UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

PROYECTO FIN DE CARRERA

EMISIÓN DE CLASES PRESENCIALES: PRODUCCIÓN AUTOMÁTICA Y APLICACIÓN BÁSICA

-PROYECTO FIN DE CARRERA-

José Luis Gamo Revilla
Abril 2012

EMISIÓN DE CLASES PRESENCIALES: PRODUCCIÓN AUTOMÁTICA Y APLICACIÓN BÁSICA

AUTOR: José Luis Gamo Revilla
TUTOR: Jesús Bescós Cano

Video Processing and Understanding Lab
Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Abril 2012

Palabras clave

Emisión de clases presenciales, seguimiento activo, análisis de vídeo, SURF, CAMShift, *tracking* de objetos, cámara IP, cámara PTZ, *streaming* de vídeo.

Resumen

El presente PFC tiene como objetivo el estudio de las distintas situaciones que requieren la emisión de clases presenciales a puntos remotos y el desarrollo de una aplicación que las resuelva, además de dotarla de la funcionalidad añadida de realizar un seguimiento automático del profesor.

Para poder llevar a cabo el seguimiento automático del profesor se plantea el estudio de las distintas técnicas de *tracking* de objetos en secuencias de vídeo, e implementación de un algoritmo basado en estas técnicas que trabaje con la secuencia capturada por la cámara en tiempo real.

Abstract

The scope of the current PFC has the object of studying the various situations that require the issuance of classes to remote points and developing an application that resolve them, as well give it the added functionality of automatically tracking of the teacher.

To carry out the automatic tracking of the teacher focuses on the study of the different techniques for tracking objects in video sequences, and implementation of an algorithm based on these techniques to work with the sequence captured by the camera in real time.

Agradecimientos

En primer lugar quiero agradecer a mi tutor, Jesús Bescós, el apoyo y la dedicación que me ha brindado en la realización de este proyecto, gracias a su inestimable ayuda he llegado al final del camino.

A José María Martínez, que junto con Jesús, han permitido que realice este proyecto en el VPULab y me han ayudado durante toda la carrera. También agradecer a los compañeros del laboratorio que de una manera u otra se han interesado por mi proyecto.

A todos mis amigos, a los que ya tenía y a los que he conocido en la universidad, por los momentos que hemos pasado durante este tiempo y por los momentos que espero pasar con vosotros.

A toda mi familia, y muy especialmente a mis padres, ya que son los que me han aguantado durante todo este tiempo y me han estado apoyando incondicionalmente. Sé que el haber terminado esta carrera es el mejor regalo que os puedo dar.

A Verónica, que durante este tiempo que ha durado el proyecto me has apoyado en todo momento y has confiado en mí, gracias por todo tu cariño. Por fin puedo abrir nuevas puertas junto a tí.

José Luis Gamo Revilla

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

En este primer capítulo se realizará una descripción de la base de la idea de la que parte este proyecto, marcando unos objetivos y una estructuración de la memoria.

1.1 Motivación

Debido al gran avance de las tecnologías de la información en los últimos años podemos tener acceso a diferentes sistemas de información desde nuestra propia casa a través de internet. A veces es simplemente una comodidad el tener esta opción de acceso, pero en otros casos se convierte en algo necesario para ciertas personas. Es por ello que se cree necesario tener un sistema de videoconferencia en las aulas, adaptado al tipo de cámara que se instala en ellas.

A través del sistema de videoconferencia instalado en las aulas podremos dar acceso a la sesión que se está impartiendo en ella a cualquier alumno, que por causas justificadas (no se quiere reducir la asistencia a las clases), no sea posible que asista a la sesión. También sería posible que en ponencias cualquier componente del tribunal que se encuentre demasiado lejos, pueda visualizarla y otorgar la nota correspondiente. Finalmente sería también una buena opción para en conferencias dar acceso a ciertas personas acreditadas que tengan problemas para asistir.

A la hora de utilizar el sistema de videoconferencia sería un gran avance que nadie se tuviese que hacer cargo de mover la cámara para seguir siempre al profesor o estar enfocando la zona de actividad en la que se está moviendo. Es en este momento en el que se hace necesario un sistema de seguimiento automático del profesor. Haciendo uso de las características de la cámara, que en este caso es una cámara PTZ (*Pan, Tilt, Zoom*, permite movimientos horizontales, verticales y ajustes del *zoom*) y creando el sistema de localización del profesor, podremos ordenar a la cámara que realice los movimientos necesarios hacia cada nueva posición del profesor.

El sistema de localización se deberá realizar basándose en el procesado del flujo de vídeo que la cámara está grabando en cada instante, ya que es la única fuente de información que tenemos en este caso. El procesado deberá realizarse en tiempo real y acercarse a procesar los 25 fps (*frames* por segundo) que será la mayor velocidad con la que la cámara nos irá sirviendo cada *frame*. En el caso de que esta velocidad de procesado no se pudiese alcanzar

debido a que no se encuentre una forma de detectar el movimiento del profesor, nos conformaríamos con una velocidad no inferior a 5 fps. Si no se alcanzasen estos 5 fps correríamos el riesgo de que el profesor al realizar desplazamientos durante la explicación, se saliese del encuadre de la cámara mientras la aplicación se encuentra procesando el *frame* anterior. Una vez se produzca la salida del profesor del cuadro, la cámara lo perdería completamente porque comenzaría a procesar y a buscar al profesor en *frames* en los que no está, y por lo tanto ordenando diversos movimientos erróneos a la cámara haciendo impredecible la región en la que estará enfocando.

La motivación de este proyecto se basa en la necesidad de conseguir un sistema de videoconferencia, con un seguimiento automático del profesor que trabaje en tiempo real, y se adapte a las características de una cámara específica (en este caso la cámara es SONY SNC-RZ50P) exprimiendo al máximo sus posibilidades de utilización.

1.2 Objetivos

El principal objetivo de este PFC es el desarrollo de una o varias aplicaciones de videoconferencia con las cuales conseguir las funcionalidades de emisión de las clases presenciales y producción automática del video (modo seguimiento, modo estático, modo escritorio). Se desarrollarán aplicaciones con funcionalidades limitadas específicas para cada uno de los distintos casos. En el caso de ser miembro de un tribunal, se necesitará una aplicación con permisos de control del movimiento de la cámara pero no sería necesario un seguimiento automático. En el caso de un alumno, únicamente se otorgarán permisos para la visualización. Será en el caso de la aplicación para el profesor en la que deberemos contar con todas las funcionalidades posibles, ya que él sí necesitará un seguimiento automático y la posible selección del modo escritorio.

Para lograr los retos planteados en este proyecto será necesaria la realización de las siguientes tareas:

i. **Estudio del estado del arte actual:** Inicialmente, se realizará un estudio de las distintas plataformas y librerías existentes y se valorará su adecuación al problema. Se estudiarán las distintas técnicas de producción y de seguimiento de objetos aislados en entornos de iluminación controlada. Se recopilará información sobre *software* gratuito que pueda sernos útil en vista de sus funcionalidades. Se estudiarán las técnicas de seguimiento de

objetos en secuencias de vídeo y las distintas posibilidades para realizar el *streaming* de vídeo.

ii. **Arquitectura del sistema:** Una vez seleccionado el *software* que nos es funcional (en el caso de que haya alguno que cumpla todos los requisitos), se estudiará la mejor arquitectura que resuelve el problema y se planteará el desarrollo de las aplicaciones necesarias.

iii. **Sistema de captura:** Se estudiará el de funcionamiento de la cámara seleccionando la configuración más adecuada para la interacción con las aplicaciones a desarrollar. Se comprenderán las distintas funciones implementadas en las librerías para poder utilizarlas correctamente en dichas aplicaciones.

iv. **Subsistema de seguimiento activo:** Se implementarán dos algoritmos para la localización del profesor y mediante la fusión de ambos se obtendrá un algoritmo general más robusto y con mejores resultados frente a la pérdida del profesor. Tendremos el seguimiento mediante puntos característicos (SURF) y el seguimiento mediante histograma de color.

v. **Subsistema de emisión:** Se estudiará la funcionalidad de las librerías Gstreamer para la emisión del flujo de vídeo y para la producción del vídeo en modo escritorio. Tras el estudio se integrarán dichas librerías en la aplicación general obteniendo el sistema completo de emisión y producción automática.

vi. **Análisis de resultados y conclusiones:** Se extraerán una serie de conclusiones de cada uno de los algoritmos por separado y una valoración del sistema general con la fusión de ambos. También se valorarán las pruebas realizadas con cada una de las aplicaciones desarrolladas.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Capítulo 1:** Introducción, objetivos y motivación del proyecto.

- **Capítulo 2:** Análisis de plataformas, evaluación e su adecuación, estudio de las técnicas de producción y de seguimiento de objetos, recopilación de *software* gratuito, estudio del seguimiento de objetos en secuencias de vídeo y del *streaming* de video.

- **Capítulo 3:** Descripción del sistema y planteamiento de las aplicaciones a desarrollar.

- **Capítulo 4:** Descripción y configuración de la cámara, obtención de *frames* y envío de comandos de movimiento.

- **Capítulo 5:** Subsistema de seguimiento con la integración de OpenCV para el procesado y estudio de los algoritmos de seguimiento, determinación de la nueva posición y movimiento de la cámara.

- **Capítulo 6:** Subsistema de emisión con la integración de Gstreamer.

- **Capítulo 7:** Evaluación de los resultados obtenidos por los algoritmos y por la emisión-recepción del flujo de vídeo.

- **Capítulo 8:** Conclusiones y trabajo futuro.

2 Estado del arte

En este capítulo se van a analizar las distintas librerías, técnicas, y aplicaciones que ya están desarrolladas y que se ajusten en cierta medida a los requerimientos de este problema.

2.1 Análisis de plataformas y librerías disponibles

Las primeras librerías de las que hay que hablar son SonyNetworkCamera4.dll y sncstrm.dll. Son las librerías necesarias para la realización de aplicaciones que mantengan una conexión con la cámara que se va a utilizar. SonyNetworkCamera4.dll proporciona funciones específicas para el control de la cámara, para su configuración y para la conexión con la misma, mientras que sncstrm.dll proporciona las funciones necesarias para el manejo del flujo de vídeo (recepción en la aplicación y presentación en la región predeterminada).

Se utilizarán las librerías de OpenCV para realizar todo el procesado de imágenes y del flujo de vídeo recibido para la resolución del seguimiento automático. Se podrán resolver tanto el problema del seguimiento mediante puntos SURF como el seguimiento por histograma de color. También servirá para presentar por pantalla los resultados que se van obteniendo de los seguimientos y nos facilitará las tareas de presentación y selección de una región determinada de la imagen que será el objeto a seguir.

Una vez tenemos el control total sobre la cámara (gracias a las librerías SonyNetworkCamera4.dll y sncstrm.dll) y la posibilidad de realizar el procesado del flujo de vídeo para el seguimiento automático (gracias a las librerías de OpenCV), falta por dotar a la aplicación de la funcionalidad de emitir un flujo de vídeo a un punto remoto. Para esta funcionalidad se ha pensado en primera opción, en la utilización de las librerías de VLC, ya que VLCmediaPlayer resuelve en parte la emisión del flujo de vídeo. Digo que las resuelve en parte porque en las pruebas realizadas sólo se ha conseguido realizar la emisión en una red de área local. En segunda opción están las librerías de Gstreamer, a través de las cuales se puede construir un *pipeline* que reciba el flujo de la aplicación desarrollada y lo emita a través de internet a una dirección determinada (o a varias en el caso de poder utilizar multicast).

2.2 Evaluación de su adecuación al problema a resolver

En cuanto a la adecuación al problema a resolver, está claro que tanto SonyNetworkCamera4.dll y sncstrm.dll son totalmente funcionales y además totalmente necesarias ya que sin ellas sería imposible realizar la comunicación con la cámara, tanto para recepción del flujo de vídeo como para realizar los movimientos necesarios de la cámara.

De estas librerías se utilizarán las siguientes funciones que ya están programadas internamente:

- `SNC::LoadLibrary()`: se encarga de inicializar y cargar todas las funciones existentes en las librerías para poder hacer uso de ellas.

- `SNC::OpenHandle()`: crea los módulos necesarios para trabajar con la cámara.

- `SNC::SetVideoCodec(sncHandle, VIDEO_CODEC_JPEG)`: configura la librería para que trabaje con la codificación especificada en el parámetro de la función (no cambia la configuración de la cámara, ésta, estará configurada con una codificación principal y otra secundaria, cada aplicación podrá seleccionar una u otra según le sea necesario).

- `SNC::CreateScreen(sncHandle, windowHandle, 280, 10, width, height)`: crea la pantalla en la que se va a presentar el flujo de vídeo recibido de la cámara. Se pueden modificar las coordenadas de posición y el tamaño de la pantalla, pero siempre teniendo en cuenta que el tamaño de la interfaz gráfica de la aplicación debe de ser suficientemente grande para que quepa completamente.

- `SNC::SetNetwork(sncHandle, &netInfo)`: con esta función se configuran los parámetros de conexión con la cámara. En la estructura netInfo se encuentran los campos necesarios para introducir el nombre del usuario, la contraseña, la dirección IP de la cámara y el puerto al que se quiere conectar (hay algunos parámetro más pero no ha sido necesaria su utilización).

- `SNC::EnableVideoStream(sncHandle, true)`: gracias a esta función se puede activar o desactivar el *stream* de vídeo según el argumento que se ponga.

- `SNC::EnableVideoPlay(sncHandle, true)`: se activa o desactiva la visualización en la pantalla predeterminada del flujo de vídeo recibido según el argumento de la función.

- `SNC::EnableAudioStream(sncHandle, true)`: se activa o desactiva el *stream* de audio según el argumento de la función.

- `SNC::EnableAudioPlay(sncHandle, true)`: se activa o desactiva la escucha del audio recibido según el argumento de la función.

- `SNC::EnableVideoDecode(sncHandle, true)`: se activa o desactiva la decodificación del vídeo para recibirlo en la aplicación, codificado o decodificado, según el argumento de la función.

- `SNC::SetCallback(sncHandle, CALLBACK_DEC_VIDEO, videoCallback, NULL)`: se realiza la asignación de la función a la que hay que llamar en cada momento de interrupción por decodificación del vídeo. De esta manera tendremos la imagen actual en nuestra aplicación para poder realizar sobre ella cualquier tarea que sea necesaria en ese momento. En este caso para realizar el procesado de imagen para el seguimiento, tanto por puntos SURF como por histograma de color.

- `SNC::SetMessageEnd(sncHandle, windowHandle, WM_SNC_END, 0)`: asigna el código que se devolverá a la rutina principal en el caso de finalizar la conexión con la cámara, y de esta manera poder finalizar también la rutina principal.

- `SNC::Start(sncHandle)`: comienza la conexión con la cámara y la presentación del flujo de vídeo en caso de estar activada. A partir de aquí se podrá utilizar cualquier función de movimiento de la cámara o de ajuste del *zoom* para colocarla en la posición que más interese.

- `SNC::SendCGICommand(sncHandle, "POST", "/command/ptzf.cgi", comando, sResponse, sizeof(sResponse), &ret_len)`: función utilizada para realizar el envío de cualquier comando a la cámara. En este caso se envía un comando de movimiento en el cual podrá estar descrito un movimiento relativo, o absoluto a un valor determinado tanto de posición como de *zoom*.

- `SNC::Stop(sncHandle)`: esta función es la encargada de finalizar todos los *streams* que se estén realizando así como la presentación del flujo recibido.

- `SNC::CloseHandle(sncHandle)`: elimina los módulos creados en la inicialización de la librería.

- `SNC::FreeLibrary()`: una vez se haya desinicializado la librería hay que liberarla, y es precisamente lo que realiza esta función. Será la última que se utilizará.

Para resolver todo el asunto del seguimiento automático del profesor se hace necesaria la utilización de funciones de procesado de imágenes, para ello obtenemos una completa funcionalidad de las librerías de OpenCV, que se adecúan completamente al problema. En el caso de la utilización de las funciones de OpenCV tenemos que dividir las funciones en tres partes, las funciones de utilización general, las funciones relacionadas con el seguimiento por puntos SURF y las funciones relacionadas con el seguimiento por histograma.

Funciones generales:

- `cvNamedWindow("Asignacion de puntos")`: crea una nueva ventana en la que presentar imágenes y la pone el título que se pasa como argumento de la función. La utilizaremos tanto para presentar los resultados como para poder seleccionar alguna sección de la imagen para el seguimiento automático.

- `cvMoveWindow("Asignacion de puntos", 100, 200)`: mueve la ventana creada previamente con cierto título a la posición en la pantalla que se indica como argumentos de la función.

- `cvDestroyWindow("Asignacion de puntos")`: destruye la ventana con el título que se indica como argumento de la función.

- `cvShowImage("Asignacion de puntos", frame_inicial)`: muestra en la ventana con el título indicado la imagen se pasa como argumento a la función.

- `cvSetMouseCallback("Asignacion de puntos ", onMouseCalibracion, 0)`: activa la función a la que hay que llamar cuando se produce una interrupción por algún evento producido por el ratón sobre la ventana con el título indicado en los argumentos de la función. Se utilizará para seleccionar la región en la que se encuentra inicialmente el profesor y poder obtener las coordenadas.

- `cvCopy(frame_actual, frame_inicial)`: copia una IplImage en otra quedando dos IplImage independientes. En este caso copiaría la frame_actual en la frame_inicial.

- `cvRectangle(frame_modificado, p1, p2, CV_RGB(255, 255, 255), 1, 8, 0)`: dibuja un rectángulo en una IplImage fijando como esquina superior izquierda p1 y como esquina inferior derecha p2, del color, grosor y tipo de línea definidos en los argumentos de la función. Se utilizará para dibujar la región en la que se encuentra el objeto que se está siguiendo en cada *frame*.

- `cvCreateImage(cvSize(640, 480), 8, 3)`: crea una IplImage del tamaño introducido en el primer argumento, con una profundidad de color de 8 bits y con 3 canales en este caso. En el caso de ser en escala de grises se pondrá únicamente un canal. Esta función devuelve un puntero a la dirección de memoria reservada para la imagen.

- `cvReleaseImage(&frame)`: libera la memoria reservada previamente por cvCreateImage.

- `cvSetImageROI(image, ROIimage)`: configura sobre la IplImage pasada como argumento una región de interés como la indicada en el segundo argumento de la función. En el momento que se configura una ROI (Region Of Interest), cada

vez que se utiliza la imagen, no se trabaja con la imagen completa sino con la ROI únicamente.

- `cvResetImageROI(image)`: elimina la región de interés configurada en la IplImage pasada como argumento, quedando la imagen completa para trabajar con ella.

- `cvWaitKey(wait)`: función que espera tantos milisegundos como indica el argumento que se pasa a la función. Será utilizada para que la cámara llegue a su posición final antes de seguir procesando los siguientes *frames*.

Funciones relacionadas con el seguimiento por puntos SURF:

- `cvCircle(imagen, cvPoint(10, 10), 2, cvScalar(255, 0, 0), 1, 8, 0)`: función que dibuja un círculo en la imagen, en la posición (10, 10), con un radio de 2 píxeles y el color, grosor y tipo de trazo indicados en los argumentos de la función.

- `cvCreateMat(height, width, CV_8UC3)`: crea una matriz de tamaño y tipo de datos definido por los argumentos de la función.

- `cvGetReal2D(matriz, 10, 10)`: devuelve el valor que hay en la posición (10, 10) de la matriz pasada como argumento a la función.

- `cvCvtColor(image1, image2, CV_BGR2GRAY)`: convierte color de la image1 al espacio de color definido como argumento y se guarda en la image2.

- `cvSURFParams(500, 0)`: crea una estructura de parámetros para la función cvExtractSURF con el umbral que se va a utilizar y si se que quieren 64 descriptores o si se quieren 128 descriptores. En este ejemplo tendremos el umbral de 500 y 64 descriptores para la obtención de puntos SURF.

- `cvExtractSURF(image, 0, &imageKeypoints, &imageDescriptors, storage, params)`: la función extrae de la imagen los puntos de interés obtenidos mediante el algoritmo de extracción de características SURF inicializado por los parámetros pasados como argumentos. Los puntos de interés y los descriptores asociados se almacenan en storage y se

tiene un puntero a los puntos (imageKeypoints) y un puntero a los descriptores (imageDescriptors) para poder trabajar con ellos.

Funciones relacionadas con el seguimiento por histograma de color:

- `cvCreateHist(1, hdims, CV_HIST_ARRAY, hranges, 1)`: crea un histograma vacío reservando la memoria necesaria.

- `cvInRangeS( hsv, lower, upper, mask)`: crea una máscara con los píxeles de hsv cuyo valor de color se encuentra entre los valores lower y upper.

- `cvSplit( hsv, hue, 0, 0, 0)`: separa los canales de una imagen, en este caso la imagen es hsv y se guarda el primer canal en hue.

- `cvCalcHist(&hue, hist, 0)`: calcula el histograma de color de una imagen y lo guarda en la variable hist que se ha pasado como argumento a la función.

- `cvCalcBackProject(&hue_image, backproject, hist)`: calcula la retroproyección del histograma sobre el canal seleccionado de la imagen y guarda los datos en la variable backproject.

- `cvCamShift(backproject, track_window, cvTermCriteria( CV_TERMCRIT_EPS | CV_TERMCRIT_ITER, 10, 1), track_comp, track_box)`: busca dentro de la track_window la posición del objeto buscado haciendo uso de la retroproyección. Al finalizar las iteraciones determinadas como argumento, se devuelven tanto en track_comp como en track_box los resultados de la posición encontrada.

Para poder hacer uso de todas estas funciones primero es necesario instalar las librerías OpenCV en el PC e integrarlas en el proyecto. En el Anexo A se encuentra explicado el procedimiento a seguir.

Las librerías de VLC se han desestimado para ser utilizadas ya que no se puede conseguir el resultado esperado y se tienen las librerías de Gstreamer que parecen ser más funcionales para resolver el problema.

Para utilizar las librerías Gstreamer es necesaria su instalación en el PC y su integración en el proyecto que se está desarrollando en visual C++. En el Anexo B se encuentra explicado el proceso de instalación de Gstreamer, y en el Anexo C se explica el modo de utilizar y las funcionalidades de algunos módulos de Gstreamer que pueden ser interesantes para este proyecto.

Una vez integradas las librerías Gstreamer en el proyecto se han comenzado a realizar las pruebas pertienentes. Se ha conseguido un *pipeline* que envía un flujo de vídeo de test y otro flujo de audio de test a un punto remoto configurado previamente. Gracias a este *pipeline* tenemos resuelto la parte de emisión (la recepción se realiza a través del VLCMediaPlayer, que abriendo un medio de red en el protocolo y puerto de emisión, reproduce el vídeo o el audio), pero quedan aspectos que resolver para que sea totalmente resuelto el problema completo. Es necesario que el *pipeline* reciba el flujo, tanto de vídeo como de audio, en vez de un fichero de test, de la propia aplicación una vez recibido desde la cámara. En este punto entra en juego el objeto AppSrc de Gstreamer que se encarga de introducir estos flujos al *pipeline*, pero esto sólo es la idea ya que aún no se ha conseguido programar. También falta por resolver que el vídeo y el audio vayan sincronizados o dentro del mismo *pipeline* para que en recepción baste con un solo reproductor de medios.

Gstreamer también provee de elementos que se encargan de obtener el flujo de vídeo que se está viendo en la pantalla del ordenador, de manera que uniendo estos elementos y seleccionando unas veces el flujo de la cámara y otras veces el flujo de estos elementos, se puede llegar a conseguir que el profesor emita lo que le interesa en cada momento, ya sea una presentación a través del proyector o sea lo que está escribiendo en la pizarra.

2.3 Técnicas básicas de producción

La técnica de producción de vídeo que más se ajusta al problema que se quiere resolver en este proyecto es la utilización de un mezclador de fuentes. Su función es crear una salida master de una grabación de vídeo en tiempo real o de difusión. Por lo general, estos mezcladores se utilizan para eventos en vivo, o cualquier evento en el que múltiples fuentes deben ser mezcladas en tiempo real (a diferencia de la edición y post-producción). Precisamente este es el aspecto que prima en este proyecto, “deben de ser mezcladas en tiempo real”. Se cuenta con dos fuentes únicamente, el flujo de vídeo recibido desde la cámara y el flujo de vídeo que se recoge de la pantalla del ordenador. En este punto el

profesor deberá seleccionar que flujo es el que se introduce en el *stream* de vídeo para ser enviado a los puntos remotos.

La selección del flujo principal se deberá hacer añadiendo un nuevo módulo al *pipeline* de Gstreamer y será este el encargado de cambiar de uno a otro en función de la selección que se realice por parte del profesor en la interfaz gráfica de la aplicación.

El hecho de que la selección se tenga que hacer dentro del *pipeline* se debe a que el elemento que recoge el vídeo de la pantalla del ordenador se encuentra en el interior del *pipeline*, y por lo tanto tendrá que ser un módulo posterior al que le lleguen tanto este flujo de vídeo como el flujo de vídeo de la cámara (a través del modulo AppSrc) el que se encargue de dar paso a uno u otro flujo.

2.4 Seguimiento de objetos aislados en entornos de iluminación controlada

La necesidad de automatización de muchos de los procesos que nos encontramos en la vida real y la consiguiente sustitución de personas por máquinas, hace necesaria la evolución de las técnicas de visión artificial que dichas máquinas nos pueden ofrecer. La posibilidad de tener ordenadores potentes y la disponibilidad de cámaras de alta calidad han generado un gran interés hacia los algoritmos de seguimiento automático de objetos. Podemos encontrar varios campos hacia los que van orientados dichos algoritmos [1]: detección de objetos en movimiento, seguimiento de objetos *frame* a *frame* y el análisis del movimiento del objeto para reconocer su comportamiento.

Para poder realizar el seguimiento de un objeto de interés, habrá que fijarse tanto en su forma como en su apariencia, para poder decidir correctamente si es él y no otro objeto el que aparece en la escena de seguimiento, y además poder decidir acerca de su situación en la escena.

Debido a la enorme amplitud de este campo de estudio, se ha optado por referenciar únicamente las técnicas más adecuadas para resolver el problema de este proyecto. Las representaciones del objeto que más cercanas a este proyecto se encuentran en cuanto a forma se refiere [1] son las siguientes:

- **Puntos:** el objeto se puede representar por un punto llamado centroide o por un conjunto de puntos representativos del objeto. La representación por puntos se usa en seguimiento de objetos que ocupen un pequeño espacio en la escena de seguimiento.

- **Formas geométricas primitivas:** se representa el objeto por un rectángulo, una elipse, o un conjunto de varias formas de las anteriores representando cada una de ellas una parte diferente del objeto. Este tipo de representación se utiliza para realizar el seguimiento de objetos que no sean rígidos.

- **Silueta y contorno:** el contorno representa la frontera del objeto y la silueta es la región interior del contorno. La representación por silueta y contorno se utilizará en seguimientos de objetos complejos.

En cuanto a la apariencia del objeto se refiere la única característica que nos interesa de todas las que se citan en [1] es la siguiente:

- **Densidad de probabilidad de la apariencia del objeto:** esta densidad de probabilidad puede ser estimada de varias maneras. En este proyecto nos interesa el histograma de color del objeto. Esta característica puede ser obtenida de regiones especificadas por los modelos de forma del objeto.

Una vez decidido el modelo de representación del objeto, el algoritmo de seguimiento se encargará de localizar la posición en la que se encuentra en cada *frame* del vídeo en función de la representación. Por lo tanto habrá distintos algoritmos de detección del objeto, uno para cada tipo de representación. En nuestro caso se utilizará una representación por puntos del objeto y por lo tanto la estructura del algoritmo de seguimiento será, en línea con lo expuesto en [5], la siguiente:

1) Inicialización:

a) Extracción de los puntos de interés del objeto en el primer *frame*: tras seleccionar el objeto a seguir, se obtiene un pequeño *frame* del que se obtienen los puntos de interés (o puntos característicos) del mismo y sus correspondientes descriptores.

b) **Obtención del cuadro que contiene al objeto:** al tener el objeto que se ha seleccionado tenemos también la región en la que se encuentra situado dentro del *frame* completo.

2) **Bucle principal (para cada nuevo *frame*):**

a) **Extracción de los puntos de interés del nuevo *frame*:** en cada nuevo *frame* que se tenga habrá que obtener los puntos de interés de la imagen completa y sus correspondientes descriptores.

b) **Comparación de los nuevos puntos con los puntos del objeto:** se comparan los descriptores de los puntos del objeto obtenidos anteriormente y se crea una asociación con los más similares de la nueva imagen completa (*matching* de puntos).

c) **Obtención del nuevo cuadro que contiene al objeto y sustitución del anterior:** una vez se tienen las asociaciones se sabe donde se encuentra situado el objeto en el nuevo *frame* con lo que se puede obtener un cuadro que lo contenga. Este cuadro obtenido sustituirá al que teníamos anteriormente guardado.

d) **Actualización del objeto y su modelado por puntos:** se obtiene el nuevo objeto del cuadro obtenido y sus correspondientes puntos de interés guardándolos para el procesado del siguiente *frame* que se obtenga.

Cuando se ha terminado de realizar el paso 2.c se está en disposición de estimar el movimiento que ha sufrido el objeto dentro de la imagen respecto del *frame* anterior. Para esto, una opción es calcular el centro de gravedad de los puntos que se han referenciado en la nueva imagen y en él se centrará el cuadrado que contiene al nuevo objeto. Calculando el desplazamiento que sufre el centro de gravedad se tiene una medida del movimiento global del objeto respecto de la imagen. Con la obtención del movimiento que realiza el objeto *frame* a *frame* queda finalizado el trabajo del algoritmo para seguimiento de objetos.

Es a partir de este momento donde entra en juego la cámara PTZ, que es la utilizada en nuestro caso. El movimiento realizado por el objeto en cada *frame* deberá ser compensado por la cámara realizando el movimiento necesario para dejar centrado el objeto en la imagen.

La extracción de puntos de interés (*keypoints*) y sus correspondientes descriptores será explicada en la sección 2.6.1 que trata sobre Extracción de puntos por SURF.

También se utilizará como forma complementaria de detección de la posición del objeto otro algoritmo basado en la apariencia del objeto, haciendo una búsqueda por histograma de color. Este algoritmo será explicado en la sección 2.6.2.

2.5 Recopilación de software gratuito

Se ha buscado *software* gratuito que se podría utilizar para ciertas funcionalidades que buscamos en nuestra aplicación. El hecho de tener un *software* en paralelo con la aplicación que se va a desarrollar no es del todo funcional, no obstante se va a analizar cada una de las aplicaciones encontradas y clasificar en función de su funcionalidad.

2.5.1 Control de escritorio remoto

Se ha buscado *software* de control de escritorio remoto pensando en la posibilidad de dar la clase con un profesor remoto. De esta manera el profesor que no pueda asistir al aula tendrá la posibilidad de impartir la clase desde el lugar en que se encuentre. Este tipo de aplicación le permitirá controlar el PC del aula desde el suyo propio y realizar cualquier tipo de presentación, manejo de cualquier programa o incluso resolver cualquier duda desde el chat que proporciona alguna de las aplicaciones. El único aspecto desfavorable que tienen estas aplicaciones a la hora de utilizarlas para este cometido, es que es necesaria la participación de un encargado de encender el PC del aula y lanzar la aplicación de control remoto. También controlará el chat en el caso de tenerlo la aplicación.

Como aplicaciones de control de escritorio remoto se han encontrado las siguientes:

- **ScreenStream:** a través de este programa podemos mostrar al resto de usuarios lo que pasa en nuestro escritorio con sólo poner en el navegador la dirección IP del mismo. También podremos mostrarles el contenido grabado por la *webcam* y el audio del micrófono. No tiene una interfaz que permita la conexión con la cámara IP que utilizamos, solamente reconoce cámaras instaladas en el ordenador. No obstante se podría lanzar la aplicación de visualización del flujo de la cámara y utilizando screenstream emitirlo a través del escritorio.

En la Figura 2-1 se observa la aplicación cuando comienza a emitir lo que sucede en el escritorio y en la Figura 2-2 se observa el navegador una vez a cargado la dirección IP correspondiente y nos muestra la ventana que está activa en el ordenador remoto. En este caso se ha querido mostrado una manera de visualizar lo que la cámara está grabando a través de la aplicación básica de visualización.

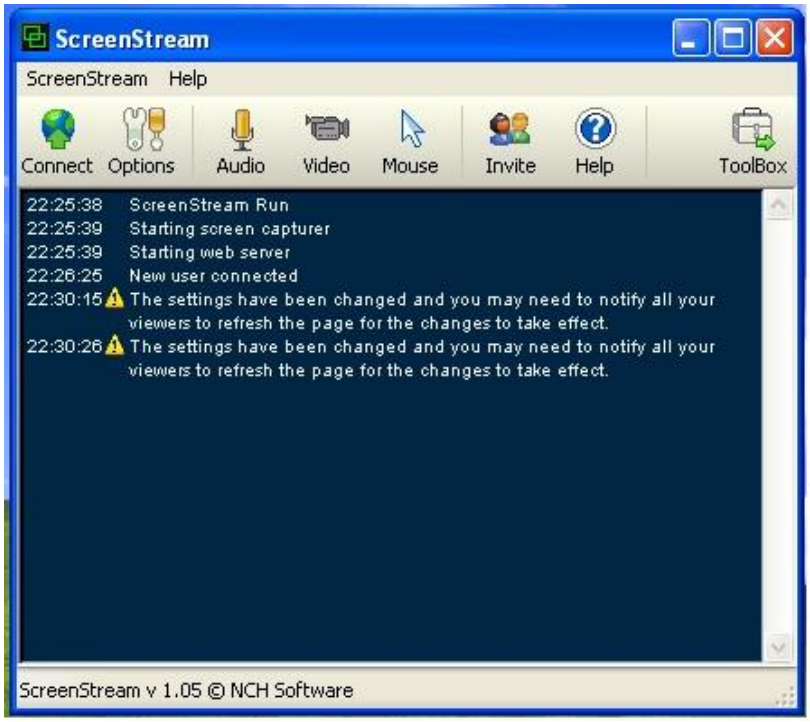


Figura 2-1: Emisión ScreenStream.

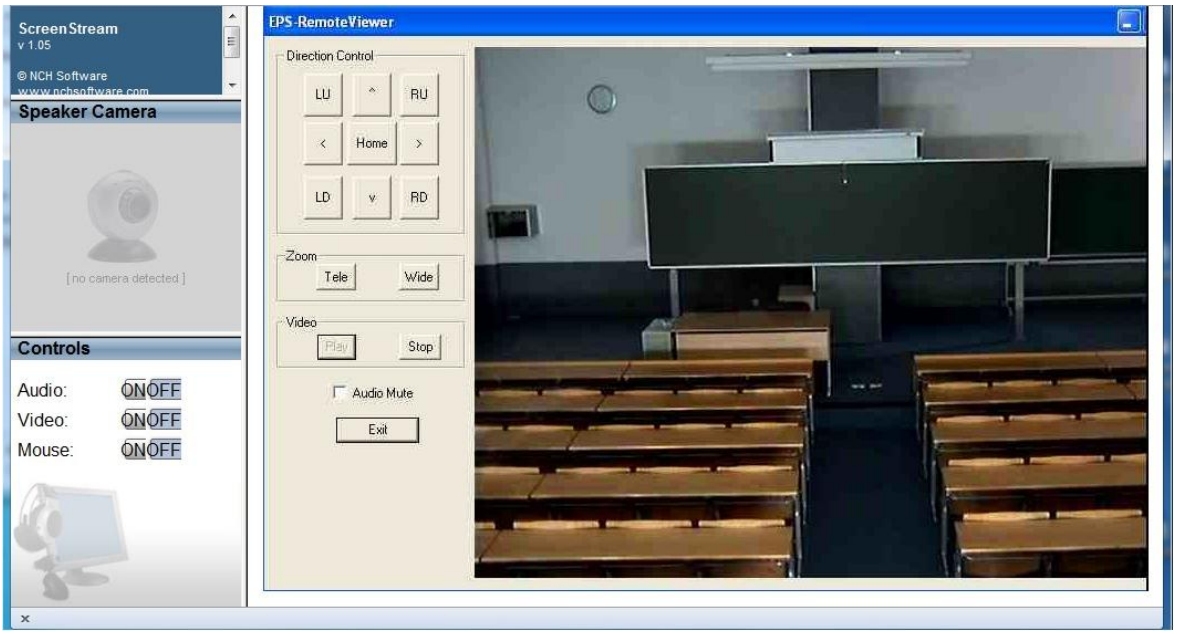


Figura 2-2: Recepción ScreenStream.

- **Logmein:** utilizando esta aplicación podemos obtener un control total sobre el PC remoto. Una vez instalada la aplicación en dicho PC, podremos acceder a él desde un navegador teniendo control sobre el ratón, lanzamiento de aplicaciones, transferencia de archivo y chat, obteniendo así una total funcionalidad. Gracias a esta aplicación podremos impartir una clase en el aula con el profesor en un PC remoto. También nos permite compartir únicamente el escritorio con varias personas invitándolas previamente, con lo que obtendríamos la funcionalidad tanto de profesor remoto como alumno remoto. Por último, si lanzamos la aplicación de visualización del flujo de vídeo de la cámara del aula, y compartimos el escritorio con alumnos que no hayan podido asistir al aula, estos alumnos podrán disfrutar de la clase y si surgiese cualquier duda podrían comunicarse a través del chat con el profesor o con la persona encargada del control del PC.

  En la Figura 2-3 observamos cómo a través del navegador obtenemos el control total del PC remoto y podemos visualizar el flujo de la cámara, habiendo lanzado previamente la aplicación de visualización.

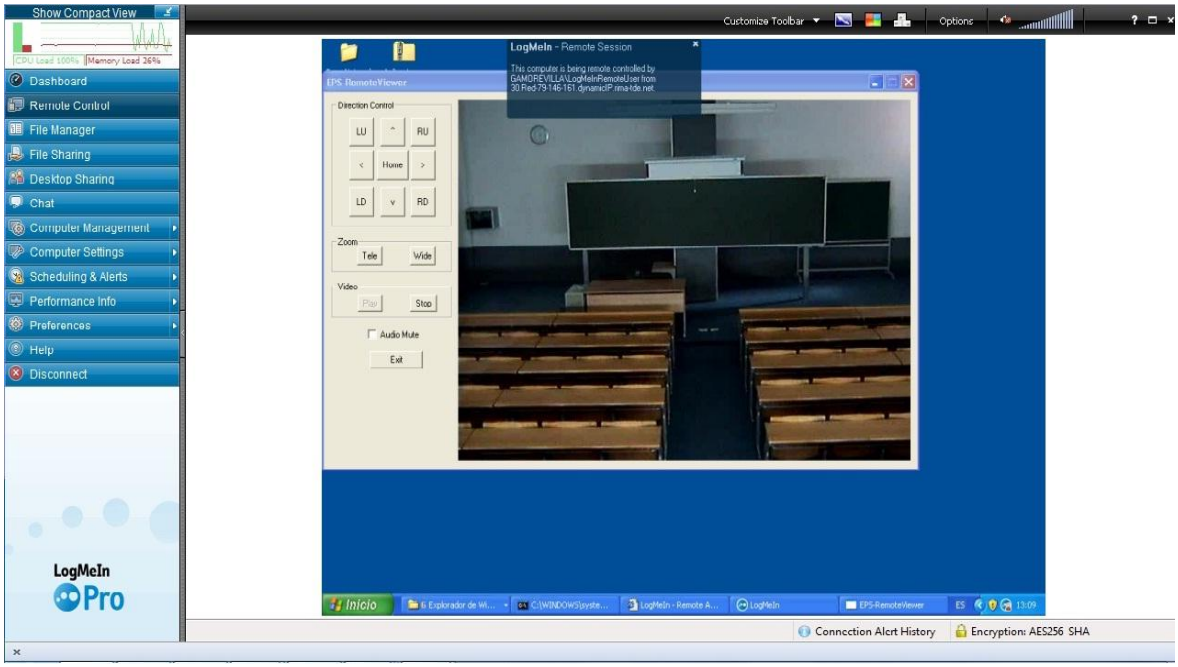


Figura 2-3: Control remoto Logmein.

- **Zsoporte:** a través de esta aplicación obtendremos el control total sobre el ordenador remoto. Configurando los permisos correctamente en el cliente, tendremos las funcionalidades de control del teclado y del ratón, lanzamiento de

aplicaciones, envío y recepción de ficheros, y también cuenta con un chat integrado a través del cual podremos comunicarnos con el usuario para requerirle cualquier acción que sea necesaria, o simplemente para comentarle las acciones que estamos llevando a cabo. Es una aplicación que se ajustaría a nuestro proyecto desde el punto de vista de que el profesor podría estar en su casa manejando el ordenador del aula y lo que se muestra en el proyector. Desde el punto de vista de videoconferencia no se ajustaría ya que para que los alumnos pudiesen ver el flujo de la cámara tendrían que ser ellos los que tuviesen el control remoto del PC del aula y además solamente podría verlo un alumno.

En la Figura 2-4 se puede ver el panel de control de Zsoporte. Cuando recibimos una conexión entrante del cliente, como en este caso es la del PC con IP 102.168.1.36, podemos realizar varias acciones que son la de control del escritorio remoto, exploración de archivos y la de inicio del chat. Una vez seleccionada la opción de ver escritorio remoto pasamos a ver la siguiente ventana, que se muestra en la Figura 2-5, y moviendo el ratón sobre ella podemos realizar el lanzamiento de cualquier aplicación que se encuentre instalada en el PC, e incluso instalar nuevas aplicaciones que necesitemos.

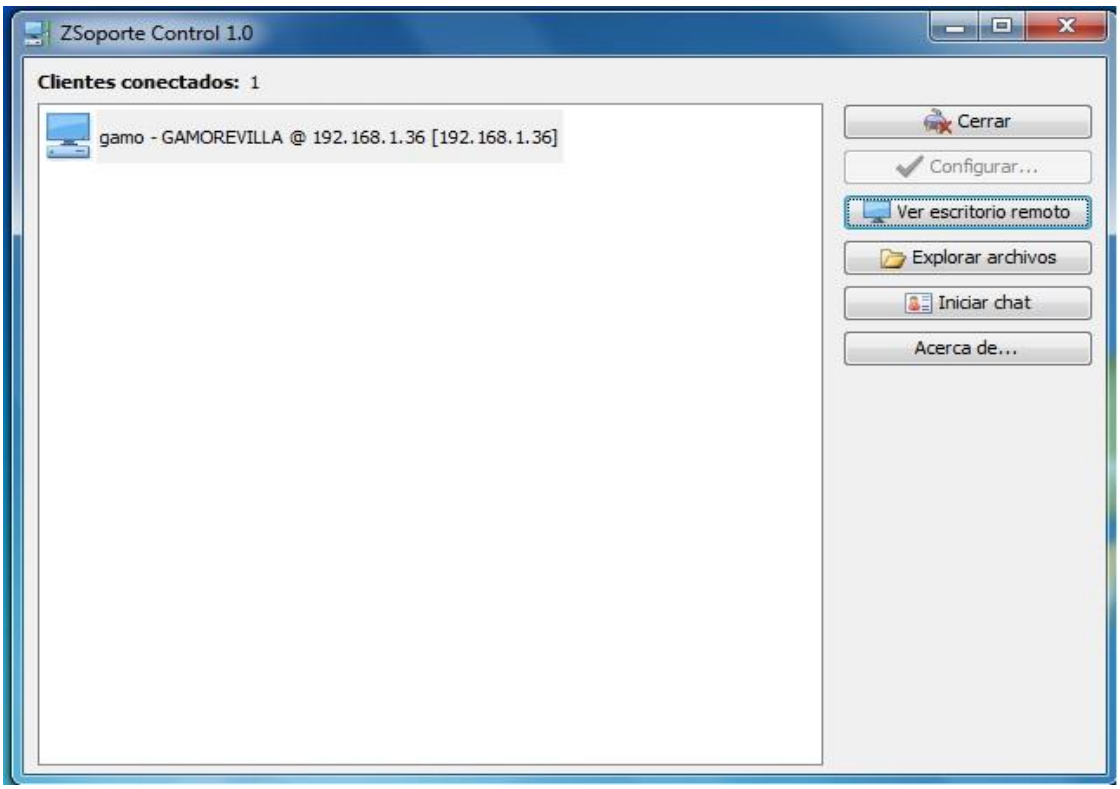


Figura 2-4: Panel de control de Zsoporte.

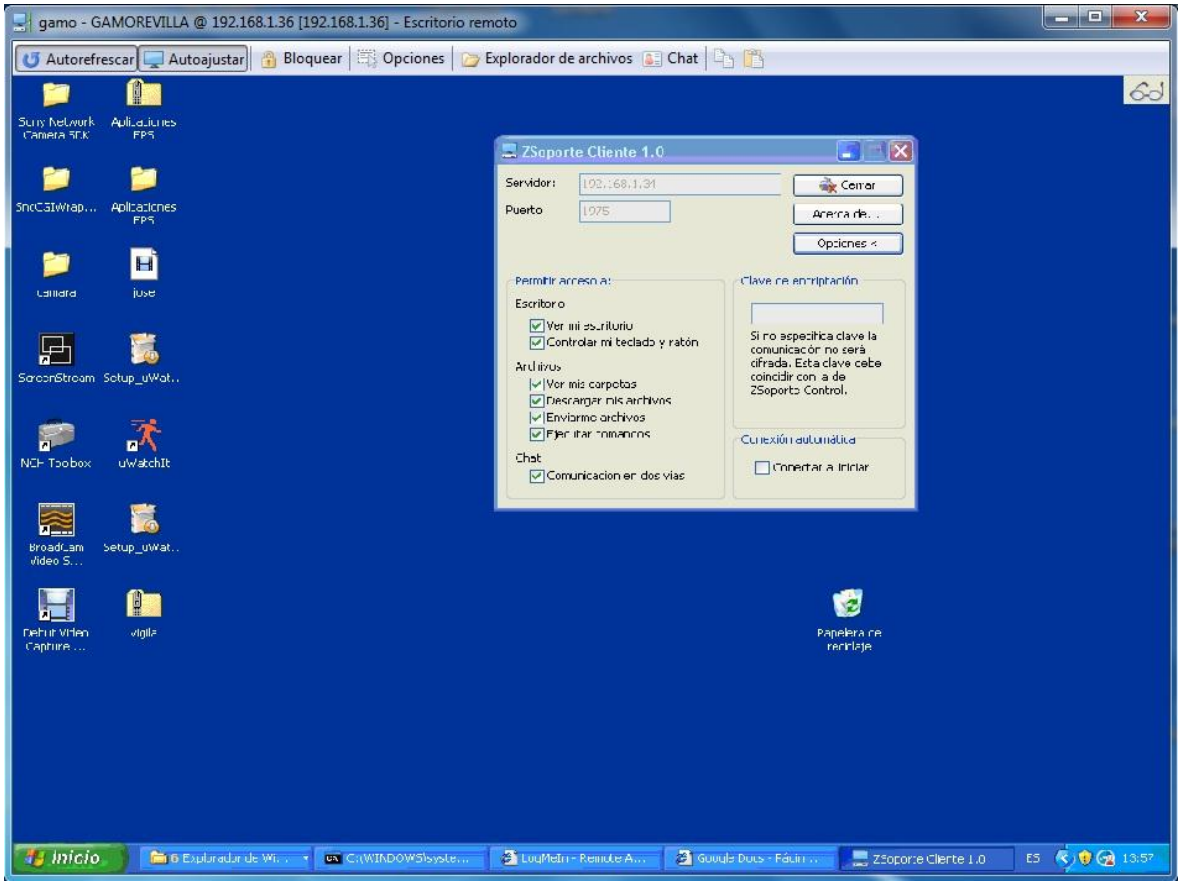


Figura 2-5: Control remoto de Zsoporte.

- **TeamViewer:** esta aplicación nos da la posibilidad de conectarnos a cualquier otro ordenador que también la tenga instalada y a través de un ID y un código de acceso, podremos tener el control total del PC remoto. Podremos lanzar cualquier aplicación que sea necesaria en la explicación o realizar cualquier función de instalación o mantenimiento del PC. También tendremos la posibilidad de realizar la conexión en modo presentación de manera que podremos ir explicando a los alumnos cada paso que vamos dando.

  Esta aplicación se podría incluir en las aplicaciones de videoconferencia ya que nos da la posibilidad de realizar el envío del audio y del vídeo grabado por la cámara del ordenador tanto en el sentido cliente-servidor como servidor-cliente. El aspecto negativo es que no podremos realizar una conexión con la cámara IP del aula ya que no tiene la posibilidad de recibir el flujo de vídeo, aunque al igual que con las otras aplicaciones podremos lanzar la aplicación desarrollada para visualización del aula.

En las siguientes figuras se muestran ciertas funcionalidades que la aplicación nos da. En la Figura 2-6 se ve cómo a través del ID de asociado tendremos acceso al ordenador remoto en modo control remoto, transferencia de archivo o VPN (tenemos que conocer el ID y la contraseña de conexión del ordenador al que queremos conectarnos).

Figura 2-6: TeamViewer interfaz de inicio.

En la Figura 2-7 se observa cómo se pueden ir lanzando las distintas aplicaciones, abrir carpetas para buscar archivos y todas las funcionalidades que se tienen en un ordenador local. También se puede observar el chat, la transferencia de audio y la transferencia del vídeo grabado por la cámara instalada en el PC.

Finalmente en la Figura 2-8 se puede ver la aplicación en modo presentación en vez del modo de control remoto. Tendremos un ordenador que será el que haga la presentación y tendrá la opción de utilizar una pizarra, para ir dibujando sobre el escritorio, señalar con un puntero las distintas carpetas o archivos a los que se va haciendo referencia en la videoconferencia, o incluso colocar bocadillos explicativos de qué se va haciendo en cada paso o para qué sirven ciertas aplicaciones. Este último aspecto se ve muy claramente en la Figura 2-8, un bocadillo que explica que accediendo a esa carpeta se podrá lanzar la aplicación de control de escritorio remoto.

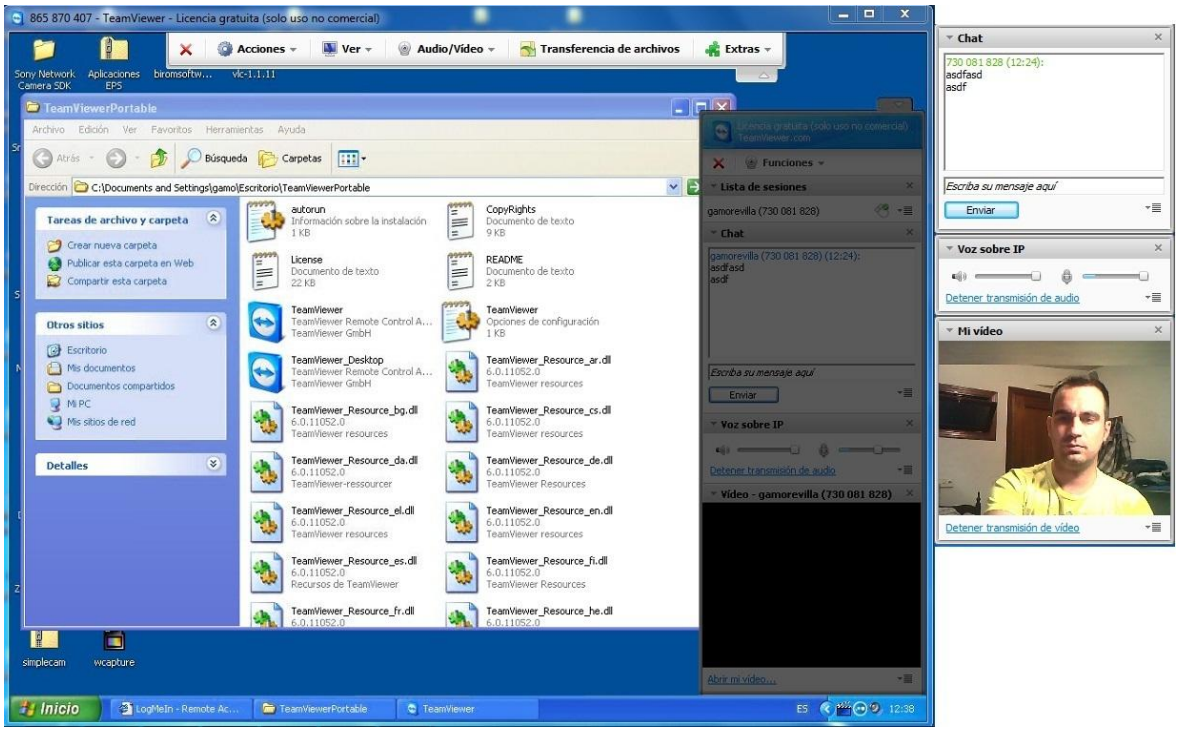


Figura 2-7: TeamViewer, control remoto y videoconferencia.

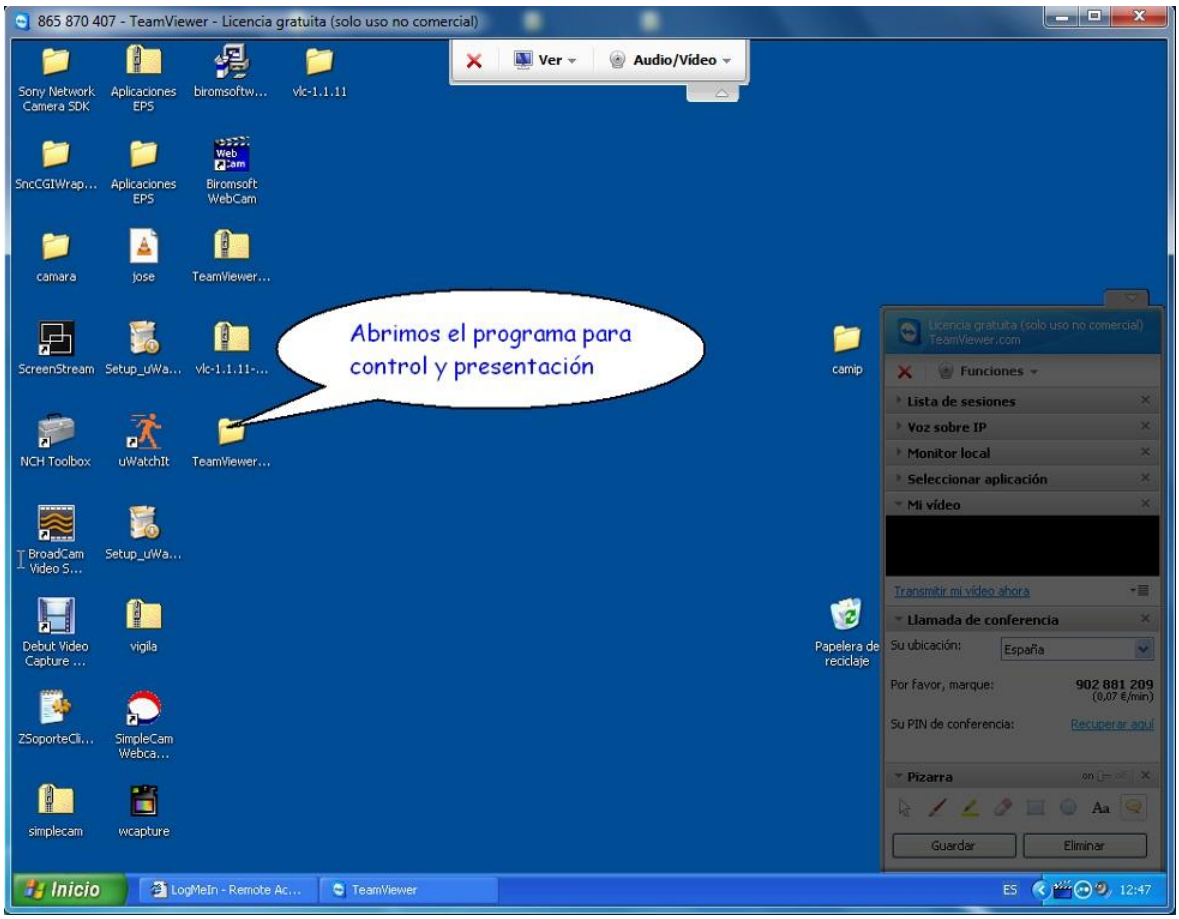


Figura 2-8: TeamViewer modo presentación.

2.5.2 Videoconferencia

Muchas son las aplicaciones que nos permiten realizar una videoconferencia entre dos ordenadores, pero no hay ninguna que tenga implementada una conexión con una cámara como la que vamos a utilizar. Esto nos limita a la hora de usar el flujo de vídeo de la cámara y enviarlo al otro ordenador. En las pruebas realizadas con cada una de ellas se ha conseguido que alguna pueda coger el flujo de vídeo desde la dirección "http://host-camara/image" y lo emita. El problema que se ha encontrado en esta prueba es que el audio que la cámara graba se pierde. No obstante, la cámara cuenta con un servidor de MPEG4, que se encuentra en la dirección "http://host-camara/mpeg4", en el que supuestamente se recibiría el flujo de audio-vídeo completo, pero no se ha conseguido obtener respuesta por parte de la cámara desde ninguna de las aplicaciones.

Se ha buscado *software* gratuito que puede ajustarse a nuestras necesidades de videoconferencia y las aplicaciones que merece la pena comentar son las siguientes:

- **SimpleCam**: a través de esta aplicación se publica en internet el flujo de vídeo de la cámara web permitiendo el acceso al resto de usuarios a través de la dirección IP del ordenador que hace de servidor. No tiene una interfaz que permita la conexión con la cámara IP que utilizamos, solamente reconoce cámaras instaladas en el ordenador por lo que no será válida para resolver nuestro problema.

- **Uwatchit:** con esta aplicación podemos visualizar el flujo de vídeo de varias cámaras tanto IP como cualquier otra cámara conectada al PC para poder vigilar varias zonas al mismo tiempo. No es válida para usarla en videoconferencia ya que no tiene la función de servidor *streaming* y la conexión con nuestra cámara no es posible, por lo que se desestima su funcionalidad para nuestro problema.

- **Vigila 8:** esta aplicación permite la monitorización de hasta 8 cámaras IP, pero al igual que la aplicación anterior no es válida por la falta del servidor *streaming* integrado y por no ser posible la conexión con la cámara que vamos a utilizar.

- **Wcapture:** permite publicar el contenido grabado por la cámara y permite un acceso al servidor a través de http. No tiene una interfaz que permita la conexión con la cámara IP que utilizamos, únicamente reconoce cámaras instaladas en el ordenador.

- **Biromsoft webcam:** captura de imágenes a través de la cámara conectada al ordenador y nos permite el acceso al contenido a través del navegador. No tiene una interfaz que permita la conexión con la cámara IP que utilizamos, solamente reconoce cámaras instaladas en el ordenador por lo que no será útil para este proyecto.

- **Yawcam:** recibe el flujo de vídeo de una cámara web, tanto si está instalada en el ordenador como si es una cámara IP como la que utilizamos en este proyecto. Una vez recibido el flujo lo podemos emitir en distintos protocolos, y hacer que sea accesible para el resto de usuarios a través de la dirección IP del ordenador. De esta manera se convierte la aplicación en un servidor de *streaming* de vídeo con control de usuarios y que se adecua al problema que queremos resolver. El problema viene en el momento que también queremos enviar el audio de la cámara, que como comentaba anteriormente, sería necesario conectar la aplicación al servidor MPEG4 de la cámara, pero no se ha conseguido. Únicamente es capaz de obtener el flujo de vídeo del servidor de imágenes de la cámara.

  En la Figura 2-9 se muestra la interfaz de esta aplicación con la función de *stream* activada, que será la que nos permite visualizar en un navegador el flujo de vídeo que estamos recibiendo. También observamos una ventana de previsualización del flujo que estamos emitiendo hacia el navegador.

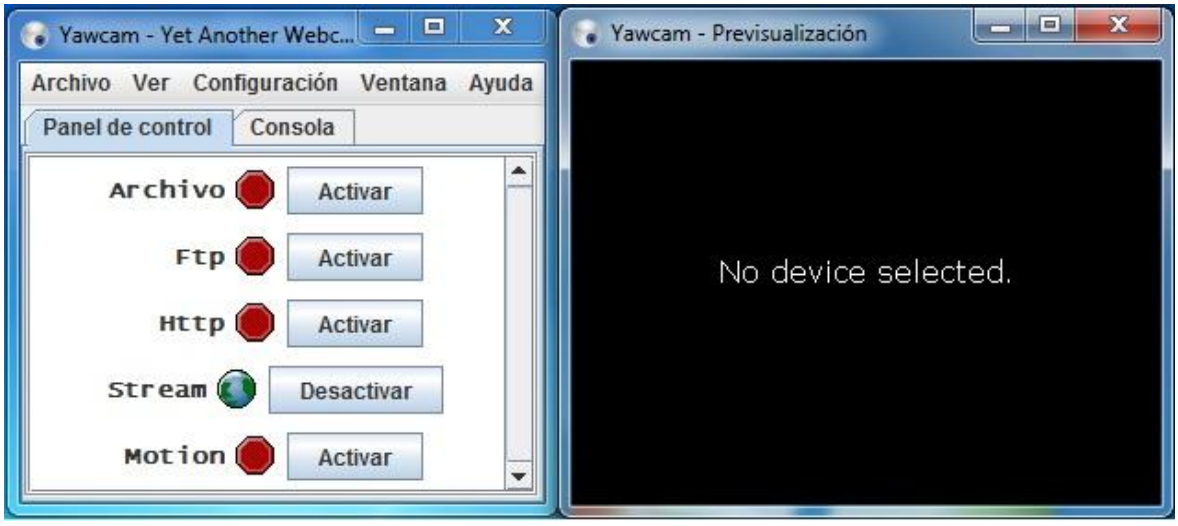


Figura 2-9: Yawcam.

- **Webcam7:** al igual que la aplicación anterior permite la conexión a la cámara con la que trabajamos y también hace de servidor *streaming* con control de usuarios. Los resultados obtenidos no son buenos, ya que debido a la interfaz de conexión con la cámara, la emisión se hace a una *frame rate* muy inferior a la deseada.

  En la Figura 2-10 se muestra la interfaz gráfica de webcam7 observando que se pueden poner varias fuentes e ir seleccionando la que nos interesa cada vez, ya sea la cámara principal, otras cámaras o ficheros de vídeo grabados anteriormente.

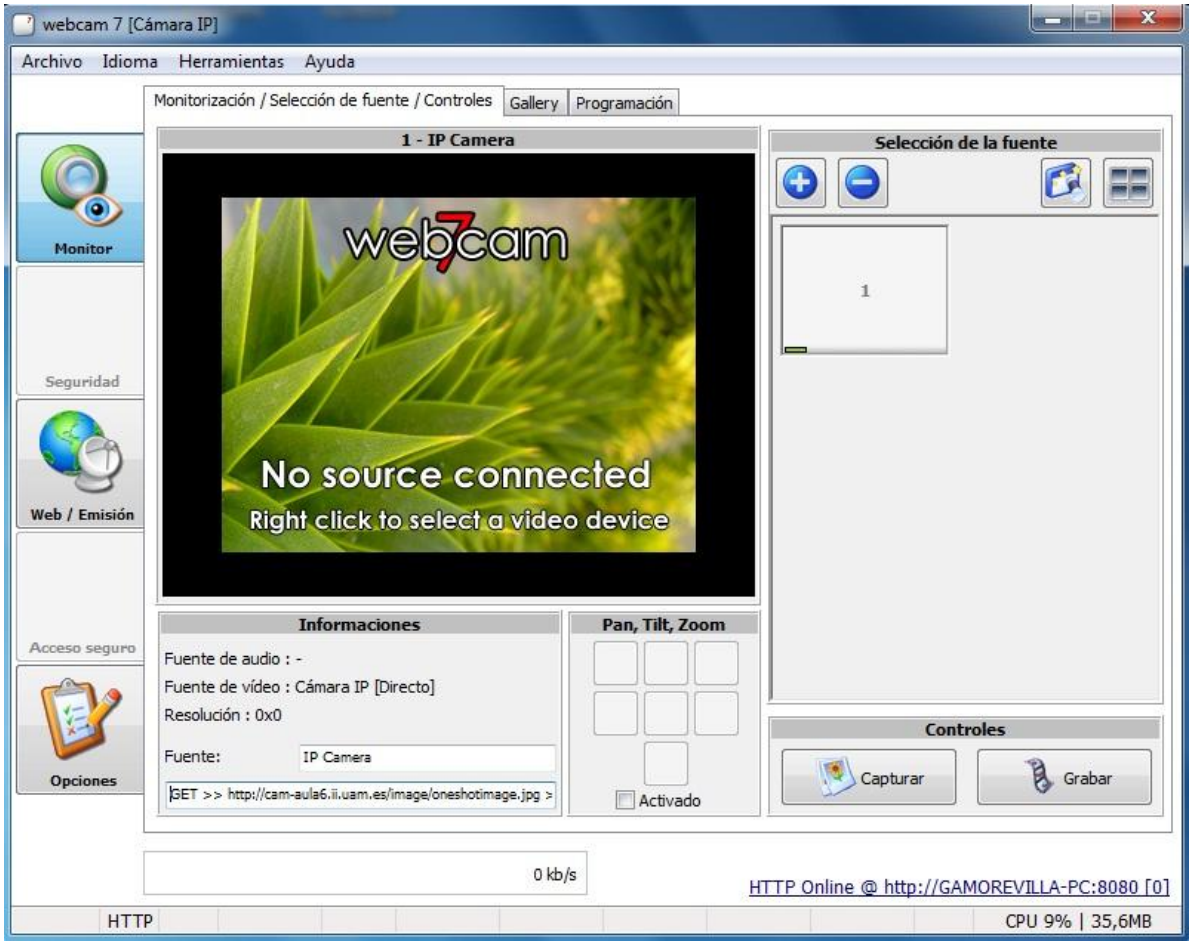


Figura 2-10: Webcam 7.

- **VLC Media Player:** esta aplicación es un reproductor multimedia que podremos utilizar para la emisión del flujo de vídeo ya que cuenta con varios protocolos de *streaming*. Conectándonos a la cámara IP obtendremos el flujo deseado y también lo podremos emitir a través de varios protocolos diferentes a distintas IP´s determinadas, o en multicast. También tendremos la posibilidad de codificar el

vídeo en otro formato que nos interese más para reducir la carga de la red y mejorar su reproducción en remoto.

Al igual que alguna de las aplicaciones estudiadas anteriormente, esta aplicación no nos permite la conexión con el servidor MPEG4 de la cámara, quedando su funcionalidad reducida a la emisión únicamente del vídeo (para emitir el audio habría que utilizar un método paralelo, pero no se garantizaría la sincronización de audio y vídeo en recepción).

En la Figura 2-11 se ilustra la interfaz gráfica principal de la aplicación, mientras que en la Figura 2-12 observamos cómo se puede conectar la aplicación con la cámara para poder recibir el flujo de vídeo que se está grabando.

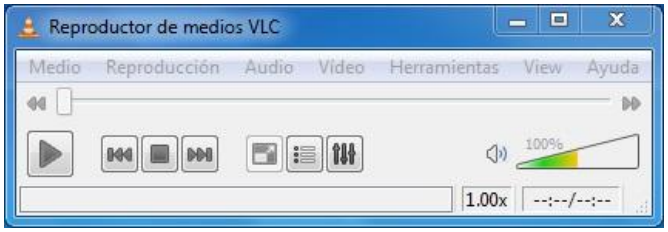


Figura 2-11: VLC Media Player.

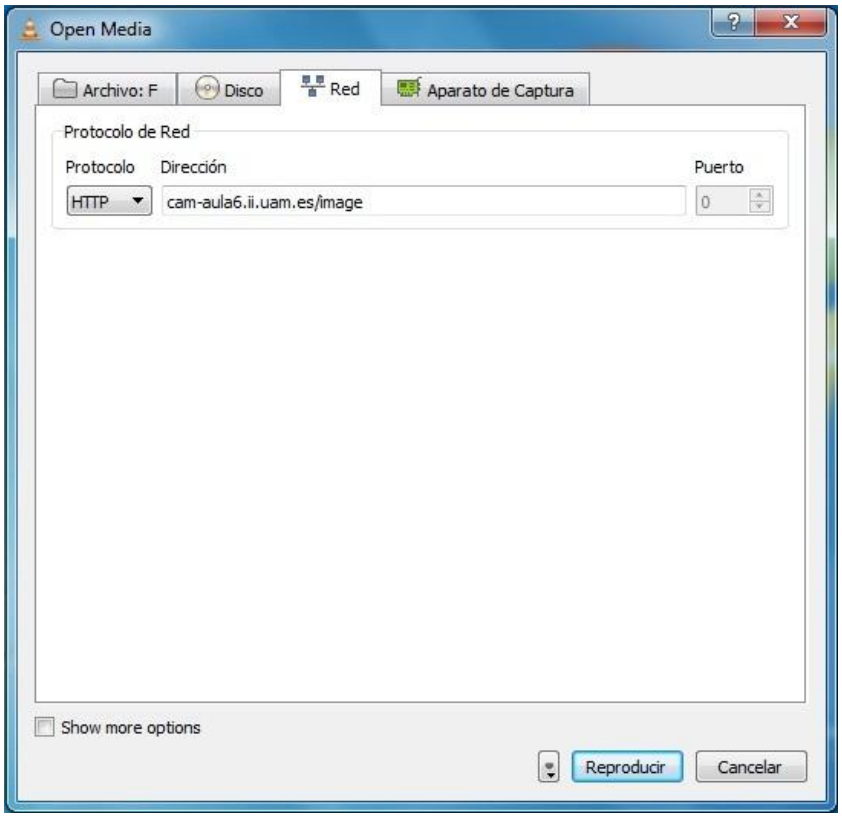


Figura 2-12: VLC conexión con la cámara.

- **Skype:** aplicación de videoconferencia con otros usuarios. Nos permite enviar y recibir el vídeo capturado por las cámaras de los ordenadores, pero no nos permite una conexión a la cámara IP que estamos utilizando, por ello no nos valdrá. También nos permite compartir el escritorio y mostrar al otro usuario las acciones que estamos llevando a cabo. Por lo tanto la funcionalidad de la aplicación que sería válida para este proyecto será la de compartición del escritorio previo lanzamiento de la aplicación de visualización del aula, o para mostrar en el aula el escritorio de un profesor remoto. También se podría utilizar en paralelo con las aplicaciones que sólo son capaces de conectarse al flujo de vídeo sin audio, y poder así también enviar el audio en paralelo.

  En la Figura 2-13 se muestra la interfaz gráfica inicial de skype.

Figura 2-13: Skype.

- **TeamViewer:** aunque la aplicación ha sido ya estudiada en el apartado de control de escritorio remoto, puede ser incluida perfectamente en este apartado ya que da una gran funcionalidad para videoconferencia. Ya se ha dicho que no es capaz de

conectarse con la cámara IP del aula, pero a través del lanzamiento de la aplicación de visualización, se podrá hacer uso de ese flujo de vídeo y los demás alumnos podrán verlo.

Gracias a esta aplicación podremos resolver una nueva situación, que el profesor esté en un PC remoto y que al mismo tiempo haya alumnos que no hayan podido asistir al aula. El profesor tendrá que poner la aplicación en modo presentación e invitar a los distintos alumnos para que puedan conectarse, como también será necesario invitar al PC del aula para que los alumnos que hayan asistido a clase puedan ver en el proyector la presentación que realiza el profesor. Otra forma de resolver este aspecto será que el profesor realice una conexión en modo control remoto con el PC del aula, y a través de esa conexión controlar ese PC. Tendrá que lanzar una nueva conexión en modo presentación en el PC del aula e invitar desde ese PC a los alumnos que lo necesiten. De esta manera la presentación se realiza desde el aula, pero controlada desde el PC del profesor, y se envía a los alumnos que no estén.

- **CamApplication:** es la aplicación que se está desarrollando en este proyecto, la cual nos permite un control total sobre la cámara y la recepción y visualización del vídeo en la propia aplicación. Llevará integrado un módulo de seguimiento automático del profesor a través del procesado del flujo de vídeo recibido, que se encargará de enviar las instrucciones de movimiento que sean pertinentes para mantener siempre enfocado al profesor en uno de los modos de producción automática. Para la consecución de la funcionalidad de *streaming* de vídeo se añadirá en la medida de lo posible un nuevo módulo basado en las librerías de Gstreamer que nos permitirá tanto transmitir el flujo de vídeo a usuarios remotos como seleccionar lo que se quiere enviar, ya sea el flujo de la cámara o sea el escritorio del PC con alguna presentación que se esté haciendo en el aula.

  En la Figura 2-14 se muestra la interfaz gráfica de la aplicación que se ha desarrollado. A la izquierda se pueden observar los botones de control del movimiento de la cámara así como los de *zoom*, vemos los campos para los datos de conexión con la cámara y los botones de comienzo del *streaming* del flujo de vídeo. Abajo podemos observar los botones de todas las posiciones

predeterminadas que se hayan configurado y los de activación y finalización del seguimiento automático del profesor.

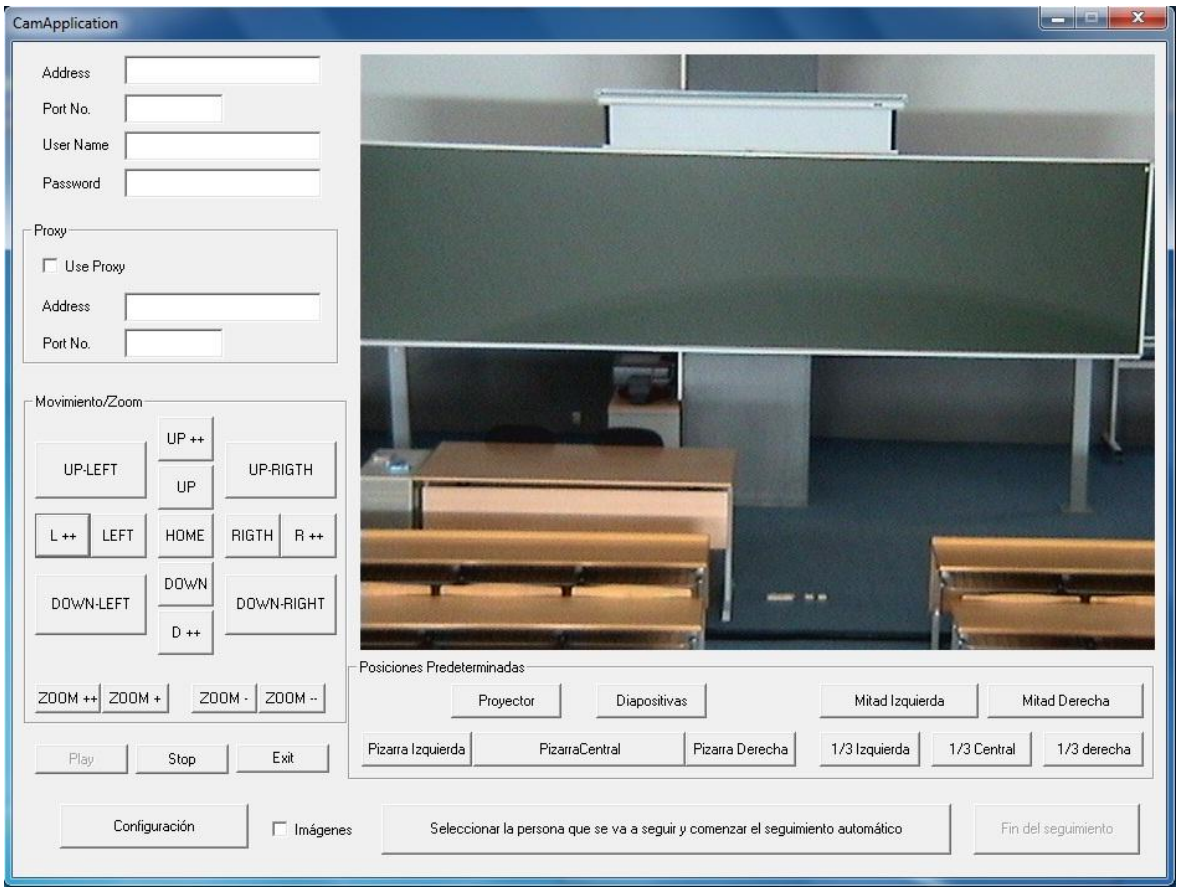


Figura 2-14: CamApplication.

- **EPS-RemoteViewer:** es una aplicación desarrollada en este proyecto que a su vez es una simplificación de CamApplication. Gracias a esta aplicación podremos conseguir recibir el flujo de vídeo de las cámaras en cualquier punto remoto, pudiendo también controlarlas en caso de tener los permisos necesarios en la fecha indicada. Facilitando esta aplicación junto con ciertos permisos (solo visualización y fecha de la sesión) a los alumnos, se podrá conseguir que el día que no puedan asistir al aula, el profesor les facilite esos permisos para que desde su casa puedan visualizar todo lo que el profesor quiera mostrarles.

  En la Figura 2-15 se muestra la aplicación con los botones de control de movimiento, aunque no serán funcionales si los permisos no están activados. Tanto los datos de conexión como la fecha o rango de fechas en los que poder conectarse a la cámara se encuentran cifrados en un archivo de configuración. Este archivo se

crea desde la aplicación creada también en este proyecto EPS-CfgGenerator la cual se puede observar en la Figura 2-16.

Figura 2-15: EPS-RemoteViewer.

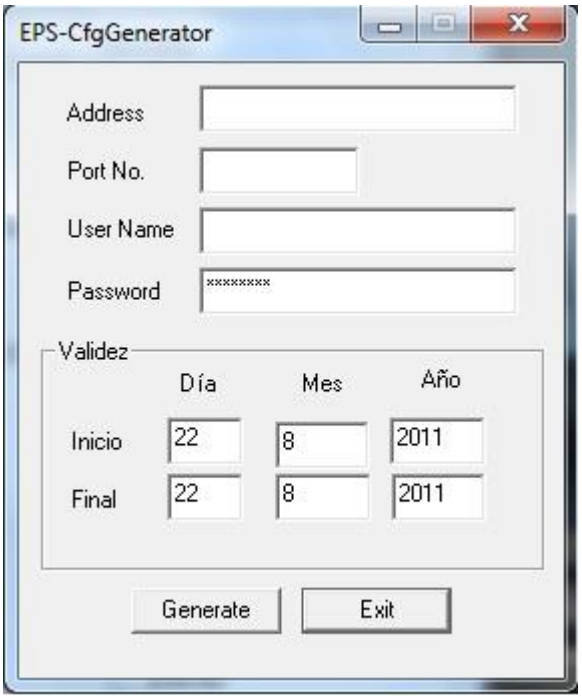


Figura 2-16: EPS-CfgGenerator.

2.5.3 Estudio de su adecuación para uso en docencia en aula con profesor remoto

Para el uso en docencia en aula con profesor remoto será necesario al menos tener una aplicación que tenga la funcionalidad de controlar el PC del aula, a no ser que a través de videoconferencia se comunique con un encargado del manejo del PC (lo cual es un poco pesado teniendo la otra posibilidad). Las aplicaciones más adecuadas para este cometido son Logmein, Zsoporte y TeamViewer que nos dan un control total sobre el PC remoto, sobresaliendo por los resultados y la funcionalidad TeamViewer.

Para usar estas aplicaciones es necesario seguir un cierto protocolo. Tanto en Logmein como en Zsoporte como en TeamViewer es necesario que algún encargado del aula encienda el PC y lance la aplicación Logmein, el cliente de Zsoporte o la aplicación TeamViewer (en caso de no estar configuradas en el arranque del PC).

A partir de aquí, si utilizamos Logmein, el profesor tendrá el control del PC desde el navegador del suyo, conectándose a la página http://www.logmein.com e introduciendo su cuenta aparecerán los distintos ordenadores sobre los que tiene control. Seleccionará el del aula correspondiente y comenzará a manejarlo. Podrá lanzar cualquier tipo de aplicación ya sea para una presentación o para mostrar un ejemplo de programación. También es interesante la posibilidad de que el profesor lance en paralelo la aplicación de control de la cámara para poder tener él una realimentación de lo que está sucediendo en el aula y de lo que se está viendo en el proyector. Así mismo, a través de Skype podría mostrar su cara a los alumnos y transmitir el audio para que los alumnos puedan escucharlo en el aula.

Si se usa Zsoporte sería necesario que el encargado introdujese en el cliente la dirección del PC del profesor para que a través de la aplicación servidor pueda obtener el control del PC del aula. Una vez conectadas las aplicaciones cliente y servidor el profesor puede lanzar cualquier tipo de aplicación en el aula y seguirá el mismo proceso anteriormente mencionado.

Si se utiliza TeamViewer, el encargado deberá comunicarnos el ID del PC del aula y la contraseña de conexión. A partir de este momento tendremos el control del PC del aula excepto en aspectos de configuración de TeamViewer, que serán ocultos a ese control. Por lo tanto si queremos además de controlarlo realizar una presentación, el encargado tendrá

que poner TeamViewer también en modo presentación y realizar las invitaciones necesarias.

Con Logmein y Zsoporte se tiene un sistema de control del escritorio remoto y videoconferencia que cumple todas las funcionalidades que cualquier profesor podría realizar estando en el aula. El único aspecto que no se puede simular es la escritura en la pizarra, pero valiéndonos de cualquier otra aplicación de dibujo quedaría resuelto. En cuanto a que el profesor reciba las dudas de los alumnos se puede resolver a través del micro del aula, que al estar conectado a la cámara, el profesor lo escuchará en la aplicación desarrollada para visualización y control de la cámara. En cuanto a TeamViewer se refiere, el aspecto de la pizarra quedaría resuelto con el modo pizarra que nos ofrece, los otros aspectos se resolverían de la misma manera que hemos comentado para Logmein y Zsoporte.

2.6 Técnicas de seguimiento de objetos en secuencias de vídeo

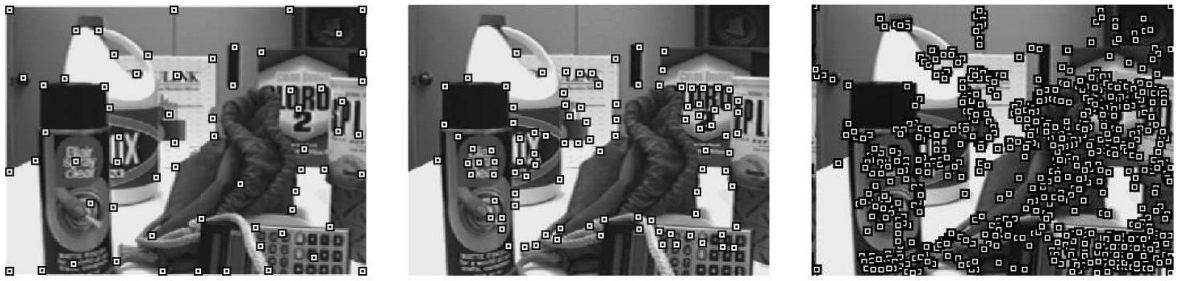
Uno de los requerimientos de este proyecto es dotar a la aplicación que se ha desarrollado de un sistema de seguimiento automático del profesor, de manera que una vez haya sido lanzado, el profesor pueda moverse libremente por el aula hasta que él mismo finalice el seguimiento.

Para hacer este seguimiento realidad se busca un procesado de la secuencia de vídeo grabada por la cámara para detectar la posición actual del profesor, que combinada con el envío de las instrucciones de movimiento pertinentes se conseguirá un algoritmo de seguimiento con cámara en movimiento.

Tal y como se estudia en [1], para la realización del *tracking* de un objeto es necesario un mecanismo de detección del objeto para cada *frame* de la secuencia. Una aproximación para la detección de objetos es usar la información de un único *frame*, pero algunos métodos suelen usar la información temporal obtenida de una secuencia de *frames* para reducir el número de falsas detecciones. La selección de la característica para el *tracking* es muy importante, en este proyecto nos centramos en la detección de puntos y en el histograma de color del objeto. Con estas dos características se conseguirá una complementación que de un mejor resultado en casos de que unos de los dos métodos esté fallando en la detección. A través de una medida de calidad en cada uno de ellos se podrá decidir qué resultado es mejor o si se ha perdido el objeto.

2.6.1 Extracción de puntos por SURF

Los detectores de puntos [1] se utilizan para encontrar los puntos de interés en imágenes que tienen una textura expresiva en zonas localizadas. Existen varios detectores de puntos entre los que hay que destacar Moravec's Interest Operator [3], Harris Interest Point Detector [2], KLT Detector [6] y SIFT Detector [4].



(a) (c) (c)

Figura 2-17: Puntos de interés. (a) Harris, (b) KLT, (c) SIFT.

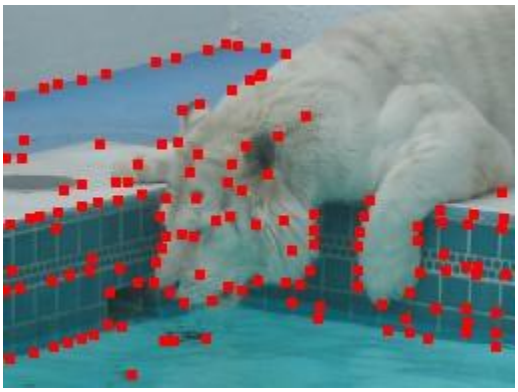


Figura 2-18: Puntos de interés. Moravec´s Interest Operator.

El *tracking* por puntos se puede formular como la correspondencia del objeto detectado representado por puntos a lo largo de los *frames* [1]. La correspondencia por puntos es un problema complicado en los casos de oclusión, de detecciones incorrectas y de entradas y salidas del objeto del cuadro.

Con la utilización de métodos determinísticos de correspondencia podemos definir un coste de correspondencia usando una combinación de las siguientes limitaciones:

- **Proximidad:** se asume que el objeto no ha cambiado notablemente su posición en el *frame* respecto de la posición que tenía en el *frame* anterior.

- **Máxima velocidad:** define el límite superior de la velocidad a la que se va a mover el objeto, por lo tanto se puede determinar una región circular alrededor del objeto en la que se va a encontrar en el nuevo *frame*.

- **Pequeño cambio de velocidad:** esta limitación asume que la dirección y velocidad que lleva el objeto no cambia drásticamente.

- **Movimiento común:** la velocidad del objeto y la de los objetos que se encuentran en una región cercana será similar. Esta limitación se puede usar para objetos representados por múltiples puntos.

- **Rigidez:** asumiendo que los objetos son rígidos en el mundo 3D, la distancia de entre cualquier par de puntos en el objeto actual permanecerá sin cambio.

- **Uniformidad proximal:** es una combinación de la proximidad y del pequeño cambio de velocidad.

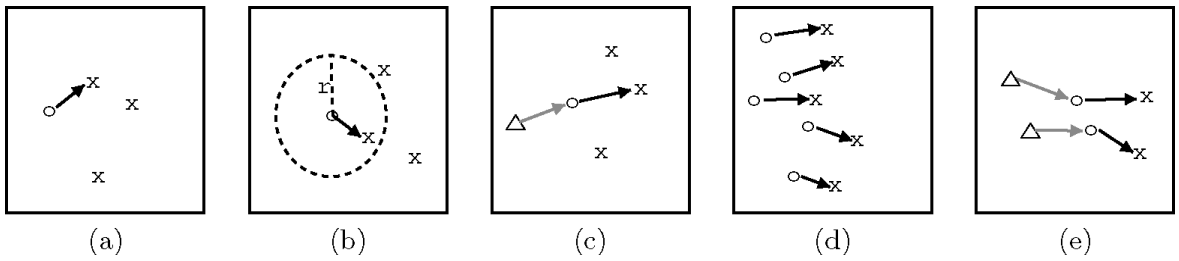


Figura 2-19: Limitaciones. (a) Proximidad, (b) máxima velocidad, (c) pequeño cambio de velocidad, (d) movimiento común, (e) rigidez.

Una vez se han extraído los puntos de interés del nuevo *frame*, podremos hacer corresponder los puntos de interés del objeto con los del *frame*, habiendo filtrado previamente los puntos del *frame*. Esto quiere decir que para cada punto del objeto, iremos aplicando las distintas limitaciones sobre los puntos del nuevo *frame*, para quedarnos con un grupo menor de puntos sobre el que hacer la correspondencia. Por ejemplo, en el caso de la máxima velocidad, descartaremos todos aquellos que estén fuera del círculo que determina esa máxima velocidad y finalmente nos quedaremos con uno de los puntos que están dentro, y ese punto del *frame* será el que corresponderá con el del objeto que estamos buscando.

Para la detección de puntos de interés se va a utilizar SURF (Speed Up Robust Features) que es un detector de puntos y descriptores invariante a escala y rotación. SURF está

inspirado en el detector de puntos y descriptores SIFT, pero es varias veces más rápido y más robusto frente a diferentes transformaciones de la imagen. En este caso, la velocidad del procesado es crítica, ya que necesitamos que el algoritmo de seguimiento del profesor sea a tiempo real, procesando entre 10 y 25 *frames* por segundo.

2.6.2 Histogramas de color

A parte de los puntos característicos, otra de las características de un objeto en la que nos podemos centrar para la realización del *tracking* es el color del objeto, o la distribución de colores que este tiene. Es en este punto en donde entra en juego el histograma de color del objeto que se quiere seguir.

El color aparente de un objeto está influenciado, a priori, por dos factores físicos, 1) la distribución espectral de energía de la fuente y 2) las propiedades reflectantes de la superficie del objeto. [1] En el procesado de imagen, el espacio de color RGB (rojo, verde, azul) se utiliza normalmente para representar el color, sin embargo, no es un espacio perceptualmente uniforme, además las dimensiones RGB están altamente correladas. Por el contrario HSV (color, saturación, brillo), sí es una aproximación a un espacio de color perceptualmente uniforme, aunque es un espacio sensible al ruido.

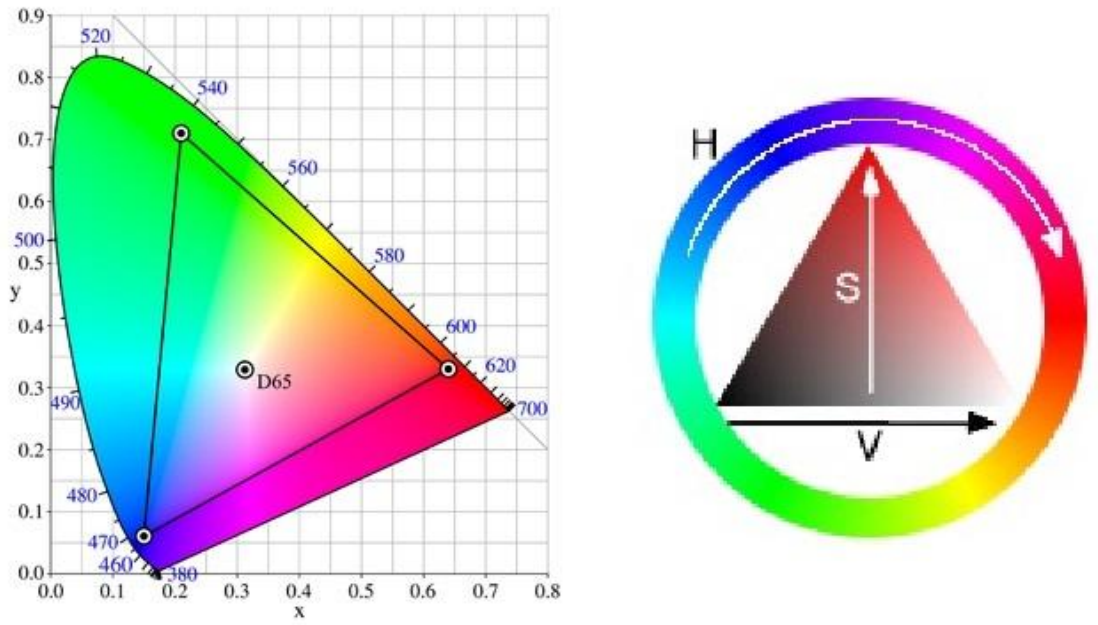


Figura 2-20: a la izquierda el modelo RGB, a la derecha el modelo HSV.

Para el *tracking* del objeto en las distintas imágenes se utilizará el modelo HSV, ya que en él encontramos una componente que es directamente el color del pixel, mientras que en el RGB tenemos una mezcla de las tres componentes para darnos el valor del color.

- **Mean-Shift Clustering:** para el problema de segmentación de imagen, Comaniciu y Meer proponen el enfoque mean-shift para encontrar grupos en la unión espacio-espacio de color, [l, u, v, x, y], donde [l, u, v] representan al color y [x, y] representan la localización espacial. Dada una imagen, el algoritmo se inicializa con un largo número de hipotéticos centros de grupos elegidos aleatoriamente de los datos. Entonces, cada centro de grupo se mueve al medio de los datos, quedando dentro de un elipsoide multidimensional centrado en el centro del grupo. El vector definido por el viejo y el nuevo centro de grupo se llama *mean-shift vector*. El mean-shift vector es procesado iterativamente hasta que los centros de grupo no cambian sus posiciones. Se nota que mientras se suceden las iteraciones de mean-shift algunos grupos pueden llegar a fusionarse.

Una vez realizado el mean-shift clustering, tenemos la imagen segmentada en regiones de distintos colores, y cada una de esas regiones tendrá un histograma de color asociado, al igual que el objeto que estamos buscando tendrá el suyo. En este punto se realiza una comparación de los histogramas de color a través de la *backprojection* (retroproyección) del histograma del objeto, y localizamos el clúster cuyo histograma corresponde en mayor porcentaje al del objeto que se está buscando en la imagen. El centro de ese clúster será el nuevo centro del objeto.

2.7 Streaming de vídeo

Otra parte que es interesante para el desarrollo de este proyecto es la emisión del flujo de vídeo de la cámara a través de internet. Gracias al *streaming* de vídeo podremos hacer que cualquier alumno que se encuentre en su casa pueda ver lo que sucede en el aula.

Ya se ha hablado de EPS-RemoteViewer y ha quedado claro que a través de esta aplicación se puede conseguir que los distintos usuarios puedan acceder al flujo de vídeo, pero el hecho de que cada uno necesite esta aplicación lo complica un poco todo. Lo que se quiere estudiar en esta sección es otra forma de servir ese flujo de vídeo a los usuarios, teniendo acceso al mismo a través de alguna aplicación estándar o desde alguna página

web, evitándose así cualquier incompatibilidad entre el *software* y el sistema que esté utilizando cada usuario.

En la búsqueda de aplicaciones que resolviesen el problema del *streaming* de vídeo se encontró una página web, http://livestream.com, a partir de la cual se pueden obtener diversas aplicaciones y privilegios en función del tipo de miembro que seamos. Al ser un miembro gratuito se puede tener acceso a la aplicación "Livestream Procaster", que se encargará de enviar el flujo de vídeo a los servidores de Livestream. Este flujo de vídeo se compone en un principio de lo que está sucediendo en el escritorio del PC, pero se puede configurar de varias maneras según lo que queramos transmitir. El hecho de poder enviar el escritorio nos resuelve el problema de la conexión de la aplicación con la cámara (ya que como el resto de aplicaciones estudiadas en este proyecto no tiene una interfaz de conexión), al poder lanzar previamente la aplicación EPS-RemoteViewer y seleccionando el área de interés emitirlo desde Livestream Procaster.

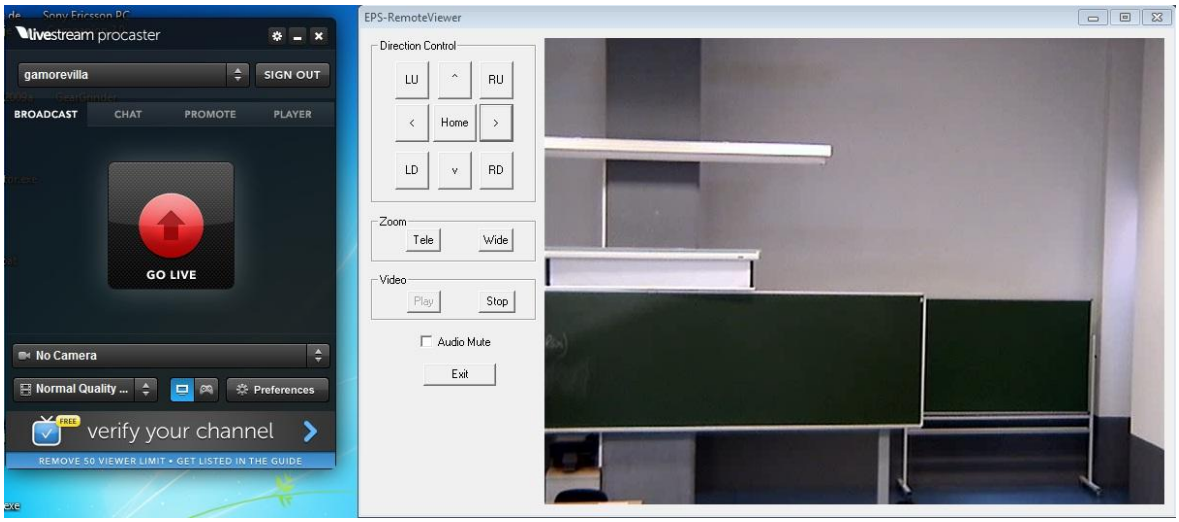


Figura 2-21: Livestream Procaster y EPS-RemoteViewer.

Una vez lanzadas ambas aplicaciones sólo nos falta activar el *broadcasting* pulsando el botón "GO LIVE" que se observa en la Figura 2-21. En el momento que lo hayamos pulsado el flujo comenzará a emitirse, pero para que no se emitan cosas que no queremos, se debe ajustar el rectángulo verde que nos aparece en la pantalla a la zona en la que se muestra la imagen recibida por EPS-RemoteViewer. En la Figura 2-22 se puede observar como se ha ajustado dicho rectángulo a la zona y en la parte inferior izquierda aparece el flujo de vídeo que se está emitiendo a los servidores. También es de gran funcionalidad el

chat que aparece ya que permite una comunicación en ambos sentidos con las personas que están accediendo al contenido a través del canal de Livestream.

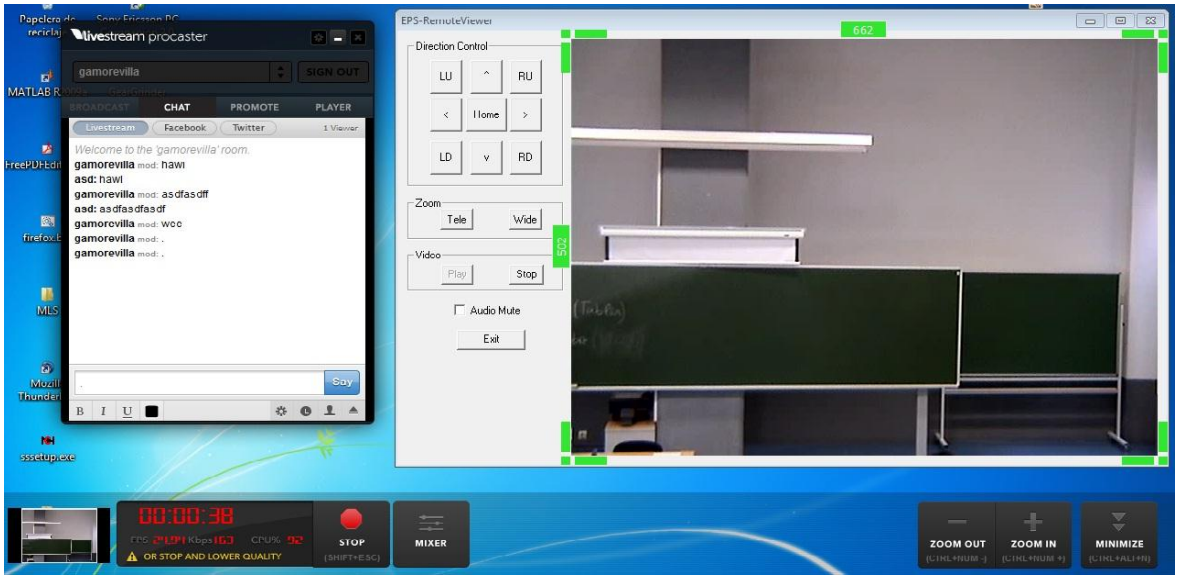


Figura 2-22: Livestream Procaster emitiendo la zona seleccionada en verde.

El acceso al contenido se realiza a través de la página web livestream.com/, añadiendo a la dirección el nombre del canal en el que se está emitiendo. En la Figura 2-23 observamos cómo se recibe el flujo de vídeo a través de la dirección http://www.livestream.com/gamorevilla.

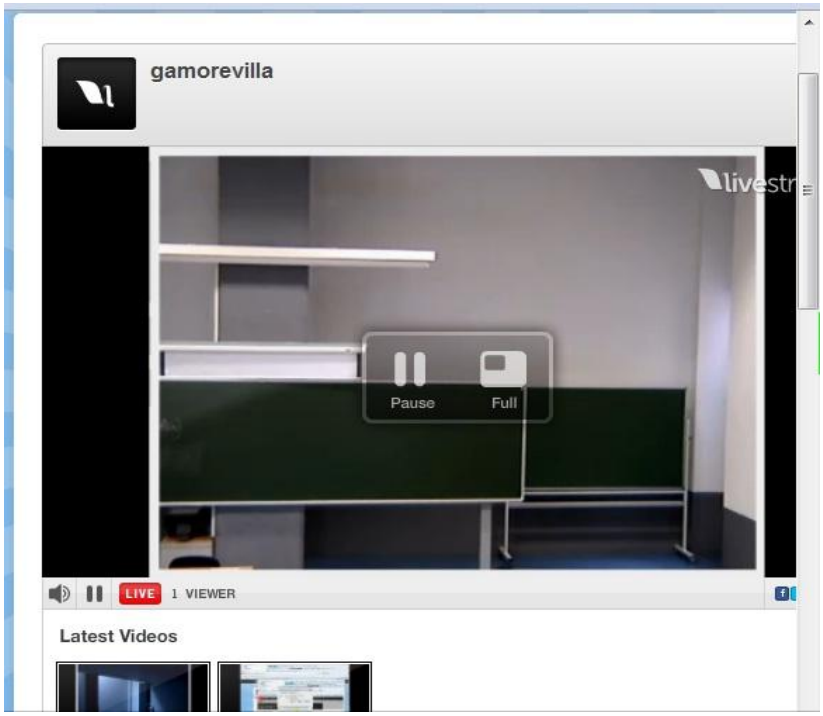


Figura 2-23: Visualización del flujo en el navegador a través del canal en livestream.com.

En este caso el canal es *gamorevilla* porque el usuario gratuito que se ha creado es *gamorevilla*. En el caso de las aulas de la universidad habría que crear un usuario y luego lanzar un canal para cada una de ellas, y quizás también convendría crear una página web a través de la cuál tener acceso a todos los canales, ya sea por un enlace que nos redirija a Livestream.com o integrando en la web, por código HTML, el reproductor que nos facilitan. De esta manera se tendría una universidad virtual en la que podríamos estar en varias clases a la vez, o simplemente observar la que más nos interese.

Livestream Procaster tiene la posibilidad de, además de transmitir el flujo en tiempo real, poder dejar guardada la sesión en el servidor, de manera que se puede acceder a sesiones anteriores en las distintas aulas.

En este proyecto nos centramos en *software* gratuito, por lo cual estamos limitados a lo que cada página web nos ofrece. En el caso de Livestream se limita el ancho de banda y calidad con la que se transmite el flujo de vídeo, también se limita el número de usuarios concurrentes a 50. Otra limitación que sería interesante que no existiese es la de poder otorgar o denegar el acceso al contenido a usuarios que no nos interese que lo visualicen. Obteniendo un usuario Premium se levantan todas esas limitaciones y la obligación de visualizar publicidad en los primeros segundos.

Otra opción que existe para el *streaming* del flujo de vídeo grabado por la cámara es la de integrar en la propia aplicación, un servidor *streaming*. Como ya se ha estudiado, las librerías de Gstreamer nos permiten solucionar la mayor parte de las funcionalidades que queremos en el *streaming* de vídeo, ya que su integración en la aplicación la podemos programar como mejor nos parezca.

3 Arquitectura del Sistema

En este capítulo se va a describir cuál ha sido la arquitectura utilizada para diseñar el sistema, el funcionamiento de las distintas aplicaciones que se han desarrollado en este proyecto y los algoritmos que se han utilizado para la realización del seguimiento automático incluido en la aplicación principal.

3.1 Descripción general

La idea principal de este proyecto es conseguir desarrollar una aplicación que realice tres funciones básicamente: controlar el movimiento de la cámara, realizar un procesado del flujo de vídeo recibido para poder conseguir un seguimiento automático del profesor y por último realizar el envío del flujo de vídeo a puntos remotos a través de un *stream* de vídeo.

En el esquema de funcionamiento del sistema general se puede observar el funcionamiento de la idea más ambiciosa del sistema, incluyendo el funcionamiento del seguimiento automático del profesor y la emisión vía *streaming* con composición del vídeo.

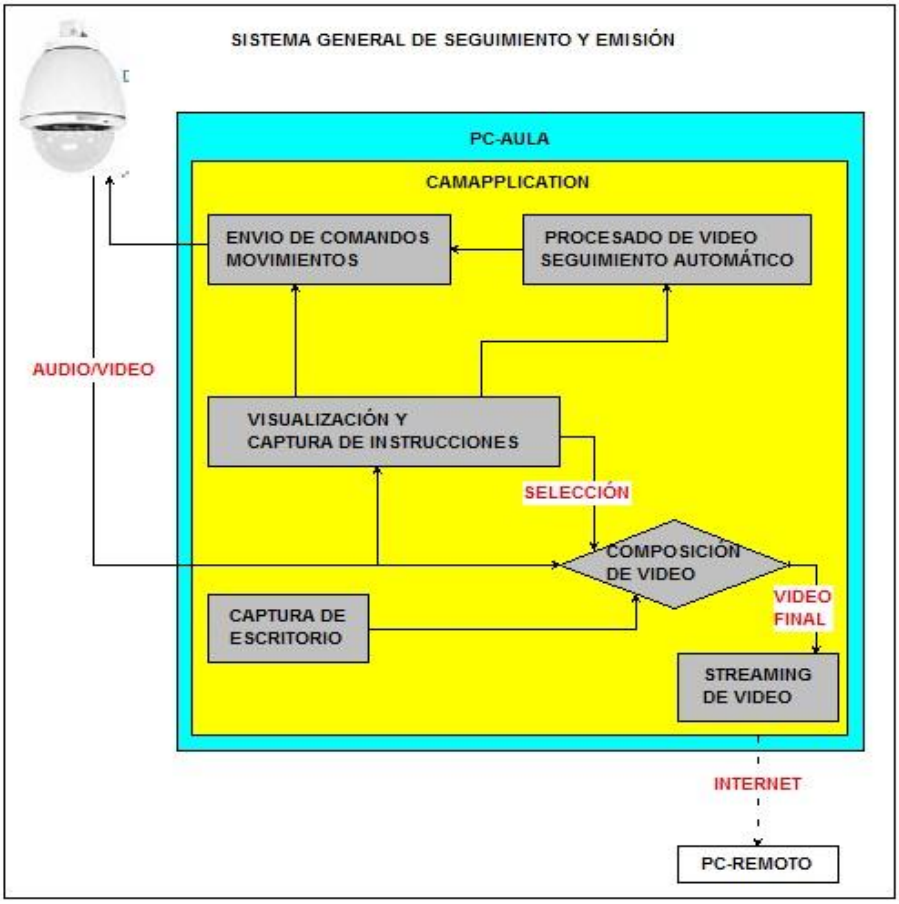


Figura 3-1: Esquema del sistema general de seguimiento automático y emisión.

En las siguientes secciones se comentará la posibilidad de reducir el esquema general de la aplicación, evitando colocar el módulo de *streaming*, el módulo de captura de escritorio y el módulo de composición de vídeo, pero consiguiendo el mismo resultado a través de la aplicación Livestream Procaster estudiada en la sección 2.7, sobre *Streaming* de vídeo.

- **Visualización y captura de instrucciones:** este módulo es la interfaz gráfica de la aplicación general, en la que se reproducen tanto el flujo de vídeo como el flujo de audio recibidos desde la cámara. En esta interfaz se recogen a través de botones las instrucciones que el profesor quiera realizar, y en función de las mismas se activarán los distintos módulos requeridos (procesado de vídeo y seguimiento automático del profesor, selección de la composición del vídeo y emisión del flujo de vídeo). También se recogerán las instrucciones pertinentes de movimiento de la cámara, ya sean movimientos relativos, posiciones predeterminadas de la cámara o cambios del *zoom*. Estas instrucciones serán enviadas a la cámara a través del módulo de envío de comandos de movimiento.

- **Envío de comandos de movimiento:** este módulo es el encargado de enviar a la cámara los comandos de movimientos que sean requeridos, bien por instrucciones recogidas en el módulo de visualización y captura de instrucción o bien por instrucciones generadas en el módulo de procesado de vídeo y seguimiento automático del profesor. En el caso de provenir las instrucciones del módulo de captura de instrucciones, el envío del comando se realizará en modo multihilo para que la aplicación pueda seguir reproduciendo el flujo de vídeo y sea posible visualizar el recorrido que ha realizado la cámara. También permite varias instrucciones de movimiento mientras se realiza el movimiento, y de esta manera irá enlazando los distintos movimientos ordenados. En el caso de provenir la instrucción del módulo de procesado de vídeo y seguimiento automático del profesor, el envío del comando no se realiza en multihilo, sino que la aplicación se queda esperando a que se realice por completo, ya que en el procesado de vídeo no nos interesa recoger ninguna imagen de la transición, sino obtener la imagen de la posición final del movimiento para, a partir de ahí, continuar buscando al profesor en la secuencia.

- **Procesado de vídeo y seguimiento automático:** este módulo se activa o desactiva por orden del módulo de captura de instrucciones. Cuando está activado se encarga

de calcular la nueva posición del profesor en la imagen y dependiendo de en qué zona se encuentre ordenar el movimiento, o no, de la cámara. Para ello recibe el flujo de vídeo de la cámara desde el módulo de visualización y captura de instrucciones, realiza el procesado *frame* a *frame* y obtiene los datos de posiciones y calidades obtenidas por los métodos de extracción de puntos por SURF e histograma de color, y en función de esas calidades calculadas se decide si mover la cámara a la posición obtenida por SURF, a la posición obtenida por histograma de color o si no ha de moverse la cámara ya que no hay un movimiento considerable del profesor. Una vez calculado el movimiento necesario, se envía al módulo de envío de comandos de movimiento la instrucción pertinente para que la cámara se mueva a la posición calculada.

- **Captura de escritorio:** es el módulo encargado de obtener el flujo de vídeo que se está presentando en la pantalla del ordenador (si es posible desarrollar este módulo se hará basándose en las librerías de Gstreamer), para poder enviarlo al módulo de composición de vídeo y que pueda hacer con él lo que sea pertinente.

- **Composición de vídeo:** básicamente es un módulo que se encarga de multiplexar el flujo de vídeo recibido del módulo de captura de escritorio y el flujo de vídeo recibido desde la cámara. En función de la opción de producción seleccionada en la interfaz gráfica, se recibirá la orden pertinente de selección desde el módulo de captura de instrucciones, para poder insertar un flujo u otro en el flujo de vídeo final que será enviado al módulo de *streaming* de vídeo. Este módulo será necesario o no en función de si es posible el desarrollo del módulo de captura de escritorio y del módulo de *streaming* de vídeo.

- ***Streaming* de vídeo:** a través de este módulo se pretende dar salida al flujo de vídeo hacia un PC-remoto a través de internet. Se recibe el flujo de vídeo final del módulo de composición de vídeo y gracias a las librerías Gstreamer se consigue transmitir en el protocolo RTP a uno o varios PC´s remotos. Al igual que los dos módulos anteriores, queda pendiente de un desarrollo satisfactorio ya que la integración en la aplicación general es complicada y aún no hay un resultado claro de los tres módulos. Como ya se ha comentado anteriormente, en el caso de que no se puedan desarrollar satisfactoriamente, se solventaría el problema utilizando la aplicación Livestream Procaster, que soluciona la composición del vídeo final de

manera sencilla, la emisión del flujo de manera interna (con lo que no habría que preocuparse de ella) y la visualización en un navegador web ya la tiene implementada.

3.2 Aplicaciones asociadas

Debido a que se tienen distintas situaciones en las que interesa usar la cámara del aula para recoger los acontecimientos que suceden, es necesaria la creación de varias aplicaciones que se ajusten a lo requerido en cada una de esas situaciones.

Las aplicaciones que se han desarrollado son EPS-RemoteViewer, EPS-CfgGenerator y CamApplication. Tanto EPS-RemoteViewer como CamApplication son aplicaciones de conexión, visualización y control de la cámara en función del tipo de usuario que se introduzca en los datos de conexión, las diferencias entre una aplicación y otra son que CamApplication cuenta con el seguimiento automático integrado, la posibilidad de integrar el servicio de *streaming* y una funcionalidad añadida al control del movimiento ya que gracias a las posiciones predefinidas se puede colocar la cámara en ciertas regiones con un solo *click.*

El proceso que hay que seguir para poner en funcionamiento cada una de las aplicaciones se encuentra definido en el Anexo D. El desarrollo de estas aplicaciones se ha llevado a cabo trabajando con las cámaras instaladas en las aulas 5 y 6, aunque también se han dejado configuradas las cámaras instaladas en la sala de grados A-121.

3.2.1 Aplicación de control de conexión

Esta aplicación, EPS-CfgGenerator, simplemente se encarga de crear un archivo, *EPS-RemoteViewer.cfg*, en el que va cifrada la información de conexión con la cámara.

Figura 3-2: EPS-CfgGenerator, introducción de datos de conexión.

En ella se introducen el usuario y la contraseña, la dirección y puerto de conexión con la cámara y las fechas entre las cuales el usuario introducido será válido. El usuario introducido tendrá que haber sido previamente configurado en la cámara desde el navegador, habiéndole otorgado los privilegios requeridos. Una vez se ha creado el archivo de configuración, copiándolo en la carpeta en que se encuentra cualquiera de las otras dos aplicaciones, al ejecutar una de ellas se accederá al archivo obteniendo así los datos de conexión, y en función del usuario y de las fechas obtenidas se decidirá si se permite al usuario la conexión o no. En el caso de haber permitido la conexión, la propia cámara será la encargada de dar privilegios de control del movimiento, o no, al usuario (según esté configurado).

El objetivo de esta aplicación es ocultar, por motivos de seguridad, datos sensibles sobre la configuración de la cámara, esencialmente su dirección IP y los nombres de usuario, y limitar temporalmente su uso a lo estrictamente necesario, para evitar problemas de privacidad.

3.2.2 Aplicación básica de control de la cámara (sin seguimiento automático)

La aplicación básica de control de la cámara se ha llamado EPS-RemoteViewer, ya que con ella conseguiremos visualización del flujo de vídeo de la cámara y el control de la misma desde cualquier ordenador, ya sea el del mismo aula, o un ordenador remoto.

Colocando el archivo de configuración *EPS-RemoteViewer.cfg* en la misma carpeta que la aplicación, al lanzarla tendremos el acceso a la cámara que en él se encuentra configurado. De esta manera se resuelve el problema de que los alumnos puedan visualizar lo que en el aula ocurre desde sus casas, habiéndoles facilitado la aplicación y un archivo de configuración con los datos adecuados para ver cierta sesión y ninguna otra, y evitando que tengan la capacidad de mover la cámara al pulsar los botones (ya que en este caso será el profesor el que la mueva cuando le parezca necesario o realice un seguimiento automático). También se resuelve el problema de que sea el profesor el que está en un ordenador remoto y tiene que impartir una clase o actuar como miembro de un tribunal en un determinado aula, ya que teniendo la aplicación y el archivo (ahora sí, con capacidad de movimiento de la cámara), podrá visualizar lo que sucede en el aula y con las aplicaciones de control remoto de escritorio realizar la sesión como él desee.

En la Figura 3-3 vemos la aplicación EPS-RemoteViewer en funcionamiento. En este momento se está mostrando el flujo de vídeo recogido por la cámara en tiempo real y la posibilidad de pulsar cualquier botón de control del movimiento, ajuste del *zoom* o incluso seleccionar si se quiere o no escuchar el audio recogido por el micrófono inalámbrico del aula (el cual se encuentra conectado a la entrada de audio de la cámara). En el caso de tener un usuario restringido, por mucho que se pulsasen los botones de movimiento o ajuste del *zoom*, la cámara no haría caso de las instrucciones, sólo serían funcionales los botones de Play, Stop, Audio Mute y Exit de la aplicación.

Figura 3-3: EPS-RemoteViewer.

En este caso el diagrama de bloques que define la aplicación es bastante más reducido y simple que el diagrama del sistema general. Como se puede observar en la Figura 3-4, el diagrama cuenta únicamente con el bloque de visualización y captura de instrucción y el bloque de envío de comandos de movimiento.

El bloque de envío de comandos de movimiento realiza las mismas funciones descritas en el caso del diagrama del sistema general, mientras que en el bloque de visualización y captura de instrucciones se ha añadido la funcionalidad de obtener los datos de conexión con la cámara y decidir en función de ellos si el usuario tiene o no acceso a la cámara en la fecha actual.

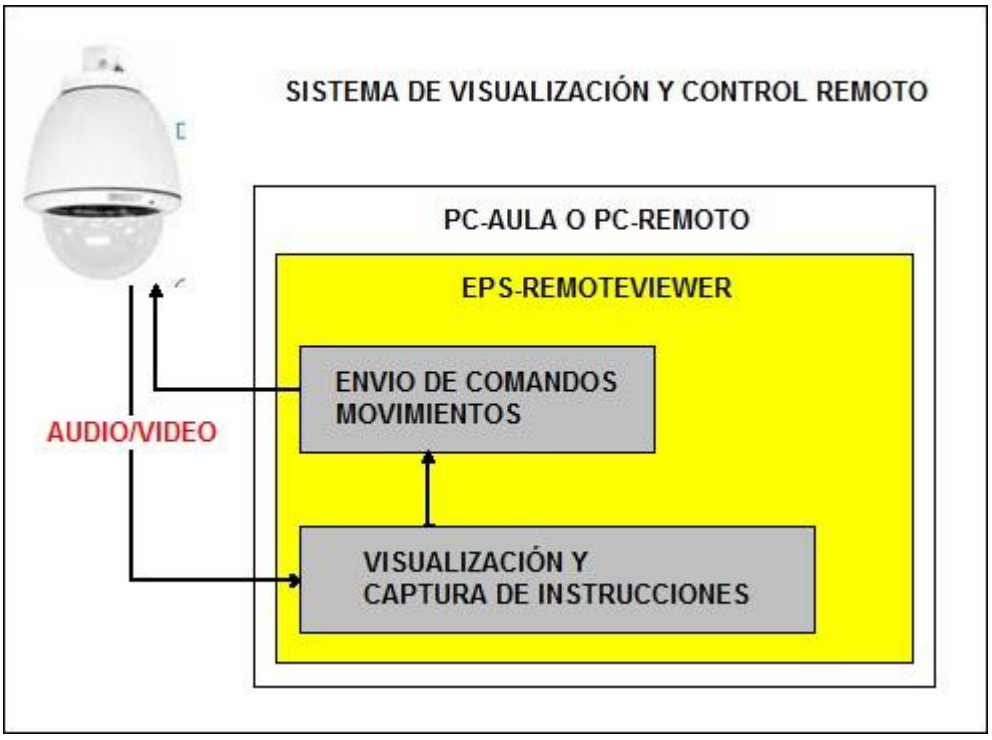


Figura 3-4: Sistema de visualización y control remoto.

3.2.3 Aplicación principal

Como aplicación principal se ha desarrollado CamApplication, que permite la conexión con la cámara, visualización, control del movimiento, ajuste del *zoom*, selección de posiciones predefinidas, seguimiento automático y la posibilidad de integrar el servicio de *streaming*.

El desarrollo de CamApplication se ha realizado en función al esquema del sistema general de seguimiento automático y emisión mostrado en la Figura 3-1.

En la Figura 3-5 se muestra la interfaz gráfica de la aplicación principal, CamApplication, pudiéndose ver en ella la zona de visualización del flujo de vídeo capturado por la cámara. Se observa que aparecen unos campos para introducir texto, que son los que habrá que rellenar con los parámetros de conexión con la cámara. En este caso no es necesario encriptar los datos de conexión ni restringir el horario de uso del usuario, ya que esta aplicación estará únicamente en el ordenador de cada aula. Quizás fuese interesante la posibilidad de crear una aplicación con el campo de dirección y puerto de conexión configurados internamente para cada aula, y dejando sólo la posibilidad de introducir usuario y contraseña. De esta manera cada aplicación creada se conectaría a una única cámara específica. La decisión de que esta aplicación se usara únicamente desde el ordenador del aula en el que se encuentra la cámara, se ha tomado porque el seguimiento automático sólo es funcional desde la misma aula en la que se encuentra el profesor a seguir, ya que para inicializar el seguimiento el profesor se debe mantener quieto mientras se selecciona a sí mismo.

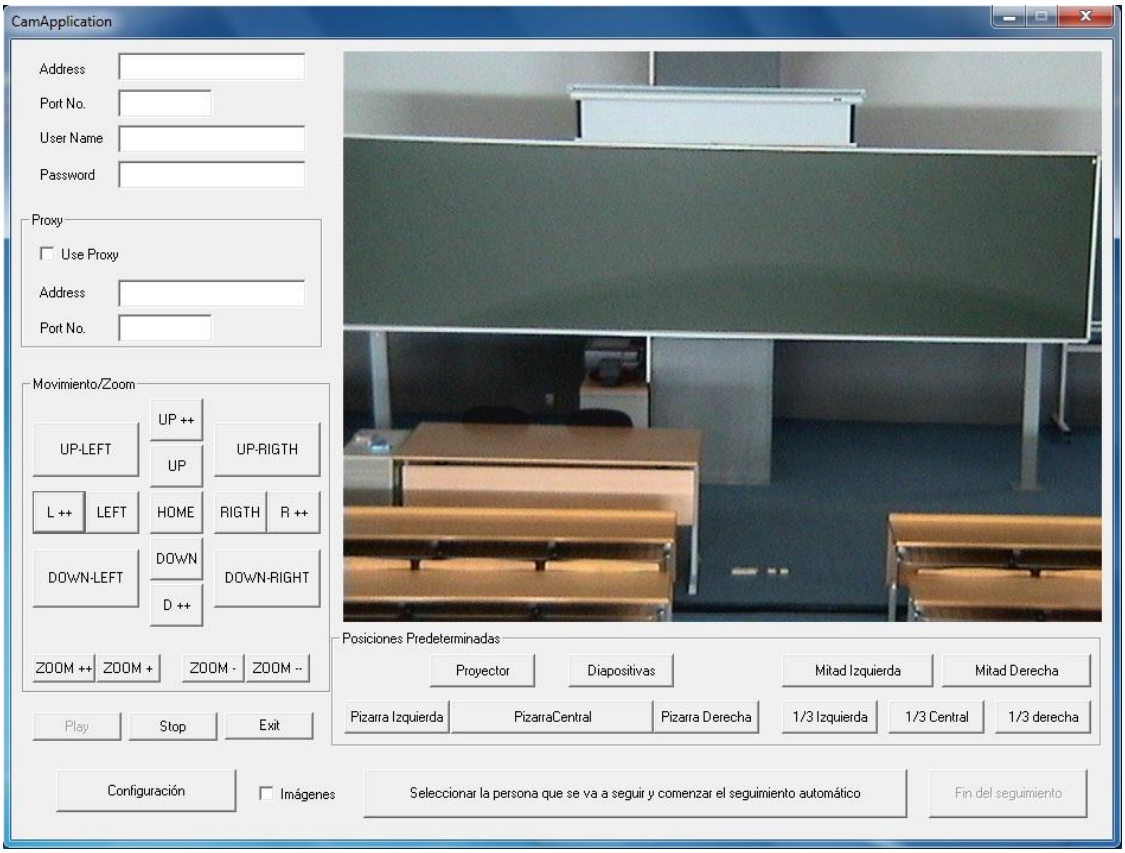


Figura 3-5: Interfaz gráfica de CamApplication.

También se puede observar un grupo de botones, desde los cuales se capturan las instrucciones de movimiento de la cámara y ajuste del *zoom* de la misma, llamado

Movimiento/Zoom. Otro grupo de botones que llama la atención es el llamado Posiciones Predeterminadas, gracias al cual se puede colocar la cámara enfocando una cierta región de la pizarra, el proyector o la pantalla de diapositivas con un solo *click*.

El botón de Configuración sirve para configurar el seguimiento automático (calibración del movimiento y *zoom* adecuado) y las regiones a las que enfocará cada una de las posiciones predeterminadas.

Finalmente tenemos el botón de selección de la persona a seguir y comienzo del seguimiento automático y el de finalización del seguimiento. Con ellos se inicializará el proceso de seguimiento automático seleccionando la cara de la persona a seguir; a partir de ahí la aplicación ordenará los movimientos necesarios a la cámara para tener siempre enfocado al profesor, y pulsando el botón de finalización en cualquier momento se parará el seguimiento y se pasará a tener que ordenar los movimientos por pulsado de botones.

3.2.3.1 Parámetros de conexión a la cámara

Para realizar la conexión con la cámara hay que hacer uso de ciertos parámetros que nos permitirán acceder a ella a través de un navegador web para poder visualizar el flujo de vídeo, realizar cualquier movimiento o configurar cualquier aspecto de la misma. Los mismos parámetros de conexión necesitaremos para realizar la conexión a través de cualquiera de las aplicaciones desarrolladas en este proyecto.

Los parámetros necesarios para realizar la conexión serán la dirección IP de la cámara a la que nos vamos a conectar y el número de puerto que siempre serán los mismos para una misma cámara y por otro lado el nombre de usuario y contraseña correspondiente que hayan sido configurados en la cámara previamente (o los de administrador en caso de serlo).

En el caso de estar utilizando un proxy entre la aplicación y la cámara habrá que indicar a la aplicación que se está utilizando e introducir tanto la dirección IP como el puerto al que hay que conectarse del proxy. En nuestro caso no ha sido necesaria la utilización de estos parámetros ya que no se ha utilizado ningún proxy, pero los campos se han dejado puestos en la aplicación para que si en algún momento son necesarios se puedan utilizar correctamente.

3.2.3.2 Uso de las opciones PTZ de la cámara

Al ser una cámara del tipo PTZ (Pan, Tilt, Zoom), lo que quiere decir que permite realizar movimientos en horizontal, vertical y ajustes del *zoom*, es necesario conocer la forma de ordenar estos movimiento y ajustes desde la aplicación (ya que desde el navegador web ya están creados los botones necesarios para realizar estas funciones).

La forma que se ha utilizado para ordenar movimientos y ajustes del *zoom* a la cámara es a través de una función que viene implementada en la librería sncstrm.dll a través de la cual se envían a la cámara ciertos comandos.

```
SNC::SendCGICommand(sncHandle, "POST", "/command/ptzf.cgi",
"RelativePanTilt=FFD8, 0000, 24", sResponse,
sizeof(sResponse), &ret_len);
```

Al usar la función SendCGICommand estaremos enviando al *script* ptzf.cgi de la cámara el comando que se haya introducido como parámetro. En este caso se está utilizando RelativePanTilt, que es para realizar un movimiento relativo de la cámara desde la posición en la que se encuentra. Tras pulsar cualquier botón de movimiento será enviado el correspondiente comando a través de esta función. Si lo que se quiere es ajustar el *zoom*, pulsando el botón de aumento o disminución del *zoom* se enviará el comando RelativeZoom con sus correspondientes datos de ajuste.

3.2.3.3 Posiciones predeterminadas

La creación de unas posiciones predeterminadas se hace necesaria ya que facilitará mucho el trabajo del profesor en cuanto al manejo de la aplicación se refiere. Si tuviese que colocar la cámara enfocando a una de las pizarras pequeñas, le costaría pulsar muchas veces los botones de movimientos relativos, tanto en horizontal como en vertical y además ajustar el *zoom*. Cada vez que cambiase la posición de la cámara tendría que estar ajustando dicha posición de una manera muy tediosa. A la hora de realizar estos cambios de posición, si tenemos un botón con una posición predeterminada asignada a él, con un solo *click*, la cámara se cambiará a la posición deseada. La labor tediosa se reduce a la configuración de cada una de las posiciones predeterminadas, pero una vez se hayan configurado todas ellas, la configuración se guarda en un fichero que siempre deberá

acompañar a la aplicación, y así, se cargarán cada vez las posiciones tal y como se han configurado.

Las posiciones predeterminadas son las siguientes:

- **HOME:** vista general del aula en la que se enfoca toda la zona de movimiento del profesor, viéndose todas las pizarras, las dos zonas de proyección (diapositivas y proyector) y algunas filas de pupitres.
- **POYECTOR:** vista de la zona de proyección a través del proyector posterior de aula.
- **DIAPOSITIVAS:** vista de la zona de proyección de diapositivas.
- **PIZARRA IZQUIERDA:** vista de la pizarra pequeña situada a la izquierda del aula.
- **PIZARRA CENTRAL:** vista completa de la pizarra central.
- **PIZARRA DERECHA:** vista de la pizarra pequeña situada a la derecha del aula.
- **MITAD IZQUIERDA:** vista de la zona mitad izquierda de la pizarra central.
- **MITAD DERECHA:** vista de la zona mitad derecha de la pizarra central.
- **TERCIO IZQUIERDO:** vista de la zona que ocupa el tercio izquierdo de la pizarra central.
- **TERCIO CENTRAL:** vista de la zona que ocupa el tercio central de la pizarra central.
- **TERCIO DERECHO:** vista de la zona que ocupa el tercio derecho de la pizarra central.

En la Figura 3-6 se observa cada una de las zonas enfocadas por la cámara tras hacer *click* sobre el botón correspondiente a su posición predeterminada.

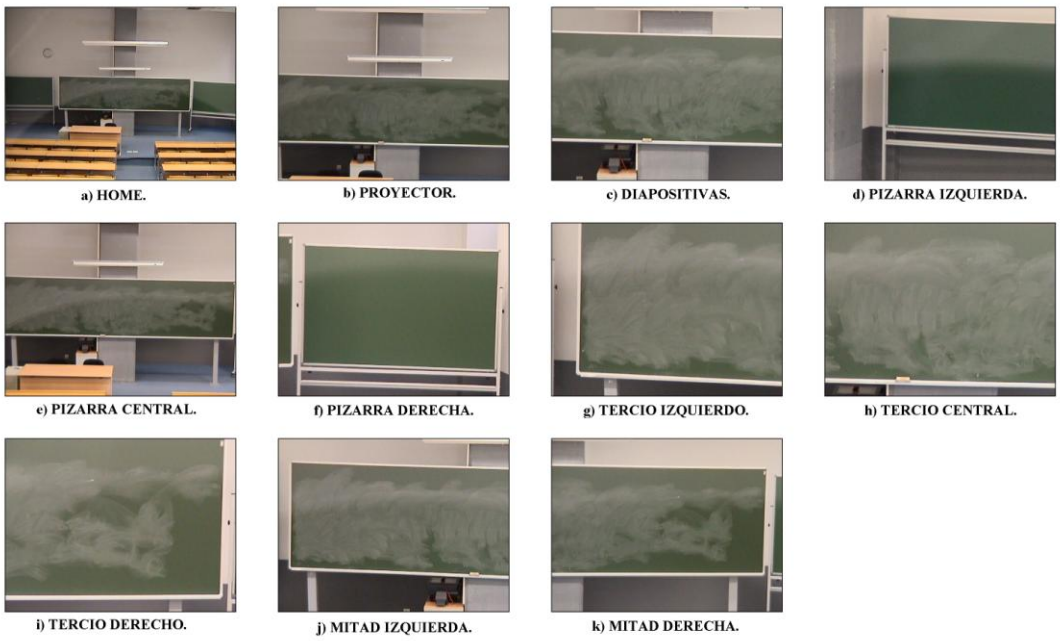


Figura 3-6: Visualización de las posiciones predeterminadas de la cámara.

3.2.3.4 Seguimiento del profesor

El seguimiento del profesor se realiza mediante la unión de la posibilidad de movimiento que brinda la cámara y un sistema de seguimiento en secuencias de vídeo. El sistema de seguimiento en secuencias de vídeo está compuesto por un seguimiento por puntos característicos (SURF) y un seguimiento por histograma de color, los cuales serán explicados más adelante. Ahora lo que nos interesa, es que a cada uno de esos seguimientos hay que facilitarles el *frame* actual y el objeto a seguir (en este caso, la cara del profesor). Para ello, en la aplicación, se obtiene el *frame* actual gracias a la función videoCallback y posteriormente se extrae de ella el objeto.

Una vez obtenido el *frame* actual (actual en el momento de pulsar el botón de inicio del seguimiento), se muestra en una ventana generada por OpenCv en la que están activados los eventos del ratón. Con el ratón se recuadrará la cara del profesor de manera que quede completamente dentro del recuadro (tal y como se muestra en la Figura 3-7), teniendo en cuenta que las partes del fondo de la imagen sean minoritarias en el recuadro.

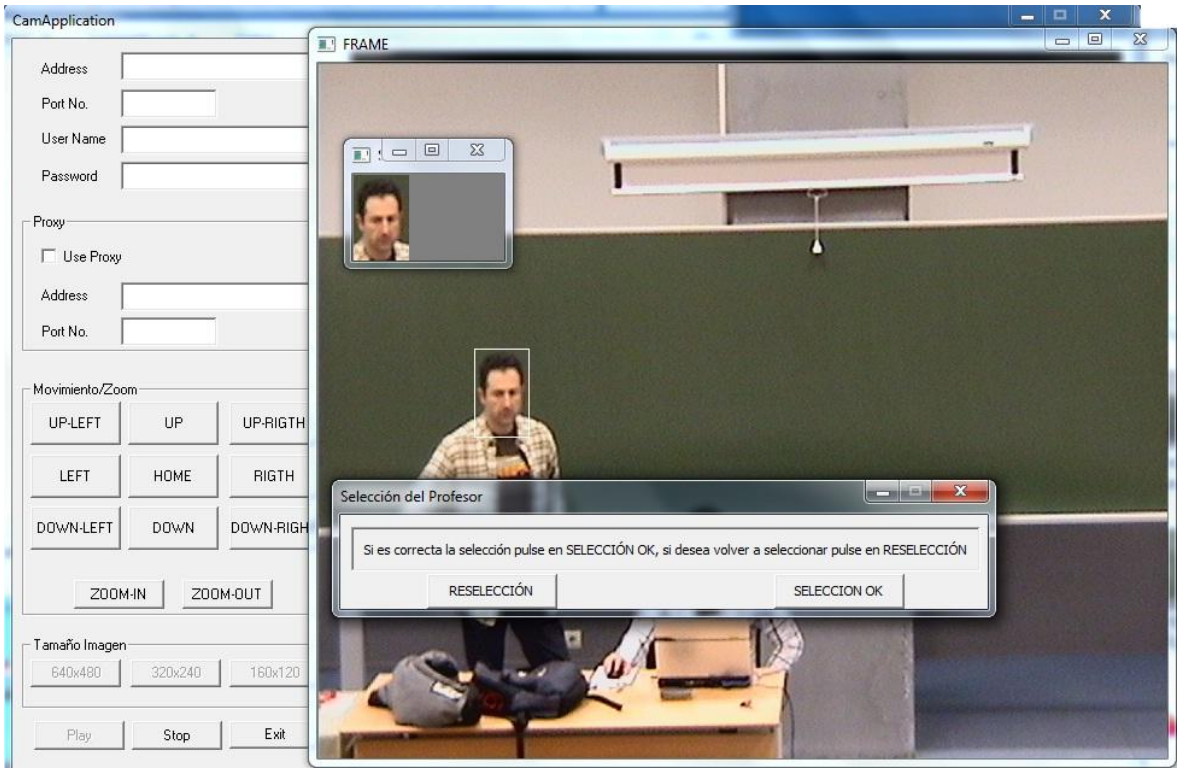


Figura 3-7: Selección de la cara del profesor para el seguimiento automático.

Una vez recuadrada correctamente la cara se hará la comprobación de que tenga los suficientes puntos característicos para que el seguimiento por puntos característicos funcione correctamente. En cuanto se tenga una pequeña imagen de la cara del profesor, con suficientes puntos característicos, serán pasadas a ambos seguimientos tanto la imagen del *frame* actual completa, como la de la cara del profesor. A partir de aquí ambos algoritmos funcionarán en paralelo, obteniendo cada uno de los nuevos *frames* actuales y actualizando su propio objeto a seguir en cada una de las iteraciones. Para cada iteración, en cada nuevo *frame*, se obtendrá una nueva posición del objeto y una medida de la calidad con la que se ha obtenido esa posición, de manera que se dará por válida la posición con la mayor calidad de los dos algoritmos. Finalmente se calcula el desplazamiento que ha sufrido el objeto que se está siguiendo.

Una vez calculado el desplazamiento del objeto de interés, solamente queda desplazar la cámara de manera que la nueva posición del objeto quede centrada en la imagen. Para ello, durante la configuración de la cámara, se ha hecho una calibración del seguimiento automático (a partir de una serie de movimientos de la cámara en cada dirección y con el *zoom* adecuado para el seguimiento, se calcula una relación de los píxeles que se desplaza un punto muy característico del fondo en cada mínimo desplazamiento de la cámara), a

partir de la cual se obtiene la relación de movimiento que hay entre la magnitud del comando y la cantidad de píxeles que se desplaza la cámara.

Si para una magnitud de movimiento horizontal de 20 se desplaza 60 píxeles el punto de interés, tendremos una relación de 60/20=3, por lo tanto, si al calcular el desplazamiento tenemos que hay que moverse 21 píxeles, se tendrá que enviar el comando de movimiento con una magnitud de 21/3=7.

Por último se enviará el comando de movimiento RelativePanTilt con las magnitudes de movimiento horizontal y vertical calculadas teniendo en cuenta el desplazamiento y la relación de movimiento de la cámara.

El proceso de calibración y ciertos fallos que se han detectado en el momento de mover la cámara y continuar con el procesado para el seguimiento, serán explicados a fondo en la sección 5.5 que tratará el Movimiento de la cámara.

4 Subsistema de Captura

Este capítulo está dedicado a conocer la cámara con la que se ha trabajado en este proyecto, viendo sus posibles configuraciones y la manera en la que se realiza la comunicación entre las aplicaciones desarrolladas y la cámara, tanto para la realización de movimientos como para la obtención de cada *frame* en la aplicación.

4.1 Descripción de la cámara

El modelo de cámara utilizado en este proyecto es una cámara PTZ, la SONY SNC RZ50P. La cámara se encuentra conectada a la red IP de la Escuela y permite el acceso desde cualquier PC conectado a internet.

Figura 4-1: Cámara SONY SNC RZ50P.

La cámara SONY SNC RZ50P es una cámara PTZ, que gracias a estar motorizada permite los movimientos en horizontal y vertical. Tiene integrado un *zoom* óptico de hasta 26x y luego permite seguir haciendo *zoom* gracias al *zoom* digital. Alcanza una tasa de imágenes de 25 fps y una resolución máxima de 704x576 píxeles, aunque la que se ha utilizado en el desarrollo de este proyecto ha sido VGA (640x480 píxeles) debido a que las librerías de desarrollo no trabajan con la máxima resolución. La cámara cuenta también con la posibilidad de trabajar en modo noche.

En cuanto a codificación de vídeo se refiere, es capaz de trabajar con dos codificaciones simultáneas, MPEG4 y JPEG (obteniendo cada *frame* individualmente para poder procesarlo). También cuenta con la posibilidad de tener una codificación en H.264, pero al utilizar esta codificación no puede trabajar con otra distinta al mismo tiempo.

4.2 Configuración de la cámara

La cámara viene con una configuración por defecto que nos permite conectarnos a ella a través de la dirección IP 192.168.1.100. Para realizar la conexión es necesario conectar un cable Ethernet entre el PC y la cámara y configurar la tarjeta Ethernet del PC con una IP correspondiente a la misma red que la cámara (por ejemplo 192.168.1.102), a partir de aquí con poner en el navegador la dirección de la cámara aparecerá un menú de visualización y configuración, tal y como se puede ver en la Figura 4-2.

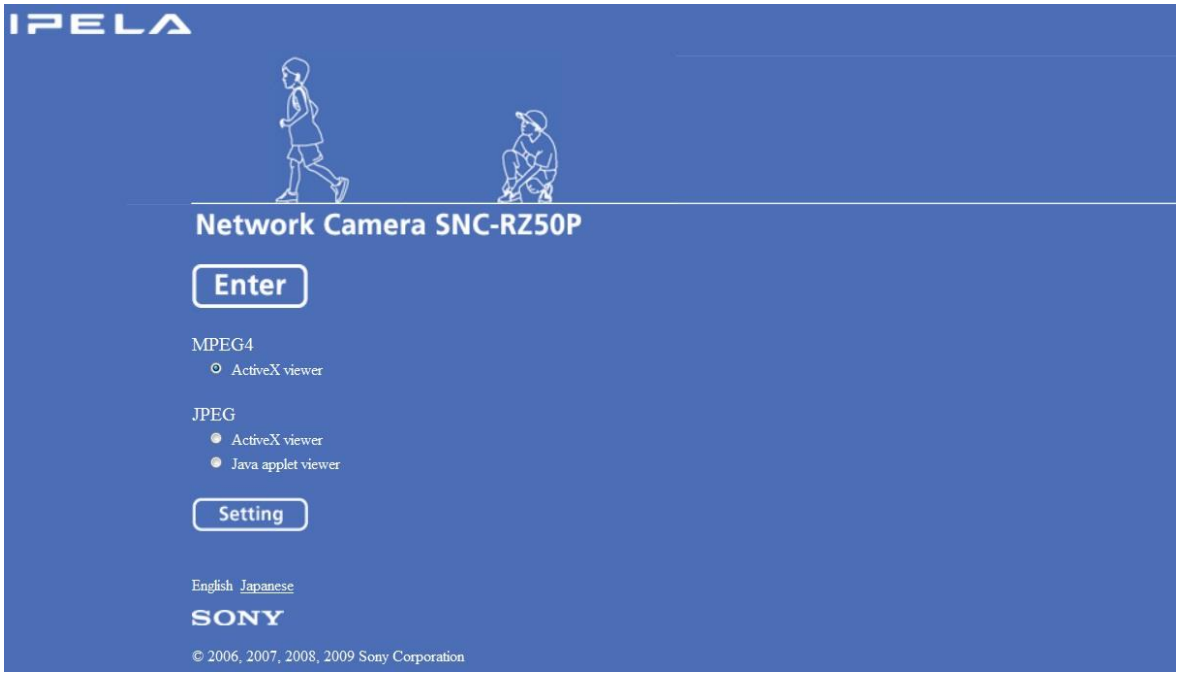


Figura 4-2: Menú principal de la cámara.

En este menú podemos seleccionar el tipo de codificación y aplicación con la que visualizar el flujo de vídeo de la cámara; también se puede acceder al menú de configuración seleccionando el botón de *Setting* (que es lo que nos interesa en esta sección).

Al acceder al menú de configuración de la cámara nos encontramos muchos aspectos que se pueden configurar. Para el desarrollo de este proyecto sólo ha sido necesario configurar varios de ellos y se han dejado con la configuración por defecto los demás. En la Figura 4-3 se puede observar el formato del menú y cada uno de los aspectos configurables, que son accesibles a través de los botones situados a la izquierda y las sub-secciones de cada aspecto son accesibles a través de las pestañas situadas en la parte superior de la Figura 4-3.

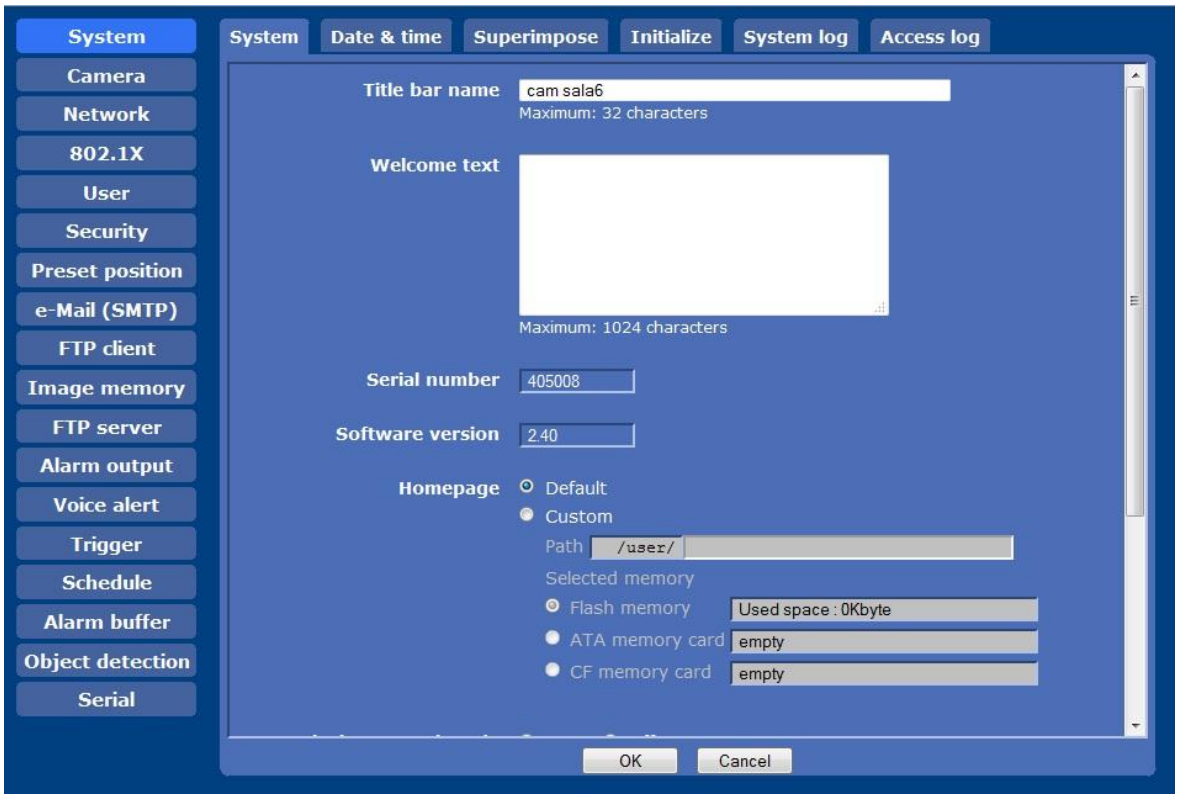


Figura 4-3: Menú de configuración de la cámara.

- **System:** en esta sección se pueden configurar aspectos generales del sistema como son el nombre otorgado a la cámara, la fecha actual, la posibilidad de superponer en el vídeo algún tipo de texto que se desee, se podrá resetear la cámara al estado por defecto o guardar algún *backup* de la configuración. En este proyecto únicamente se ha configurado el nombre, haciéndolo descriptivo del aula en la que se encuentra ubicada la cámara, el tipo de funcionamiento de la opción PTZ seleccionado en modo normal y se ha desactivado el control exclusivo (esta opción no es necesaria, pero en el proyecto se ha utilizado para realizar distintas pruebas con varias aplicaciones al mismo tiempo).

- **Camera:** es la sección a través de la cual se pueden configurar aspectos relacionados con el flujo de vídeo grabado y generado por la cámara. El tipo de codificación utilizado tanto para vídeo como para audio y la posibilidad de utilizar el modo nocturno integrado en la cámara. En la sub-sección Common se ha configurado el modo de imagen en automático con imagen en color y el *zoom* total (óptico + digital). El enfoque de la cámara se realizará automáticamente seleccionando el Focus mode en Auto. El estabilizador estará desactivado ya que es necesario tener el total del *frame rate*. En cuanto a la configuración del micrófono,

lo primero será activarlo y seguido seleccionar la ganancia, que dependerá de cada aula, seleccionando la que mejor resultado de escucha proporcione. La codificación de audio será G.711 (64kbps), que es la máxima calidad permitida por la cámara. De la sub-sección Day/Night seleccionaremos el modo automático. En la sub-sección Video códec se configura el modo Dual códec con prioridad MPEG4. Para la codificación JPEG, el tamaño de imagen será de 640x480 píxeles con una *frame rate* de 25 fps, una calidad de imagen máxima y un control de ancho de banda ilimitado para poder obtener la mejor calidad para el procesado de las imágenes. Para la codificación MPEG4 tendremos un tamaño de 640x480 píxeles, al igual que en JPEG, una *frame rate* de 16 fps y una *bit rate* de 1024 kbps. La codificación H.264 no ha sido utilizada en este proyecto.

- **Network:** esta sección hace referencia a los datos de conexión de la cámara con la red de la escuela. En ella se configurarán la dirección IP asignada por la escuela (que permita acceso desde el exterior de la red interna de la escuela), la máscara de red correspondiente y la dirección de la puerta de enlace, también se configurarán los servidores DNS. Cabe la posibilidad de configurar el puerto de conexión, pero se ha dejado el 80 para poder acceder directamente desde el navegador y evitar tener que ponerlo cada vez que se quiera acceder.

- **User:** aquí se configurarán los datos de acceso tanto del administrador como los de cualquier usuario que se quiera configurar. Se asignará un nombre de usuario y una contraseña para cada uno de ellos, y además, en los que no sean el administrador habrá que otorgar un cierto privilegio que irá, desde únicamente tener permitida la visualización, hasta tener un control total del movimiento y de la configuración (al igual que el administrador).

- **Preset position:** esta sección está dedicada a poder configurar ciertas posiciones de interés a las que poder mover la cámara con un solo *click*. En este proyecto no se han configurado internamente en la cámara ya que se ha programado la posibilidad de configurar y guardar las posiciones en la propia aplicación.

El resto de secciones y sub-secciones que no se han mencionado han de quedar configuradas tal y como vienen configuradas por defecto.

4.3 Obtención de frames de audio y vídeo para análisis y emisión

Para realizar el procesado necesario en el seguimiento automático es necesario tener en la aplicación desarrollada el *frame* actual (obtención de *frames* en tiempo real); también será necesario para la opción de emisión (al introducir el módulo de *streaming*) el poder tener en la aplicación tanto el *frame* actual como el audio actual.

Para poder obtener el *frame* de vídeo correspondiente al instante de tiempo actual, la librería de la cámara genera una interrupción en el momento en que se tiene la imagen completa y decodificada. Para poder tener acceso a ella, hay que configurar previamente a qué función llamará la aplicación en cada una de las interrupciones que se generen por decodificación de la imagen actual. Para ello se hace uso de la función:

```
SNC::SetCallback(sncHandle, CALLBACK_DEC_VIDEO,
videoCallback, NULL);
```

En ella se especifica el tipo de interrupción, en este caso CALLBACK_DEC_VIDEO, que significa interrupción por decodificación del vídeo, y la función a la que se dirigirá la interrupción, videoCallback. La función videoCallback deberá tener un formato determinado, ya que la interrupción generará una estructura específica que será pasada como argumento a la función.

```
void __stdcall videoCallback(VIDEOINFO *pVideoInfo, unsigned
long param)
```

En la estructura pVideoInfo tendremos los datos relacionados con el *frame* actual, en pBuf tenemos un puntero a la cadena que contiene la imagen completa, en UserData tendremos datos como el tiempo identificativo del momento en que ha sido capturada, la *frame* rate o la posición de la cámara, en VideoCodecInfo se encuentra el tipo de codificación y el tamaño de la imagen. Lo que realmente nos interesa para el procesado es pBuf, con lo cual se guarda en la variable frame_actual que es usada por los algoritmos del seguimiento automático.

En el caso de la emisión, y dado que se necesitan tanto vídeo como audio, se utilizaría la función:

```
SNC::SetCallback(sncHandle, CALLBACK_RAW_BOTH, AVRawCallback,
(unsigned long)windowHandle);
```

En este caso se utiliza CALLBACK_RAW_BOTH para indicar que la interrupción se hará cuando tanto el vídeo como el audio estén codificados, y a través de la función AVRawCallback se podrá tener acceso a las estructuras correspondientes. Dicha función tendrá la siguiente cabecera:

```
void __stdcall AVRawCallback(VÍDEOINFO *pVídeoInfo, AUDIOINFO
*pAudioInfo, unsigned long param)
```

En los argumentos de la función se observa la misma estructura para el vídeo que en el caso descrito anteriormente además de una estructura específica para el audio. En pAudioInfo se puede tener información del tipo de codificación utilizado, como son el número de bits por muestra, el número de canales, el número de muestras por segundo y en pBuf tendremos un puntero a la cadena que contiene los datos codificados del audio. Accediendo tanto al pBuf del vídeo como al pBuf del audio tendremos los datos necesarios para la emisión del *streaming* de vídeo.

4.4 Envío de comandos de movimiento a la cámara

Una vez obtenemos en la aplicación toda la información descendente de la cámara a la aplicación, ya sólo falta la comunicación ascendente, que en este caso será únicamente para ordenar los movimientos y los ajustes del *zoom* pertinentes a la cámara.

Para el envío de comandos a la cámara se utiliza la función:

```
SNC::SendCGICommand(sncHandle, "POST", "/command/ptzf.cgi",
"RelativePanTilt=FFD8,0000,24", sResponse, sizeof(sResponse),
&ret_len);
```

A través de esta función y configurando correctamente la cadena de caracteres que en ella se coloca como cuarto argumento, se puede conseguir cualquier tipo de movimiento de la cámara o cualquier ajuste del *zoom*.

Se podrán realizar movimientos tanto relativos a la posición actual como absolutos en cualquiera de los ejes de la cámara y ajustar el *zoom*, también de manera relativa a la posición actual o absoluta. Para ello utilizaremos los siguientes comandos:

- **RelativePanTilt:** realiza el movimiento relativo en horizontal y vertical. Es usado en los movimientos realizados por los botones de movimiento de la interfaz gráfica, y en la ordenación de movimientos desde el módulo de seguimiento automático.

- **AbsolutePanTilt:** realiza el movimiento hacia una posición definida en las coordenadas que acompañan al comando. Este comando se usa al pulsar cualquier botón de posición predeterminada.

- **RelativeZoom:** realiza el ajuste del *zoom* desde la posición en la que se encuentra, y dependiendo de la magnitud introducida aumentará el *zoom* o lo disminuirá. Si se aumenta lo suficiente el *zoom*, pasará de ser un *zoom* óptico a ser un *zoom* digital sobre la imagen capturada. Este comando es utilizado por los botones de ajuste del *zoom* de la interfaz gráfica.

- **AbsoluteZoom:** realiza el ajuste del *zoom* haciendo que se coloque en una posición determinada por la magnitud que lo acompaña. Se utiliza al colocar la cámara en las posiciones predeterminadas ya que requieren también un *zoom* sobre la zona.

En el caso del comando RelativePanTilt, las magnitudes que lo acompañan significan la cantidad de movimiento que realizan en cada eje y la velocidad a la que se va a realizar. Dichas magnitudes están representadas en hexadecimal:

- **Pan:** es la primera magnitud que acompaña al comando y significa la cantidad de movimiento que realiza en horizontal desde la posición actual. Si se introduce un número positivo, 0028h, se moverá una magnitud de 40 hacia la derecha, si se introduce un número negativo, FFD8h, se moverá una magnitud de 40 hacia la izquierda.

- **Tilt:** es la segunda magnitud que acompaña al comando y significa la cantidad de movimiento que realiza en vertical desde la posición actual. Si se introduce un número positivo, 0028h, se moverá una magnitud de 40 hacia arriba, si se introduce un número negativo, FFD8h, se moverá una magnitud de 40 hacia abajo.

- **Speed:** es la tercera magnitud que acompaña al comando y significa la velocidad a la que realizará el movimiento la cámara. Está representada en hexadecimal y

puede oscilar entre 00h y 24h, siendo 00h la menor velocidad y 24h la máxima velocidad de movimiento.

En el caso del comando AbsolutePanTilt, las magnitudes que lo acompañan significan la posición final de la cámara a la que se quiere desplazar la misma. Dichas magnitudes están representadas en hexadecimal:

- **Pan:** es la primera magnitud que acompaña al comando y es la coordenada horizontal a la que se va a mover la cámara. Las coordenadas horizontales que se pueden introducir varían entre F670h, que corresponde a una rotación de 170º hacia la izquierda, y 0990h que corresponde a una rotación de 170º hacia la derecha, pasando por 0000h que es la posición de referencia de la cámara, tal y como se muestra en la Figura 4-4.

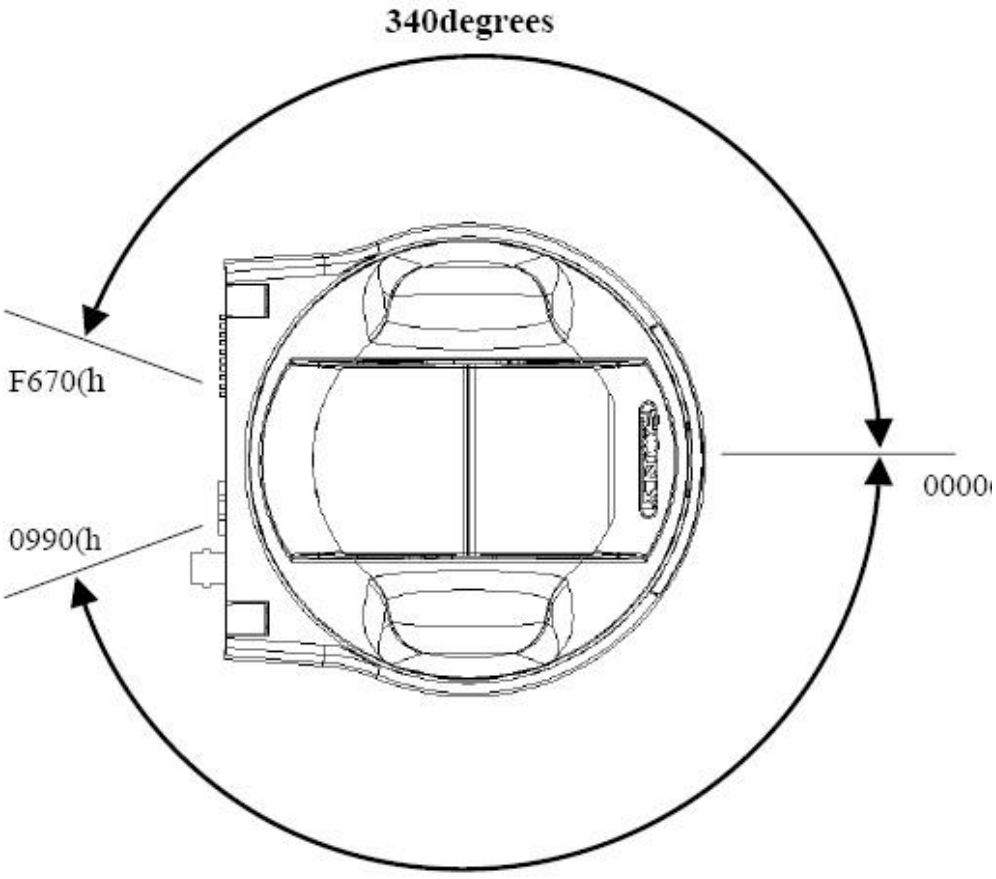


Figura 4-4: vista superior de la cámara. Rango de los ángulos horizontales que puede tomar la cámara.

- **Tilt:** es la segunda magnitud que acompaña al comando y es la coordenada vertical a la que se va a mover la cámara. Las coordenadas verticales que se pueden introducir varían entre FCC4h, que corresponde a una rotación de 57,5º hacia abajo, y 033Ch que corresponde a una rotación de 57,5º hacia arriba, pasando por 0000h que es la posición de referencia de la cámara, tal y como se muestra en la Figura 4-5.

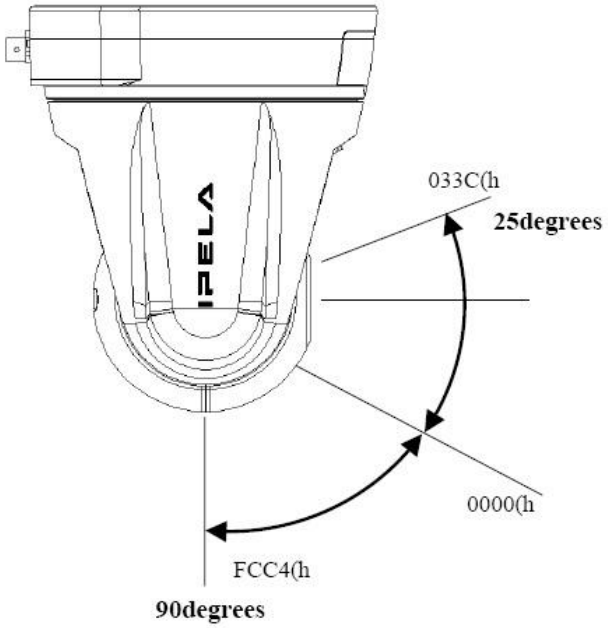


Figura 4-5: Vista lateral de la cámara. Rango de los ángulos verticales que puede tomar la cámara.

- **Speed:** es la tercera magnitud que acompaña al comando y significa la velocidad a la que realizará el movimiento la cámara. Está representada en hexadecimal y puede oscilar entre '00' y '24', siendo '00' la menor velocidad y '24' la máxima velocidad de movimiento.

Hay que decir que las posiciones absolutas no siempre serán las mismas en distintas cámaras, dependerán de la colocación que tenga cada una de ellas, por ejemplo, el punto de referencia horizontal, 0000h, no siempre estará enfocando justamente al centro de la pizarra. Es por ello que para cada aula habrá que hacer una pre-configuración de las posiciones predeterminadas.

En el caso del comando AbsoluteZoom, la magnitud que lo acompaña significa la posición final del *zoom* de la cámara. Dicha magnitud está representada en hexadecimal y puede variar entre 0000h y 4000h pasando por los valores indicados en la Tabla 4-1 para obtener

un *zoom* óptico, aunque si seguimos aumentando la magnitud pasaremos a tener un *zoom* digital. No es necesario introducir exactamente los valores que se muestran ya que si introducimos un valor intermedio tendremos un *zoom* intermedio entre las Zoom Ratio.

Zoom Ratio	Optical Zoom Position Data
x1	0000
x2	1760
x3	214C
x4	2722
x5	2B22
x6	2E20
x7	3080
x8	3278
x9	3426
x10	359E
x11	36EE
x12	381C
x13	392E
x14	3A26
x15	3B08
x16	3BD4
x17	3C8C
x18	3D2E
x19	3DBC
x20	3E58
x21	3EA2
x22	3F00
x23	3F4E
x24	3F92
x25	3FCC
x26	4000

Tabla 4-1: Códigos asociados a los distintos zoom.

5 Subsistema de Seguimiento Activo

El subsistema de seguimiento activo desarrollado en este proyecto se basa en la localización de la cara del profesor en la imagen completa usando los puntos obtenidos mediante el algoritmo SURF. Para complementar esta localización se utiliza un algoritmo secundario que funciona en paralelo, utilizando el histograma de color de la cara del profesor para la localización en la imagen completa. En las distintas secciones de este capítulo se describirán tanto el esquema como el funcionamiento interno de cada uno de los algoritmos, y cómo llegar a generar el movimiento necesario de la cámara para mantener enfocado al profesor.

5.1 Esquema general

El esquema general del seguimiento activo del profesor constará de dos métodos de búsqueda que se complementan pudiendo seleccionar uno u otro en función de la medida de calidad que se haya obtenido de cada uno de ellos.

Figura 5-1: Sistema general del seguimiento activo.

Lo primero que observamos en la Figura 5-1 es el módulo encargado de obtener el *frame* actual desde la cámara, que seguidamente es enviado a los dos algoritmos de localización implementados. La existencia de dos algoritmos trabajando en paralelo sobre el mismo *frame* se basa en la idea de seguir puntos mientras se puede, pero cuando los puntos desaparecen o dejan de ser fiables, se sigue una 'nube de color', en vez de puntos concretos. Por lo tanto es necesario que ambos algoritmos sean capaces de localizar la posición del objeto buscado (en nuestro caso la cara del profesor), determinando el

movimiento que ha realizado respecto del cuadro anterior y una medida de calidad (fiabilidad del proceso) con la que ha sido localizado. Con estos parámetros devueltos por ambos algoritmos, será el módulo decisor el encargado de decidir el movimiento que debe realizar la cámara en función de la calidad obtenida en cada algoritmo. Finalmente será el módulo de envío de comandos el que envíe a la cámara el comando de movimiento necesario.

El motivo de necesitar una medida de calidad o fiabilidad es poder utilizar ambos métodos alternativos y cambiar de uno a otro cuando la calidad del principal cae por debajo de un umbral o cuando la calidad del secundario es superior a la calidad del principal.

5.2 Seguimiento mediante puntos característicos (SURF)

En esta sección se hablará del seguimiento mediante puntos característicos de la imagen obtenidos mediante SURF (*Speed Up Roboust Features*). SURF es un detector de puntos de interés y sus correspondientes descriptores, que se declaran invariantes a la rotación y al escalado del punto de interés. De esta manera podremos localizar los puntos de interés de la cara de la persona en el siguiente *frame* que se obtenga de la cámara, pudiendo así determinar el movimiento global que han sufrido puntos homólogos entre dos *frames* consecutivos y poder ejecutar las órdenes necesarias de movimiento de la cámara.

En la Figura 5-2 se puede observar el esquema completo de funcionamiento del sistema de localización por puntos SURF trabajando individualmente. En el caso de trabajar en paralelo con el sistema de localización por histograma de color, la actualización de objeto y posición se realizará en un lazo externo, común a ambos algoritmos debido a que unas veces se considerará como posición válida la calculada por un algoritmo, y otras veces la calculada por el otro dependiendo de las calidades obtenidas en cada uno de ellos. Por lo tanto es necesario que la actualización de objeto y posición anterior sean controladas externamente a ambos algoritmos. De esta manera será una actualización dependiente de la decisión final tomada respecto al movimiento sufrido y no del resultado interno del algoritmo.

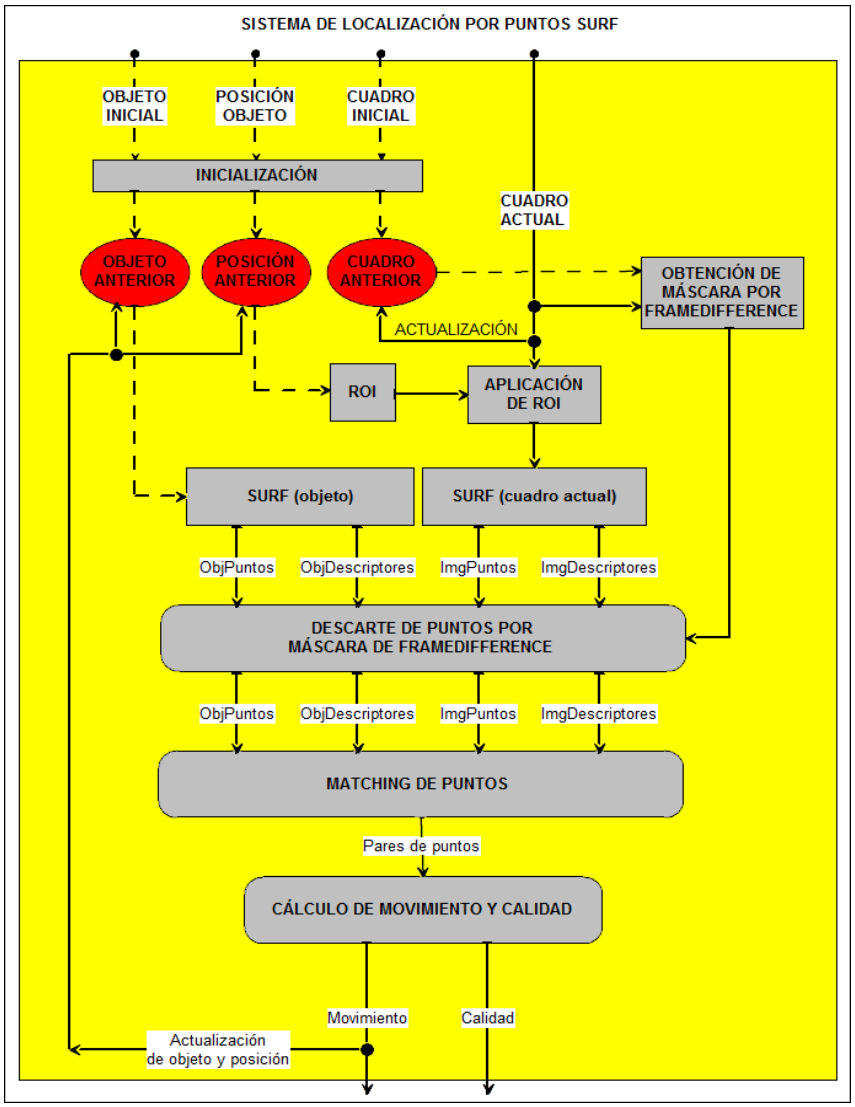


Figura 5-2: Sistema de localización por puntos SURF.

5.2.1 Obtención del *frame*

La obtención del *frame* actual que la cámara está grabando la realizaremos a través un *callback* establecido en las librerías de manejo de la cámara. Mediante la activación de dicho *callback*, cada vez que la cámara obtenga un nuevo *frame*, se realiza una llamada a la función "videoCallback" tal y cómo se ha configurado en la siguiente sentencia:

```
SNC::SetCallback(sncHandle, CALLBACK_DEC_VIDEO,
videoCallback, NULL);
```

Una vez configurada la función "videoCallback", cada vez que recibamos la llamada correspondiente a un nuevo *frame*, obtendremos en la estructura "pVideoInfo" toda la información referente al *frame* (tamaño del *frame*, datos obtenidos, codificación

utilizada…). Es ahora cuando estamos en posición de analizar la imagen obtenida ya que podemos utilizar toda esa información para pasársela a las siguientes funciones del proceso de análisis y estamos seguros de que es la imagen actual. Copiaremos la información que nos interesa en una IplImage, que es la estructura que utiliza OpenCV para almacenar imágenes, y que se utilizará para todos los análisis que necesitamos hacer.

La cabecera de la función videoCallback será la siguiente:

```
void __stdcall videoCallback(VIDEOINFO *pVideoInfo, unsigned
long param);
```

5.2.2 Extracción de puntos

La extracción de puntos de interés la vamos a realizar mediante la llamada a la función "cvExtractSURF" de OpenCV previa configuración de los parámetros necesarios.

```
CvSURFParams params = cvSURFParams(THRESHOLD, DESCRIPTORES);

cvExtractSURF(surf.obj, 0, &objectKeypoints,
&objectDescriptors, storage, params);
```

La extracción de puntos se hará tanto de la imagen de la cara como de la región obtenida en el *frame* actual al asignarle la ROI, para posteriormente realizar el *matching* o correspondencia de los puntos. Para una obtención de puntos realmente característicos es necesario configurar el parámetro "THRESHOLD" de manera que el resultado de la búsqueda sea un compromiso entre la cantidad de puntos obtenidos y la calidad de cada uno de ellos (un punto con mayor calidad será el correspondiente a ojos, comisura de los labios, la nariz, y los de menor calidad serán puntos en zonas homogéneas a partir de los cuales no podemos tomar una decisión fiable). Hay que tener en cuenta que cuantos más puntos y de mejor calidad tengamos, menor será el error que cometeremos en el *matching* y por consiguiente más fiable será la localización de la cara del profesor. El parámetro se configurará con un número mayor que 0 teniendo en cuenta que cuanto menor sea el parámetro más puntos se obtendrán pero de menor interés. La recomendación de OpenCV es poner un valor de 500, pero en nuestro caso sería necesario un cálculo empírico de este valor dado que la imagen de la cara tendrá unas dimensiones pequeñas y habrá un menor número de puntos de interés en una región menor, o dejando fijo el valor en 500 se irá ajustando el *zoom* óptico, obteniendo una mayor resolución en la cara y la posibilidad de

tener mayor número de puntos hasta llegar al mínimo. Para la realización de este ajuste impondremos una cantidad mínima de 10 puntos característicos en la imagen de la cara, de manera que si tenemos menos puntos de interés en la región seleccionada se indicará al administrador que debe realizar un aumento del *zoom* de seguimiento. Una vez se haya ajustado este *zoom* se tendrá un tamaño mínimo de la región que contiene la cara con al menos diez puntos de interés. A partir de este punto se utilizará siempre el mismo *zoom* y el mismo tamaño de la cara para todo el proceso de análisis. De esta forma, cuando obtengamos los puntos de interés en la ROI[1] del nuevo *frame*, obtendremos al menos los mismos diez puntos, o muy parecidos a los de la cara, y además los puntos que se obtengan en los márgenes; así, aunque la cara se haya movido, estaremos seguros de recogerla en esa región ampliada, ya que el movimiento *frame* a *frame* de la persona será muy pequeño.

5.2.3 *Matching* de puntos del objeto con los puntos del *frame*

El *matching* de puntos se basa en encontrar para cada punto del objeto su correspondiente punto en la región ampliada del siguiente *frame*. Para ello utilizaremos la distancia euclídea entre los descriptores de un punto del objeto y los descriptores de cada uno de los puntos del *frame*, habiendo descartado previamente los que tengan un valor del laplaciano distinto al del laplaciano del punto que buscamos (si no tienen el mismo laplaciano no pueden ser el mismo punto o uno parecido y al descartar puntos del *matching* se reduce el tiempo de procesado). Iremos calculando cada distancia euclídea y quedándonos con las dos distancias menores. Para dar validez al punto encontrado, se tendrá que cumplir que la distancia entre descriptores con el punto más parecido y la distancia entre descriptores con el segundo punto más parecido disten al menos un 60% (dis1 < 0.6*dist2), de esta manera conseguimos que el punto obtenido sea el más parecido y tenemos un margen considerable con todos los demás puntos.

En el caso de que se cumpla el requerimiento del 60% en las dos distancias menores, nos quedaremos con el punto correspondiente a la menor distancia y así habremos obtenido el primer par de puntos. En el caso de que no se cumpla el requerimiento del 60% en las dos distancias menores, descartaremos el punto modelo que estamos buscando ya que no

[1] ROI: region of *interest* (región de interés en la realizaremos la búsqueda de los puntos para realizar el *matching*). Será una cantidad mayor de pixeles por cada lado que el tamaño del *frame* objeto, y centrada en la posición calculada en la iteración anterior, reduciéndose de esta manera el tamaño de la zona de búsqueda y consiguientemente se reduce el tiempo de procesado (tanto en obtención de los puntos de interés como en el *matching* de los mismos).

tenemos las suficientes garantías de que el punto de menor distancia sea realmente el que le corresponde. Al descartarle haremos que no compute en el cálculo del movimiento global de la cara y evitaremos posibles errores en el movimiento de la cámara.

Una vez obtenidas todas las correspondencias válidas entre los puntos del objeto y los puntos del *frame*, se pensó en hacer una segunda correspondencia buscando asignaciones de todos los puntos del *frame* en los puntos del objeto. De esta manera se podría comprobar que se asignan los mismos pares de puntos que en el paso anterior. Esta idea, finalmente se reduce a comprobar que dos puntos del objeto no tengan el mismo punto del *frame* asignado, en cuyo caso se comprobará cuál de las dos asignaciones ofrece una menor distancia de descriptores (el par de puntos que más parecido es), eliminando de la lista de cómputo de movimiento el otro par de puntos.

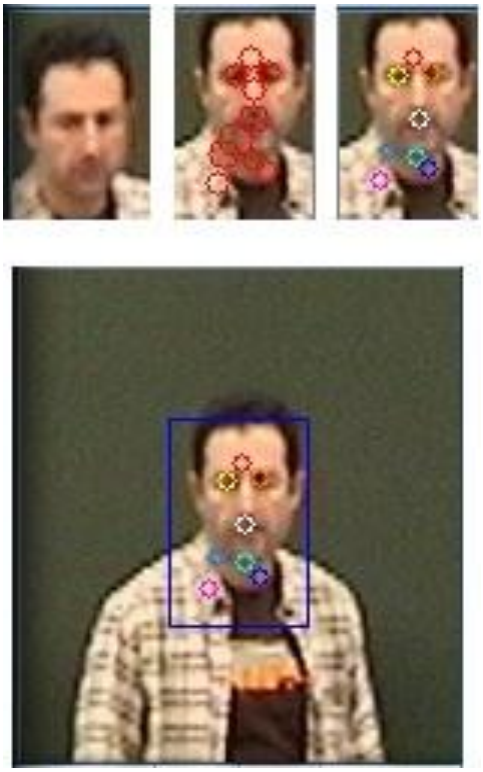


Figura 5-3: Asignación final de puntos de interés.

En la Figura 5-3 se puede observar un ejemplo del objeto inicial del seguimiento (arriba, izquierda), el objeto correspondiente al *frame* anterior (arriba, centro) con sus puntos de interés marcados con círculos rojos, el mismo objeto (arriba, derecha) con los puntos que han obtenido una relación de semejanza y la ROI (abajo) con la asignación de puntos y la localización del objeto. Se nota que se han perdido dos puntos de los rojos (nariz y barbilla) en las asignaciones debido a que en el cálculo de las distancias de descriptores no

se ha encontrado ninguna que cumpliese las restricciones anteriormente mencionadas. Es difícil encontrar diferencias notables entre la imagen del objeto del *frame* anterior con la imagen correspondiente a la ROI ya que son de *frames* contiguos y el movimiento de la persona en el tiempo entre *frames* es muy leve.

Una vez tenemos la lista que nos relaciona todos o al menos la mayor parte de los puntos del objeto con los puntos del *frame*, ya que algunos se han desestimado, estamos en disposición de calcular las distancias espaciales entre cada par de puntos (distancia en píxeles tanto en el eje x como en el eje y de la imagen) y, calculando la media del desplazamiento en cada uno de los ejes tendremos una estimación del movimiento global del objeto. En este momento habría que enviar el comando de movimiento necesario a la cámara.

5.2.4 Uso de *frame-difference*

Tras varias pruebas fallidas de calcular correctamente el desplazamiento sufrido por el objeto entre dos *frames* consecutivos se observó que los puntos del fondo de la escena, que eran mayoritarios en fondos texturados y que permanecían estáticos mientras que la cara se movía, a la hora de realizar el cómputo de movimiento hacían una aportación notable pero errónea (la asignación de los puntos era la correcta, pero al ser estáticos no deberían ser considerados en la estimación del posible movimiento ya que la media de los movimientos individuales se ve afectada).

Para poder descartar los puntos del fondo primero hay que saber que son de fondo, por lo tanto, y asumiendo que en la escena considerada lo único que se mueve en la región de interés es el profesor (mientras la cámara no recibe la instrucción de movimiento), realizando un *frame-difference* entre los dos *frames* consecutivos, obtendremos una imagen diferencia en la cual se puede analizar qué partes se han movido o no lo han hecho dentro de la ROI.

Previo paso a determinar qué parte se ha movido o no, hay que decidir cuál es el valor de ruido obtenido entre dos *frames* consecutivos. Para este cálculo utilizaremos la región del *frame* que no es de interés (*frame* completo menos la región de interés), calculando sobre ella la mediana del ruido. El valor que utilizaremos para umbralizar la imagen resultado del *frame-difference* será el valor de la mediana del ruido más 40 (margen que utilizamos para estar seguros de dejar por debajo todo el ruido al completo). Para mejorar la decisión en

una zona, antes de umbralizar con el valor calculado, realizamos la media de los píxeles de alrededor, asignado a cada pixel la media de una zona de conectividad 35. Una vez hecho todo esto tenemos una imagen que muestra en negro todas las zonas que asumimos que no se han movido y en blanco todas las zonas que si lo han hecho (en la zona del cuerpo, dependiendo de qué textura encontremos en las ropas del profesor, obtendremos en blanco todo o solamente los bordes). Nosotros queremos que toda la zona del cuerpo y la cara del profesor sea blanca completamente, por lo tanto hay que hacer una pequeña modificación sobre esa imagen, recorreremos la imagen de fila en fila hasta encontrar el primer pixel blanco, buscaremos si hay en la misma fila un último pixel en blanco, y así tendremos los dos extremos del profesor en esa fila. Ahora solo es necesario rellenar con blanco los píxeles que hay entre ambos blancos, y así lo haremos con todas las filas de la imagen (sólo es necesario recorrer la imagen por filas ya que el movimiento que realizará el profesor será mayoritariamente en horizontal).

Tras este proceso obtenemos una imagen en la que aparecen en blanco las zonas en las que se encuentra el cuerpo, o cualquier otra zona en la que se haya realizado algún tipo de movimiento. Dado que en el *frame* anterior tenemos localizada la región de interés, este aspecto no nos produce ningún problema ya que asumimos que lo que se ha movido dentro de esta zona es la persona que estamos siguiendo y lo que se haya movido fuera de esta zona no nos interesa.

Ahora ya estamos en posición de eliminar los puntos que hemos asumido que son de fondo de las listas de puntos para el *matching*, tanto de la imagen objeto como de la imagen completa, ya que en la imagen resultado del proceso anterior tendremos el valor ‘0’ si es de fondo y el valor ‘255’ si es parte de la persona o de la zona que se ha movido.

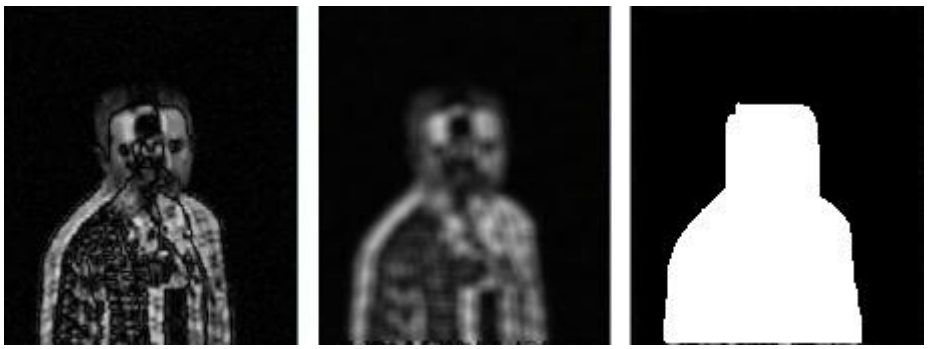


Figura 5-4: Consecución de la máscara de movimiento

En la Figura 5-4 se puede observar en el primer paso la imagen resta de dos *frames* consecutivos en la zona de la ROI, llegando a verse algunos píxeles del fondo que son claros debido al ruido de la cámara. En el segundo paso estos píxeles ya no se ven ya que se ha quitado el ruido, además se ve la figura de la persona difuminada debido al cálculo de la media con conectividad 35. En el último paso observamos claramente la máscara de movimiento obtenida. La figura conseguida es más ancha que el cuerpo de la persona ya que consta de dos partes, la posición en el *frame* anterior y la posición en el nuevo *frame*. A partir de esta máscara es fácil discriminar los puntos que no se han movido en la escena ya que si están en negro podremos descartarlos para la estimación del movimiento.

5.2.5 Localización del objeto

Una vez tengamos las asignaciones correctas de los puntos característicos de la cara, resultado del *matching* de puntos combinado con el resultado del proceso de *frame-difference* para la eliminación de puntos indeseados, podremos calcular cuál ha sido el movimiento global de la persona u objeto relacionando las coordenadas (x, y) de cada par de puntos y sacando la media del movimiento de los puntos en cada eje. Finalmente obtendremos un resultado similar al que muestra la Figura 5-5.

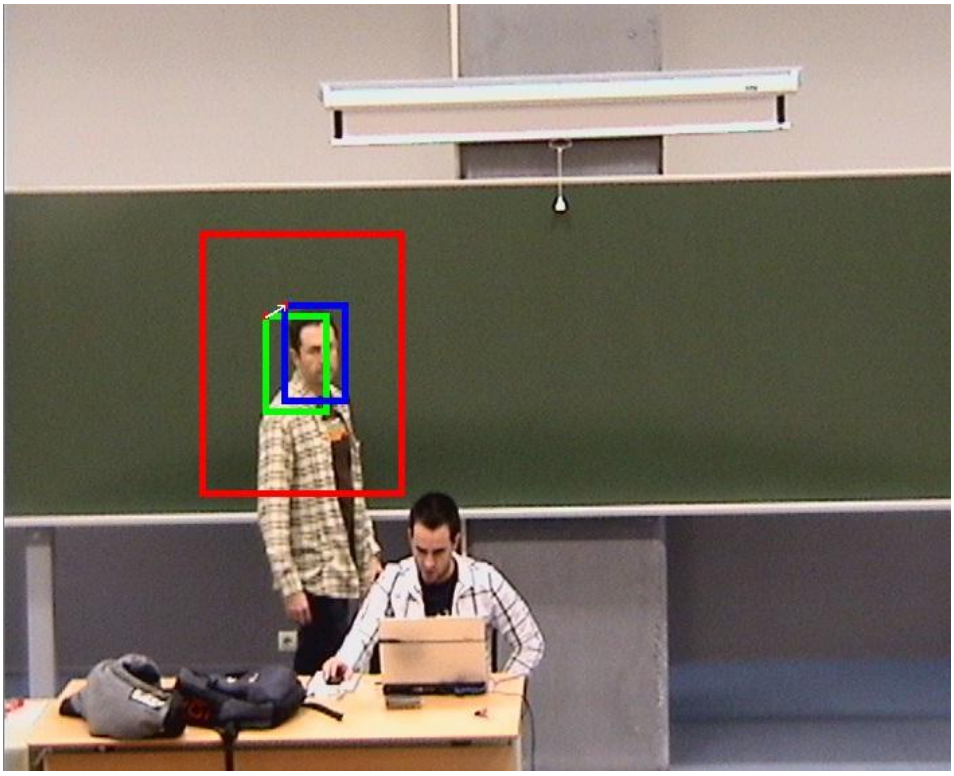


Figura 5-5: Relación entre la ROI, la posición anterior del objeto y la nueva posición del objeto.

Siendo el rectángulo rojo la ROI de búsqueda calculada a partir de la posición anterior del objeto (marcada con el rectángulo verde), y el rectángulo azul será la posición que se ha calculado que ocupa el objeto en la iteración actual. De esta manera el desplazamiento se obtiene calculando la distancia entre las esquinas superiores izquierdas de las posiciones anterior y actual del objeto.

Para este cálculo conocemos la posición de la esquina superior izquierda del *frame* actual del objeto relativa al *frame* anterior completo. A partir de esa esquina, en el nuevo *frame* podemos trazar su correspondiente ROI, que será de un tamaño superior al objeto en todas las direcciones. Los puntos característicos del nuevo *frame*, nos dan unas coordenadas relativas a la ROI, de manera que si un punto del nuevo *frame* con coordenadas relativas a la ROI (60,60), que ha hecho *matching* con uno del *frame* objeto con coordenadas (10,10), y sabiendo que la ROI es una extensión de 40 píxeles sobre la posición del objeto anterior, podremos calcular el desplazamiento que ha sufrido ese punto característico de la siguiente manera:

X_desp = x_nueva – margen_x – x_obj

X_desp = 60 – 40 – 10 = 10

Y_desp = 60 – 40 – 10 = 10

De esta manera tendremos un desplazamiento de (10,10) asumiendo positivo los desplazamientos hacia la derecha y hacia abajo y negativos los desplazamientos hacia la izquierda y hacia arriba del objeto. Realizamos el cálculo de los desplazamientos de cada uno de los puntos válidos y vamos acumulando el valor para realizar la media en X e Y dividiendo el valor suma de cada una de las dos coordenadas entre el número de puntos válidos. Así habremos obtenido la media de todos los desplazamientos en cada una de las coordenadas.

Finalmente sumaremos a las coordenadas de la esquina superior izquierda que tenía el objeto anterior los desplazamientos correspondientes a cada eje, obteniendo así dónde se encuentra situada, en relación con el *frame* completo, la nueva esquina superior izquierda. Este nuevo valor se quedará guardado para que en la siguiente iteración se pueda actualizar el objeto (ya que debido a los movimientos de la cara, del cuerpo o a los gestos del profesor, el objeto irá siendo distinto en cada iteración). Al actualizar el objeto en cada

iteración conseguimos un mejor resultado (se guarda un nuevo objeto sacado de la región que se ha calculado que ocupa en el *frame* actual para poder tener unos puntos característicos actualizados para la siguiente iteración), ya que *frame* a *frame* la diferencia es menor que si cogemos un objeto inicial y siempre buscamos ese objeto en todos los *frames*, que dará lugar a situación de pérdida del objeto en los momentos en que el profesor se dé la vuelta para escribir en la pizarra.

Además de para actualizar la representación o descripción del objeto, al conocer cuánto se ha movido el objeto, estamos en disposición de ordenar a la cámara los movimientos pertinentes para mantenerlo siempre centrado en la imagen, logrando de este modo un seguimiento activo.

5.2.6 Fiabilidad del proceso

Se utilizará la medida de fiabilidad del proceso para poder decidir entre el resultado de un método de localización o el del otro método. En el método de localización por puntos SURF se basa el cálculo de esta medida en el posible error cometido en los *matching* de los puntos. La forma que se ha utilizado para el cálculo de la fiabilidad es la utilización de las medias de las distancias obtenidas entre los descriptores de cada par de puntos válido.

Una vez calculada la calidad referente al método surf, será necesaria una normalización de este valor para poder realizar una comparación con la calidad del método de localización por histograma de color. Se hará un relación entre los valores 0 – 1 de la gráfica con los valores 0.1 y 0 de las distancias. Siendo la distancia media 0 el máximo valor de calidad 1 y siendo la máxima distancia permitida de 0.1 el valor mínimo de calidad 0.

Calidad = f(dis) = (-1/0.1)*dis + 1;

En la Figura 5-6 se puede observar la gráfica que describe esta función, en la cuál se ve claramente que cuanto mayor sea la distancia media de los descriptores obtenida menor será la calidad resultante hasta la distancia máxima de 0.1 que se le asigna la calidad mínima. La distancia mínima en la gráfica aparece con el valor 0 aunque a lo largo del proceso de seguimiento automático es casi imposible llegar a esa distancia de descriptores y por consiguiente la calidad nunca será la máxima posible. La única forma de obtener esa máxima calidad sería buscando el objeto en el mismo *frame* del que ha sido sacado, pero

en nuestro caso siempre se busca un objeto del *frame* anterior, que aunque no se haya movido nada en la escena siempre tendremos el ruido producido por la cámara al grabarla.

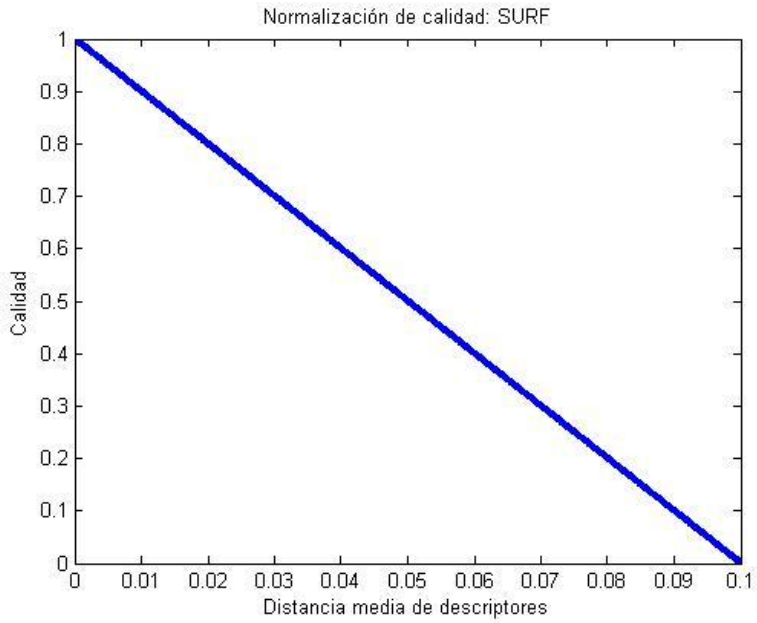


Figura 5-6: Normalización de la calidad obtenida.

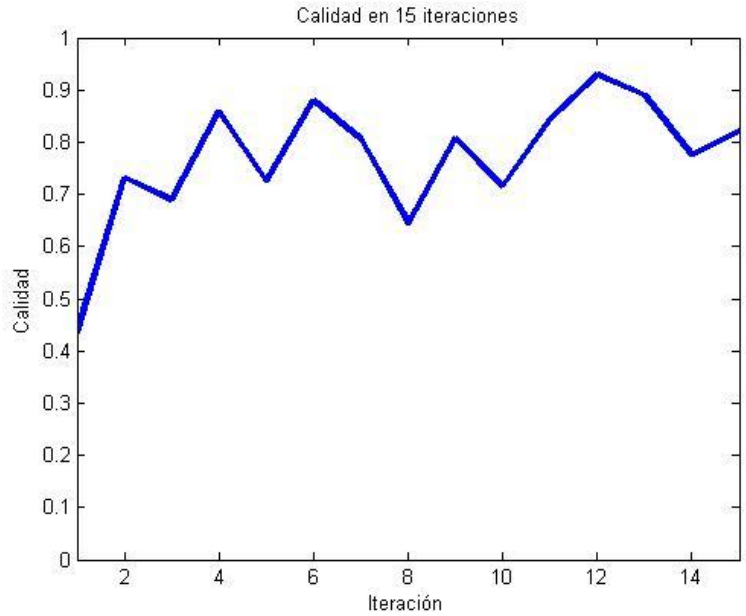


Figura 5-7: Ejemplo de evolución de la calidad normalizada.

La Figura 5-7 nos muestra las calidades obtenidas a lo largo de 15 iteraciones (localización en 15 *frames* consecutivos) del seguimiento automático por procesado de puntos SURF. En ella se observan varios picos de bajada de calidad, que coinciden en el eje temporal con

cambios más bruscos de la posición del profesor (giros hacia la pizarra, aceleraciones del movimiento).

5.3 Seguimiento mediante histograma de color

Al igual que en la sección anterior se ha hablado del principal método de seguimiento del profesor, seguimiento por puntos SURF, en esta sección se va a hablar del método secundario o complementario para el seguimiento del profesor, seguimiento mediante histograma de color. En este caso se hará una búsqueda en el nuevo *frame* de la zona que tiene una distribución de colores lo más semejante posible a la distribución de colores que el objeto tiene.

Para ello se obtendrá el histograma de color del objeto, y mediante la llamada a las funciones de OpenCV backproject y camshift se obtendrá la posición en la que se encuentra el cuadro más semejante al objeto que se está siguiendo.

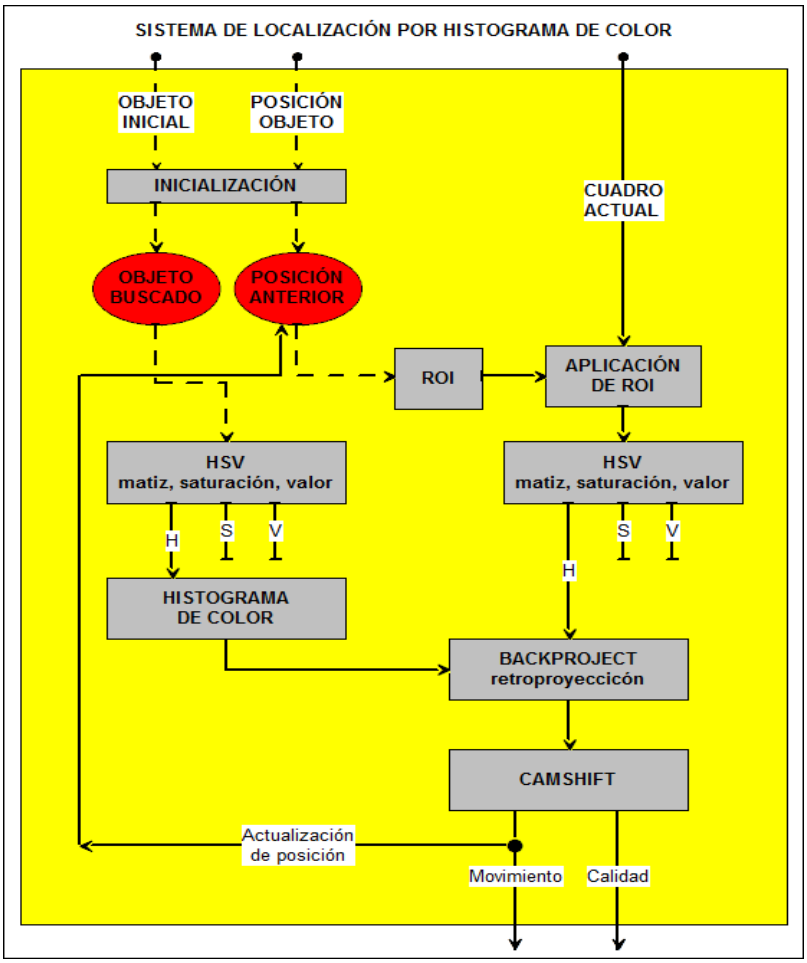


Figura 5-8: Sistema de localización por histograma de color.

Al igual que en el sistema de localización por puntos SURF, este esquema tiene también una lazo de actualización de la posición del objeto, que tal y como se muestra en la Figura 5-8 funciona internamente si se trabaja con este algoritmo únicamente. En el caso de trabajar en paralelo con la localización por puntos SURF, la actualización de posición se realizará en un lazo externo, común a ambos algoritmos debido a que unas veces se considerará posición válida la calculada por un algoritmo, y otras veces la calculada por el otro dependiendo de las calidades obtenidas en cada uno de ellos. Por lo tanto es necesario que la actualización de posición anterior sea controlada externamente a ambos algoritmos.

5.3.1 Obtención del *frame*

El proceso de obtención del nuevo *frame* para el seguimiento mediante histograma de color se realiza de la misma manera que se ha descrito en la sección de obtención del *frame* para el seguimiento por puntos SURF. Este proceso es común para ambos métodos de seguimiento, la misma imagen obtenida se pasa tanto a un método como al otro.

5.3.2 Extracción del histograma de color

Lo primero que se necesita hacer para comenzar la búsqueda del objeto es cambiar la descripción del color que se tiene del modelo RGB al modelo HSV, ya que el modelo HSV tiene una componente que es directamente el color o tinte del pixel (siendo las otras dos componentes saturación y brillo), tal y como se muestra en la Figura 5-9.

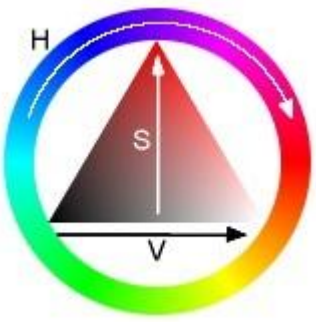


Figura 5-9: Modelo HSV.

De esta manera al calcular el histograma sobre la componente H, obtenemos la distribución de los colores en el objeto. Y a partir de aquí se podrá buscar la zona con una distribución semejante en sucesivos *frames* mediante la llamada a las correspondientes funciones de OpenCV backproject y camshift.

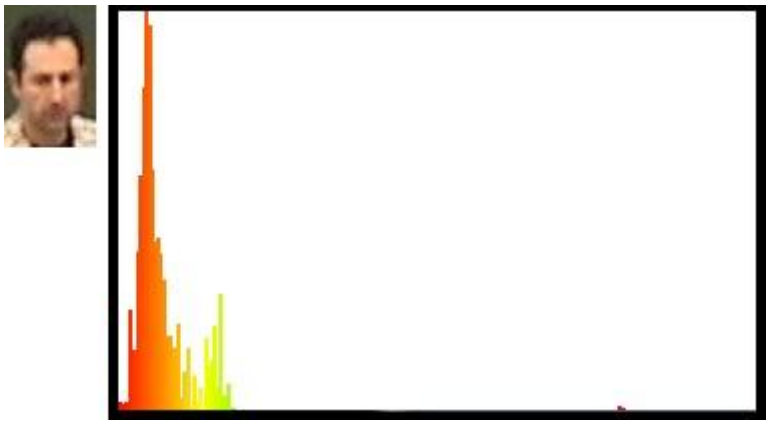


Figura 5-10: Cara que se está siguiendo e histograma de color correspondiente.

En la Figura 5-10 observamos cómo queda el histograma de color de la cara sacado del canal H (matiz) del modelo HSV. Este será el histograma que se utilizará para determinar la posición de la cara dentro del nuevo *frame*.

5.3.3 Búsqueda de la región del *frame* con mayor probabilidad de ser el objeto

Una vez se tiene el histograma de color del objeto que se desea seguir, será necesaria la llamada a las funciones backpoject y camshift que serán las encargadas de calcular la retroproyección del histograma en la imagen completa y de obtener a partir de la retroproyección la región con una distribución más parecida a la distribución que el histograma nos muestra, por lo tanto con una mayor probabilidad de ser el objeto que buscamos.

La función camshift directamente nos devuelve el rectángulo que engloba esta región, y por lo tanto directamente de ese rectángulo tendremos la nueva esquina. También nos devuelve un parámetro a partir del cual se puede calcular una calidad. Este parámetro hace referencia a la cantidad de área del rectángulo que se asocia al objeto, por lo tanto se puede sacar una probabilidad de que ese rectángulo sea el objeto buscado.

Esta calidad directamente se encuentra entre los valores 0 y 1 siendo el 0 la menor calidad posible y 1 la mayor. Ahora estamos en disposición de comparar las calidades de ambos procesos de seguimiento y decidir cuál está siendo mejor en cada una de las iteraciones en función a las calidades obtenidas.

5.3.4 Localización del objeto

Una vez tengamos la región del *frame* con mayor probabilidad de ser el objeto que se busca, resultado del seguimiento por histograma de color, podremos calcular cuál ha sido el movimiento global de la persona objeto realizando unas sencillas operaciones entre las coordenadas de la esquina superior izquierda anterior y la obtenida en esta iteración. Hay que tener en cuenta que las nuevas coordenadas obtenidas vienen en referencia a la ROI, por tanto hay que hacer un cambio de coordenadas y ponerlas en relación al *frame* completo. Una vez estén las coordenadas en relación al *frame* completo simplemente habrá que hallar el desplazamiento entre las anteriores y las calculadas en la iteración actual.

En la Figura 5-5 se pueden volver a observar las relaciones de coordenadas que existen entre el *frame* completo, la ROI, el objeto anterior, el objeto encontrado y el desplazamiento sufrido.

Si tenemos que las coordenadas de la esquina superior izquierda del objeto anterior son (100, 100) respecto al *frame* completo; en la iteración actual se ha calculado que la esquina superior izquierda del objeto tiene unas coordenadas (30, 30) respecto de la ROI; por otro lado, tenemos que la posición de la ROI se ha calculado a partir de la posición del objeto anterior aumentando un margen de 40 píxeles hacia cada lado del objeto, por lo tanto, las coordenadas de la esquina superior izquierda de la ROI son (100-40= 60, 100-40= 60). Finalmente las coordenadas del objeto actual respecto al *frame* completo serán (60+30=90, 60+30=90) y el desplazamiento habrá sido de (90-100= -10, 90-100= -10).

X_ROI = x_obj – 40 = 100 – 40 = 60;

Y_ROI = y_obj – 40 = 100 – 40 = 60;

X_nueva = X_ROI + x= 60 + 30 = 90;

Y_nueva = Y_ROI + y= 60 + 30 = 90;

X_desp = X_nueva – x_obj = 90 – 100 = -10;

Y_desp = Y_nueva – y_obj = 90 – 100 = -10;

De esta manera tendremos un desplazamiento de (-10,-10) del objeto, asumiendo positivo los desplazamientos hacia la derecha y hacia abajo y negativos los desplazamientos hacia la izquierda y hacia arriba.

Finalmente se guardarán las coordenadas de la esquina superior izquierda calculada en la iteración actual respecto del *frame* completo para comparar en la siguiente iteración únicamente, ya que en este proceso de seguimiento no se actualiza el objeto, se hace un seguimiento en función al primer objeto seleccionado (es un proceso complementario al de seguimiento por puntos SURF y una forma de recuperación, en ningún caso es el proceso principal).

En este momento se estaría en disposición de ordenar a la cámara el movimiento pertinente para mantener enfocado el objeto en el caso de que la calidad del proceso de seguimiento principal bajase por debajo de la que se ha obtenido en este proceso.

5.3.5 Fiabilidad del proceso

En el proceso de seguimiento por histograma de color es más fácil saber la fiabilidad que tenemos a la hora de encontrar la posición del objeto ya que son las propias funciones las que nos devuelven el parámetro de área de la componente conexa; dividiendo este área entre el área que ocupa el objeto, tenemos una relación de la cantidad de objeto que se ha encontrado y el tamaño total, relación que podemos asumir como un indicador de calidad entre 0 y 1, siendo 0 la menor calidad y 1 la mayor.

En la Figura 5-11 se observa un ejemplo de cómo quedaría la gráfica de calidades del seguimiento por histograma de color a lo largo de 15 iteraciones del algoritmo (localización en 15 *frames* consecutivos). Se observa una oscilación muy pequeña en torno a la calidad 0.35. Esto se debe a que el algoritmo de búsqueda por histograma de color la mayor parte del tiempo encuentra la cara, pero dependiendo de hacia dónde esté mirando el profesor variará el valor obtenido. Si el profesor gira un poco la cara, tendremos una región menor con el color de la cara, ya que siempre se busca el color de cara del objeto inicial. El hecho de que el valor de calidad esté en torno a los 0.35 se debe a que el algoritmo no cuenta como componente conexa la zona de pizarra, el pelo y la zona de la camisa, y es mucho área que se desprecia respecto al tamaño total del objeto.

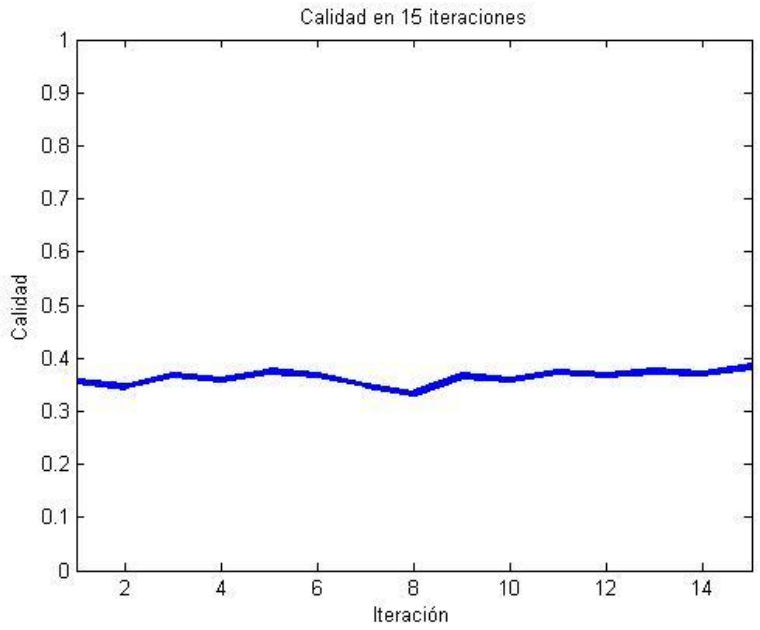


Figura 5-11: Calidades obtenidas en el seguimiento por histograma de color.

5.4 Determinación de la posición del objeto

Cada uno de los procesos de seguimiento descritos anteriormente devuelve tanto una localización del objeto en el nuevo *frame* a través de su esquina superior izquierda como una calidad o fiabilidad con la que se ha calculado esa localización.

Para determinar la nueva posición del objeto se comparan las calidades obtenidas en cada uno de los procesos y se decide a partir de ellas cuál de los dos procesos ha funcionado mejor, quedándonos como nueva posición la obtenida por el proceso que mayor fiabilidad nos ha dado. Si el seguimiento por puntos SURF obtiene una calidad superior a la obtenida por el seguimiento por histograma de color, nos quedaremos con la posición obtenida por ese proceso, y viceversa.

En el caso de que de que las calidades sean exactamente iguales, nos quedaremos con la posición obtenida por el seguimiento por puntos SURF, ya que se le ha otorgado la condición de procedimiento principal de seguimiento del objeto.

En el caso de que ambos procedimientos obtengan unas calidades demasiado bajas (por debajo de 0.2), habrá que decidir qué hacer en este momento, ya que las fiabilidades no son muy significativas de que el objeto encontrado sea exactamente el que buscamos, y quizás no sea ni parecido. En este momento tendremos la posibilidad de decidir no mover la

cámara o decidir finalizar el seguimiento ya que todas las iteraciones siguientes serán probablemente erróneas. Otra posibilidad que existe a la hora de detectar una merma de calidad en ambos algoritmos es lanzar un algoritmo de recuperación que permita al sistema seguir funcionando correctamente, del cual será explicada la idea principal en la sección 5.6.

5.5 Movimiento de la cámara

Una vez se tiene la nueva posición del objeto y por lo tanto el desplazamiento sufrido desde el anterior *frame*, el último paso para completar el seguimiento activo será el de mover la cámara de manera que el objeto se mantenga centrado en la imagen capturada. Evidentemente este movimiento será hacia una posición en la que se espera que esté el objeto, ya que en el tiempo transcurrido durante el procesado y movimiento de la cámara la persona ha podido variar levemente su posición.

La realización de los movimientos de la cámara lleva consigo varios aspectos que hay que solventar. En primer lugar, es necesario saber qué desplazamiento en píxeles se produce en la imagen que toma la cámara en relación con la magnitud que se envía en el comando de movimiento. En segundo lugar, dado que el tiempo que tarda en moverse la cámara es muy superior al intervalo entre cuadros, es necesario adoptar ciertas medidas para que el seguimiento funcione correctamente.

5.5.1 Calibración del movimiento de la cámara

Se ha añadido un módulo de calibración automática del movimiento de la cámara en el que se define un *zoom* óptimo para el seguimiento y se obtienen las relaciones de movimiento para ese *zoom*. Esta calibración sólo será necesaria realizarla una vez en la aplicación del aula ya que se creará un fichero con la relación del movimiento en píxeles de la cámara con la unidad de magnitud introducida en el comando.

Para llevar a cabo la calibración se ordenan varios movimientos a la cámara, cambiando en cada uno de ellos tanto la magnitud del comando como la dirección del movimiento a realizar. De esta manera se obtendrán distintas relaciones para distintos ejes y se espera que las relaciones de movimientos dentro del mismo eje sean similares para distintas magnitudes (incluyendo movimientos negativos de la cámara).

Lo primero que se hace es seleccionar el *zoom* óptimo. Este proceso se realizará por parte de la persona que este configurando inicialmente la aplicación. Esta persona se encargará de ir haciendo *zoom* y seleccionando la región que ocupa su propia cara hasta obtener un tamaño de región suficientemente grande para que el seguimiento funcione correctamente (si la imagen es demasiado pequeña no se obtendría una cantidad suficiente de puntos para el seguimiento por SURF). En este caso se ha definido un tamaño mínimo de 45x70 píxeles para la región de la cara ya que en las pruebas realizadas el seguimiento comenzó a funcionar a partir del *zoom* correspondiente a este tamaño (en las pruebas el tamaño era de 46x71, pero como es muy difícil ajustar a un píxel la selección, el tamaño se ha redondeado a 45x70 y se utilizará como corte por parte de la aplicación).

Una vez definido y configurada la cámara con el *zoom* de seguimiento adecuado, para poder decidir la cantidad de movimiento que la cámara ha descrito es necesario tener un sistema que reconozca cuánto se ha movido cierto punto tras el envío del comando. Este sistema se basa en la extracción de puntos por SURF (aprovechando que ya está desarrollado en la aplicación). Se extraen los puntos de interés del primer *frame* y tras realizar el movimiento completo se extraen los puntos de interés del *frame* actual y se hace un *matching* entre los puntos del primer *frame* y los puntos del *frame* actual. En este momento existen dos maneras de proceder:

- **Administrador:** entraría en juego el administrador de la cámara o quién la esté configurando en ese momento y se encargaría de seleccionar el mismo punto tanto en la imagen inicial como en la imagen final, después de realizar el movimiento de la cámara. De esta manera se tendrá con certeza el movimiento total sufrido por el punto seleccionado y por consiguiente el movimiento total de la escena. Repitiendo este proceso en cada una de las direcciones se obtendrán las distintas relaciones de movimiento de la cámara.

- **Automático:** se podría realizar una operación parecida a la anterior pero automáticamente. Se trata de configurar la extracción de puntos SURF con un THRESHOLD de 4500 para asegurarnos de que los puntos son pocos pero muy significativos (esquinas, puntos de corte de varios objetos…). Se comienza colocando la cámara en posición HOME y seguido se aplica el *zoom* correspondiente al seguimiento. Una vez ha llegado a su posición final se obtiene el *frame* y se almacena, se aplica un movimiento en cualquiera de las direcciones y se

vuelve a obtener el *frame* cuando haya llegado a su posición final. Se repite el proceso en cada una de las cuatro direcciones y una vez se tengan los cuatro *frames* iniciales y los cuatro *frames* finales, se realiza la extracción de puntos y el *matching* en cada dirección entre los *frames* iniciales y los finales. En cada dirección se calcula la media de movimiento sufrido por los puntos que han hecho *matching* y se obtiene la relación de movimiento de la cámara en cada una de las direcciones.

Finalmente se ha optado por incluir el proceso automático para evitar trabajo al administrador de la cámara a la hora de la calibración. En el momento de probarlo se hicieron varias pruebas y se obtuvieron los resultados de las siguientes tablas:

	Comando		Movimiento en 20 iteraciones		Relación
	PAN	TILT	Horizontal	Vertical	final
Derecha	A (10)	0	-220	-6	-1,1
	7	0	-150	-6	-1,1
	4	0	-83	-3	-1,0
	2	0	-43	-2	-1,1
	1	0	-22	-1	-1,1
	0	0	1	0	0,0
Izquierda	FFFF (-1)	0	22	0	1,1
	FFFE (-2)	0	44	0	1,1
	FFFC (-4)	0	87	1	1,1
	FFF9 (-7)	0	153	-1	1,1
	FFF6 (-10)	0	218	-1	1,1

	Comando		Movimiento en 1 iteración		Relación
	PAN	TILT	Horizontal	Vertical	final
Derecha	C8 (200)	0	-214	-7	-1,1
	8C (140)	0	-153	-5	-1,1
	50 (80)	0	-86	-2	-1,1
	28 (40)	0	-45	-1	-1,1
	14 (20)	0	-23	0	-1,2
	0	0	0	0	0,0
Izquierda	FFEC (-20)	0	22	1	1,1
	FFD 8(-40)	0	44	1	1,1
	FFB0 (-80)	0	88	1	1,1
	FF74 (-140)	0	151	2	1,1
	FF38 (-200)	0	217	2	1,1

	Comando		Movimiento en 20 iteraciones		Relación
	PAN	TILT	Horizontal	Vertical	final
Arriba					
	0	4	-3	87	1,1
	0	2	-2	45	1,1
	0	1	-1	22	1,1
	0	0	0	0	0,0
Abajo	0	FFFF (-1)	0	-22	-1,1
	0	FFFE (-2)	2	-44	-1,1
	0	FFFC (-4)	1	-88	-1,1
	0	FFF9 (-7)	0	-153	-1,1
	0	FFF6 (-10)	3	-220	-1,1

Tabla 5-1: Calibraciones horizontales y vertical sin *zoom*.

	Comando		Movimiento en 20 iteraciones		Relación
	PAN	TILT	Horizontal	Vertical	final
Derecha					
	4	0	-224	-4	-2,8
	2	0	-123	-2	-3,1
	1	0	-62	-1	-3,1
	0	0	1	0	0,0
Izquierda	FFFF (-1)	0	61	1	3,1
	FFFE (-2)	0	124	2	3,1
	FFFC (-4)	0	243	7	3,0
	FFF9 (-7)	0	423	4	3,0

	Comando		Movimiento en 1 iteración		Relación
	PAN	TILT	Horizontal	Vertical	final
Derecha					
	50 (80)	0	-243	-3	-3,0
	28 (40)	0	-123	-2	-3,1
	14 (20)	0	-61	-1	-3,1
	0	0	0	0	0,0
Izquierda	FFEC (-20)	0	61	1	3,1
	FFD8 (-40)	0	120	2	3,0
	FFB0 (-80)	0	241	4	3,0
	FF74 (-140)	0	422	5	3,0

	Comando		Movimiento en 20 iteraciones		Relación
	PAN	TILT	Horizontal	Vertical	final
Arriba					
	0	4	-4	257	3,2
	0	2	-2	129	3,2
	0	1	-1	64	3,2
	0	0	0	0	0,0
Abajo	0	FFFF (-1)	1	-64	-3,2
	0	FFFE (-2)	2	-127	-3,2
	0	FFFC (-4)	4	-254	-3,2

Tabla 5-2: Calibraciones horizontales y vertical con *zoom* 2159.

De estos datos se pueden sacar varias conclusiones que facilitarán el funcionamiento del algoritmo de seguimiento. Se puede observar que mientras nos movamos dentro del mismo eje, la relación se mantiene en magnitud (evidentemente se ven con signo distintos ya que se mueve en distinto sentido), por lo tanto, sólo hay que guardar una de las relaciones en cada eje. También se ve claramente que para un distinto *zoom* tendremos una relación distinta de movimiento, lo que era de esperar, ya que el hecho de cambiar la configuración óptica de la cámara no hace que se modifique el movimiento de los motores de la misma. Esta última conclusión no facilita nada el seguimiento, es más, lo complica en cierto modo. Tendremos que tener para cada posible *zoom* una relación de movimiento de cada eje y en el caso de poder hacer *zoom* durante el seguimiento habría que estar cambiando de una relación a otra. En este proyecto se ha definido un *zoom* constante para el seguimiento de manera que este proceso de cambio de relación se evita, aunque en un futuro se podría mejorar e integrar el cambio de *zoom* durante el seguimiento automático (a la vista de las calibraciones parece ser que también se mantiene una relación entre el *zoom* y la cantidad de píxeles que se mueve en función del comando enviado).

En las tablas se observa que al hacer un movimiento únicamente horizontal, se produce también un pequeño desplazamiento vertical. Esto es debido al cambio de perspectiva que

se realiza sobre la escena al girar una cierta cantidad la cámara (o a que la plataforma de soporte de la cámara no esté correctamente nivelada). Como el profesor se va a desplazar mayoritariamente en el eje horizontal y siempre en un plano paralelo a la pizarra, se podrá pasar por alto este pequeño desplazamiento ya que en el seguimiento el profesor seguirá dentro de la ROI debido a que el margen que añade la misma es mucho mayor.

Tras el proceso de calibración, con la obtención de las relaciones de movimiento, y con el seguimiento automático activado, con su cálculo del desplazamiento del objeto, nos disponemos a mover la cámara a la posición adecuada. Para ello habrá que calcular las magnitudes que hay que colocar en el comando de movimiento. Simplemente habrá que realizar unas sencillas operaciones:

PAN = - desplazamiento_X / relación_X;

TILT = desplazamiento_Y / relación_Y;

Con ello conseguimos que la cámara, tras la ejecución del comando de movimiento, se quede enfocando la región donde se ha calculado que estaba el objeto.

5.5.2 Gestión del retardo en el movimiento

En un principio se realizaba el movimiento de la cámara tras cada iteración del algoritmo de seguimiento automático, pero después de muchas pruebas se comprobó que no era posible realizarlo de esta manera. En la Tabla 5-3 se puede observar que esto es debido a que la cámara tarda alrededor de 900 ms en cada movimiento (dependiendo de lo grande que sea el desplazamiento tardará un poco más o un poco menos de los 900 ms). Si a estos 900 ms se le suma lo que tarda el algoritmo de seguimiento automático, unos 75 ms de localización por puntos SURF y 4 ms por localización por histograma de color, nos ponemos cerca del segundo en cada iteración. Por lo tanto la premisa que teníamos de que el objeto a seguir se desplaza muy poquito *frame* a *frame* (40 ms entre *frames* y que trabajamos a 25 fps) hay que desecharla ya que en un segundo no podemos decir lo mismo, el objeto podría haberse movido lo suficiente como para salirse de la ROI, perdiéndose el seguimiento automático. Además de que al realizar movimientos en cada iteración la cámara nunca está estática y no deja una buena sensación.

En la Tabla 5-3 se muestran los tiempos obtenidos a lo largo de 20 iteraciones del seguimiento automático con sus respectivos movimientos de cámara cuando han sido

necesarios. Notar que cuando el tiempo del movimiento es de 0 ms es debido a que no ha habido movimiento del objeto o a que las calidades tanto del seguimiento por SURF y del seguimiento por histograma de color no cumplían los requisitos.

Iteración	Total (ms)	Movimiento (ms)	Seguimiento (ms)	SURF (ms)	Histograma (ms)
1	1331	744	587	583	4
2	996	932	64	60	4
3	71	0	71	67	4
4	73	0	73	69	4
5	842	774	68	64	4
6	72	0	72	68	4
7	75	0	75	71	4
8	859	787	72	68	4
9	1070	992	78	75	3
10	998	930	68	65	3
11	800	733	67	63	4
12	1001	933	68	64	4
13	997	931	66	62	4
14	799	735	64	60	4
15	870	803	67	64	3
16	1109	1037	72	68	4
17	1077	1001	76	72	4
18	1039	976	63	59	4
19	1059	990	69	66	3
20	999	936	63	59	4

Tabla 5-3: Tiempos de procesado y movimiento de cámara para 20 iteraciones.

En la tabla de tiempos podemos observar que en la primera iteración el procesado por puntos SURF tarda muchísimo tiempo, lo cual es debido a que en la primera iteración no tenemos una ROI, si no que trabajamos con el *frame* completo y hay demasiados puntos para hacer *matching*. En las siguientes iteraciones, con la asignación de la ROI se reduce significativamente el tiempo de este procesado.

La posible solución a este problema reside en mover la cámara el menor número de veces, solamente las imprescindibles. Para ello se propone la solución por movimiento acumulado. Esto quiere decir que en cada iteración del seguimiento automático se van acumulando los desplazamientos, pero no moviendo la cámara, de manera que cuando la acumulación en alguno de los ejes supere cierta cantidad (esta cantidad se regirá por el margen que añade la ROI sobre la posición del objeto, de manera que cuando el objeto se acerque al borde de la ROI, sin llegar a salirse, se efectúe el movimiento), el algoritmo

enviará el comando necesario para realizar el movimiento. De esta manera se reducirán los puntos de ruptura del seguimiento automático al reducir el número de veces que hay que mover la cámara y también se mejora la sensación visual al tener la cámara estática un mayor tiempo. Por otro lado se mejora la localización del objeto en el *frame* al transcurrir menor tiempo entre *frame* y *frame*.

Otro aspecto que puede influir en el funcionamiento del seguimiento automático es el hecho de que en cada cambio de posición de la cámara, en un principio, no tendremos la posibilidad de usar el *frame-difference* para descartar puntos no válidos ya que la diferencia entre el *frame* anterior (imagen de la posición anterior al movimiento de la cámara) y el nuevo *frame* (imagen de la nueva posición de la cámara) será demasiado grande y por lo tanto daría como resultado una discriminación errónea. Habría que desechar el *frame* anterior al movimiento, realizar una iteración en la que sólo se guarde el *frame* para en la siguiente iteración poder utilizar ese *frame* y el nuevo *frame* actual para el *frame-difference*.

Como conclusión a esta sección se podría decir que el seguimiento automático del profesor en el aula es factible, aunque habría que llegar a una mejor compenetración de los algoritmos de localización del objeto con la cámara. Se ha visto que los algoritmos de localización funcionan correctamente en una secuencia de vídeo señalando la posición del profesor en cada *frame*, por lo tanto esta parte del seguimiento estaría resuelta por completo; habría que mejorar la tardanza de la cámara en las transiciones de los movimientos y establecer un procedimiento para retomar el seguimiento después de un movimiento de la cámara.

5.6 Recuperación en caso de fallo en el seguimiento

El algoritmo de recuperación en caso de fallo que se plantea tiene como base la localización por histograma de color. Habrá que situar la cámara en la posición HOME (para asegurarnos de que aparece el profesor en la escena) y realizando sucesivas iteraciones del algoritmo se esperará a que aparezca la cara del profesor y que la localización por histograma de color de un buen resultado. Una vez localizada la cara del profesor se volverá a ajustar el *zoom*, dejándolo configurado para el seguimiento automático, y se centrará la cámara en la región que ocupa la cara. Se realizará una nueva iteración del histograma de color para volver a localizar la cara con el *zoom* adecuado

(*zoom* específico para el seguimiento activo) y poder determinar tanto la posición inicial como el nuevo objeto a seguir para inicializar de nuevo los algoritmos de seguimiento. A partir de aquí estaremos en el mismo punto del que se partía cuando se seleccionaba la cara manualmente en el lanzamiento del seguimiento automático. Tendremos un nuevo objeto de búsqueda para la localización por puntos SURF, ya que necesita siempre un objeto actualizado, y para el histograma de color seguiremos con el objeto inicial del principio, ya que siempre va a ser más ajustado a la búsqueda que el que se pueda generar en una nueva localización por el algoritmo.

Este algoritmo de recuperación del sistema de seguimiento automático no ha sido desarrollado en este proyecto. Aquí se ha plasmado la idea principal para recuperarlo, dando los pasos que se creen necesarios para la consecución de un algoritmo estable.

6 Subsistema de Emisión

Para realizar la emisión del flujo de vídeo a distintos puntos remotos será necesaria la utilización de las aplicaciones que se han explicado en anteriores capítulos, o dotar a la aplicación principal desarrollada en este proyecto de la posibilidad de emisión del flujo que luego pueda gestionar un proveedor de servicios. Para ello se optó inicialmente por utilizar las librerías de Gstreamer como base para la creación del módulo de emisión del flujo de vídeo. Decir que, debido a la complejidad de las herramientas involucradas y a las restricciones temporales del proyecto, aún no se ha llegado a la solución final del módulo de emisión, solamente se ha conseguido hacer funcionar ciertas partes independientemente.

6.1 Introducción a Gstreamer

Gstreamer es una biblioteca escrita en el lenguaje de programación C que sirve para desarrollar aplicaciones dedicadas al manejo de distintos medios. Las aplicaciones que nos permite crear van desde aplicaciones audiovisuales para la reproducción de audio y vídeo hasta aplicaciones más complejas para mezclar audio y vídeo, codificar e incluso realizar un *streaming* de los mismos.

Gstreamer basa su funcionamiento en complementos, los cuales pueden proveer a nuestra aplicación de los *codecs* necesarios para su correcto funcionamiento o de otras muchas funcionalidades. A través del siguiente ejemplo se podrá entender más fácilmente el funcionamiento de Gstreamer. Para este ejemplo usaremos la siguiente línea de comandos que servirá para reproducir un archivo MP3:

```
gst-launch-0.10 filesrc location="archivo.mp3" ! decodebin ! alsasink
```

En esta línea se especifica el nombre del programa que se usará (gst-launch-0.10) y los distintos elementos de los que estará compuesto el *pipeline*.

- **filesrc location="archivo.mp3":** es la ruta del archivo con el que trabajaremos.
- **!:** indica un *enlace a* y sirve para marcar las distintas conexiones entre los módulos que se van a usar.
- **decodebin:** permite identificar el tipo de datos o archivo con los que se va a trabajar.

- **alsasink:** este módulo permite conectar el sistema creado a nuestro sistema de sonido. Alsasink es el utilizado en Linux, en Windows se podrá utilizar autoaudiosink para realizar la salida del audio.

En la Figura 6-1 se pueden observar los distintos módulos utilizados en el ejemplo y las correspondientes conexiones que se han creado entre cada uno de ellos.

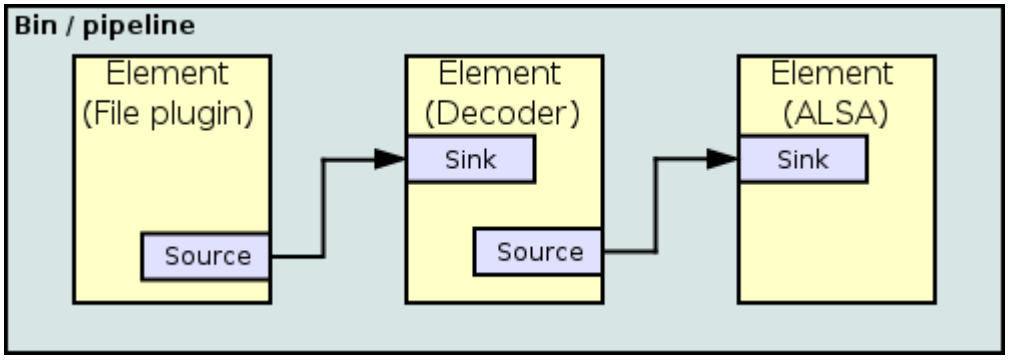


Figura 6-1: Ejemplo de *pipeline* en Gstreamer.

La creación de cualquier otra aplicación se basa en la correcta combinación de los distintos módulos que necesitamos y la realización de las conexiones pertinentes entre ellos, pero antes de poder desarrollar una aplicación o simplemente ejecutar un comando como el del ejemplo será necesario realizar una instalación de Gstreamer y su integración en VC++. Todo el proceso de instalación y configuración del proyecto en VC++ se encuentra descrito en el Anexo B y los distintos módulos de interés de Gstreamer para este proyecto se encuentran explicados en el Anexo C.

6.2 Creación del pipeline

En Gstreamer existe un elemento que se encarga de la conexión, sincronización y transmisión de información entre los distintos módulos integrados en él, este elemento es el *pipeline* de Gstreamer. En este elemento iremos añadiendo los distintos módulos que se van a utilizar y posteriormente configuraremos el orden de conexión en el que tendrán que trabajar. El propio *pipeline* se encargará de gestionar los posibles errores en la información introducida, incompatibilidades entre módulos y demás fallos que puedan surgir.

Para resolver el problema de transmisión de los flujos de vídeo y audio que en este proyecto se plantean se puede diseñar el *pipeline* de dos maneras. La primera se tratará del envío de ambos flujos recibidos en la aplicación desde la cámara, y la segunda sería similar

a la primera pero introduciendo un multiplexor, que nos permita seleccionar la fuente de flujo de vídeo que dejamos pasar al resto del *pipeline* para poder enviar en ciertos momentos un flujo de vídeo alternativo procedente de capturas del escritorio del PC del aula (en el que puede haber una presentación u otra información relevante).

En las siguientes figuras se muestran las posibles maneras de construir el *pipeline*.

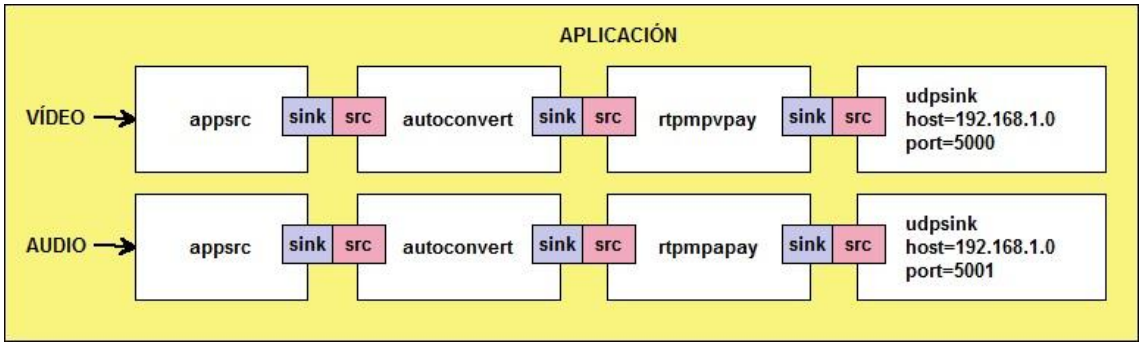


Figura 6-2: Pipelines independientes para audio y vídeo.

En esta primera construcción tenemos dos *pipelines* independientes, cada uno de ellos encargado de transmitir o el vídeo o el audio a un puerto diferente. Esta solución podría dar problemas de sincronización ya que la recepción se haría también de manera independiente en dos ejecuciones del VLCMediaPlayer. Para intentar solucionar este posible problema se ha pensado en otra solución en la que se utiliza un único *pipeline* con dos entradas y una única salida hacia internet.

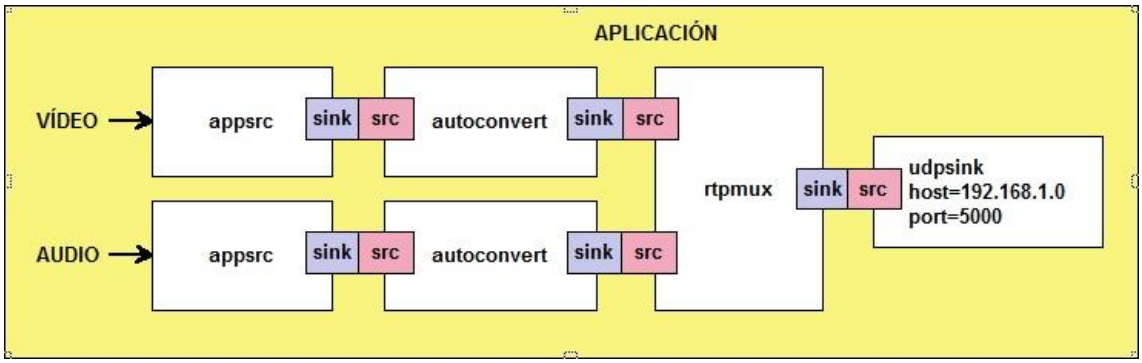


Figura 6-3: Pipeline conjunto con multiplexor para audio y vídeo.

Como solución más completa tenemos la que se muestra en la siguiente figura ya que cuenta con la posibilidad de enviar lo que está sucediendo en el escritorio del PC. Se ha añadido un módulo de entrada de captura de escritorio (dshowvideosrc) que genera el flujo de vídeo a partir del escritorio. También se ha añadido un multiplexor selector de vídeo

para poder seleccionar la entrada deseada a la parte de vídeo del *pipeline*. De esta manera con sólo cambiar la configuración del selector cambiaríamos el vídeo enviado de lo que está grabando la cámara a lo que está sucediendo en el escritorio. Para el audio sólo tendremos una entrada ya que el único audio existente en la aplicación es el recibido desde la cámara.

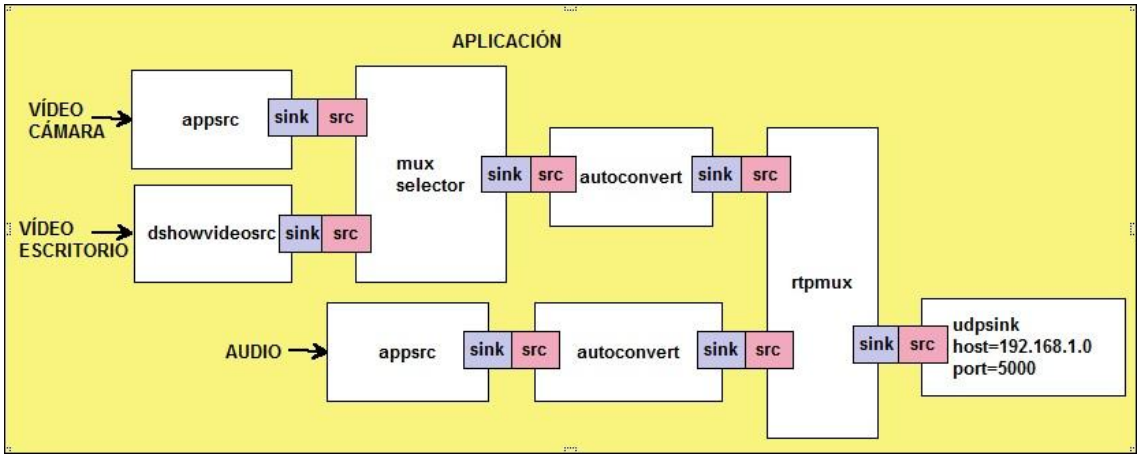


Figura 6-4: Pipeline con selección de vídeo.

Estas construcciones se han intentado probar, pero la introducción del elemento appsrc genera un problema que aún no se han solucionado. El problema que plantea este elemento es cómo realizar la alimentación del *pipeline* a medida que se va solicitando nueva información. Una vez se haya comprendido el funcionamiento de appsrc y resuelto el problema, contaremos con el punto de entrada de información al *pipeline* y podremos introducir los flujos de audio y vídeo en el *pipeline* que corresponda.

6.3 Pruebas realizadas

Las pruebas que se han realizado en este proyecto se han basado en configurar el *pipeline* de distintas maneras y comprobar el funcionamiento del mismo. En estas pruebas sólo se ha podido comprobar el envío y recepción de flujos de vídeo y audio obtenidos desde un fichero o desde los propios archivos de test que proporciona Gstreamer, ya que no ha sido posible la integración completa del elemento appsrc para recoger los flujos recibidos en la aplicación desde la cámara.

Se han realizado dos tipos de pruebas, unas desde la línea de comandos de Cygwin (simulación del sistema operativo Linux sobre Windows para poder ejecutar el lanzador de

Gstreamer correctamente) y otras probando distintas configuraciones del *pipeline* desde el proyecto en desarrollo en VC++ en Windows.

Ejecuciones desde la línea de comandos de Cygwin:

- gst-launch -v audiotestsrc ! autoconvert ! autoaudiosink: con esta configuración del *pipeline* se reproduce un fichero de pruebas de audio de Gstreamer. Se obtiene una correcta reproducción del tono puro que contiene el fichero.

- gst-launch -v videotestsrc ! autoconvert ! autovideosink: configuración del *pipeline* para la reproducción del fichero de pruebas de vídeo de Gstreamer. Con esta configuración se reproduce correctamente el vídeo de pruebas, en la siguiente figura se puede ver la ventana de reproducción creada por el elemento autovideosink.

Figura 6-5: Reproducción por Gstreamer del vídeo de pruebas.

- gst-launch playbin uri=file://archivo.mp3: configuración para reproducción de un fichero de audio introducido a través del *plugin* playbin. También funciona correctamente con un fichero de audio y vídeo simultáneamente.

- gst-launch filesrc location="archivo.avi" ! decodebin ! autoaudiosink: configuración para reproducción del audio de ficheros. Reproducción correcta del audio que contiene el fichero.

- gst-launch filesrc location="archivo.avi" ! decodebin ! autoconvert ¡ autovideosink: configuración para reproducción del vídeo de ficheros. Reproducción correcta del vídeo que contiene el fichero.

- gst-launch filesrc location="archivo.mp3" ! decodebin ! autoconvert ! rtpmpapay ! udpsink host=localhost port=5000: configuración para el envío de flujos de audio desde un fichero a una dirección de internet a través del protocolo RTP. Recepción en VLC 1.0.5 con apertura de volcado de red en udp://:5000. Se escucha muy cortado. Fallo en la reproducción ya que seguramente reproduce cada paquete recibido a una velocidad superior a la que se debería, sin embargo, si transmitimos el fichero de audio de pruebas se escucha el tono con algún corte, pero mucho más uniforme que el fichero mp3.

- gst-launch videotestsrc ! autoconvert ! rtpmpvpay ! udpsink host=localhost port=5000: configuración para el envío de flujos de vídeo desde el fichero de prueba a una dirección de internet a través del protocolo RTP Reproducción en vlc 1.0.5 con apertura de volcado de red en rtp://:5000. Reproducción correcta.

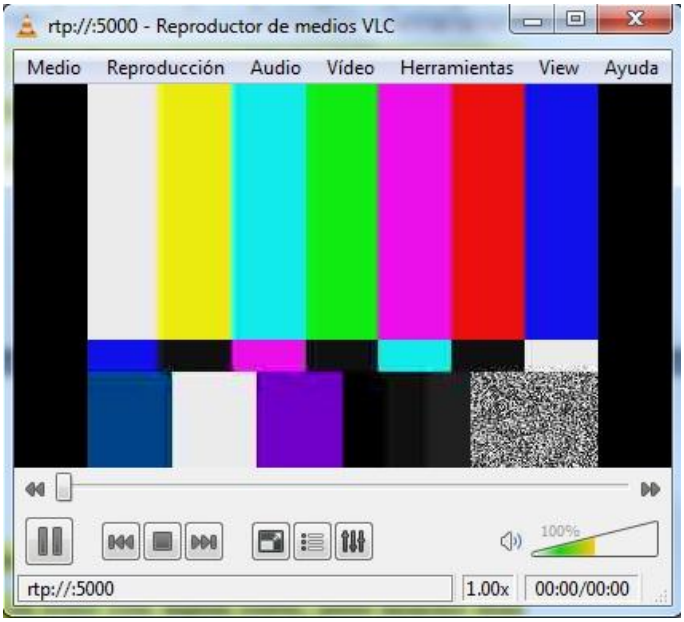


Figura 6-6: Recepción del vídeo de test en VLC 1.0.5.

Configuraciones de *pipeline* desde VC++:

- audiotestsrc->autoconvert->rtpmpapay->udpsink: configuración del *pipeline* para transmisión del flujo de audio del fichero de prueba. Reproducción correcta (con algunos cortes momentáneos) en vlc 1.0.5 con apertura de volcado de red en udp://:5001.

- videotestsrc->autoconvert->rtpmpvpay->udpsink: configuración del *pipeline* para transmisión del flujo de vídeo del fichero de prueba. Reproducción correcta en vlc 1.0.5 con apertura de volcado de red en rtp://:5000.

- Si dentro del mismo *pipeline* se configuran dos ramas independientes realizando las siguientes conexiones:

    1. audiotestsrc->autoconvert->rtpmpapay->udpsink (port 5001)

    2. videotestsrc->autoconvert->rtpmpvpay->udpsink (port 5000)

    Mediante dos ejecuciones simultáneas de vlc 1.0.5, se pueden obtener la reproducción de los dos flujos independientemente. Esta forma de recepción de los flujos no sería la óptima ya que se produciría una desincronización entre el vídeo y el audio. Al ser dos ejecuciones independientes de vlc no tendríamos ningún tipo de control sobre las reproducciones para poder sincronizar ambas.

También se probó la configuración del *pipeline* sustituyendo el elemento de entrada por el elemento appsrc, para intentar introducir flujos desde la propia aplicación, pero no fue posible obtener un buen resultado ya que daba error el nuevo elemento introducido y no se pudo solventar.

En el transcurso de la investigación del funcionamiento de Gstreamer como reproductor, transmisor y receptor de flujos de audio y vídeo se han realizado muchas otras pruebas que no se encuentran aquí definidas ya que no obtuvieron un resultado correcto, tanto para configuraciones del *pipeline* por línea de comandos en CygWin como para configuraciones realizadas desde la VC++.

6.4 Avances necesarios para el desarrollo del módulo de emisión

Al tener que enviar el flujo de vídeo grabado por la cámara y no tener Gstreamer un módulo específico de conexión que funcione con nuestra cámara, será necesaria la obtención del flujo *frame* a *frame* desde la aplicación desarrollada para poder pasar este flujo recibido al módulo de emisión. Dado que la aplicación tiene una parte de seguimiento automático que deja bloqueada la recepción de los *frames* mientras que está procesando el último, será necesario crear el módulo de emisión en una segunda aplicación que será la encargada de obtener todos los *frames* para la emisión y pasárselos al *pipeline* de Gstreamer. Esta segunda aplicación será lanzada desde la aplicación principal al activar la emisión del flujo.

Para el funcionamiento de la aplicación de emisión habrá que utilizar el módulo de entrada de datos al *pipeline* de Gstreamer, *appsrc*. Este módulo leerá de un *buffer* los *frames* que en él se irán insertando y los irá introduciendo en el *pipeline* a medida que el mismo los vaya solicitando, activando las señales *push-buffer*, *need-data* y *enough-data* cuando sea necesario.

En este momento nos encontramos con un problema que hay que resolver. Hay que enviar tanto *frames* de vídeo como de audio y para ello es necesaria la utilización de dos *pipelines*, uno para vídeo y otro para audio. La creación de los dos *pipelines* no es un problema ya que cada uno es independiente del otro y se podría conseguir enviar por un lado el vídeo y por otro el audio, pero no se garantiza la recepción en remoto sincronizada de los dos flujos. Sería mejor la utilización de un único *pipeline* con una multiplexación de ambos flujos ya que se tendría una recepción en remoto sincronizada (esta solución está plasmada en la Figura 6-3).

6.5 Conclusiones

Como conclusión a esta sección cabría decir que la utilización de Gstreamer como base del módulo de emisión sería una opción factible, pero dado que en este proyecto no se ha llegado al completo desarrollo de dicho módulo, a continuación se van a comentar las soluciones alternativas que existen para el envío del flujo de vídeo.

Una opción sería dotar al usuario remoto de una conexión directa con el servidor de la cámara, ya sea a través de la aplicación EPS-RemoteViewer o a través de una página web (aula web) en la que se colocarían los *scripts* de conexión y visualización web para cada una de las cámaras. El uso del aula web permitiría seleccionar la visualización del aula deseada, la gestión de los usuarios a los que se les da acceso a cada aula y la activación o desactivación de ciertas aulas, de manera que tenemos un control de la visualización más eficiente que si utilizamos EPS-RemoteViewer ya que habría que configurar cada distribución de la aplicación de acuerdo con la sesión que se desea visualizar.

Otra opción sería el uso de Livestream, que a través de la aplicación Procaster nos permite compartir el escritorio o una zona del mismo con los usuarios interesados. Esta opción de emisión es más adecuada que la anteriormente mencionada porque evitamos las conexiones concurrentes al servidor de la cámara y una posible sobrecarga del mismo. De esta manera, sólo una aplicación será la que se conecte con la cámara, CamApplication en el caso de querer realizar un seguimiento automático o EPS-RemoteViewer en el caso de querer únicamente visualizar lo que la cámara está capturando. Al haber una única conexión concurrente con la cámara no tendremos que preocuparnos por el ancho de banda que hay en la conexión con la misma ya que será suficiente, lo mismo pasa con el ancho de banda entre el PC del aula y el servidor de Livestream, que al haber una única conexión será suficiente (Procaster será la aplicación encargada de realizar esta conexión y del envío correcto del flujo de vídeo que se quiere compartir). En este momento será Livestream el que nos impondrá limitaciones tanto en el número de usuarios concurrentes en la visualización del vídeo como en el ancho de banda con el que será servido el vídeo a los usuarios. Ambas limitaciones se podrán solventar obteniendo un tipo de usuario *Premium*, además de permitirnos un control sobre qué usuarios pueden o no pueden visualizar el vídeo que se está compartiendo.

Finalmente se concluye que la mejor solución para el envío del flujo de vídeo es una mezcla del aula web comentado con la utilización de Livestream. Esta mezcla reside en coger la idea del aula web pero sustituyendo los *scripts* de conexión directa con la cámara por los *scripts* de conexión con los canales de emisión creados previamente en Livestream.

7 Resultados: pruebas de funcionamiento de la aplicación

En las siguientes secciones de este capítulo se expondrán los resultados obtenidos de las pruebas realizadas a las distintas aplicaciones que se han desarrollado en este proyecto y se realizará una valoración de su funcionalidad final.

7.1 Funcionalidad de las aplicaciones finales desarrolladas

Las pruebas realizadas a las distintas aplicaciones que se han desarrollado en este proyecto son las siguientes:

- **Aplicación básica:** la finalidad de esta aplicación es ofrecer al observador remoto la opción de visualización del flujo capturado por la cámara, además de darle la posibilidad de mover la cámara para centrarse en la zona que más le interese. Como alternativa tenemos la opción de que el observador pueda visualizar el flujo capturado por la cámara, pero sin tener opción de movimiento. En las pruebas realizadas a esta aplicación se obtienen unos muy buenos resultados en los casos de visualización del vídeo y reproducción del audio (tanto desde la red de la universidad como desde una red particular lejana), teniendo entre ellos una sincronización perfecta. En cuanto a la funcionalidad de movimiento de la cámara se obtienen también buenos resultados ya que la cámara tiene una respuesta casi inmediata (incluso desde una red particular lejana) y un movimiento bastante uniforme gracias a la configuración de la velocidad del movimiento.

- **Aplicación principal:** esta aplicación es una extensión de la aplicación básica, a la que se le han ido añadiendo más opciones. Las nuevas opciones son el movimiento de la cámara hacia una posición predeterminada y el seguimiento automático del profesor; también se considera como tal la opción de enviar el flujo de vídeo a una dirección IP remota. En cuanto a los resultados obtenidos por esta aplicación, en los aspectos que ya contenía la aplicación básica, se han obtenido exactamente los mismos resultados ya que están desarrollados de la misma forma. En cuanto al movimiento a posiciones predeterminadas se obtiene una respuesta instantánea del comienzo del movimiento respecto de la pulsación en el botón, y un rápido movimiento de la cámara hasta su posición final ya que se ha configurado este tipo

de movimiento para que sea realizado a la máxima velocidad de la cámara. Los resultados del seguimiento automático serán comentados en los siguientes apartados, y en cuanto a la emisión del flujo de vídeo se refiere ya se ha comentado en el capítulo 6, sobre el Subsistema de Emisión, que los resultados no han sido satisfactorios; no obstante se cuenta con la solución alternativa planteada que ha dado un buen resultado en la transmisión del flujo de vídeo.

Como conclusión a este apartado se puede decir que la aplicación básica y la aplicación principal sin seguimiento automático y sin emisión a puntos remotos funcionan perfectamente y se obtienen los resultados que cabría esperar de ellas. La utilización de cualquiera de estas dos aplicaciones se limitará a la visualización del vídeo del aula y al posicionamiento de la cámara en la zona de actividad manualmente (se podrá realizar el posicionamiento automático dependiendo de los resultados finales obtenidos por el algoritmo de seguimiento automático).

7.2 Funcionalidad del sistema de seguimiento activo

En esta sección se expondrán los resultados obtenidos por los módulos en los que está basado el seguimiento activo, probando su funcionalidad tanto de manera independiente como de manera conjunta.

Para obtener una medida objetiva de los resultados se ha utilizado un método que se basa en medir la cantidad de píxeles que se solapan entre la localización realizada por el algoritmo y una localización introducida manualmente para cada *frame*. Esta localización introducida manualmente es la que se cree que se ajusta más a la hora de contener la cara completa del profesor. Una vez se ha introducido la localización para cada uno de los *frames,* se ejecuta el algoritmo de localización y se compara el resultado con la localización introducida. Se obtiene una cantidad de píxeles que se han solapado entre ambas localizaciones y se obtiene el porcentaje de área solapada. Este porcentaje es el que nos dirá cuánto de buena es la localización obtenida por el algoritmo de una manera objetiva.

Los niveles de calificación que se van a utilizar para calificar cada una de las localizaciones obtenidas por el algoritmo son los siguientes:

- PERFECTA: para porcentajes de solapamiento mayores de 80%.

- BUENA: para porcentajes de solapamiento entre el 60% y el 80%.
- ACEPTABLE: para porcentajes de solapamiento entre el 40% y el 60%.
- MALA: para porcentajes de solapamiento inferiores al 40%.

Para la obtención de los siguientes resultados se han probado los distintos algoritmos utilizados con una secuencia de vídeo grabada desde la cámara del aula, simulando el movimiento que un profesor podría realizar durante una explicación. Para hacer más completa la secuencia se ha realizado un bucle que procesa la secuencia enlazándola tres veces. De esta manera se obtiene un cambio de posición más brusco de lo normal de la cara del profesor al pasar del último *frame* al primer *frame* de nuevo.

7.2.1 Resultados del procesado por puntos SURF

El algoritmo de seguimiento automático se puede dividir en varias partes a la hora de evaluar los resultados obtenidos. La primera parte es la localización del profesor por puntos SURF.

En la Figura 7-1 se puede observar una selección de imágenes en la que se muestra cómo ha ido funcionando la localización de la cara por puntos SURF a lo largo de la secuencia de prueba. Podemos observar cómo a medida que van sucediéndose los cuadros, la posición de la cara localizada por el algoritmo (recuadro azul en la Figura 7-1) y la posición real de la cara (recuadro blanco en la Figura 7-1) difieren cada vez más hasta llegar al *frame* 116, en el que el solape es del 0% y se considera totalmente perdida la cara. El hecho de que cada vez se vaya cometiendo más error en la localización es debido a que en el *matching* de puntos se permite una cierta distancia de descriptores, lo que implica un cierto error en el desplazamiento calculado y consecuentemente un error acumulativo al actualizar el objeto buscado *frame* a *frame*.

En esta secuencia, a partir del *frame* 116 no se vuelve a recuperar la cara del profesor en ningún momento ya que se está realizando la búsqueda de un objeto que no corresponde con la cara del profesor, además de que no se garantiza que este objeto actualizado contenga un mínimo de puntos SURF.

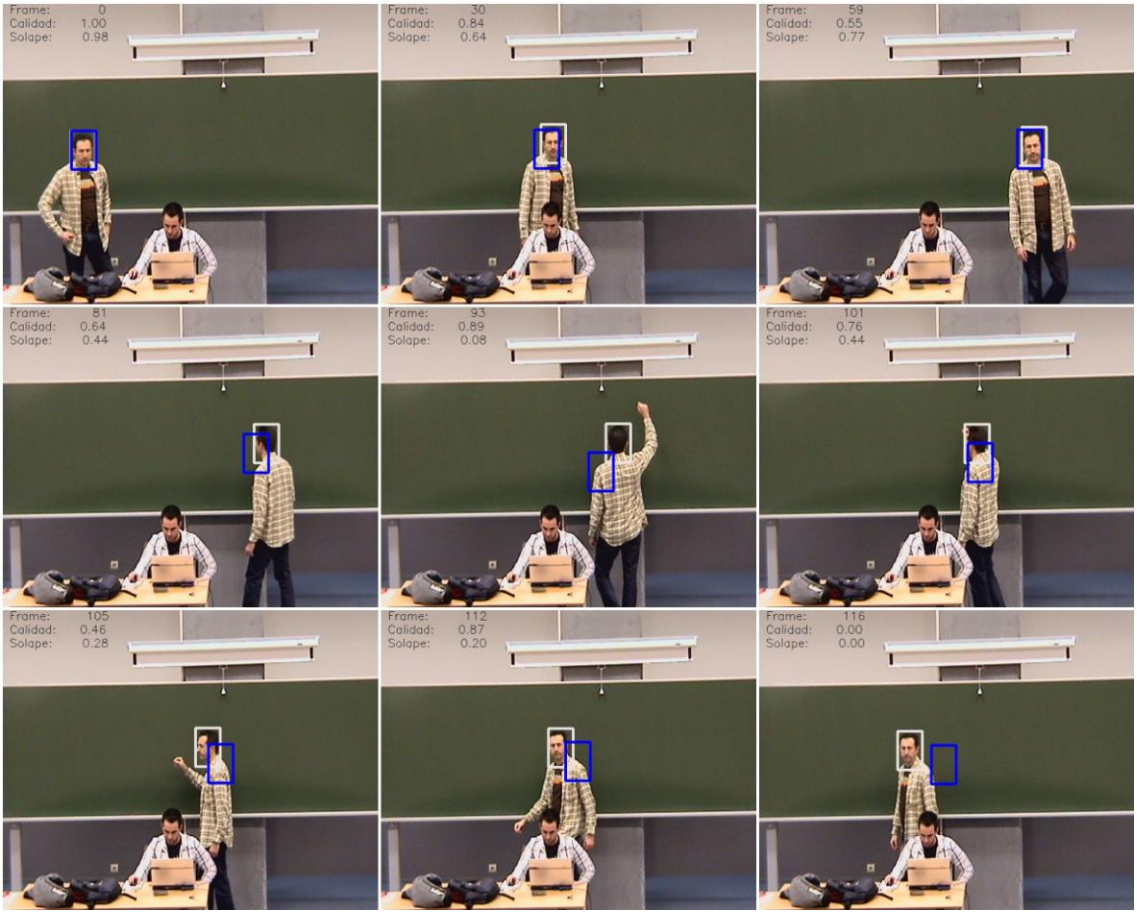


Figura 7-1: Secuencia de localización por puntos SURF.

A partir de las localizaciones obtenidas por el algoritmo de localización por puntos SURF y las localizaciones de la cara introducidas manualmente obtenemos la Figura 7-2 en la que se muestra el tanto por uno de área que se encuentra solapado entre una localización y la otra.

Se puede observar cómo a partir del cuadro 80 el solape comienza a disminuir considerablemente, momento en el que el profesor se ha dado la vuelta hacía la pizarra y los puntos característicos de la cara han desaparecido. En ese momento se obtienen unas asignaciones de puntos en el *matching* posiblemente erróneas que generan un desplazamiento erróneo. En el cuadro 99 se observa una recuperación del solape, que bien podría haberse producido por un nuevo error en el *matching* de puntos, pero en este caso ha sido un error favorable. A partir del cuadro 116 el algoritmo se ha perdido completamente, obteniéndose un solape el 0%, aunque a partir de ese momento encontramos cuatro picos en la gráfica que son debidos a que el profesor vuelve a pasar por la posición que tiene calculada el algoritmo (que en este momento ya es fija porque no encuentra ninguna asignación de puntos válida). Los dos primeros picos encontrados son

correspondientes a la pasada del profesor hacia la derecha y hacia la izquierda de nuevo de la segunda iteración de la secuencia de prueba, y el tercer y cuarto pico son los correspondientes a la tercera iteración de la secuencia de prueba.

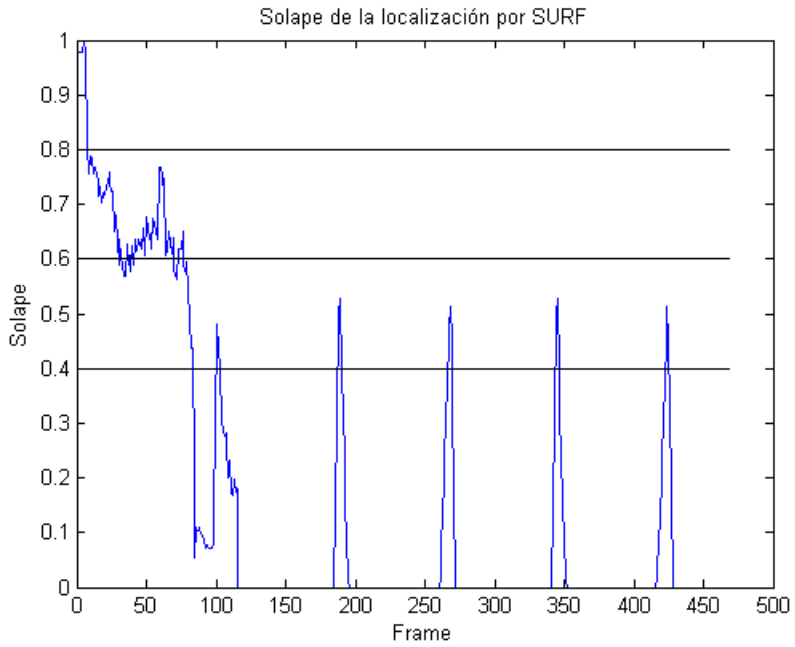


Figura 7-2: Solape obtenido entre la localización real y la localización calculada por puntos SURF.

En la figura anterior se observan las líneas de corte correspondientes a las distintas calificaciones. A partir de los datos mostrados en la figura obtenemos la Tabla 7-1 que contiene la cantidad de cuadros en los que la localización calculada obtiene una cierta calificación.

CALIFICACIÓN DE FRAMES (SURF)				
PERFECTA	BUENA	ACEPTABLE	MALA	TOTAL
8	58	30	372	468

Tabla 7-1: Calificaciones de las localizaciones por puntos SURF.

Observamos cómo tenemos ciertos cuadros con una calificación máxima que corresponden a los cuadros iniciales de la secuencia, donde aún no se ha comenzado a realizar ningún movimiento significativo. En cuanto el profesor comienza a moverse y mantiene un movimiento uniforme, obtenemos una calificación mayoritariamente buena, hasta el momento en que el algoritmo comienza a perderse, que comenzamos a tener calificaciones

malas. Notar que en la tabla se cuentan 372 calificaciones malas debido a que el algoritmo una vez se ha perdido por completo no se recupera.

En vista de estos resultados se puede decir que el algoritmo de localización por puntos SURF no funciona tan bien como cabría esperar al trabajar independientemente. Para mejorar su funcionamiento sería necesario un sistema de recuperación del algoritmo.

7.2.2 Resultados del procesado por histograma de color

La segunda parte a evaluar del algoritmo de seguimiento automático es la localización del profesor por Histograma de Color.

En la Figura 7-3 se puede observar una selección de imágenes en la que se muestra cómo ha ido funcionando la localización de la cara por histograma de color a lo largo de la secuencia de prueba. En este caso podemos observar cómo a medida que van sucediéndose los cuadros, la posición de la cara localizada por el algoritmo (recuadro verde en la Figura 7-3) y la posición real de la cara (recuadro blanco en la Figura 7-3) tienen una diferencia estable, solamente en ciertos cuadros de la secuencia se observa una diferencia mayor entre ambas posiciones, estos cuadros son el 97 y el 156. El cuadro 97 se corresponde con el momento en el que el profesor se da la vuelta hacia la pizarra, lo cual influye en la detección de la cara ya que ha desaparecido. El cuadro 156 se corresponde con la transición entre la primera iteración de la secuencia y la segunda iteración, lo cual significa un cambio más brusco en la posición de la cara entre ambos *frames* ya que no es un movimiento continuo del profesor sino un salto en la secuencia. Este aspecto de movimientos bruscos puede influir en el correcto funcionamiento del algoritmo ya que según se observa, la localización obtenida en el cuadro 156 no es exactamente la cara.

Hay que destacar que en la secuencia se observa que aunque en el cuadro 156 se pierde parcialmente la cara del profesor, en el cuadro 158 se ha vuelto a recuperar. Esto es un aspecto muy positivo del algoritmo de localización por histograma de color ya que al no actualizarse el objeto de búsqueda en ningún momento no se produce el error acumulativo producido en la localización por puntos SURF.

Gracias a que el algoritmo es capaz de recuperarse de una mala localización puntual de la posición que ocupa la cara, se puede decir que el algoritmo no se pierde por completo en ningún momento en esta secuencia, y dado que se simula un movimiento bastante brusco

del profesor y el giro hacia la pizarra, se podría decir que es difícil que se pierda en cualquier otra secuencia.

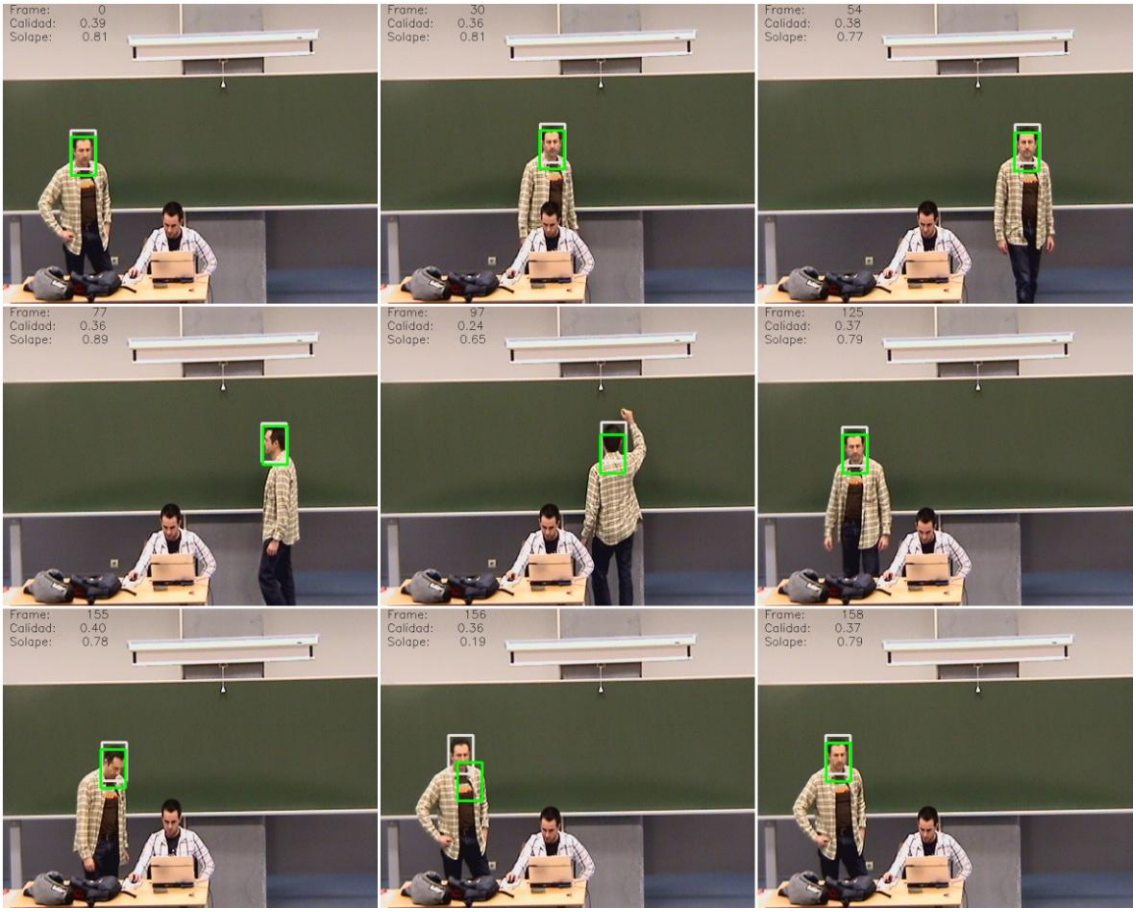


Figura 7-3: Secuencia de localización por Histograma de Color.

A partir de las localizaciones obtenidas por el algoritmo de localización por histograma de color y las localizaciones de la cara introducidas manualmente obtenemos la Figura 7-4 en la que se muestra el tanto por uno que se encuentra solapado entre una localización y la otra.

Se puede observar cómo casi el completo de la secuencia obtiene una calificación entre perfecta y buena, exceptuando los momentos puntuales correspondientes al *frame* 156 en las dos uniones de las secuencias (*frames* 156 y 312), que se puede observar cómo se obtiene una mala calificación. También cabe notar los tres picos de bajada del solape correspondientes a las tres pasadas de la secuencia por el giro hacia la pizarra del profesor. Estas tres bajadas se centran en los *frames* 97, 253 y 409 de la secuencia completa.

Como ya se ha dicho anteriormente el algoritmo nunca llega a la pérdida completa de la cara, y eso se ve reflejado en que nunca se obtiene un solape del 0%. En los peores

momentos se obtienen unas bajadas puntuales del solape seguidas de una rápida recuperación del algoritmo.

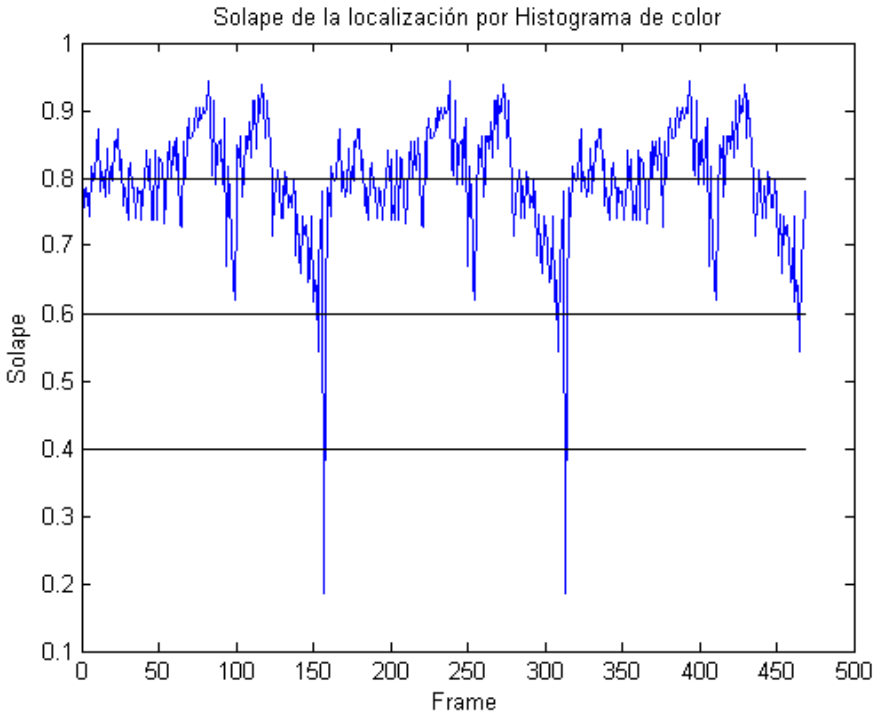


Figura 7-4: Solape obtenido entre la localización real y la localización calculada por Histograma de Color.

En la figura anterior se observan las líneas de corte correspondientes a las distintas calificaciones. A partir de los datos mostrados en la figura obtenemos la Tabla 7-2 que contiene la cantidad de cuadros en los que la localización calculada obtiene una cierta calificación.

CALIFICACIÓN DE FRAMES (HISTOGRAMA)				
PERFECTA	BUENA	ACEPTABLE	MALA	TOTAL
232	229	5	2	468

Tabla 7-2: Calificaciones de las localizaciones por Histograma de Color.

Como ya se ha comentado, se obtienen unas calificaciones de entre perfecta y buena la mayor parte del tiempo, esto también se ve reflejado en la tabla anterior, obteniéndose una mala calidad en dos *frames* únicamente de 468.

En función a estos resultados se puede decir que el algoritmo de localización por histograma de color da un buen resultado funcionando independientemente a lo largo de

toda la secuencia de prueba. Este algoritmo podría ser un buen algoritmo para solventar las pérdidas producidas en el algoritmo de localización por puntos SURF.

7.2.3 Resultados del procesado conjunto

Finalmente se va evaluar el algoritmo de seguimiento automático conjunto que consta de tanto del algoritmo de localización por puntos SURF como del algoritmo de localización por histograma de color, trabajando conjuntamente para solventar los fallos que puedan surgir individualmente.

En la Figura 7-5 se puede observar la selección de imágenes en la que se muestra cómo ha funcionado la localización de la cara por el procesado conjunto a lo largo de la secuencia de prueba. En este último caso podemos observar cómo a medida que van sucediéndose los cuadros, la posición de la cara localizada por el algoritmo (recuadro verde o azul en la Figura 7-5) y la posición real de la cara (recuadro blanco en la Figura 7-5) tienen una diferencia aceptable visualmente, al igual que en la localización por histograma de color, solamente en ciertos cuadros de la secuencia se observa una diferencia mayor entre ambas posiciones, estos cuadros son el 98 y el 156.

En este caso hay que destacar que en la secuencia hay momentos en los que el recuadro que representa la localización calculada por el algoritmo mayoritariamente aparece en azul, aunque hay momentos puntuales en los que aparece en verde. Los momentos en los que esto sucede se corresponden con los momentos en los que la localización por puntos SURF no obtiene una buena calidad y se opta por la localización obtenida por histograma de color. Se puede observar claramente que gracias a este método de recuperación del algoritmo, podemos continuar realizando las localizaciones por puntos SURF una vez se ha realizado la recuperación, evitando así que el algoritmo elegido como principal se quede completamente perdido, como se ha observado que pasaba en el funcionamiento individual.

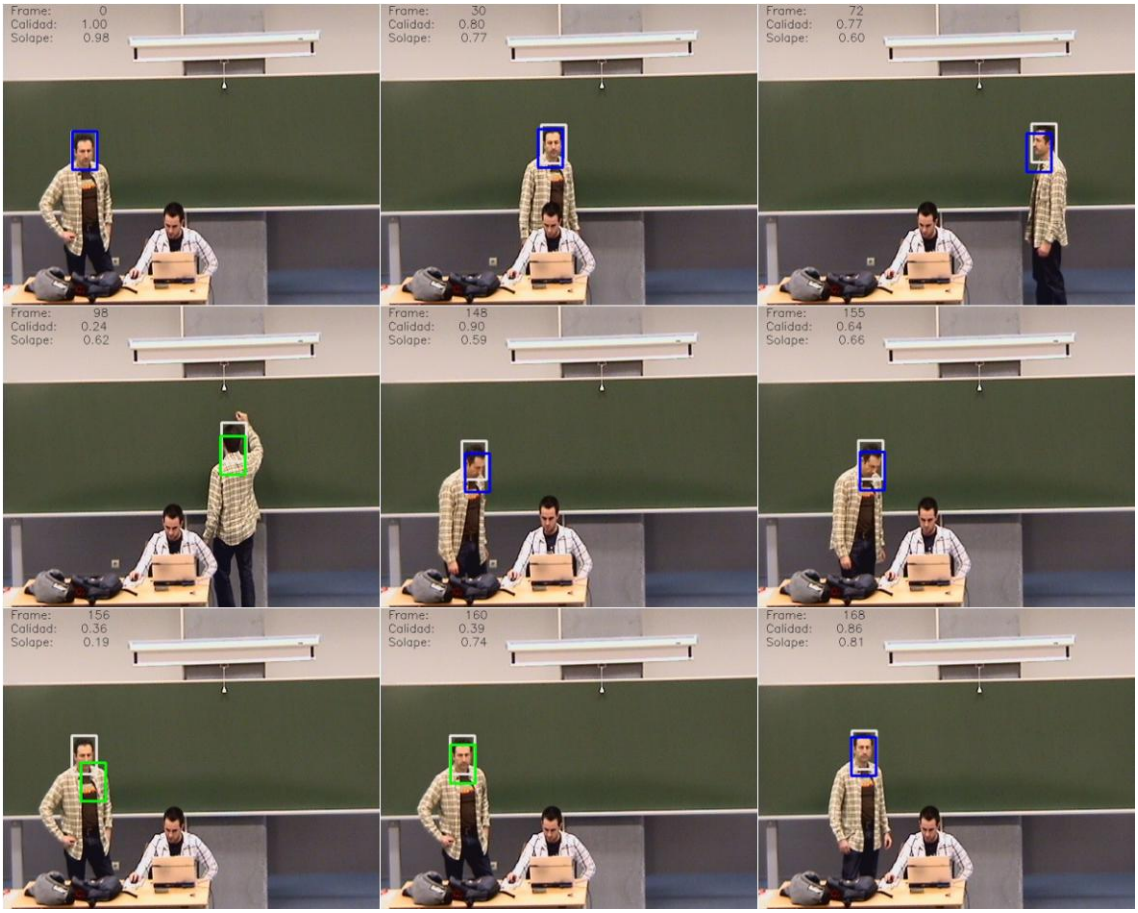


Figura 7-5: Secuencia de localización Conjunta.

A partir de las localizaciones obtenidas por el algoritmo y las localizaciones de la cara introducidas manualmente obtenemos la Figura 7-6 en la que se muestra el tanto por uno de área que se encuentra solapado entre una localización y la otra.

Se puede observar cómo la mayor parte de la secuencia obtiene una calificación buena, pero oscilando continuamente y llegando a obtener calificaciones de perfecta y aceptable en un menor número, e incluso llegando a calificaciones de mala en una minoría de las veces. En este caso, en ningún momento de la secuencia se llega a tener un 0% de solape, por lo tanto nunca se llega a perder el algoritmo.

Se debe notar que después de una bajada del solape se obtiene una rápida recuperación del algoritmo, al igual que pasaba con la localización por histograma de color, y que dichas bajadas coinciden con los puntos en los que se conectan las secuencias y los puntos en los que el profesor se gira hacia la pizarra.

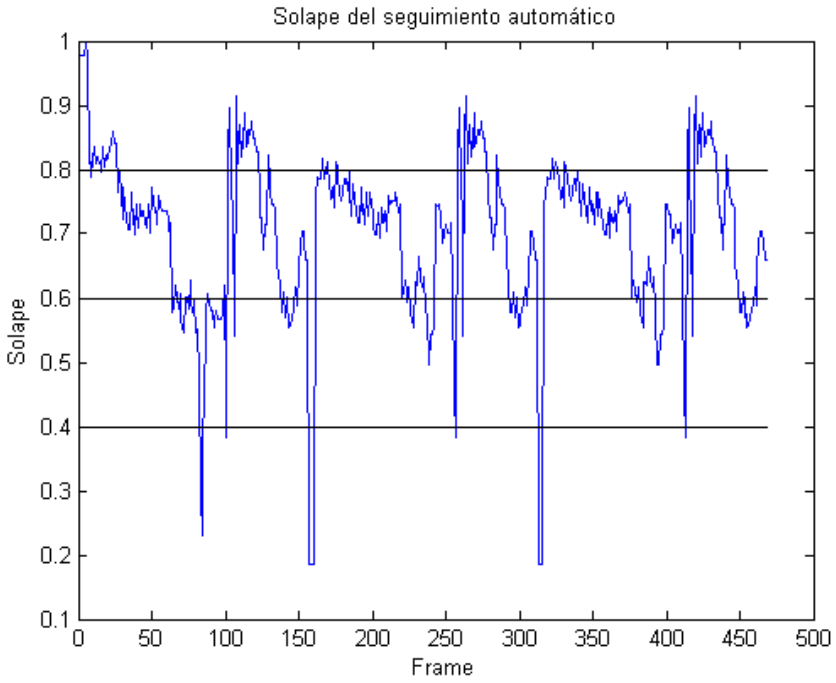


Figura 7-6: Solape obtenido entre la localización real y la localización calculada por el algoritmo conjunto.

En la figura anterior se observan las líneas de corte correspondientes a las distintas calificaciones. A partir de los datos mostrados en la figura obtenemos la Tabla 7-3 que contiene la cantidad de cuadros en los que la localización calculada obtiene una cierta calificación.

CALIFICACIÓN DE FRAMES (CONJUNTO)				
PERFECTA	BUENA	ACEPTABLE	MALA	TOTAL
89	285	81	13	468

Tabla 7-3: Calificaciones de las localizaciones por algoritmo conjunto.

En esta tabla se puede observar que la mayoría de los *frames* obtienen una calificación buena, obteniendo bastantes calificaciones perfecta y aceptable, obteniendo únicamente en 13 *frames* de 468 una mala calificación.

Hay que comentar que se obtienen unos resultados peores que si se utilizase únicamente el algoritmo de localización por histograma de color (más abajo veremos cómo mejorar estos resultados dando más peso a la localización por histograma de color), pero de esta manera se obtiene un sistema en teoría más robusto al contar con dos formas independientes de localización, y basadas en características totalmente distintas de la imagen como son puntos característicos y la distribución de color de la cara. Es evidente que la secuencia

utilizada para realizar las pruebas no permite valorar la hipotética robustez de esta aproximación.

En la Figura 7-7 se muestra una gráfica que compara el solape obtenido en las ejecuciones independientes y la ejecución conjunta de los algoritmos de localización.

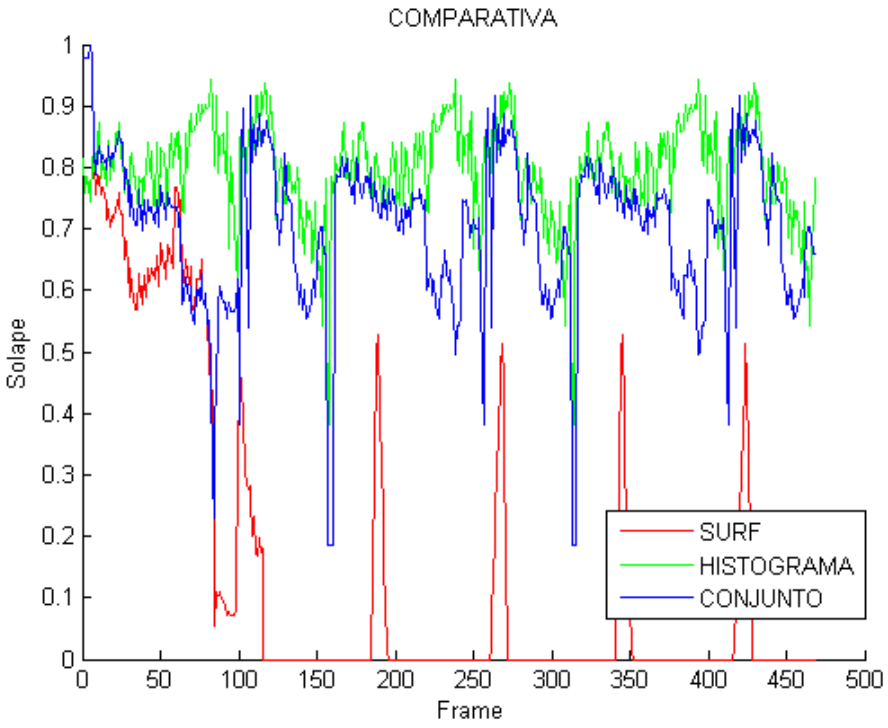


Figura 7-7: Comparativa de algoritmos de localización.

En esta gráfica se puede observar todo lo mencionado con anterioridad: que el algoritmo de localización conjunta mejora bastante la localización por puntos SURF, pero no llega a ser tan buena como la localización por histograma de color; sólo en algunos momentos se iguala. Decir que para la consecución de la mejora respecto de la localización por puntos SURF solamente ha sido necesaria la intervención del algoritmo de localización por histograma de color en 16 cuadros de 468.

Ahora pasamos a dar un peso mayor en el algoritmo conjunto a las localizaciones calculadas por el algoritmo de localización por histograma de color. Este peso mayor se consigue introduciendo en la decisión un margen, de manera que cuando la calidad de la localización por puntos SURF sea mayor que la calidad de la localización por histograma de color mas el margen (en este caso se ha usado un margen de 0.4) se seleccionará la

localización por puntos SURF, el resto de las veces se seleccionará la localización por histograma de color.

En la Figura 7-8 se muestra el comportamiento del algoritmo de localización conjunto dando preferencia a la localización por histograma de color en comparación con el comportamiento anterior, cuando se daba preferencia a la localización por puntos SURF. Se puede observar cómo casi en la totalidad de los *frames* el algoritmo con preferencia por la localización por histograma de color se encuentra por encima.

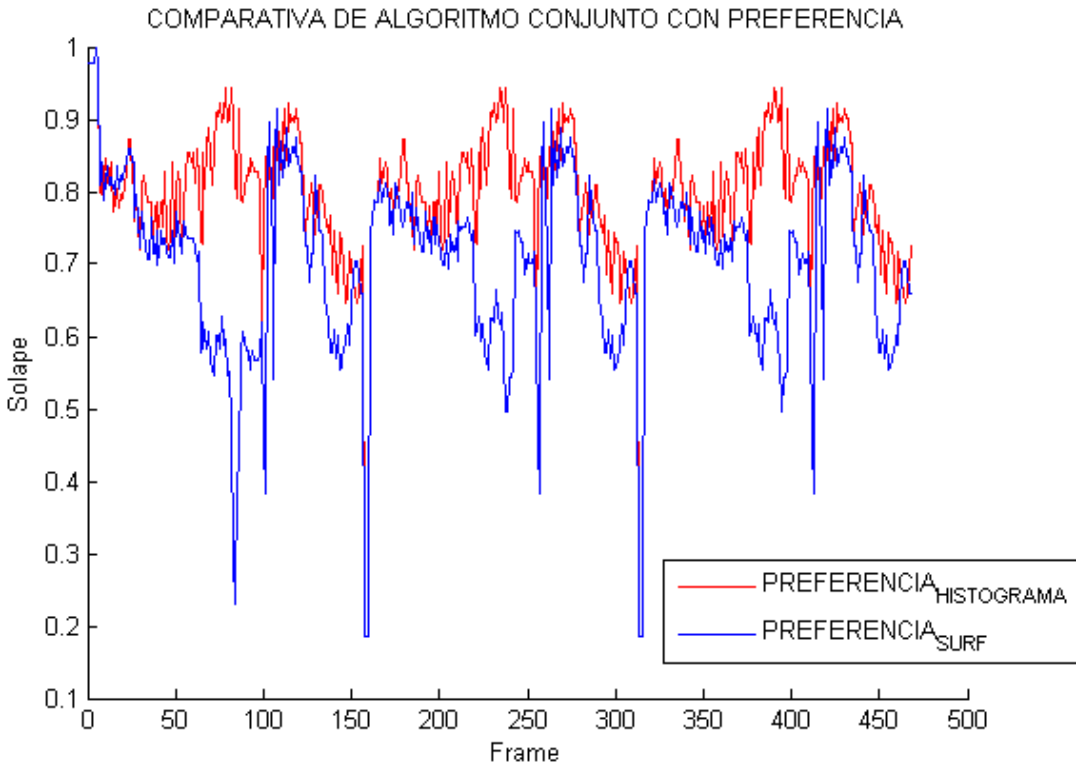


Figura 7-8: Comparativa de algoritmo conjunto con preferencia en algoritmos de localización.

En la Tabla 7-4 podemos ver cómo las calificaciones obtenidas han mejorado bastante en comparación con la Tabla 7-3, habiéndose reducido de 13 a 8 las malas calificaciones y aumentando considerablemente las calificaciones perfectas, pasando de 89 a 253 gracias a la desaparición de las calificaciones aceptables una merma en las calificaciones buenas.

CALIFICACIÓN DE FRAMES (PREFERENCIA HISTOGRAMA)				
PERFECTA	BUENA	ACEPTABLE	MALA	TOTAL
253	207	0	8	468

Tabla 7-4: Calificaciones de las localizaciones por algoritmo conjunto con preferencia por histograma.

En este caso el algoritmo conjunto se ha decidido por la localización por histograma de color 245 veces de 468 *frames* en vez de las 16 veces que se había decidido sin dar preferencia a la localización por histograma de color.

Como conclusión se podría decir que se ha conseguido un sistema de seguimiento activo basado en los puntos característicos del objeto de búsqueda (localización por puntos SURF) y en la distribución de color del mismo (localización por histograma de color), del cual se obtienen unos resultados bastante buenos gracias a la complementación que entre ellos existe, consiguiendo un funcionamiento estable a lo largo de toda la secuencia de prueba. Dependiendo del tipo de secuencia con la que se trabaje (cambios o no de iluminación y escala del objeto) y del tipo de objeto de objeto a seguir (objeto con puntos característicos o sin ellos), convendrá dar más peso dentro del algoritmo conjunto a la localización por puntos SURF o a la localización por histograma de color.

8 Conclusiones y trabajo futuro

Este capítulo se va a dedicar a dar una visión de qué se ha conseguido en este proyecto y qué se puede hacer para intentar mejorar los distintos aspectos que no han dado unos resultados satisfactorios.

8.1 Conclusiones

En este proyecto se ha desarrollado una aplicación básica que ofrece la posibilidad de visualizar, de manera remota, lo que la cámara del aula está capturando en ese mismo instante. También permite el movimiento de la cámara hacía la zona que más le interese visualizar al profesor.

La aplicación avanzada que se ha desarrollado en este proyecto tiene como punto de partida la aplicación básica, a la que se han ido añadiendo distintos módulos. El primer módulo que se ha añadido permite configurar unas posiciones predefinidas en las que poder colocar la cámara con un solo *click.* Por otro lado está el módulo de seguimiento activo, que daría la funcionalidad de realizar un seguimiento del profesor automáticamente a la aplicación, pero los resultados obtenidos en su estudio desvelan que por las características de la cámara no ha sido posible realizar un seguimiento en tiempo real del profesor. Por último está el módulo de *streaming* de vídeo, que pretende dotar a la aplicación de la posibilidad de enviar el flujo de vídeo capturado por la cámara o el capturado del escritorio del PC a un punto remoto. Este último módulo pretendía usar las librerías Gstreamer para realizar la composición y el envío del vídeo, pero la complejidad de esta herramienta ha hecho inviable desarrollar por completo el módulo en el marco de este Proyecto.

Como conclusión a este proyecto final de carrera se puede decir que aunque la aplicación principal desarrollada no satisface por sí sola todas las funcionalidades inicialmente planteadas, debido a que el desarrollo de ciertos módulos se ha complicado en cierta manera, se han planteado varias formas de resolver satisfactoriamente cada una de esas funcionalidades que faltan por medios alternativos o complementarios a la propia aplicación. El sistema completo no cuenta con una sola aplicación, sino que se compone de la ejecución de varias aplicaciones (desarrolladas en este proyecto o aplicaciones gratuitas encontradas) que se complementarán y en conjunto darán una funcionalidad completa. En

el siguiente apartado se explicará qué caminos seguir para poder dotar a la aplicación principal de toda la funcionalidad por sí sola.

8.2 Trabajo futuro

Para dotar de una funcionalidad completa a la aplicación principal que se ha desarrollado en este proyecto y que llegue a satisfacer todos los requisitos que inicialmente se han planteado es necesario el trabajo en ciertos módulos de la misma.

- **Mejoras del seguimiento activo:** para mejorar el seguimiento activo del profesor la primera posibilidad que se plantea es la de realizar una segmentación del fondo teniendo en cuenta que hay que dividir el escenario de varias zonas y posicionando la cámara en cada una de ellas siempre de la misma manera. Con esto conseguimos limitar el número de fondos que tenemos y el número de posiciones en que se puede encontrar la cámara, y así, en cada una de ellas obtener la posición del profesor comparando el *frame* actual con el fondo guardado para esa posición. Los fondos de cada una de las posiciones se pueden obtener fácilmente con el aula vacía, aunque a la hora de concurrir varias personas en la escena podría haber problemas y también pueden ocurrir a la hora de cambiar de una zona a otra. Habría que utilizar alguna forma de histéresis para evitar el cambio continuo de la cámara de una zona a otra y la posible pérdida del seguimiento en los puntos de ruptura que suponen los cambios de zona. Además de mejorar el resultado del seguimiento activo se le podría añadir una nueva funcionalidad que consistiría en la realización de diferentes planos, para ello hay que realizar un estudio de cómo funciona el seguimiento activo si se van haciendo cambios de *zoom* en función del movimiento que realice el profesor. Esta nueva función supondría un incremento de los posibles problemas que pueden surgir, por ejemplo, que un leve movimiento del profesor provoque que se salga del plano, o que el mínimo movimiento de la cámara no sea tan pequeño como el movimiento que realice la cara del profesor, ya que el movimiento de la cámara es independiente del *zoom* y si el mínimo movimiento sin *zoom* es de un píxel, con un *zoom* mayor el mínimo movimiento será mayor también. Por otro lado, ya que el punto de ruptura más importante se encuentra en el momento de mover la cámara, que al no ser lo suficientemente rápida da problemas, cabría la posibilidad de realizar pruebas con una cámara más rápida pero eso supondría cambiar ciertas partes de la aplicación para acomodarla a las

librerías que utilice la nueva cámara. Por último, se podrían explorar otras técnicas de seguimiento (diferentes a puntos SURF e histograma de color), profundizando en la estimación de su calidad y en su integración en un único algoritmo.

- **Mejoras del módulo de emisión:** la principal mejora de la emisión del flujo de vídeo reside en el completo desarrollo del módulo de emisión basado en Gstreamer y su integración en la aplicación principal. Por otro lado la implementación de una página web para gestionar los contenidos que se permiten visualizar a cada usuario. Otro aspecto que cabría mencionar acerca de este módulo de emisión es la posibilidad de que una empresa gestora de contenidos obtuviese el flujo completo y se encargase de distribuirlo de la manera más adecuada, para ello sería necesario que la aplicación fuese capaz de suministrar este flujo de la manera requerida por la empresa.

Referencias

[1] A.Yilmaz, O. Javed, and M. Shah, “Object tracking: a survey”, ACM Computing Surveys, Volume 38 Issue 4, 2006, 54 pages.

[2] Harris, C. and Stephens, M., “A combined corner and edge detector”, in 4th Alvey Vision Conference, 1988, Pages147–151.

[3] Moravec, H., “Visual mapping by a robot rover”, in Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1979, Pages 598–600.

[4] Lowe, D., “Distinctive image features from scale-invariant keypoints”, Int. J. Comput. Vision 60, 2, 2004, Pages 91–110.

[5] Rémi Trichet, Bernard Mérialdo, “Generic object tracking for fast video annotation”, VISAPP 2007, 2nd International Conference on Computer Vision Theory and Applications, 8-11 March, 2007, Barcelona, Spain, Page 2.

[6] Shi, J. and Tomasi, C., “Good features to track”, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1994, Pages 593–600.

Glosario

Backprojection	Retroproyección. Asigna a cada pixel del cuadro de búsqueda la densidad de probabilidad que tiene el color del pixel en el histograma de color del objeto de búsqueda.
Callback	Código ejecutable que lleva a cabo iteraciones sobre objetos con propiedades específicas para llevar a cabo operaciones que desde el punto de vista del sistema son equivalentes, pero sobre elementos distintos.
CAMSHIFT	Algoritmo de segmentación de imágenes por color. (Continuously Adaptive Mean Shift)
CGI	Tecnología que permite transferir datos entre el cliente y el programa ejecutado. (Del inglés Common Gateway Interface)
C++	Lenguaje de programación orientado a objetos.
Dirección IP	Etiqueta numérica que identifica un dispositivo en una red que utilice el protocolo IP.
DLL	Biblioteca de enlace dinámico. Término con el que se refiere a los archivos con código ejecutable que se cargan bajo demanda de un programa por parte del sistema operativo. (Del inglés Dinamic-Link-Library)
FPS	Tasa de cuadros de la secuencia. (Del inglés Frames Per Second)
Frame	Cada cuadro de la secuencia de vídeo.
Gstreamer	Librerías usadas para la creación de aplicaciones audiovisuales.
Host	Nombre en internet del PC. Alternativa a la dirección IP.
Keypoints	Puntos característicos.

Matching	Asignación a los puntos característicos del objeto de sus puntos afines en *frame* actual.
OpenCV	Librerías usadas para el procesado de imágenes.
Pipeline	Elemento usado en Gstreamer para transmisión del flujo y control de la transmisión de un elemento a otro.
PTZ	Cámara que permite movimientos en los ejes horizontal y vertical además de ajustes en el *zoom*. (Del inglés Pan Tilt Zoom)
ROI	Región de interés en la que se realiza la búsqueda de puntos SURF para el *matching.*(Del inglés Region Of Interest)
RTP	Protocolo de transmisión en tiempo real. (Del inglés Real Time Protocol)
Script	Guión, archivo de órdenes para interactuar con el sistema operativo.
Streaming	Transmisión de contenido en tiempo real.
SURF	Algoritmo para extracción de puntos característicos en imágenes. (Del inglés Speed Up Roboust Features)
VC++	Visual C++, entorno de desarrollo para C++.
VPU-UAM	Grupo de Tratamiento e Interpretación de Vídeo de la UAM.

Anexos

Anexo A. Integración de OpenCV para el análisis

Para realizar el procesado de imagen, que es vital para poder conseguir un seguimiento correcto del profesor, es necesaria la integración de ciertas librerías que ayuden y faciliten la tarea. Se han elegido las librerías OpenCV ya que consta de funciones pre-programadas que facilitan la realización del análisis al completo.

Lo primero que hay que hacer es instalar el paquete OpenCV2.1 en el ordenador; para ello se descarga esta versión de internet y se ejecuta el instalador obtenido (http://sourceforge.net/projects/opencvlibrary/files/opencv-win/2.1/). Una vez instalado habrá que realizar una conexión entre el proyecto que se está creando y las librerías instaladas que se van a utilizar. Para ello tendremos que configurar el VC++ abriendo las siguientes pestañas: Herramientas->Opciones->Proyectos y soluciones->Directorios de VC++, y seleccionando Archivos de inclusión, se añadirá "C:\OpenCV2.1\include\opencv" que dependiendo del directorio en que hemos instalado OpenCV podrá variar la ruta. Lo siguiente que hay que añadir es el *path* de los archivos de biblioteca añadiendo "C:\OpenCV2.1\lib", y finalmente los archivos de código fuente, que serán:

- "C:\OpenCV2.1\src\cv"
- "C:\OpenCV2.1\src\cvaux"
- "C:\OpenCV2.1\src\cxcore"
- "C:\OpenCV2.1\src\highgui"
- "C:\OpenCV2.1\src\ml"

En la Figura A-1 se puede observar la ventana de configuración de opciones de VC++ y en ella las distintas opciones que se han ido seleccionando hasta llegar a incluir las direcciones de las distintas librerías de OpenCV. En el desplegable situado arriba a la derecha se encuentran los distintos tipos de archivos que se pueden incluir, y seleccionando uno de ellos se pueden añadir todas las direcciones de los archivos necesarios de ese tipo.

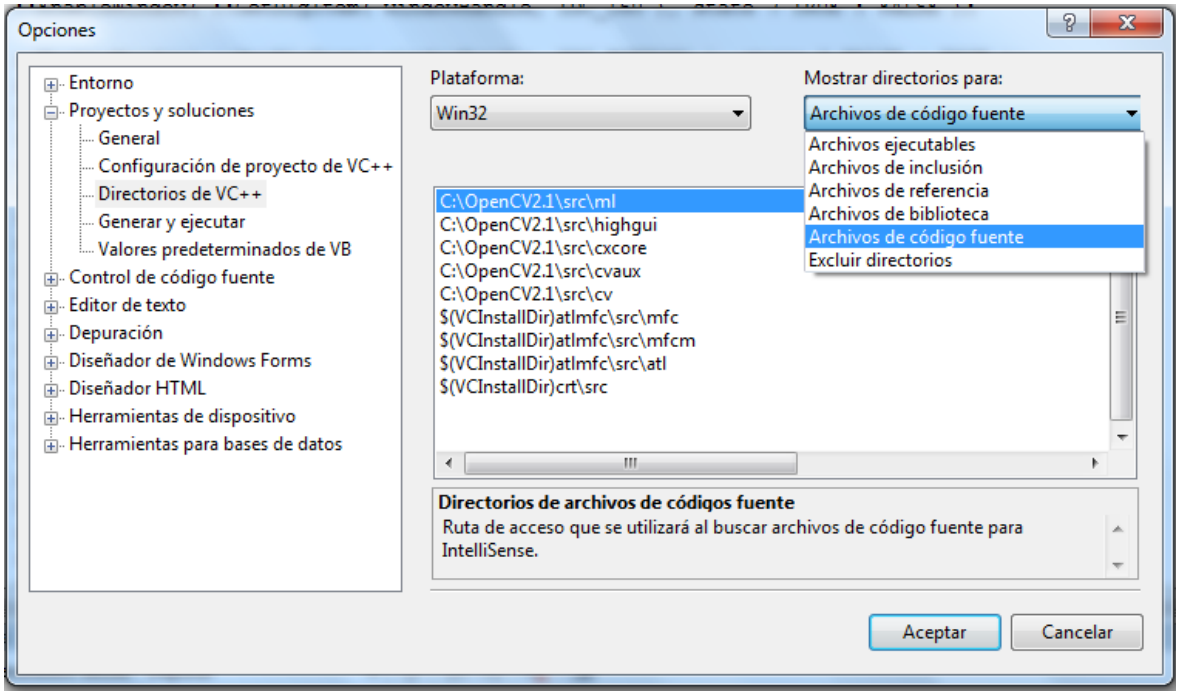


Figura A-1: Configuración VC++ con OpenCV2.1.

Una vez hechas las referencias del programa habrá que configurar las referencias de las librerías que utilizaremos en el proyecto que se está desarrollando. Para lo cual es necesario abrir las pestañas de Proyecto->Propiedades, y en Propiedades de configuración seleccionar Vinculador para poder añadir en Entradas las librerías que vamos a utilizar:

- "cv210.lib"
- "cxcore210.lib"
- "highgui210.lib"

En la Figura A-2 se observa la ventana de configuración del proyecto que se está desarrollando. En el cuadro de la izquierda se puede ver la ruta seguida hasta llegar a las propiedades del vinculador del proyecto para poder añadirle las distintas librerías que se utilizarán. En el cuadro de la derecha, en Dependencias adicionales se añadirán las tres librerías antes mencionadas, que son las necesarias para la utilización de las distintas funciones de OpenCV que en este proyecto se han utilizado. En el caso de que fuese necesario utilizar funciones distintas para otro tipo de finalidad, quizás fuese necesario añadir alguna librería más de OpenCV en estas Dependencias adicionales para que el

vinculador pueda cargarlas correctamente a la hora de crear el ejecutable de la aplicación desarrollada.

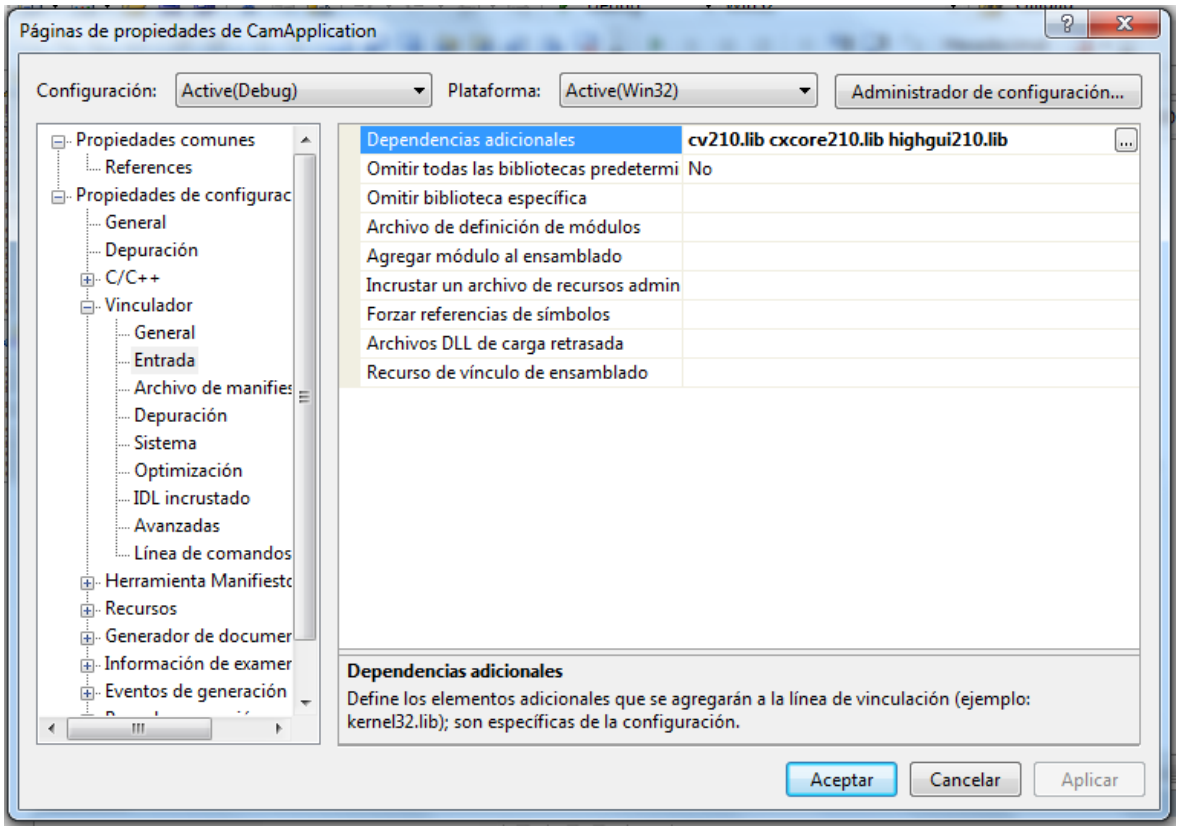


Figura A-2: Configuración de proyecto con OpenCV.

Una vez realizados estos pasos se encuentra correctamente configurado el VC++ para poder utilizar las librerías de OpenCV en el proyecto. Solamente habrá que ir incluyendo las cabeceras correspondientes en el código fuente que vayamos creando para poder utilizar las funciones requeridas.

Todas las funciones utilizadas de las librerías OpenCV y sus correspondientes cabeceras han sido explicadas en la sección 2.2**.**

La información utilizada para la correcta instalación y configuración de OpenCV en este proyecto ha sido obtenida de la página web:

- http://opencv.willowgarage.com/wiki/VisualC%2B%2B_VS2008

Anexo B. Integración de Gstreamer para la emisión

Lo primero que se debe hacer para poder comenzar a utilizar Gstreamer es integrar las correspondientes librerías en el proyecto de Visual C++ que estamos desarrollando. Para ello es necesario descargar el instalador para Windows Gstreamer-WinBuilds-GPL-x86.mse que encontramos en la siguiente página web:

- http://code.google.com/p/ossbuild/downloads/detail?name=GStreamer-WinBuilds-GPL-x86.msi

Una vez descargado se instalará en el PC de desarrollo en la ruta que más interese, teniendo así una carpeta en la que se encuentran todas las librerías necesarias para el funcionamiento de Gstreamer en Windows y las librerías necesarias para el desarrollo de aplicaciones.

Ahora nos queda indicarle a Visual C++ donde encontrar estas librerías para poder utilizarlas. En Herramientas->Opciones->Proyectos y soluciones->Directorios de VC++->Archivos de referencias, y Archivos de biblioteca añadiremos la siguiente línea:

- C:\Program Files\OSSBuild\GStreamer\v0.10.6\sdk\lib

De manera que quede tal y como se muestra en la Figura A-3. En la Figura A-3 se muestra sólo la inclusión de la dirección en archivos de referencia, por lo que habría que obtener un resultado similar al incluirla en archivos de biblioteca.

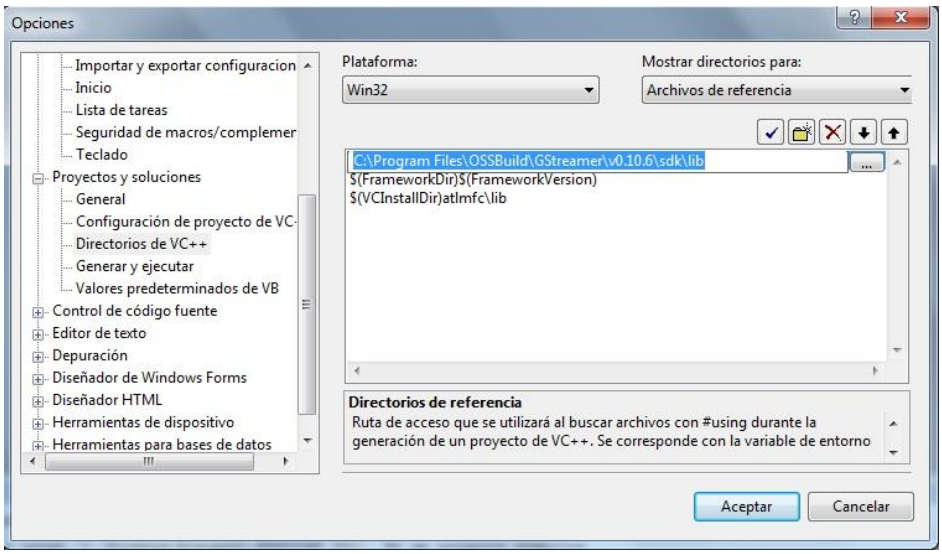


Figura A-3: Configuración de VC++ con Gstreamer.

Para la utilización de los distintos *plugins* de Gstreamer, será necesario el ir añadiendo los *paths* necesarios en archivos de código fuente. En este caso se ha añadido el *plugin* para servidor RTSP, añadiendo la siguiente línea:

- C:\Program Files\OSSBuild\GStreamer\v0.10.6\sdk\include\gstreamer-0.10\gst\rtsp-server

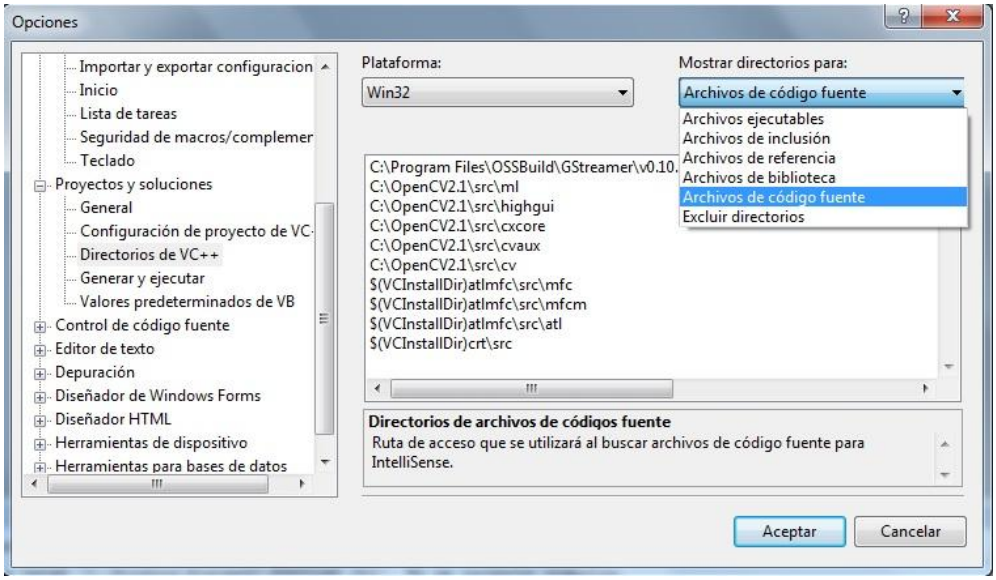


Figura A-4: Inclusión de *plugins*.

Finalmente habrá que añadir dentro del proyecto que se está desarrollando las distintas librerías que se van a utilizar. En este caso se han añadido las siguientes, aunque para otros proyectos quizás no sean necesarias algunas de ellas:

- gstreamer-0.10.lib
- glib-2.0.lib
- gstsignalprocessor-0.10.lib
- gobject-2.0.lib
- gstbasevideo-0.10.lib
- gstinterfaces-0.10.lib
- gstbase-0.10.lib

Para añadirlas es necesario acceder a Propiedades del proyecto->Propiedades de configuración->Vinculador->Entrada->Dependencias adicionales, y ahí iremos añadiendo cada una de las librerías que vamos a utilizar en nuestro proyecto tal y como se muestra en la Figura A-5.

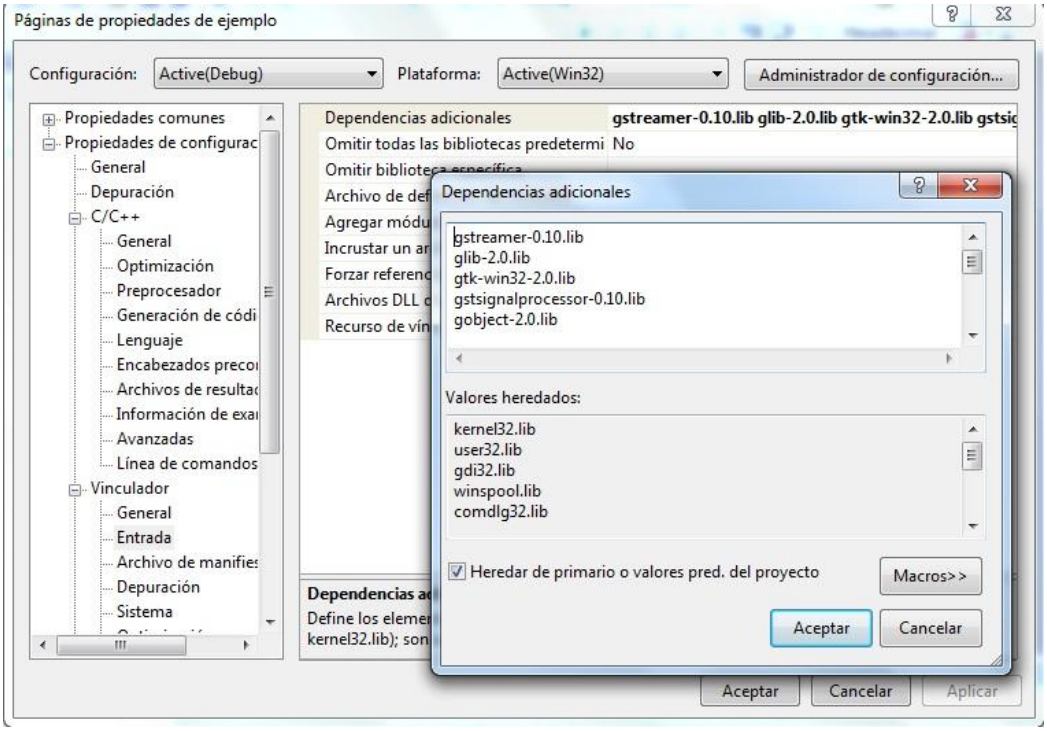


Figura A-5: Configuración de librerías en proyecto.

Anexo C. Módulos de interés de Gstreamer

Para el desarrollo del módulo de envío del flujo de vídeo nos interesan ciertos módulos de Gstreamer que son necesarios para esta función. En el módulo de envío tendremos tres partes que resolver, la primera será la introducción de los flujos de vídeo y audio al módulo de envío, la segunda será la codificación de estos flujos para mejorar las prestaciones del envío por la red y la tercera será la parte de salida del flujo a la red a través de un módulo que sea capaz de transmitir en el protocolo RTP (Real Time Protocol) que es un protocolo de transmisión en tiempo real.

Para la creación de todos los módulos que se van a utilizar se hace uso de la función gst_element_factory_make ("filesrc", "file-source"), en la que se introducirá el tipo de elemento y el nombre que se le va a asignado. En este caso el tipo de elemento es "filesrc" y el nombre asignado "file-source".

Elementos utilizados para introducción de flujos:

- **filesrc:** este elemento es el encargado de obtener de un fichero el flujo y transmitirlo al siguiente elemento. El fichero podrá ser de audio, de vídeo o de audio y vídeo simultáneamente. En el caso de contener sólo audio o vídeo será necesaria la introducción después de este elemento de otro que se encargue de la decodificación correspondiente. En el caso de contener audio y vídeo habrá que introducir un demultiplexor que se encargue de separar y seleccionar el flujo deseado.

- **audiotestsrc:** se obtiene un flujo de audio de prueba para poder comprobar el funcionamiento del *pipeline* creado.

- **videotestsrc:** se obtiene un flujo de vídeo de prueba para poder comprobar el funcionamiento del *pipeline* creado.

- **dshowvideosrc:** este elemento obtiene un flujo de vídeo a partir de capturas de pantallas del escritorio del PC. Este elemento se podría utilizar para que cuando el profesor lo desee pasar de enviar el flujo de la cámara a enviar lo que está sucediendo en el PC.

- **udpsrc:** se obtiene el flujo que se está recibiendo en un cierto puerto UDP del PC. Podría tratarse tanto de vídeo como de audio o de ambos, por eso, para la utilización de este elemento es necesario conocer que medio se está recibiendo para poder configurar correctamente el *pipeline*. Se podría utilizar para crear una aplicación para la recepción del vídeo enviado desde la aplicación principal, aunque esta parte queda resuelta gracias a VLCMediaPlayer que es capaz de recibir y reproducir el flujo recibido.

- **appsrc:** gracias a este elemento tenemos la opción de insertar un flujo desde la aplicación en desarrollo al *pipeline* de Gstreamer. Gracias a la activación de las señales “need-data” y “enough-data” el *pipeline* le comunica al elemento cuándo debe añadir información al *buffer* de lectura o cuando no necesita más información, de esta manera la aplicación le pasará la información necesaria al elemento appsrc y este a su vez se la pasará al *pipeline*.

- **ffdemux_mov_mp4_m4a_3gp_3g2_mj2:** este elmento es el encargado de separar (demultiplexar) los flujos de audio y vídeo que vienen contenidos en los formatos que expresa en el nombre. Al ser un demultiplexor, tiene una entrada y varias salidas, la conexión de este elemento se complica un poco ya que no se sabe a priori qué flujos contiene. En el caso de contener tanto audio como vídeo, habrá que realizar dos conexiones o seleccionar una de las salidas del elemento.

- **ffdemux_avi:** este es el demultiplexor para los formatos avi. Se conectará de la misma manera que el anterior demultiplexor.

Elementos utilizados para la decodificación de flujos:

- **ffdec_mp4:** este elemento se encargará de decodificar el flujo de vídeo recibido desde el demultiplexor para que el siguiente elemento en la cadena pueda realizar con él las tareas que se requieran. Como el nombre indica será un decodificador FFmpeg para el formato mp4.

- **dshowvdec_mp4:** es el decodificador de DirecthShow para formatos mp4.

Elementos utilizados para la salida de flujos:

- **autoconvert:** este elemento es una matriz de conversión entre elemento. Conoce el tipo de formato que acepta el siguiente elemento y en función del formato que emite el anterior elemento realiza la conversión automáticamente para el correcto funcionamiento del *pipeline*.

- **autovideosink:** es el elemento encargado de reproducir en el PC el flujo de vídeo final resultado del paso por el *pipeline*. Automáticamente obtiene los *drivers* necesarios para realizar la presentación del flujo de vídeo en una ventana nueva.

- **autoaudiosink:** al igual que el anterior elemento se encarga de reproducir el resultado del *pipeline*, pero en este caso se trata de audio en vez de vídeo. También se encarga automáticamente de obtener los *drivers* de la tarjeta de audio y de reproducir el flujo.

- **udpsink:** en este caso el elemento también es de salida del *pipeline*, pero en vez de reproducir el flujo obtenido, se encarga de enviarlo por internet a través del

protocolo UDP a una dirección y puerto determinados, que habrá que configurar correctamente antes de poner en funcionamiento el *pipeline*.

- **appsink:** este elemento será el encargado de obtener el flujo final del *pipeline* y devolvérselo a la aplicación en desarrollo para poder hacer con él cualquier otra cosa externa a Gstreamer que sea necesaria.

- **rtpmpapay:** este elemento se encargará de la creación de los paquetes para emitirlos en el protocolo RTP a partir de un flujo de audio.

- **rtpmpvpay:** genera los paquetes necesarios para emitir un flujo de vídeo a internet a través del protocolo RTP.

- **rtpmux:** este elemento es un multiplexor de flujos para poder emitir a internet el flujo de vídeo y el flujo de audio sincronizados. Creará los paquetes necesarios para su transmisión por el protocolo RTP.

En algunos de los elementos utilizados será necesario configurar alguno de sus parámetros. En el elemento filesrc habrá que definir el *path* del fichero que se va a tomar como fuente. Para ello será necesaria la utilización de la siguiente función:

- g_object_set(G_OBJECT (file-src), "location", "file.avi", NULL);

En ella se indica el *path* del archivo que se quiere abrir. En el elemento udpsink habrá que indicar la dirección IP a la que se quiere transmitir y el puerto al que van dirigidos los paquetes, para ello haremos uso de las siguientes instrucciones:

- g_object_set (G_OBJECT (udp-sink), "host", “192.168.0.0”, NULL);

- g_object_set (G_OBJECT (udp-sink), "port", 5001, NULL);

En ellas se indicara en la propiedad *host* la dirección, pudiendo ser esta la dirección de una red y así obtener un envío *multicast* a los distintos ordenadores de esa red. En la propiedad *port* se indicará el puerto deseado.

Para la interpretación de las señales de alguno de los elementos usaremos las siguientes asignaciones, de manera que cuando se active la señal indicada, el programa se dirigirá a la función que se ha creado expresamente para resolver la interrupción:

- g_signal_connect (app-src, "need-data", G_CALLBACK (need_data), NULL);
- g_signal_connect (app-src, "enough-data", G_CALLBACK (enough_data), NULL);

Anexo D. Utilización de las aplicaciones desarrolladas

Para usar las aplicaciones desarrolladas en este proyecto no es necesario realizar ningún tipo de instalación de las mismas, simplemente será necesario copiar la carpeta que contiene la aplicación y sus correspondientes librerías al directorio deseado del PC manteniendo la estructura de archivos tal y como se tienen en la carpeta.

Para el funcionamiento correcto de EPS-CfgGenerator es necesario que la carpeta contenga el ejecutable EPS-CfgGenerator.exe y el fichero EPS-RemoteViewer.cfg en el que se guardarán los datos de conexión.

- **EPS-CfgGenerator**:
  - EPS-CfgGenerator.exe
  - EPS-RemoteViewer.cfg

Para el funcionamiento de EPS-RemoteViewer la carpeta contendrá la librería sncstrm.dll y otra carpeta con el ejecutable EPS-Remoteviewer.exe, la librería SonyNetworkCamera4.dll y el fichero de configuración EPS-RemoteViewer.cfg.

- **EPS-RemoteViewer:**
  - sncstrm.dll
  - **EPS-RemoteViewer:**
    - EPS-RemoteViewer.exe
    - SonyNetworkCamera4.dll
    - EPS-Remoteviewer.cfg

Para el funcionamiento de la aplicación principal CamApplication es necesario que la carpeta principal contenga la librería sncstrm.dll y otra carpeta con el ejecutable CamApplication.exe, el fichero de configuración de las posiciones predeterminadas y de la calibración con el nombre correspondiente a la IP de la cámara del aula (cam-aula5.ii.uam.es.cfg) y las librerías correspondientes.

- **CamApplication:**
    - sncstrm.dll
    - **CamApplication:**
        - CamApplication.exe
        - Ip-camara.cfg
        - SonyNetworkCamera4.dll

Teniendo estas tres c**a**rpetas con los archivos necesarios en sus correspondientes directorios, se podrá ejecutar cualquiera de las aplicaciones en cualquier PC sin necesidad de realizar ningún tipo de instalación.

En las siguientes secciones se va a explicar cómo un usuario debe proceder para poner en funcionamiento cada una de las aplicaciones desarrolladas en este proyecto.

- **EPS-CfgGenerator**

Esta aplicación es la encargada de crear un archivo, *EPS-RemoteViewer.cfg*, en el que va cifrada la información de conexión con la cámara. Sin este fichero la aplicación de visualización y control del contenido de la cámara no se podrá ejecutar.

Una vez ejecutada la aplicación EPS-CfgGenerator.exe tendremos la interfaz gráfica mostrada en la Figura A-6, en la que se deberán introducir los datos de conexión con la cámara, los datos correspondientes al usuario y el rango de fechas en el que se otorgará acceso al usuario.

En el campo de texto correspondiente a Address se introducirá el nombre de *host* o la dirección IP correspondiente a la cámara, en el campo de Port No se introducirá el puerto de conexión, que en este caso será el 80. Los campos User Name y Password contendrán

los datos de uno de los usuarios que estén previamente configurados en la cámara y en los campos correspondientes a Validez se introducirán la fecha de inicio y la fecha de final entre las cuales el usuario podrá tener acceso a la visualización del flujo y el control de la cámara (en función de los permisos configurados para el usuario en la cámara).

Figura A-6: EPS-CfgGenerator, introducción de datos de conexión.

Una vez se han introducido todos los datos requeridos por la aplicación y tras pulsar el botón Generate, se creará junto al ejecutable de la aplicación el archivo *EPS-RemoteViewer.cfg* que contendrá los datos cifrados. Este archivo habrá que copiarlo a la misma carpeta que contiene el ejecutable de la aplicación de visualización para que ésta se pueda conectar correctamente con la cámara (en caso de ser un usuario válido y estar dentro del rango de fechas previsto).

- **EPS-RemoteViewer**

Para poder ejecutar EPS-RemoteViewer correctamente deberemos tener junto al ejecutable de la aplicación el archivo de configuración *EPS-RemoteViewer.cfg* con datos válidos. De esta manera la aplicación tendrá los datos necesarios de conexión con la cámara y al ejecutarla tendremos el acceso a la cámara que en él se encuentra configurado.

En la Figura A-7 vemos la aplicación EPS-RemoteViewer en funcionamiento. Se puede observar el grupo de botones de control del movimiento de la cámara, que cuenta con movimientos fijos en todas las direcciones e incluso movimientos combinados para generar

movimientos en diagonal y por último, el botón Home hará que la cámara se coloque en la posición predefinida como tal. Otro grupo de botones que se puede observar es el de control del *zoom*, que cuenta con un botón para aumentar el *zoom* y otro para disminuirlo. Por último se puede observar el grupo de control del vídeo con los botones de Play y Stop que lo que hacen es comenzar o terminar la reproducción del audio y del vídeo que se está siendo capturando en ese instante por la cámara, aunque el audio se puede desactivar seleccionando el Audio Mute.

En el caso de tener un usuario restringido, por mucho que se pulsasen los botones de movimiento o ajuste del *zoom*, la cámara no haría caso de las instrucciones, sólo serían funcionales los botones de Play, Stop, Audio Mute y Exit de la aplicación.

Figura A-7: EPS-RemoteViewer.

- **Aplicación principal**

Como aplicación principal se ha desarrollado CamApplication, que permite la conexión con la cámara, visualización, control del movimiento, ajuste del *zoom*, selección de posiciones predefinidas, seguimiento automático y la posibilidad de integrar el servicio de *streaming*.

En la Figura A-8 se muestra la interfaz gráfica de la aplicación principal, CamApplication, pudiéndose ver en ella la zona de visualización del flujo de vídeo capturado por la cámara además de varios grupos de botones y campos de texto.

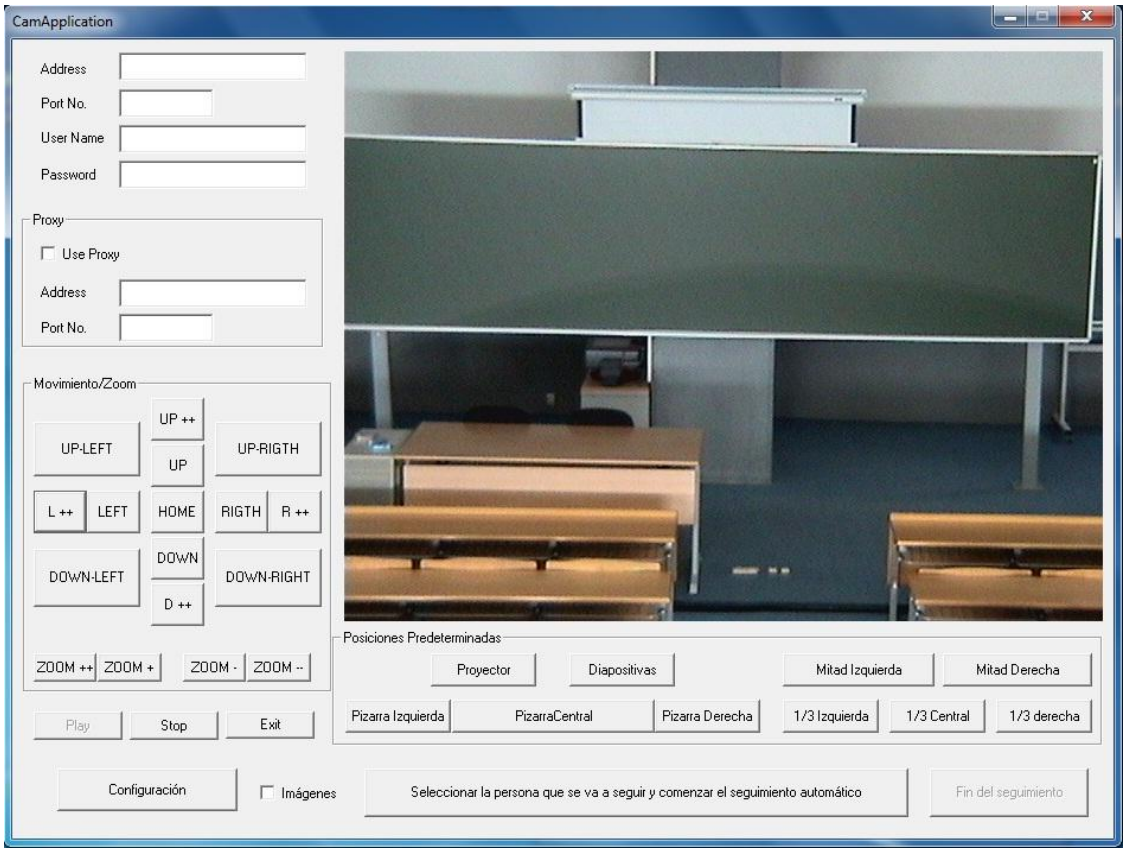


Figura A-8: Interfaz gráfica de CamApplication.

En los campos de texto se deberán introducir los siguientes datos:

- **Dirección:** se podrá introducir la dirección IP o el nombre del *host* en el caso de estar configurado.
- **Número de puerto:** se deberá introducir el número de puerto al que nos queremos conectar, que en este caso es el 80.
- **Nombre de usuario:** habrá que introducir uno de los usuarios previamente configurados en la cámara, el cual tendrá ciertos privilegios que se le hayan otorgado.
- **Contraseña:** en este parámetro se introducirá la contraseña asociada al usuario introducido en el campo anterior.

En el caso de estar utilizando un proxy entre la aplicación y la cámara habrá que indicar a la aplicación que se está utilizando e introducir tanto la dirección IP como el puerto al que hay que conectarse del proxy.

El primer grupo de botones que se observa es el de Movimiento/Zoom, que como su nombre indica servirá para controlar los movimientos de la cámara y los ajustes del *zoom* que se quieran realizar. Entre los botones de movimiento se encuentran botones de movimiento corto en todas las direcciones y unos botones de movimiento más amplio en los ejes vertical y horizontal, que podemos distinguir de los botones de movimiento corto porque en su nombre aparece la dirección de movimiento acompañada de '++', que indica que será un movimiento más amplio. Esta misma anotación se puede observar en los botones de ajuste del *zoom* que realizarán un cambio de *zoom* más grande, tanto para incrementarlo como para disminuirlo.

Otro grupo de botones que podemos observar es el llamado Posiciones Predeterminadas, gracias al cual se puede colocar la cámara enfocando una cierta región de la pizarra, el proyector o la pantalla de diapositivas con un solo *click*. Las posiciones predeterminadas hacia las que se desplazará la cámara son las siguientes:

- **HOME:** vista general del aula en la que se enfoca toda la zona de movimiento del profesor, viéndose todas las pizarras, las dos zonas de proyección (diapositivas y proyector) y algunas filas de pupitres.
- **POYECTOR:** vista de la zona de proyección a través del proyector posterior de aula.
- **DIAPOSITIVAS:** vista de la zona de proyección de diapositivas.
- **PIZARRA IZQUIERDA:** vista de la pizarra pequeña situada a la izquierda del aula.
- **PIZARRA CENTRAL:** vista completa de la pizarra central.
- **PIZARRA DERECHA:** vista de la pizarra pequeña situada a la derecha del aula.
- **MITAD IZQUIERDA:** vista de la zona mitad izquierda de la pizarra central.

- **MITAD DERECHA:** vista de la zona mitad derecha de la pizarra central.
- **TERCIO IZQUIERDO:** vista de la zona que ocupa el tercio izquierdo de la pizarra central.
- **TERCIO CENTRAL:** vista de la zona que ocupa el tercio central de la pizarra central.
- **TERCIO DERECHO:** vista de la zona que ocupa el tercio derecho de la pizarra central.

En la Figura A-9 se observa cada una de las zonas enfocadas por la cámara tras hacer *click* sobre el botón correspondiente a su posición predeterminada.

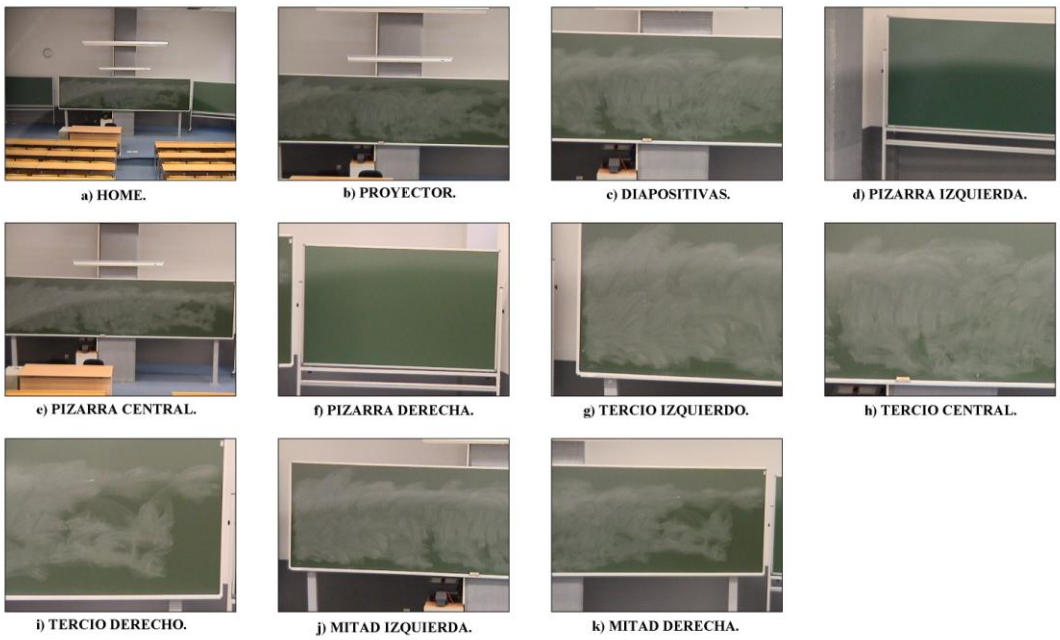


Figura A-9: Visualización de las posiciones predeterminadas de la cámara.

Estas posiciones predeterminadas se podrán configurar pulsando el botón de Configuración. Con este botón también se podrá configurar el *zoom* más adecuado y realizar la calibración del movimiento, ambas, acciones necesarias para que el seguimiento automático funcione correctamente.

Finalmente tenemos el botón de selección de la persona a seguir y comienzo del seguimiento automático y el de finalización del seguimiento. Con ellos se inicializará el

proceso de seguimiento automático seleccionando la cara de la persona a seguir; a partir de ahí la aplicación ordenará los movimientos necesarios a la cámara para tener siempre enfocado al profesor, y pulsando el botón de finalización en cualquier momento se parará el seguimiento y se pasará a tener que ordenar los movimientos por pulsado de botones.

Para la selección de la cara del profesor aparecerá una ventana con la última imagen capturada y habrá que hacer un recuadro con el puntero del ratón que recoja la cara al completo. Si hay algún fallo en la inicialización del seguimiento automático, un mensaje nos indicará que hay que realizar una selección más adecuada.

En la Figura A-10 se puede observar cómo se debe seleccionar la cara, y una vez hecha una selección correcta pulsaremos el botón SELECCIÓN OK del diálogo mostrado.

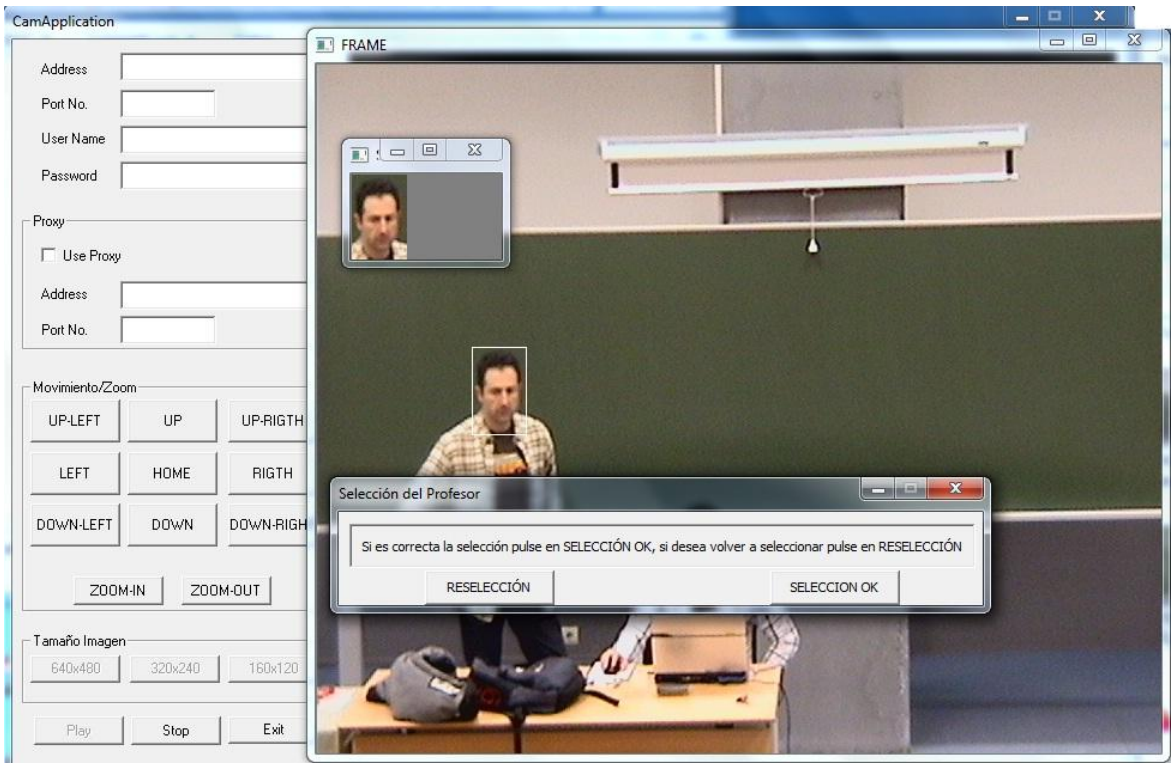


Figura A-10: Selección de la cara del profesor para el seguimiento automático.

En este momento se lanzará el algoritmo de seguimiento automático, y sólo se podrá detener pulsando el botón de Fin del Seguimiento.

PRESUPUESTO

1) Ejecución Material

- Compra de ordenador personal (Software incluido)..... 2.000 €
- Compra de cámara IP..... 2.000 €
- Alquiler de impresora láser durante 6 meses ..... 50 €
- Material de oficina ..... 150 €
- Total de ejecución material..... 4.200 €

2) Gastos generales

- 16 % sobre Ejecución Material ..... 672 €

3) Beneficio Industrial

- 6 % sobre Ejecución Material ..... 252 €

4) Honorarios Proyecto

- 840 horas a 15 € / hora..... 12600 €

5) Material fungible

- Gastos de impresión..... 80 €
- Encuadernación..... 200 €

6) Subtotal del presupuesto

- Subtotal Presupuesto..... 18004 €

7) I.V.A. aplicable

- 18% Subtotal Presupuesto ..... 3240,72 €

8) Total presupuesto

- Total Presupuesto..... 21244,72€

Madrid, abril de 2012

El Ingeniero Jefe de Proyecto

Fdo.: José Luis Gamo Revilla
Ingeniero Superior de Telecomunicación

PLIEGO DE CONDICIONES

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de un sistema de emisión de clases presenciales. 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.