza un *scrambling* a cada una de las ROIs, por lo tanto, se puede guardar la *"key"*de cada uno de los diferentes rostros para hacer una desencriptación selectiva posteriormente.

Referencias

[1] Senior, A., Pankanti, S., Hampapur, A., Brown, L., Tian, Y., Ekin, A., ... Lu, M.. "Enabling Video Privacy through Computer Vision". Tech. Rep. RC22886, IBM T.J.Watson Research Center, NY 10598, 2003.

[2] Senior, Andrew, "Protecting Privacy in Video Surveillance", Dordrecht. Springer, 2009.

[3] Winkler, Thomas, Bernhard, Rinner, "Security and Privacy Protection in Visual Sensor Networks: A Survey", Alpen-Adria Universita¨t Klagenfurt and Lakeside Labs, 2014.

[4] Adam Erdelyi, Thomas Winkler, and Bernhard Rinner, "Serious Fun: Cartooning for Privacy Protection", in Proc. Of the MediaEval Workshop., 2013.

[5] M. Boyle, C. Edwards, and S. Greenberg. "The effects of filtered video on awareness and privacy", in Proc. of ACM Conference on Computer Supported Cooperative Work, pages 1–10, Philadelphia, PA, December 2000.

[6] Dominique Maniry, Esra Acar, Sahin Albayrak, "MediaEval 2013 Visual Privacy Task: Representing People with Foreground Edges", in Proc. Of the MediaEval 2013 Workshop, 2013.

[7] L. O'Gorman, "Video privacy filters with tolerance to segmentation errors for video conferencing and surveillance", in Proc. Of Pattern Recognition (ICPR), Int. Conf. on, pages 1835–1838, 2012.

[8] Pavel Korshunov, Touradj Ebrahimi, "MediaEval 2014 Visual Privacy Task: Geometrical Privacy Protection Tool", in Proc. Of MediaEval 2014 Workshop Barcelona, Spain, 2014.

[9] Pavel Korshunov, and Touradj Ebrahimi, "MediaEval 2013 Visual Privacy Task: Warping-based Privacy Protection Tool", in Proc. Of MediaEval 2013Visual Privacy Task.

[10] Cheung, S.S, Zhao, J., Venkatesh, M.V, "Efficient Object-based Video Inpainting" in Proc. Of Image Processing, IEEE International Conference, 2006.

[11] Encryptation: https://en.wikipedia.org/wiki/Encryption

[12] Pavel Korshunov, and Touradj Ebrahimi. “Using Face Morphing to Protect Privacy”. In Proc. Of 10th IEEE International Conference on Advanced Video and Signal Based Surveillance, 2014.

[13] P. J. Benson. “Morph transformation of the facial image”. Image and Vision Computing, 12(10):691–696, 1994.

[14] A. Martínez-Ballesté, Hatem A. Rashwan, D. Puig and A. Paniza Fullana, “Towards a Trustworthy Privacy in Pervasive Video Surveillance Systems”, PERCOM Workshops, pp.920–925, IEEE Comp. Soc., 2012.

[15] Wenjun Zeng, Member, IEEE, and Shawmin Lei, Senior Member, IEEE. “Efficient Frequency Domain Selective Scrambling of Digital Video”, 2008.

[16] Frédéric Dufaux, Member, IEEE, and Touradj Ebrahimi, Member, IEEE. *H.264/AVC,* “Video Scrambling for Privacy Protectcion”, 2008.

[17] Frédéric Dufaux, Member, IEEE, and Touradj Ebrahimi, Member, IEEE, “Scrambling for Privacy Protection in Video Surveillance Systems”, 2008.

[18] http://es.wikipedia.org/wiki/RFID#Uso_actual

[19] Instituto Nacional de Tecnologías de la Información - Agencia Española de Protección de Datos, “Guía sobre seguridad y privacidad de la tecnología RFID”, 2010.

[20] https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients

[21] Dalal, Navneet, and Trigg, Bills. “Histogram of oriented gradients for human detection”. In Proc. Of CVPR, 886-893, 2005.

[22] Felzenszwalb, Pedro F., et al. “Object detection with discriminatively trained part-based models”, IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(9): 1627-1645, 2010.

[23] J. C. San Miguel and J. M. Martínez. “Strategies for object segmentation, detection and tracking in complex environments for event detection in video surveillance and monitoring. DiVA Documentation D1.2v1. TEC2011-25995 Event Video, 2012-2014.

[24] Qt project. http://qt-project.org/.

[25] OpenCV. http://opencv.org

[26] Sistemas de Vídeo Vigilancia. “Integración en Cliente – DiVA y diseño de GUI”.

[27] Qt (biblioteca). https://es.wikipedia.org/wiki/Qt_(biblioteca)

[28] Carlos Sánchez Bueno, Jose María Martínez Sánchez. “Entorno de desarrollo de aplicaciones de vídeo-seguridad multicámara”. Proyecto Fin de Carrera. Ing. Telecomunicación, Escuela Politécnica Superio, Universidad Autónoma de Madrid, Junio 2014

Bibliografía adicional

- MediaEval Benchmarking Initiative for Multimedia Evaluation. 2014, from http://www.multimediaeval.org/mediaeval2013/visualprivacy2013/
- VideoSense Network of Excellence. 2014, from http://www.videosense.eu/
- Jehan Wickramasuriya, Mohanned Alhazzazi, Mahesh Datt, Sharad Mehrotra, and Nalini Venkatasubramanian. "Privacy-Protecting Video Surveillance." (2005).
- Andrea Melle, Jean-Luc Dugelay. "Shape and Color-aware Privacy Protection". MediaEval 2013 Visual Privacy Task.
- Sebastian Schmiedeke, Pascal Kelm, Thomas Sikora. "Reversible Scrambling with colour-preservative Characteristic". MediaEval 2013 Visual Privacy Task.
- Tomas Piatrik, Virginia Fernandez, and Ebroul Izquierdo. "The Privacy Challenges of In-Depth Video Analytics." Multimedia Signal Processing (MMSP), 2012 IEEE 14th International Workshop on (2012).
- Mohamed Sedky, Claude Chibelushi, Mansour Moniri. "Physics-Based Technique for Protecting Privacy in Surveillance Videos". MediaEval 2013 Visual Privacy Task.
- Choudri, S.; Ferryman, J.M. ; Badii, A. "Robust background model for pixel based people counting using a single uncalibrated camera". Performance Evaluation of Tracking and Surveillance (PETS-Winter), 2009 Twelfth IEEE International Workshop on 2009.
- Atta Badii, Ahmed Al-Obaidi, Mathieu Einig. "Holistic Evaluation Framework for Privacy by Co-Design Impact Assessment". MediaEval 2013 Visual Privacy Task.
- Dominique Maniry, Esra Acar, Sahin Albayrak. "Representing People with Foreground Edges". MediaEval 2013 Visual Privacy Task.
- Isabel Martínez-ponte , Xavier Desurmont , Jerome Meessen , Jean-françois Delaigle. "Robust human face hiding ensuring privacy". (2005).
- Newton, E.M.; Sweeney, L. ; Malin, B. "Preserving privacy by de-identifying face images". 2005.
- Volker Eiselein, Tobias Senst, Ivo Keller, Thomas Sikora. "Using Adaptive Edge

Detection for Privacy in Surveillance Videos". MediaEval 2013 Visual Privacy Task.

- Andrew Senior , Sharath Pankanti , Arun Hampapur , Lisa Brown , Ying-li Tian , Ahmet Ekin. "Blinkering Surveillance: Enabling Video Privacy through Computer Vision". 2013.

Glosario

PFC	Proyecto Fin de Carrera
API	Application Programming Interface
HOG	Histogram of Oriented Gradients
SVM	Support Vector Machine
LSVM	Latent Support Vector Machine
DIVA	Distributed Video Analysis Framework
GT	Ground Truth
VPULab	Video Processing and Understanding Lab
ROI	Region Of Interest
AUC	Area Under the Curve
GUI	Graphical User Interface
RGB	Red Green Blue (modelo de color matemático)
YCrCb	Modelo de color basado en la luminancia (Y) y crominancia (diferencia de azul Cb, diferencia de rojo Cr)

Anexos

A. Algoritmo de scrambling en OpenCV

a. Función process

```
int process(const string& images_folder) {

   // Open windows to show the results
   namedWindow("Postdetection", CV_WINDOW_AUTOSIZE);
   namedWindow("Predetection", CV_WINDOW_AUTOSIZE);
   namedWindow("Scrambling", CV_WINDOW_AUTOSIZE);

   vector<string> images_filenames;
   readDirectory( images_folder, images_filenames );

   for( int i = 0; i < images_filenames.size(); i++ )
   {
      Mat postframe, preframe, framedct, frameidct, roi;
      Mat frame = imread( images_filenames[i] );

      if( frame.empty() )  continue;
      cout << "Process image " << images_filenames[i] << endl;

      cout << "Frame = " << numframe << endl;

      //Predetection
      preframe = predetection(frame);
      imshow("Predetection", preframe);

      //Predetection + Scrambling
      frameidct = scrambling(frame);
      imshow("Scrambling", frameidct);

      //Detection + Scrambling + Postdetection
      postframe = postdetection(frameidct);
      imshow("Postdetection", postframe);

      // If ESC is pressed, finish the loop and exit
      if(waitKey(30) == 27)
      {
         cout << "esc key is pressed by user" << endl;
         break;
      }
      numframe++;
   }
   return 0;
```

```
}
```

b. Función scrambling

```
Mat scrambling(Mat &framedetec){

   Mat rois;
   Mat
frametemp,frameidct,Bframedct,Bframeidct,Gframedct,Gframeidct,Rframedct,Rframeidct;
   framedetec.copyTo(frametemp);


   // HOG para detección de personas
   HOGDescriptor hog;
   hog.setSVMDetector(HOGDescriptor::getDefaultPeopleDetector());

   fflush(stdout);
   vector<Rect> found, found_filtered;
   vector<double> foundWeights, foundWeights_filtered;
   double t = (double)getTickCount();
   // run the detector with default parameters. to get a higher hit-rate
   hog.detectMultiScale(frametemp, found, foundWeights, 0, Size(8,8), Size(32,32), 1.05, 2);

   size_t i, j;
   for( i = 0; i < found.size(); i++ )
   {
      Rect r = found[i];
      for( j = 0; j < found.size(); j++ )
         if( j != i && (r & found[j]) == r)
            break;
      if( j == found.size() )
         found_filtered.push_back(r);
   }

   // If ROI is found apply DCT and scrambling
   if (found_filtered.size()!=0) {
      for( i = 0; i < found_filtered.size(); i++ )
      {
         Mat imageroi;
         Rect r = found_filtered[i];
         Rect r2;

         // HOG detector returns slightly larger rectangles than the real objects.
         // so we slightly shrink the rectangles to get a nicer output.
         r.x += cvRound(r.width*0.05);
         r.x = r.x+r.width/4;
         r.width = cvRound(r.width*0.8);
         r.width = r.width/2;
         r.y += cvRound(r.height*0.05);
         r.height = cvRound(r.height*0.8);
         // Selects up part of the ROI which remains to the face
```

```
        r.height = r.height/4;

        // If ROI is out of frame or negative coordinates
        if (r.x < 0 || r.y <0 ||r.x>frametemp.cols-r.width || r.y>frametemp.rows-r.height){
           return frametemp;
           //r.width = 8;
           //imageroi = frametemp(r);
        }
        else {
           imageroi = frametemp(r);
        }

        // Split ROI in RGB channels
        Mat channel[3], roiR, roiG, roiB;
        split(imageroi, channel);
        roiB = channel[0];
        roiG = channel[1];
        roiR = channel[2];

        // Call DCT and IDCT for each channel
        Bframedct = dctcall(roiB,r);
        Bframeidct = idctcall(Bframedct);
        Gframedct = dctcall(roiG,r);
        Gframeidct = idctcall(Gframedct);
        Rframedct = dctcall(roiR,r);
        Rframeidct = idctcall(Rframedct);

        // Copy scrambling results in each channel
        Bframeidct.copyTo(channel[0]);
        Gframeidct.copyTo(channel[1]);
        Rframeidct.copyTo(channel[2]);

        // Mix channels in a Mat with scrambling results
        merge(channel,3,frameidct);

        int width = frameidct.cols;
        int height = frameidct.rows;
        r2.x = r.x;
        r2.width = width;
        r2.y = r.y;
        r2.height = height;

        // If ROI is outside the frame or negative coordinates
        if (r2.x < 0 || r2.y <0 ||r2.x>frametemp.cols-r2.width || r2.y>frametemp.rows-
r2.height){
           if (r2.y > r2.height){
              frameidct.copyTo( frametemp.rowRange(r2.y-r2.height/2,
r2.y+r2.height/2).colRange(r2.x-r2.width/2, r2.x+r2.width/2));
           }
           else{
```

```
                frameidct.copyTo( frametemp.rowRange(r2.y, r2.y+r2.height).colRange(r2.x-
r2.width/2, r2.x+r2.width/2));
            }
        }
        else {
            frameidct.copyTo(  frametemp.rowRange(r2.y,  r2.y+r2.height).colRange(r2.x,
r2.x+r2.width));
        }
    }
  }
  return frametemp;
}
```

B. Definición de la clase detector

```
class detector{
public:
        detector();
        ~detector();
        void * process(IplImage *frame);
        int setParam(int parametro, int valor_int=0);
        int getParam(int parametro, int *valor_int);
        void showresults();
        int color;
        int scram;

private:
        Mat preframe, postframe, frameidct;
};
```

C. Métodos de la clase detector

```
void * detector::process(IplImage *frame){

        Mat frametemp;
        frametemp = frame;

        switch (scram)
        {
        case PRESCRAM:
                //Predetection
                preframe = predetection(frametemp);
                break;

        case POSTSCRAM:
        {
                //Predetection + Scrambling
                frameidct = scrambling(frametemp, color);
```

```
                //Detection + Scrambling + Postdetection
                postframe = postdetection(frameidct);
                break;
        }
        default:
                printf("Parametro especificado incorrectamente.\n");
                break;
        }

        return NULL;
}

void detector::showresults(){

        switch (scram)
        {
                case PRESCRAM:
                {
                        namedWindow("Pre-detection", WINDOW_NORMAL);
                        imshow("Pre-detection", this->preframe);
                        break;
                }
                case POSTSCRAM:
                {
                        namedWindow("Post-detection", WINDOW_NORMAL);
                        imshow("Post-detection", this->postframe);
                        break;
                }
        }
        waitKey(1);
}
int detector::getParam(int parametro, int *valor_int){
        switch (parametro)
        {
        case COLOR:
                *valor_int = this->color;
                break;
        case SCRAM:
                *valor_int = this->scram;
                break;
        default:
                printf("Parámetro especificado incorrectamente.\n");
                break;
        }
        return 1;
}

int detector::setParam(int parametro, int valor_int){
        switch (parametro)
        {
```

```
        case COLOR:
                this->color = valor_int;
                break;
        case SCRAM:
                this->scram = valor_int;
                break;
        default:
                printf("Parámetro especificado incorrectamente.\n");
                break;
        }
        return 1;
}
```

D. Clase DiVA_detector

```
class DiVA_detector:public DiVAAlgorithm
{
public:
        DiVA_detector(char* frameServername = NULL, int portnumber= NULL,
                        BOOL fileDumping= NULL,
                        BOOL display= NULL,
                        char* dataServerName= NULL,
                        char* contentServerName= NULL,
                        BOOL dataServerDumping= NULL,
                        BOOL contentServerDumping= NULL,
                        int captureMode=NULL);
        ~DiVA_detector();
        int processFrame(DiVAImage* pImage,void* pdata=NULL,void*
pContentData=NULL);
        int setParam(int parametro, int valor_int=0);
        int getParam(int parametro, int *valor_int);
        int refreshDisplay();
private:
        detector *Algor;
        OpenCVConverter *pconversor;
};
```

PRESUPUESTO

1) Ejecución Material

- Compra de ordenador personal (Software incluido)..... 2.000 €
- Material de oficina .....200 €
- Total de ejecución material..... 2.200 €

2) Gastos generales

- 16 % sobre Ejecución Material ..... 352 €

3) Beneficio Industrial

- 6 % sobre Ejecución Material ..... 132 €

4) Honorarios Proyecto

- 640 horas a 15 € / hora ..... 9600 €

5) Material fungible

- Gastos de impresión ..... 60 €
- Encuadernación ..... 200 €

6) Subtotal del presupuesto

- Subtotal Presupuesto ..... 12060 €

7) I.V.A. aplicable

- 21% Subtotal Presupuesto..... 2532.6 €

8) Total presupuesto

- Total Presupuesto ..... 14593,6 €

Madrid, Septiembre de 2015

El Ingeniero Jefe de Proyecto

Fdo.: Jaime Mateo Herrero

Ingeniero de Telecomunicación

PLIEGO DE CONDICIONES

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de una aplicación de preservación de privacidad en vídeo seguridad. En lo que sigue, se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa consultora con la finalidad de realizar dicho sistema. Dicha empresa ha debido desarrollar una línea de investigación con objeto de elaborar el proyecto. Esta línea de investigación, junto con el posterior desarrollo de los programas está amparada por las condiciones particulares del siguiente pliego.

Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes:

Condiciones generales

1. La modalidad de contratación será el concurso. La adjudicación se hará, por tanto, a la proposición más favorable sin atender exclusivamente al valor económico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho a declararlo desierto.

2. El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.

3. En la oferta, se hará constar el precio total por el que se compromete a realizar la obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si este se hubiera fijado.

4. La obra se realizará bajo la dirección técnica de un Ingeniero Superior de Telecomunicación, auxiliado por el número de Ingenieros Técnicos y Programadores que se estime preciso para el desarrollo de la misma.

5. Aparte del Ingeniero Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no estará obligado a aceptarla.

6. El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el contratista después de confrontarlas.

7. Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a sus facultades le hayan comunicado por escrito al Ingeniero Director de obras siempre que dicha obra se haya ajustado a los preceptos de los pliegos de condiciones, con arreglo a los cuales, se harán las modificaciones y la valoración de las diversas unidades sin que el importe total pueda exceder de los presupuestos aprobados. Por consiguiente, el número de unidades que se consignan en el proyecto o en el presupuesto, no podrá servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.

8. Tanto en las certificaciones de obras como en la liquidación final, se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.

9. Si excepcionalmente se hubiera ejecutado algún trabajo que no se ajustase a las condiciones de la contrata pero que sin embargo es admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero estime justa y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.

10. Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras

o materiales análogos si los hubiere y cuando no, se discutirán entre el Ingeniero Director y el contratista, sometiéndolos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento, se sujetarán siempre al establecido en el punto anterior.

11. Cuando el contratista, con autorización del Ingeniero Director de obras, emplee materiales de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o sustituya una clase de fabricación por otra que tenga asignado mayor precio o ejecute con mayores dimensiones cualquier otra parte de las obras, o en general, introduzca en ellas cualquier modificación que sea beneficiosa a juicio del Ingeniero Director de obras, no tendrá derecho sin embargo, sino a lo que le correspondería si hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.

12. Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ellas se formen, o en su defecto, por lo que resulte de su medición final.

13. El contratista queda obligado a abonar al Ingeniero autor del proyecto y director de obras así como a los Ingenieros Técnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y administración en su caso, con arreglo a las tarifas y honorarios vigentes.

14. Concluida la ejecución de la obra, será reconocida por el Ingeniero Director que a tal efecto designe la empresa.

15. La garantía definitiva será del 4% del presupuesto y la provisional del 2%.

16. La forma de pago será por certificaciones mensuales de la obra ejecutada, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.

17. La fecha de comienzo de las obras será a partir de los 15 días naturales del replanteo oficial de las mismas y la definitiva, al año de haber ejecutado la provisional, procediéndose si no existe reclamación alguna, a la reclamación de la fianza.

18. Si el contratista al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.

19. El contratista está obligado a designar una persona responsable que se entenderá con el Ingeniero Director de obras, o con el delegado que éste designe, para todo relacionado con ella. Al ser el Ingeniero Director de obras el que interpreta el proyecto, el contratista deberá consultarle cualquier duda que surja en su realización.

20. Durante la realización de la obra, se girarán visitas de inspección por personal facultativo de la empresa cliente, para hacer las comprobaciones que se crean oportunas. Es obligación del contratista, la conservación de la obra ya ejecutada hasta la recepción de la misma, por lo que el deterioro parcial o total de ella, aunque sea por agentes atmosféricos u otras causas, deberá ser reparado o reconstruido por su cuenta.

21. El contratista, deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa, por retraso de la ejecución siempre que éste no sea debido a causas de fuerza mayor. A la terminación de la obra, se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.

22. Hecha la recepción provisional, se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el

acta correspondiente. El Director Técnico propondrá a la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas legales establecidas.

23. Las tarifas para la determinación de honorarios, reguladas por orden de la Presidencia del Gobierno el 19 de Octubre de 1961, se aplicarán sobre el denominado en la actualidad "Presupuesto de Ejecución de Contrata" y anteriormente llamado "Presupuesto de Ejecución Material" que hoy designa otro concepto.

Condiciones particulares

La empresa consultora, que ha desarrollado el presente proyecto, lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares:

1. La propiedad intelectual de los procesos descritos y analizados en el presente trabajo, pertenece por entero a la empresa consultora representada por el Ingeniero Director del Proyecto.

2. La empresa consultora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente proyecto, bien para su publicación o bien para su uso en trabajos o proyectos posteriores, para la misma empresa cliente o para otra.

3. Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales, bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.

4. En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones así como su cantidad.

5. En todas las reproducciones se indicará su procedencia, explicitando el nombre del proyecto, nombre del Ingeniero Director y de la empresa consultora.

6. Si el proyecto pasa la etapa de desarrollo, cualquier modificación que se realice sobre él, deberá ser notificada al Ingeniero Director del Proyecto y a criterio de éste, la empresa consultora decidirá aceptar o no la modificación propuesta.

7. Si la modificación se acepta, la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta el añadirla.

8. Si la modificación no es aceptada, por el contrario, la empresa consultora declinará toda responsabilidad que se derive de la aplicación o influencia de la misma.

9. Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto, deberá comunicarlo a la empresa consultora.

10. La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.

11. La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha aplicación industrial, siempre que no haga explícita renuncia a este hecho. En este caso, deberá autorizar expresamente los proyectos presentados por otros.

12. El Ingeniero Director del presente proyecto, será el responsable de la dirección de la aplicación industrial siempre que la empresa consultora lo estime oportuno. En caso contrario, la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.